M. Van Droogenbroeck, and O. Barnich. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

ViBe: a disruptive method for background subtraction*

Marc Van Droogenbroeck and Olivier Barnich

August 2014

1 Introduction

The proliferation of the video surveillance cameras, which accounts for the vast majority of cameras worldwide, has resulted in the need to find methods and algorithms for dealing with the huge amount of information that is gathered every second. This encompasses processing tasks, such as raising an alarm or detouring moving objects, as well as some semantic tasks like event monitoring, trajectory or flow analysis, counting people, etc.

Many families of tools related to motion detection in videos are described in the literature (see [21] for tools related to the visual analysis of humans). Some of them track objects, frame by frame, by following features in the video stream. Others operate by comparing the current frame with a static background frame, pixel by pixel. This is the basis of *background subtraction*, whose principle is very popular for fixed cameras. The purpose of background subtraction is therefore formulated as the process to extract moving objects by detecting zones where a significant change occurs. Moving objects are referred to as the *foreground*, while static elements of the scene are part of the *background*. Practice however shows that this distinction is not always obvious. For example, a static object that starts moving, such as a parked car, creates both a hole in the background and an object in the foreground. The hole, named *ghost*, is usually wrongly assigned to the foreground despite that it should be discarded as there is no motion associated to it. Another definition of the background considers that background pixels should correspond to values visible most of the time, or in statistical terms, whose probabilities are the highest. But this poses problems when the background is only visible for short periods of time, like a road with heavy traffic. The diversity of scenes and specific scenarios explains why countless papers have been devoted to background subtraction, as well as additional functionalities such as the detection of shadows or the robustness to camera jitter, etc.

In this chapter, we elaborate on a background subtraction technique named ViBe, that has been described in three previous papers [1, 2, 37], and three granted patents [34, 35, 36]. After some general considerations, we describe the principles of ViBe in Section 2. We review the innovations introduced with ViBe: a background model, an update mechanism composed of random substitutions and spatial diffusion, and a fast initialization technique. We also discuss some enhancements that broaden up the possibilities for improving background subtraction techniques, like the distinction between the segmentation map and the updating mask, or a controlled diffusion mechanism. Section 3 discusses the computational cost of ViBe. In particular, we introduce the notion of *background subtraction complexity factor* to express the speed of the algorithm, and show that ViBe has a low complexity factor. Section 4 concludes the chapter.

*This is an updated version of the original chapter (the main update relates to the computation times of Table 2). These numbers were obtained with a library available at http://www2.ulg.ac.be/telecom/research/vibe/doc.

M. Van Droogenbroeck, and O. Barnich. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

1.1 What are we looking for?

The problem tackled by background subtraction involves the comparison of the last frame of a video stream with a reference background frame or model, free of moving objects. This comparison, called *foreground detection*, is a classification process that divides the image into two complementary sets of pixels (there is no notion of object at this stage): (1) the *foreground*, and (2) the *background*. This binary classification process is illustrated in Figure 1.

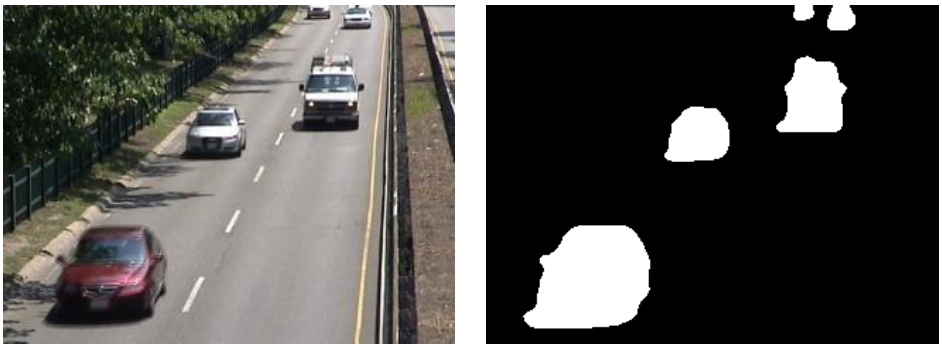

Figure 1: The image of a road with moving cars (from the baseline/highway directory of the "Change Detection" dataset [10]) and the corresponding binary segmentation map (produced by ViBe). Foreground pixels are drawn in white.

It results in a binary output map, called *segmentation map* hereafter. For humans, it might seem obvious to delineate moving objects, because humans incorporate knowledge from the semantic level (they know what a car is, and understand that there are shadows). However, such knowledge is not always available for computers to segment the image. Therefore segmentation is prone to classification errors. These errors might disturb the interpretation, but as long as their number is small, this is acceptable. In other words, there is no need for a perfect segmentation map. In addition, as stated in [27], it is almost impossible 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. Therefore, initiatives such as the i-LIDS evaluation strategy [13] are directed more towards the definition of scenarios than the performance of background subtraction.

1.2 Short review of background subtraction techniques

Many background subtraction techniques have been proposed, and several surveys are devoted to this topic (see for example [3, 5, 9]). Background subtraction requires to define an underlying background model and an update strategy to accommodate for background changes over time. One common approach to background modeling consists to assume that background pixel values (called background samples) observed at a given location are generated by a random variable, and therefore fit a given probability density function. Then it is sufficient to estimate the parameters of the density function to be able to determine if a new sample belongs to the same distribution. For example, one can assume that the probability density function is a gaussian and estimate its two parameters (mean and variance) adaptively [39]. A simpler version of it considers that the mean which, for each pixel, is stored in a memory and considered as the background model, can be estimated recursively by

$$B_t = \alpha B_{t-1} + (1-\alpha) I_t, \tag{1}$$

where B_t and I_t are the background model and current pixel values at time t respectively, and α is a constant ($\alpha = 0.05$ is a typical value). We name this filter the *exponential filter* because B_t

adapts to I_t exponentially. With this filter, the decision criterion is simple: if I_t is close to B_{t-1}, that is if the difference between them is lower than a fixed threshold, $|I_t - B_{t-1}| \leq T$, then I_t belongs to the background. This filter is one of the simplest algorithm for background subtraction, except the algorithm that uses a fixed background. There is one comparison and decision per pixel, and the background model is updated once per processed frame. To evaluate the computational speed of ViBe, we use this filter as one reference in Section 3.

The exponential filter is simple and fast. When it comes to computational speed and embedded processing, methods based on $\Sigma - \Delta$ (sigma-delta) motion detection filters [18, 20] are popular too. As in the case of analog-to-digital converters, a $\Sigma - \Delta$ motion detection filter consists of a simple non-linear recursive approximation of the background image, which is based on comparisons and on an elementary increment/decrement (usually -1, 0, and 1 are the only possible updating values). The $\Sigma - \Delta$ motion detection filter is therefore well suited to many embedded systems that lack a floating point unit.

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 [3]. Over the years, increasingly complex pixel-level algorithms have been proposed. Among these, by far the most popular is the Gaussian Mixture Model (GMM). First presented in [32], this model consists of modeling the distribution of the values observed over time at each pixel by a weighted mixture of gaussians. Since its introduction, the model has gained vastly in popularity among the computer vision community (see the excellent extended review by Bouwmans [4]), and it is still raising a lot of interest as authors continue to revisit the method.

One of the downsides 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. Also, the estimation of the parameters of the model (especially the variance) can become problematic in real-world noisy environments. This sometimes leaves one with no other practical choice than to use a fixed variance. 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 [31].

Because the determination of parameters can be problematic, and in order to avoid the difficult question of finding an appropriate shape for the probability density function, some authors have turned their attention to non-parametric methods to model background distributions. One of the strengths of non-parametric kernel density estimation methods is their ability to circumvent a part of the delicate parameter estimation step due to the fact that they rely on sets of pixel values observed in the past to build their pixel models. 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. Non-parametric 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 [8]. 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. ViBe incorporates a smoother lifespan policy for the sampled values. More importantly, it makes no assumption on the obsolescence of samples in the model. This is explained in Section 2.

The background subtraction method that is the closest to ViBe was proposed in [38]. Wang

and Suter base their technique on the notion of "consensus". They keep the 100 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. ViBe has a similar approach except that the amount of stored values is limited to 20, thanks to a clever selection policy.

While mixtures of gaussians and non-parametric models are background subtraction families with the most members, other techniques exist. In [15], each pixel is represented by a codebook, which is a compressed form of background model for a long image sequence. Codebooks are believed to be able to capture background motion over a long period of time with a limited amount of memory.

A different approach to background subtraction consists to consider that it is not important to tune a background subtraction, but instead that the results of the classification step can be refined. A two-level mechanism based on a classifier is introduced in [17]. 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 on the results of the classification. Schick et al. [28] use a similar idea and define a superpixel Markow random field to post-process the segmentation map. They show that their technique improves the performance of background subtraction most of the time; only results of background subtraction techniques that already produce accurate segmentations are not improved.

This raises the general question of applying a post-processing filter to the segmentation map. Parks et al. [25] consider several post-processing techniques that can be used to improve upon the segmentation maps: noise removal, blob processing (morphological closing is used to fill internal holes and small gaps whereas area thresholding is used to remove small blobs), etc. Their results indicate that the performance is improved for square kernel morphological filter as long as the size is not too large. The same yields for area thresholding where a size threshold well below the size of the smallest object of interest (e.g. 25%) is recommended. The post-processing is also interesting from a practical perspective. When it is applied, the segmentation step becomes less sensitive to noise, and subsequently to the exact values of the method's parameters. To produce the right-hand side image of Figure 1, we have applied a 5×5 close/open filter and a median filter on the segmentation map. For ViBe, Kryjak and Gorgon [16] propose to use a 7×7 median filter to post-process the segmentation map, and two counters per pixel, related to the consecutive classification as foreground or background, to filter out oscillating pixels.

1.3 Evaluation

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. The "Change Detection" (CD) initiative is a laudable initiative to help comparing algorithms. The CD dataset contains 31 video sequences panning a large variety of scenarios and includes groundtruth maps. The videos are grouped in 6 categories: baseline, dynamic background, camera jitter, intermittent object motion, shadow, and thermal. The CD web site (http://www.changedetection.net) computes several metrics for each video separately.

These metrics rely on the assumption that the segmentation process is similar to that of a binary classifier, and they 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. As stated by Goyette et al. [10], finding the right metric to accurately measure the ability of a method to detect motion or change without producing excessive false positives and false negatives is not trivial. For instance, the *recall* metric favors methods with a

M. Van Droogenbroeck, and O. Barnich. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

low False Negative Rate. On the contrary, the *specificity* metric favors methods with a low False Positive Rate. Having the entire precision-recall tradeoff curve or the ROC curve would be ideal, but not all methods have the flexibility to sweep through the complete gamut of tradeoffs. In addition, one cannot, in general, rank-order methods based on a curve.

One of the encountered difficulties is that there are four measurable quantities (TP, TN, FP, FN) for a binary classifier, and a unique criterion based on them will not reflect the trade-offs when parameter values are to be selected for the classifier. This is the reason why the characterization by multiple criteria helps determining the optimal parameter values. Another difficult question is the comparison of techniques. Again, there is a difference in optimizing a technique and the willingness to compare techniques. In this chapter, we do not enter the discussion of comparing ViBe to other background subtraction techniques (see [10] for a comparative study of techniques, including two variants of ViBe: PBAS and ViBe+). We prefer to concentrate on aspects proper to ViBe and to discuss some choices. Because ViBe has a conservative updating policy, it tends to minimize the amount of False Positives. As a consequence, precision is high. But simultaneously, ViBe has a inherent mechanism to incorporate ghosts and static objects into the background. For some applications, this results in an increase of the number of False Negatives. If foreground objects only represent a very small part of the image, which is common in video-surveillance, the amount of True Positives is small, and then the recall is low. Therefore, we select the Percentage of Bad Classifications (PBC), that is a combination of the four categories, as a compromise and as the evaluation criteria. It is defined as

$$PBC = 100 \times \frac{FN + FP}{TP + FN + FP + TN}. \tag{2}$$

Note the PBC assesses the quality of the segmentation map. For some applications however, the quality of segmentation is not the main concern. For example, the exact shape of the segmentation map is not relevant if the role of the background subtraction is to raise an alarm in the presence of motion in the scene. Then a criterion based on the rate of False Negatives might be more appropriate.

2 Description of ViBe

The background/foreground classification problem that is handled by background subtraction requires to define a model and a decision criterion. In addition, for real time processing of continuous video streams, pixel-based background subtraction techniques compensate for the lack of spatial consistency by a constant updating of their model parameters.

In this section, we first explain the rationale behind the mechanisms of ViBe. Then, we describe the model (and some extensions), which is intrinsically related to the classification procedure, an updating strategy, and initialization techniques.

2.1 Rationale

ViBe is a technique that collects background samples to build background models, and introduces new updating mechanisms. The design of ViBe was motivated by the following rules:

- many efficient vision-based techniques that classify objects, including the successful pose recognition algorithm of the Kinect [29], operate at the pixel level. Information at the pixel level should be preferred to aggregated features, because the segmentation process is local and dealing with pixels directly broadens up possibilities to optimize an implementation for many hardware architectures.

- we prefer the approach that collects samples from the past for each pixel, rather than building a statistical model. There are two reasons for that. Firstly, because sample values have been observed in the past, their statistical significance is higher than values that have never been measured. Secondly, the choice of a model introduces a bias towards that model. For example, if one assumes that the probability distribution function of a pixel is gaussian, then the method tries to determine its mean and variance. It does not test the relevance of the model itself, even if the distribution appears to be bi-modal.

- the number of collected samples should be kept relatively small. Taking larger numbers both increases the computational load and memory usage. In ViBe, we keep 20 background samples for each pixel. This might seem to be a large figure, but remember that at a typical video framerate is 25 or 30 Hz; keeping 20 background samples therefore represents a memory of less than 1 second.

- there should be no "planned obsolescence" notion for samples. The commonly (and exclusively) adopted substitution rule consists to replace oldest samples first or to reduce their associated weight. This assumes that old samples are less relevant than newer samples. But this assumption is questionable, as long as the model only contains background samples and is not corrupted by foreground samples. In ViBe, old and recent values are considered equally when there are replaced.

- a mechanism to ensure spatial consistency should be foreseen. Pixel-based techniques tend to ignore neighboring pixel values in their model. This allows for a sharp detection capability, but provides no warranty that neighboring decisions are consistent. ViBe proposes a new mechanism that consists to insert background values in the models of neighboring pixels. Once the updating policy decides to replace a value of the model with the current value of the pixel, it also inserts that value in the model of one of the neighboring pixels.

- the decision process to determine if a pixel belongs to the background should be kept simple because, for pixel-based classification approaches, the computational load is directly proportional to the decision process. As shown in Section 3, the overall computational cost of ViBe is about that of 6 comparisons per pixel, which is very low compared to other background subtraction strategies.

In the next section, we present the details of ViBe. There are differences with the original version of ViBe (as described in [2]) that we point out when appropriate.

2.2 Pixel classification

Classical approaches to background subtraction and most mainstream techniques rely on a probability density function (pdf) or statistical parameters of the underlying background generation process. But the question of their statistical significance is rarely discussed, if not simply ignored. In fact, the choice of a particular form for the pdf inevitably introduces a bias towards that pdf, when 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.

If we see the problem of background subtraction as a classification problem, we want to classify a new pixel value with respect to previously observed samples by comparing them. A major, and important difference in comparison with existing algorithms, is that when a new value is compared to background samples, it should be close to only a few sample values to be classified as background, instead of the majority of all sample values. Some authors believe that comparing a

M. Van Droogenbroeck, and O. Barnich. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

value with some sample values is equivalent to a non-parametric model that sums up probabilities of uniform kernel functions centered on the sample values. That is an acceptable statement for the model of [38], that considers 100 values and require a close match with at least 90 values, because the model then behaves as a pdf whose values are compared to the new value to determine if the probability is larger than 0.9. However, this reasoning is not valid for ViBe, because with the standard set of values, we only require 2 matches out of 20 possible matches, which would correspond to a probability threshold of 0.1. One can hardly claim that such a low probability is statistically significant, and consequently that the model is that of a pdf. Of course, if one trusts the values of the model, it is crucial to select background pixel samples carefully. The insertion of pixels in the background therefore needs to be conservative, in the sense that only *sure* background pixels should populate the background models.

To be more formal, let us denote by $v(p)$ the value of a feature, for example the RGB components, taken by the pixel located at p in the image, and by v_i a background feature with an index i. Each background pixel at p is then modeled by a collection of N background samples

$$\mathcal{M}(p) = \{v_1, v_2, \ldots, v_N\}, \tag{3}$$

taken in previous frames.

In order to classify a new pixel feature value $v(p)$, we compare it to all the values contained in the model $\mathcal{M}(p)$ with respect to a distance function $d()$ and a distance threshold T. The classification procedure is detailed in Algorithm 1.

Algorithm 1 Pixel classification algorithm of ViBe.

```
1  int width;                                  // width of the image
2  int height;                                 // height of the image
3  byte image[width*height];                   // current image
4  byte segmentationMap[width*height];         // classification result
5
6  int numberOfSamples = 20;       // number of samples per pixel
7  int requiredMatches = 2;        // #_min
8  int distanceThreshold = 20;
9
10 byte samples[width*height][numberOfSamples]; // background model
11
12 for (int p = 0; p < width*height; p++) {
13   int count=0, index=0, distance=0;
14
15   // counts the matches and stops when requiredMatches are found
16   while ( (count < requiredMatches) && (index < numberOfSamples) ) {
17     distance = getDistanceBetween(image[p], samples[p][index]);
18     if (distance < distanceThreshold) {
19       count++; }
20     index++;
21   }
22
23   // pixel classification
24   if (count < requiredMatches)
25     segmentationMap[p] = FOREGROUND_LABEL;
26   else
27     segmentationMap[p] = BACKGROUND_LABEL;
28 }
```

A typical choice for $d()$ is the Euclidean distance, but any metric that measures a match between two values is usable. Benezeth et al. [3] compared several metrics for RGB images. They conclude

that four of the six metrics they tested, including the common Euclidean distance, globally produce the same classification results; only the simplest zero and first order distances are less precise. In many video-surveillance applications, $v(p)$ is either the luminance or the RGB components. For the luminance, the distance $d()$ is reduced to the absolute difference between the pixel value and a sample value. As shown in the algorithm, it is important to note that, for each pixel, the classification process stops when $\#_{min}$ matches have been found; to the contrary of non-parametric methods, there is no need to compare $v(p)$ to every sample. This speeds up the classification process considerably. Discussions about the computational load of ViBe are further given in Section 3.

2.3 Updating

The classification step of our algorithm compares the current pixel feature value at time t, $v^t(p)$, directly to the samples contained in the background model $\mathcal{M}^{t-1}(p)$, built at time $t-1$. Consequently, questions regarding *which* samples have to be memorized by the model and for *how long* are essential.

Many practical situations, like the response to sudden lighting changes, the presence of new static objects in the scene, or changing backgrounds, can only be addressed correctly if the updating process incorporates mechanisms capable to cope with dynamic changes in the scene. The commonly adopted approach to updating consists to discard and replace old values after a number of frames or after a given period of time (typically about a few seconds); this can be seen as a sort of *planned obsolescence*. Although universally applied, there is no evidence that this strategy is optimal. In fact, ViBe proposes other strategies, based on random substitutions, that when combined, improve the performance over time.

General discussions on an updating mechanism

Many updating strategies have been proposed [4] and, to some extent, each background subtraction technique has its own updating scheme. Updating strategies are either intrinsically related to the model, or they define methods for adapting parameters over time. Considering the model, there are two major updating techniques. Parks et al. [25] distinguish between *recursive* techniques, which maintain a single background model that is updated with each new video frame, and *non-recursive* techniques which maintain a buffer of n previous video frames and estimate a background model based solely on the statistical properties of these frames.

Another typology relates to the segmentation results. *Unconditional* updating or *blind* updating considers that, for every pixel, the background model is updated, whereas in *conditional* updating, also called *selective* or *conservative* updating, only background pixels are updated. Both updating schemes are used in practice. Conservative updating leads to sharp foreground objects, as the background model will not become polluted with foreground pixel information. However the major inconvenience of that approach is that false positives (pixels incorrectly classified as foreground values) will continually be misclassified as the background model will never adapt to it. Wang et al. [38] propose to operate at the blob level and define a mechanism to incorporate pixels in the background after a given period of time. To the contrary, blind updating tends to remove static objects and requires additional care to keep static objects in the foreground.

Other methods, like 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. For example, Porikli et al. [26] define a GMM method and a Bayesian updating mechanism, to achieve accurate adaptation of the models.

With ViBe, we developed a new conservative updating method that incorporates three important mechanisms: (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 discussed in details in [2]. We remind their key features in the next three subsections, and provide some additional comments.

A memoryless updating 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. [38] propose the inclusion of large numbers of samples in pixel models. Others authors [8, 40] mix two temporals sub-models to handle both fast and slow changes. These approaches are effective, but then the management of multiple temporal windows is complex. In addition, they might not be sufficient for high or variable framerates.

From both practical and theoretical perspectives, we believe that it is more appropriate for background values to fade away smoothly. In terms of a non-parametric model such as ViBe, this means that the probability for a value to be replaced over time should decay progressively. Therefore, we define a policy that consists to replace sample values chosen randomly, according to a uniform probability density function. This guarantees a monotonic decay of the probability for a sample value to remain inside the set of samples, as established in [2]. There are two remarkable consequences to this updating policy: (1) there is no underlying notion of time window in the collection of samples, and (2) the results of the background subtraction are not deterministic anymore. In other words, if the same video sequence is processed several times, all the results will be slightly, but not significantly, different. Remember that this approach is necessarily combined with a conservative updating policy, so that foreground values should never be absorbed into the background models. The conservative updating policy is necessary for the stability of the process to avoid that models diverge over time.

The complete updating algorithm of ViBe is given in Algorithm 2. The core of the proposed updating policy is described at lines 18-19: one sample of the model is selected randomly and replaced by the current value. Note that this process is applicable to a scalar value, but to an RGB color vector or even more complex feature vectors as well. In fact, except for the distance calculations between values or feature vectors and their corresponding elements in the background model, all principles of ViBe are transposable as such to any feature vector.

A random updating strategy contradicts the belief that a background subtraction technique should be entirely deterministic and predictable. From our perspective, there is no reason to prefer an updating policy that would replace the oldest sample first as it reduces the temporal window dramatically. In addition, it could happen that a dynamic background has a cycle that is longer than the temporal window. The model would then not be able to characterize that background. Likewise, dynamic backgrounds with zones of different time frequencies will be impossible to handle. With our updating policy, the past is not "ordered"; one could say that the past has no effect on the future. This property is called the memoryless property (see [24]). We believe that many background subtraction techniques could benefit from this updating policy.

Until now, we have considered pixels individually; this is part of the design of pixel based methods. However, the random updating policy has introduced a spatial inhomogeneity. Indeed, the index of the sample values that is replaced depends on the result of a random test and, therefore, differs from one location to another one. This is a second important element of the design of ViBe: we should consider the spatial neighborhood of a pixel, and not only the pixel itself. Two mechanisms are used for that: (1) ensure that all the pixels are processed differently, and (2) diffuse information locally. The random selection of an index, that defines which values should be replaced, is a first method to process pixels differently. In the following, we introduce two additional mechanisms:

M. VAN DROOGENBROECK, and O. BARNICH. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

Algorithm 2 Updating algorithm of ViBe.

```
1  int width;                        // width of the image
2  int height;                       // height of the image
3  byte image[width*height];         // current image
4  byte updatingMask[*height];       // updating mask (1==updating allowed)
5
6  int subsamplingFactor = 16;       // amount of random subsampling
7
8  int numberOfSamples = 20;         // number of samples per pixel
9  byte samples[width*height][numberOfSamples]; // background model
10
11 for (int p = 0; p < width*height; p++)
12   if (updatingMask[p] == 1) { // updating is allowed
13
14     // eventually updates the model of p (in-place updating)
15     int randomNumber = getRandomIntegerIn(1, subsamplingFactor);
16     if (randomNumber == 1) { // random subsampling
17       // randomly selects a sample in the model to be replaced
18       int randomNumber = getRandomIntegerIn(0, numberOfSamples - 1);
19       samples[p][randomNumber] = image[p];
20     }
21
22     // eventually diffuses in a neighboring model (spatial diffusion)
23     int randomNumber = getRandomIntegerIn(1, subsamplingFactor);
24     if (randomNumber == 1) { // random subsampling
25       // chooses a neighboring pixel randomly
26       q = getPixelLocationFromTheNeighborhoodOf(p);
27       // diffuses the current value in the model of q
28       // (uncomment the following check to inhibit diffusion across the
             border of the updating mask)
29       // if (updatingMask[q] == 1) {
30         int randomNumber = getRandomIntegerIn(0, numberOfSamples - 1);
31         samples[q][randomNumber] = image[p];
32       // }
33     }
34
35   } // end of "if"
36 } // end of "for"
```

time subsampling, to increase the spatial inhomogeneity, and a *spatial diffusion* mechanism. We also show that the diffusion mechanism is an essential part of ViBe.

Time subsampling

With the random index selection of samples described in the previous section, we have suppressed an explicit reference to time in the model. The use of a random replacement policy allows the sample collection to cover a large, theoretically infinite, time window with a limited number of samples. In order to extend the size of the time window even more, we resort to *random time subsampling*. The main idea is that background is a slow varying information, except for the cases of a sudden global illumination change or scene cuts that need a proper handling at the frame level. Therefore, it is not necessary to update each background model for each new frame. By making the background updates less frequent, we artificially extend the expected lifespan of the background samples and ensure a spatial inhomogeneity because the decision to update or not is pixel dependent. As 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, we prefer to use a *random* subsampling policy. As shown at lines 15-16 of Algorithm 2, a random process determines if a background pixel will be inserted in the corresponding pixel model.

In most cases, we adopted a time subsampling factor, denoted ϕ, of 16: a background pixel value has 1 chance in 16 of being selected to update its pixel model. For some specific scenarios, one may wish to tune this parameter to adjust the length of the time window covered by the pixel model. A smaller subsampling factor, 5 for example, is to be preferred during the first frames of a video sequence, when there is a lot of motion in the image, or if the camera is shaking. This allows for a faster updating of the background model.

Spatial diffusion to ensure spatial consistency

One of the problems with a conservative updating policy is that foreground zones hide background pixels so that there is no way to uncover the background, and subsequently to update it. A popular way to counter this drawback is what the authors of the W^4 algorithm [11] 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 to include, in the background, groups of connected foreground pixels that have been found static for a long time, as in [7]. Some authors, like those of [38], use a combination of a pixel-level and an object-level background update.

For ViBe, we did not want to define a detection support map that would add parameters, whose determination would be application dependent, and increase the computational complexity, neither to include a mechanism that would be defined at the object level.

We believe that a progressive inclusion of foreground samples in the background models is more appropriate in general. We assume that neighboring background pixels share a similar temporal distribution and consider 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 *exclusively* classified 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 updating scheme. Lines 22 to 31 of Algorithm 2 detail the spatial diffusion mechanism. A spatial neighborhood of a pixel p is defined, typically a 4- or 8-connected neighborhood, and one location q in the neighborhood is chosen randomly. Then, the value $v(p)$

is inserted in the model of q. Since pixel models contain many samples, irrelevant information that could accidentally be inserted into a neighboring 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 containment inhibits the diffusion of errors. Note that in [12], the authors use the spatial updating mechanism of ViBe except that they insert $v(q)$, instead of $v(p)$, in the model of q. Also note that, for reasons similar to those exposed earlier, the spatial propagation method is also random. A random policy is convenient in the absence of prior knowledge of the scene, and to avoid a bias towards a scene model that would unnecessarily decrease the model flexibility.

It is worth mentioning that the original diffusion mechanism of ViBe (which uses the segmentation map as the updating mask) is not limited to the inside of background blobs. At the borders, it could happen that a background value is propagated into the model of a neighboring foreground pixel. If needed, background samples propagation into pixel models of the foreground may be prevented in at least two ways. The first way consists to prevent sample diffusion across the border of the updating mask by uncommenting lines 29 and 32 of Algorithm 2. Another technique consists to remove the inner border of the segmentation map when it serves as the updating mask, e.g. by using a morphologically eroded segmentation map as the updating mask.

An interesting question for ViBe relates to the respective effects of *in-place substitution* (a value replaces one value of its own model) and *spatial diffusion* (a value replaces one value of a neighboring model). To evaluate the impact of these strategies, we made several tests where we modified the proportion of in-place substitution and spatial diffusion. In addition, we compared the original diffusion mechanism of ViBe and a strategy that does not allow to modify models of foreground pixels; this variant, called *intra diffusion*, is obtained by uncommenting lines 29 and 32 of Algorithm 2 if the updating mask is equal to the background/foreground segmentation map. Table 1 shows the percentage of bad classification (PBCs) obtained for different scenarios; these results are averages for the 31 sequences of the Change Detection dataset.

	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
intra diffusion	3.79	3.74	3.55	3.49	3.40	3.39	3.32	3.27	3.25	3.24	3.20
inter diffusion	2.99	2.32	2.19	2.21	2.27	2.32	2.33	2.36	2.40	2.40	2.41

Table 1: Percentage of bad classification (PBC) for different updating strategies. The top row indicates the percentage of in-place substitution, with respect to the percentage of spatial diffusion. A 100% rate thus means that no values are propagated into the model of a neighboring pixel. To the contrary, in the 0% scenario, every pixel model update results from spatial diffusion. *Intra diffusion* is defined as a diffusion process that forbids the diffusion of background samples into models located in the foreground. In the original version of ViBe, diffusion is always allowed, even if that means that the value of a background pixel is put into the model of a foreground pixel, as could happen for a pixel located at the border of the background mask. This corresponds to the bottom row, named *inter diffusion*.

One can observe that, for the intra diffusion scenario, the PBC decreases as the percentage of spatial diffusion increases. By extension, we would recommend, for this dataset, to always diffuse a value and never use it to update a pixel's own model. This observation might also be applicable to other background subtraction techniques. The effects of intra diffusion are mainly to maintain a larger local variance in the model, and to mimic small camera displacements, except that all these displacements differ locally.

The last row of Table 1 (inter diffusion) provides the percentages of bad classification for the original diffusion mechanism proposed in ViBe. It consists to diffuse background values into the neighborhood, regardless of the classification result of neighboring pixels. It appears that the original spatial diffusion process is always preferable to a diffusion process restricted to the background mask (intra diffusion). In other words, it is a good idea to allow background values to cross the frontiers of background blobs. The experiments also show that the original spatial

diffusion mechanism of ViBe performs better when in-place substitution and spatial diffusion are balanced; a 50%-50% proportion, as designed intuitively in [2], is close to being optimal.

Why it is useful to define an updating mask that is different from the segmentation map

The purpose of a background subtraction technique is to produce a binary map with background and foreground pixels. Most of the time, it is the segmentation results that users are looking for. In a conservative approach, the segmentation map is used to determine which values are allowed to enter the background model. In other words, the segmentation map plays the role of an updating mask. But this is not a requirement. In fact, the unique constraint is that foreground values should never be used to update the model. In [37], we proposed to produce the updating mask by processing the segmentation map. For example, small holes in the foreground are filled to avoid the appearance of background seeds inside of background objects. Also, we remove very small foreground objects, such as isolated pixels, in order to include them in the background model as fast as possible. Another possibility consists to slow down the spatial diffusion across background borders. An idea to achieve this consists of allowing the diffusion only when the gradient value is lower than a given threshold. This prevents, or at least decreases, the pollution of background models covered by foreground objects (such as abandoned objects), while it does not affect the capability of ViBe to incorporate ghosts smoothly. Note that reducing diffusion enhances the risk that an object is never incorporated into the background.

(a) (b) (c)

Figure 2: Comparison of the effects of modifying the updating mask: (a) infrared input image, (b) segmentation map of the original version of ViBe, (c) segmentation map obtained when the updating mask inhibits samples diffusion across contrasted background borders (this inhibition mechanism is described in [37]).

These discussions clearly show that the purpose of differentiating the segmentation map and the updating mask is to induce a behavior for the background subtraction technique that is adapted to the application needs, while keeping the principles of ViBe. For example, an abandoned object might need to be left in the foreground for a certain amount of time. After that time, the application layer can decide to switch the label from foreground to background in the updating mask, in order to include the abandoned object into the background. In other words, the distinction between the updating mask and the segmentation map introduces some flexibility and permits to incorporate high-level information (such as objects blobs, regions of interest, time notions, etc) into any low-level background subtraction technique.

Initialization

Many popular techniques described in the literature, such as [8, 15], need a sequence of several dozens of frames to initialize their models. From a statistical point of view, it is logical to wait for a long period of time 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, or even to be operational from the second frame on. Furthermore, many applications require the ability to provide a mechanism to refresh or re-initialize the background model in circumstances such as in the presence of sudden lighting changes, that cannot be handled properly by the regular update mechanism of the algorithm.

A convenient solution consists 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.

In the original version of ViBe [2], we used the same assumption as the authors of [14], which is that neighboring pixels share a similar temporal distribution, to populate the pixel models with values found in the spatial neighborhood of each pixel. From our experiments, selecting samples randomly in the 8-connected neighborhood of each pixel has proved to be satisfactory. As no temporal information is available from the first frame, there seems to be no other alternative than to take values from the spatial neighborhood. However, other strategies have been tested as well. The first alternative consists to initialize $N-2$ samples of the models with values that are uniformly distributed over the possible range, and 2 samples with the current pixel value. This accounts for a clean background while providing the capability to accommodate to some motion in the next frames. In other terms, this ensures to have a background that fills the entire image and a non zero variance. If one is ready to wait for a few frames before segmenting the image, then it is also possible to simply put random values in the model. We then recommend to temporarily decrease the time subsampling factor ϕ to speed up the incorporation of background values into the model.

All these strategies have proved to be successful, and they rapidly converge to identical segmentation results when dealing with long sequences. For short time periods, the presence of a moving object in the first frame introduces an artifact commonly called a *ghost*. According to [30], 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 taken from the moving object. In subsequent frames, the object moves and uncovers the real background, which will be learned progressively through the regular model updating process, making the ghost fade over time. It appears that the propagation mechanism is capable to overcome ghost effects. In fact, as shown in [2], ghosts are absorbed rapidly compared to static objects that remain visible for a longer period of time.

Other issues relate to sudden natural illumination changes, scene cuts, or changes caused by the Automatic Gain Control (AGC) of cameras that artificially improves their "dynamic range" to produce usable images. In all these cases, changes are not due to the motion of objects, and the amount of foreground pixels indicates that there is a major change in the scene. It also happens that these changes are temporary and that the scene rapidly returns to its previous state. This is typical for IP cameras whose gain changes when an object approaches the sensor and returns to its previous state when the object leaves the scene. This situation also occurs on a pixel basis when an object stays at the same location for a short period of time, and then moves again.

The solution that we propose to cope with these problems consists to partly re-initialize the model of each pixel with the new value. More precisely, we only replace $\#_{min}$ samples of each model; the $N-\#_{min}$ other samples are left unchanged. With this mechanism, the background model is re-initialized, while it memorizes the previous background model as well. It is then possible to return to the previous scene and to get a meaningful segmentation instantaneously. Theoretically, we could even store $N/\#_{min}$ background models at a given time. This technique is similar to that of GMM based models when they would keep a few of the significant gaussians to memorize a previous model.

M. Van Droogenbroeck, and O. Barnich. **ViBe: a disruptive method for background subtraction**. In *T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7.* Chapman and Hall/CRC, pages 7.1-7.23, July 2014. [DOI: 10.1201/b17223-10]. See http://www2.ulg.ac.be/telecom/research/vibe/doc for code.

2.4 Variants of ViBe

Several variants of ViBe have been proposed recently. In [6], it was suggested to use a threshold in relation to the samples in the model for a better handling of camouflaged foreground. This adaptation was done by Hofmann et al. [12]. Their proposed background subtraction technique uses all the principles of ViBe, and add learning heuristics that adapt the model pixel classification and updating dynamics to the scene content. They also change the diffusion mechanism to guarantee that a model is never updated with a value of a neighboring pixel, by updating a neighboring model with the current value for that same location. However, previously in this section, we have discussed the diffusion mechanism and clearly showed that the original diffusion mechanism is preferable. Mould and Havlicek also propose variants of ViBe that adapt themselves to the scene content [22, 23]. Their major modification consists to apply a learning algorithm that integrates new information into the models by replacing the most outlying values with respect to the current sample collections.

While it is obvious that adapting fixed parameters to the image content should increase the performance, one can observe that a learning phase is then needed to train the parameters of the heuristics of these authors. To some extend, they have changed the dynamic behavior of ViBe by replacing fixed parameters by heuristics which have their own fixed parameters. This might be an appropriate strategy to optimize ViBe to deal with specific scene contents, but the drawbacks are that it introduces a bias towards some heuristics, and that it reduces the framerate dramatically.

In [19], Manzanera describes a background subtraction technique that extends the modeling capabilities of ViBe. The feature vectors are enriched by local jets and an appropriate distance is defined. Manzanera also proposes to regroup the samples in clusters to speed up the comparison step. These extensions do not modify the dynamic behavior of ViBe, but Manzanera shows that its method improves the performance. Enriching the feature vector slightly increases the performance, but it also increases the computation times, more or less significantly, depending on the complexity of the additional features and that of the adapted distance.

Another technique, inspired by ViBe, is described by Sun et al. [33]. They propose to model the background of a pixel not only with samples gathered from its own past, but also in the history of its spatial neighborhood.

Note that, while this Chapter focuses on the implementation of ViBe on CPUs, small modifications might be necessary for porting ViBe to GPUs, or to FPGA platforms [16]. However, the fundamental principles of ViBe are unaltered.

3 Complexity analysis and computational speed of ViBe

Except for papers dedicated to hardware implementation, authors of background subtraction techniques tend to prove that their technique runs in real time, for small or medium sized images. It is then almost impossible to compare techniques, because many factors differ from one implementation to another (CPU, programming language, compiler, etc). In this section, we analyze the complexity of the ViBe algorithm in terms of the number of operations, and then compare execution times of several algorithms. We also define the notion of *background subtraction complexity factor* that expresses the practical complexity of an algorithm.

To express the complexity of ViBe, we have to count the number of operations for processing an image. There are basically four types of operations involved: distance computation, distance thresholding, counter checks, and memory substitutions. More precisely, the number of operations is as follows for ViBe:

- Segmentation/classification step.
  Remember that we compare a new pixel value to background samples to find $\#_{min}$ matches

($\#_{min} = 2$ in all our experiments). Once these matches have been found, we step over to the next pixel and ignore the remaining background samples of a model; this is different from techniques that requires a 90% match, for example. 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 $\#_{min}$ first values of the model of a background pixel are close to the new pixel value. The algorithm then reaches its minimal number of comparisons. For foreground pixels however, the number of comparisons can be as large as the number of samples in the model, N. Therefore, if $\#F$ denotes the proportion of foreground pixels, the minimal number of comparisons per pixel N_{comp} is, on average,

$$N_{\text{comp}} = \#_{min} + (N - \#_{min})\#F. \tag{4}$$

    Each of these comparisons involves one distance calculation and one thresholding operation on the distance. For a typical number of foreground pixels in the video, this accounts for 2.8 comparisons on average per pixel: $N_{\text{comp}} = 2.8$. This is an experimentally estimated average value as computed on the Change Detection dataset (see last row, column (e) of Table 2).
  - comparisons of the counter to check if there are at least 2 matching values in the model. We need to start comparing the counter value only after the comparison between the current pixel value and the second value of the background model. Therefore, we have $N_{\text{comp}} - 1 = 1.8$ comparisons.

- Updating 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 3 additions on memory addresses per 16 background pixels.

- Summary (average per pixel, assuming that $\#_{min} = 2$ and $N_{\text{comp}} = 2.8$, that is that most pixels belong to the background):

  - 2.8 distance computations and 2.8 thresholding operations. In the simple case of a distance that is the absolute difference between two grayscale values, this accounts for 5.6 comparisons.
  - 1.8 counter checking operations. Note that a test such as $count < 2$ is equivalent to $count - 2 < 0$, that is to a subtraction and sign test. If we ignore the sign test, one comparison corresponds to one subtraction in terms of computation times.
  - $\frac{3}{16}$ addition on memory addresses.

Profiling tests show that the computation time of the updating step is negligible, on a CPU, compared to that of the segmentation step. If for the segmentation step, we approximate the complexity of a distance computation to that of a comparison, and if we assume that the image is a grayscale image, the number of operations is $3N_{\text{comp}} - 1$ per pixel; this results in 7.4 comparisons per pixel. In order to verify this number, we have measured the computation times for different algorithms. Table 2 shows measures obtained for each sequence of the Change Detection dataset.

Column (a) provides the computation times of the simple background subtraction technique with a static background, whose corresponding code is given in Algorithm 3. To avoid any bias related to input and output handling, 1000 images were first decoded, converted to grayscale, and then

Sequences	**(a)** static backgr.	**(b)** ViBe↓	**(c)** expon. filter	**(d)** ViBe	**(e)** N_{comp}	**(f)** theoretical C_F	**(g)** real C_F
highway	188	265	897	863	3.1	4.1	4.6
office	211	284	1014	822	2.7	3.6	3.9
pedestrians	177	259	981	770	2.2	2.8	4.4
PETS2006	748	1152	4619	4666	2.4	3.0	6.2
badminton	842	1125	4017	6004	3.6	4.8	7.1
boulevard	250	351	1040	1350	3.1	4.2	5.4
sidewalk	276	358	1050	1482	3.4	4.6	5.4
traffic	228	284	944	1416	4.4	6.0	6.2
boats	253	297	951	898	2.4	3.1	3.5
canoe	292	340	990	1359	3.0	4.0	4.7
fall	1385	1780	4628	7028	3.6	4.9	5.1
fountain01	334	441	1492	1402	2.4	3.0	4.2
fountain02	326	418	1466	1118	2.1	2.6	3.4
overpass	254	280	962	739	2.2	2.7	2.9
abandonedBox	338	510	1496	1493	2.9	3.8	4.4
parking	157	216	876	652	2.1	2.76	4.2
sofa	153	218	865	705	2.4	3.0	4.6
streetLight	238	336	957	821	2.8	3.7	3.4
tramstop	304	509	1460	1277	3.1	4.1	4.2
winterDriveway	215	317	934	719	2.2	2.9	3.3
backdoor	246	281	917	771	2.4	3.2	3.1
bungalows	190	257	993	997	3.2	4.3	5.2
busStation	191	268	995	811	2.4	3.2	4.2
copyMachine	655	939	3881	4265	3.0	3.9	6.5
cubicle	172	268	967	698	2.2	2.8	4.1
peopleInShade	182	276	1049	990	2.9	3.9	5.4
corridor	144	192	857	775	2.6	3.4	5.4
diningRoom	136	191	852	708	2.6	3.4	5.2
lakeSide	132	185	851	645	2.0	2.5	4.9
library	132	178	851	718	4.0	5.5	5.4
park	112	165	1134	552	2.2	2.8	4.9
Mean	**307**	**421**	**1451**	**1580**	**2.8**	**3.7**	**4.8**

Table 2: Computation times, in milliseconds and for 1000 images (converted to grayscale), of several algorithms for each sequence of the Change Detection dataset, on a Intel(R) Core(TM)2 Duo CPU T7300 @2.00GHz. The considered algorithms are: (a) background subtraction with a static background (1 distance comparison and 1 distance thresholding operation), (b) downscaled version of ViBe (1 value in the model, 1 comparison), (c) exponential filter ($\alpha = 0.05$), and (d) ViBe. The average number of distance computations per pixel is given, for ViBe, in column (e). Considering that all images were first converted to grayscale, (f) is the theoretical background subtraction complexity factor and (g) is the measured background subtraction complexity factor.

Algorithm 3 Background subtraction with a static background. This corresponds to the minimum number of operations for any pixel based background subtraction. Here, the code does not include any updating step, neither the initialization of the static background model.

```
1  int width;                                  // width of the image
2  int height;                                 // height of the image
3  byte image[width*height];                   // current image
4  byte segmentationMap[width*height];         // classification result
5
6  int distanceThreshold = 20;
7
8  byte sample[width*height];                  // static background model
9
10 ...
11 for (int p = 0; p < width*height; p++) {
12   if (getDistanceBetween(image[p], sample[p]) < distanceThreshold)
13     segmentationMap[p] = BACKGROUND_LABEL;
14   else
15     segmentationMap[p] = FOREGROUND_LABEL;
16   }
17 ...
```

stored into memory. The computation times therefore only relate to the operations necessary for background subtraction.

Values of column (a) constitute the absolute lower bound on the computation time for any background subtraction technique, because one need at least to compare the current pixel value to that of a reference, and allocate the result in the segmentation map (this last step is generally negligible with respect to that of distance computations or comparisons). In our implementation, we have converted the input video stream to grayscale to achieve the lowest computation times, because the distance calculation is then equivalent to the absolute difference between the two values. For more complex distances, the distance calculation will require more processing power than distance thresholding. We note that, on an Intel(R) Core(TM)2 Duo CPU T7300 @2.00GHz, the average computation time per image is 0.3 ms.

In [2], we proposed a downscaled version of ViBe, named ViBe↓ hereafter and in Table 2, that consists to use only one sample for each background model and, consequently, to proceed to only one comparison per pixel. Computation times of that downscaled version of ViBe are given in column (b) of Table 2. On average, the computation is 114 ms slower for 1000 images, that is about 0.1 ms per image. Because the only difference with the technique with a static background is the updating step, we can estimate the cost of the updating to a third of that of a distance computation and thresholding. Therefore, ViBe↓ is probably the fastest background subtraction technique that includes an updating mechanism. Results of ViBe↓ are shown in column (b) of Figure 3.

Columns (c) and (d) of Table 2 provide the computation times for the exponential filter method and ViBe, respectively. One can see that the computational cost of ViBe is similar to that of the exponential filter, and about 4.8 times that of the simplest background subtraction technique.

In further tests, we explored the average number of distance computations per pixel of ViBe. This number is given, for each video sequence of the Change Detection dataset, in column (e) of Table 2. Despite that the model size is 20, we have an average number of distance computations of 2.8. This means that, on average, ViBe only has to perform about less than 1 distance computation in addition to the $\#_{min} = 2$ unavoidable distance computations. While this figure is interesting for the optimization of parameters, it does not translate directly in terms of computation time because many other operations are involved in a software implementation. While these additional

(a) (b) (c)

Figure 3: Segmentation maps obtained with ViBe↓, a downscaled version of ViBe: (a) original image, (b) unfiltered segmentation map of ViBe↓, and (c) same as (b) after an area opening and a close filter.

operations, such as loops unrolling, are necessary, they are often neglected in complexity analyses. In order not to underestimate the computation times and the complexity, we define a new ratio, named the *background subtraction complexity factor*, as follows:

$$C_F = \frac{\text{measured computation time}}{\text{measured computation for one distance measure and check}}. \quad (5)$$

It corresponds to the ratio between the measured computation time and the time needed to perform the simplest background subtraction technique (with a static background). For the denominator, we calculate the time of the algorithm such as described in Algorithm 3. Because this simple algorithm has a fixed number of operations per pixel and all the operations are known, it can be used as a yardstick to measure the complexity of another algorithm. Assuming the specific case of a grayscale video stream, the algorithm requires to compute one distance and one thresholding per pixel. If we approximate these operations as two comparisons per pixel, we have a theoretical estimate of the denominator of equation 5. We can also estimate the numerator as follows. Column (e) of Table 2 corresponds to the measured number of distance computations per pixel N_{comp}. As indicated previously, we have $3N_{\text{comp}} - 1$ operations per pixel. The complexity factor can thus be estimated as $(3N_{\text{comp}} - 1)/2$. The theoretical estimates of the complexity factor for each video sequence are given in column (f) of Table 2. When we compare them to the real complexity factors (see column (g)), based on experimental measures, we see that the theoretical complexity factor is often underestimated. This is not surprising as we have neglected some operations such as loops, updating, memory reallocation, etc.

4 Conclusion

ViBe is a pixel based background subtraction technique that innovated by introducing several new mechanisms: segmentation by comparing a pixel value to a small number of previously collected samples, memoryless updating policy, and spatial diffusion. This chapter elaborates on the underlying ideas that motivated us to build ViBe. One of the main keys is the absence of any notion of time when building and updating the background models. In addition, ViBe introduces a spatial diffusion mechanism that consists to modify the model of a neighboring pixel while updating the

model of a pixel. We have showed that the use of spatial diffusion always increases the performance and that, in particular for the Change Detection dataset, it is even preferable to diffuse a value in the neighborhood of a pixel rather than to use it to update its own model. Furthermore, crossing background borders in the segmentation map to adapt models of foreground pixels is also beneficial for the suppression of ghosts or static objects.

The complexity of ViBe is also discussed in this chapter. We introduce the notion of complexity factor that compares the time needed by a background subtraction technique to the computational cost of the simplest background subtraction algorithm. The measured complexity factor of ViBe is about 4.8, on average, on the Change Detection dataset. The complexity factor even drops to 1.4 for a downscaled version of ViBe, which still produces acceptable results!

References

[1] O. Barnich and M. Van Droogenbroeck. ViBe: a powerful random technique to estimate the background in video sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 945–948, April 2009.

[2] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, June 2011.

[3] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 1–4, December 2008.

[4] T. Bouwmans. Recent advanced statistical background modeling for foreground detection - a systematic survey. *Recent Patents on Computer Science*, 4(3):147–176, 2011.

[5] 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)*, chapter 3, pages 181–199. World Scientific Publishing, January 2010.

[6] S. Brutzer, B. Höferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1937–1944, Colorado Spring, USA, June 2011.

[7] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, October 2003.

[8] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, volume 1843 of *Lecture Notes in Computer Science*, pages 751–767, London, UK, June-July 2000. Springer.

[9] S. Elhabian, K. El-Sayed, and S. Ahmed. Moving object detection in spatial domain using background removal techniques – State-of-art. *Recent Patents on Computer Science*, 1:32–54, January 2008.

[10] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. changedetection.net: A new change detection benchmark dataset. In *Change Detection Workshop (CDW)*, Providence, Rhode Island, June 2012.

[11] I. Haritaoglu, D. Harwood, and L. Davis. W^4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[12] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Change Detection Workshop (CDW)*, Providence, Rhode Island, June 2012.

[13] Home Office, Scientific Development Branch. *i-LIDS User Guide*, 2011.

[14] P.-M. Jodoin, M. Mignotte, and J. Konrad. Statistical background subtraction using spatial cues. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(12):1758–1763, December 2007.

[15] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, June 2005.

[16] T. Kryjak and M. Gorgon. Real-time implementation of the vibe foreground object segmentation algorithm. In *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 591–596, Krakow, Poland, September 2013.

[17] H.-H. Lin, T.-L. Liu, and J.-C. Chuang. Learning a scene background model via classification. *IEEE Signal Processing Magazine*, 57(5):1641–1654, May 2009.

[18] A. Manzanera. Σ-Δ background subtraction and the Zipf law. In *Progress in Pattern Recognition, Image Analysis and Applications*, volume 4756 of *Lecture Notes in Computer Science*, pages 42–51. Springer, November 2007.

[19] A. Manzanera. Local jet feature space framework for image processing and representation. In *International Conference on Signal Image Technology & Internet-Based Systems*, pages 261–268, Dijon, France, November-December 2011.

[20] A. Manzanera and J. Richefeu. A new motion detection algorithm based on Σ-Δ background estimation. *Pattern Recognition Letters*, 28(3):320–328, February 2007.

[21] T. Moeslund, A. Hilton, V. Krüger, and L. Sigal. *Visual Analysis of Humans: Looking at People*. Springer, 2011.

[22] N. Mould and J. Havlicek. A conservative scene model update policy. In *Southwest Symposium on Image Analysis and Interpretation*, Santa Fee, New Mexico, USA, April 2012.

[23] N. Mould and J. Havlicek. A stochastic learning algorithm for pixel-level background models. In *IEEE International Conference on Image Processing (ICIP)*, pages 1233–1236, September – October 2012.

[24] A. Papoulis. *Probability, random variables, and stochastic processes*. McGraw-Hill, 1984.

[25] D. Parks and S. Fels. Evaluation of background subtraction algorithms with post-processing. In *IEEE International Confersence on Advanced Video and Signal Based Surveillance*, pages 192–199, Santa Fe (New Mexico, USA), September 2008.

[26] F. Porikli and O. Tuzel. Bayesian background modeling for foreground detection. In *ACM international workshop on Video surveillance & sensor networks*, pages 55–58, Singapore, November 2005.

[27] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. *IEEE Transactions on Image Processing*, 14(3):294–307, March 2005.

[28] A. Schick, M. Bauml, and R. Stiefelhagen. Improving foreground segmentation with probabistic superpixel Markov random fields. In *Change Detection Workshop (CDW)*, Providence, Rhode Island, June 2012.

[29] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, June 2011.

[30] B. Shoushtarian and H. Bez. A practical adaptive approach for dynamic background subtraction using an invariant colour model and object tracking. *Pattern Recognition Letters*, 26(1):5–26, Januray 2005.

[31] A. Srivastava, A. Lee, E. Simoncelli, and S.-C. Zhu. On advances in statistical modeling of natural images. *Journal of Mathematical Imaging and Vision*, 18(1):17–33, January 2003.

[32] C. Stauffer and E. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 246–252, Ft. Collins, USA, June 1999.

[33] L. Sun, Q. De Neyer, and C. De Vleeschouwer. Multimode spatiotemporal background modeling for complex scenes. In *European Signal Processing Conference (EUSIPCO)*, pages 165–169, Bucharest, Romania, August 2012.

[34] M. Van Droogenbroeck and O. Barnich. Visual background extractor. European Patent Office, EP 2_015_252_B1, February 2010.

[35] M. Van Droogenbroeck and O. Barnich. Visual background extractor. Japan Patent Office, JP 2011 4699564 B2, June 2011.

[36] M. Van Droogenbroeck and O. Barnich. Visual background extractor. United States Patent and Trademark Office, US 8,009,918 B2, 18 pages, August 2011.

[37] M. Van Droogenbroeck and O. Paquot. Background subtraction: Experiments and improvements for ViBe. In *Change Detection Workshop (CDW), in conjunction with CVPR*, pages 32–37, Providence, Rhode Island, June 2012.

[38] H. Wang and D. Suter. A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognition*, 40(3):1091–1105, March 2007.

[39] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[40] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, May 2006.