Published in Image Processing On Line on 2022–11–23.
Submitted on 2022–09–30, accepted on 2022–10–01.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.434

Reviewing ViBe, a Popular Background Subtraction Algorithm for Real-Time Applications

Xavier Bou, Thibaud Ehret, Gabriele Facciolo, Jean-Michel Morel, Rafael Grompone von Gioi

Centre Borelli, ENS Paris-Saclay, France
{xavier.bou_hernandez, thibaud.ehret, gabriele.facciolo, morel, grompone}@ens-paris-saclay.fr

Communicated by Jean-Michel Morel *Demo edited by* Xavier Bou

Abstract

Background subtraction or foreground segmentation is commonly the first step in video processing pipelines, where foreground objects and moving entities are detected for further analysis. Background subtraction is still an open problem in computer vision due to the wide range of possible scenarios and challenging cases, such as weather conditions, illumination variation, dynamic backgrounds or low frame-rates. In this article, we review a well-known general algorithm for background subtraction named visual background extractor or ViBe. More than 10 years after its publication, ViBe is still widely used due to its simplicity and low computational load, as its authors claim it can process up to 200 fps. We analyze the algorithm's mechanism to segment foreground objects, comment on its strengths and drawbacks, and describe an effective variant of the method to quickly dissipate ghost effects using 3-frame difference. Furthermore, we provide an easy-to-use demo that allows to quickly test the performance of ViBe on custom data, modify its parameters and download the results.

Source Code

The ViBe algorithm is covered by several patents and its code is not provided with this publication. The reader can access a version of the code, shared for research purposes, at https://orbi.uliege.be/handle/2268/145853.

Keywords: background subtraction; foreground segmentation; change detection; motion detection; video analysis; surveillance; monitoring

1 Introduction

Technological advances in recent years result in an abundance of cameras worldwide, both at the reach of particulars and for surveillance and monitoring purposes. Consequently, the quantity of available videos has considerably increased to the point where manual analysis is no longer feasible. Hence, systems and pipelines that can automate the processing of videos have gained attention and interest. In these systems, a common early step for detection applications is background subtraction, which is often also referred to as foreground segmentation or change detection.

Background subtraction algorithms evaluate a scene and generate a mask containing only the foreground objects that do not belong to the background of the scene. These methods generally follow a three-step workflow: a model of the background of the scene is first built using a set of frames, ideally with no objects of interest present. Then, when new frames are evaluated, algorithms compute the difference between the background reference model and the observed information, classifying image pixels into foreground or background clusters. Lastly, the model is updated with the analyzed frame before moving forward onto the next.

What seemed at first an affordable problem has turned out to be rather difficult to automate. The large range of possible scenes to which the algorithm can be exposed, each with their unique attributes, makes the development of a general method for background subtraction still a fundamental problem in computer vision. Furthermore, many extreme cases present specific challenges to these methods. Some examples are bad weather conditions, such as rain or snow, dynamic backgrounds such as waving trees or water, the presence of moving shades and reflections, or sudden illumination variations [9, 3]. Finding a suitable set of descriptors that can cover all of these scenarios is not trivial. However, recent advances in deep neural networks and their capabilities to reach complex representations have been used for the background subtraction problem [4, 1, 17]. Training neural networks with annotated data to substitute one or more steps of the traditional flow has proved to yield better results than traditional, unsupervised methods, as results of the popular CDNet (http://www.changedetection.net) benchmark indicate [6, 24]. Nonetheless, these methods come with two main drawbacks. First, they are limited by their supervised nature, requiring large amounts of labelled data, something costly and time consuming. Moreover, these techniques require heavy computational resources and thus are not optimal for real-time processing [5].

The speed of the method is often disregarded in research articles, which tend to focus on the overall result at the cost of complex computational requirements. However, the application of these methods in real systems without powerful resources can lead to bottlenecks. Consequently, traditional unsupervised methods are often chosen over recent supervised methods because they are lightweight and do not need to be trained for each scene individually [5, 3]. One of the most popular background subtraction algorithms is the *visual background extractor*, or ViBe [2], which introduced a few simple yet innovative ideas at the time. ViBe is considered the fastest background subtraction method in the market and is claimed to be able to run up to 200 fps.

In this article, we review ViBe and discuss its mechanisms for building a reference background model, segmenting foreground information, initializing the model and updating it after a frame is processed. Additionally, we explain in detail a common variant of the algorithm using 3-frame difference, which is able to quickly dissipate ghost effects. Furthermore, we analyze its results in different challenging scenarios and evaluate its performance beyond well known benchmarks. Lastly, an easy-to-use demo is provided so that one can quickly test on new data and get a sense of how the algorithm responds to a particular scene. The 3-frame difference variant is included in the demo. The rest of the article includes the following sections: Section 2 provides a general view of the background subtraction problem in the literature, Section 3 describes ViBe in detail, Section 4 evaluates the performance and results of ViBe in multiple scenarios, and Section 5 describes the provided demo and details how to use it.

2 Review of Background Subtraction

As briefly introduced in Section 1, background subtraction or foreground segmentation consists in discriminating foreground objects from the background of the scene so that relevant observed entities can be further analyzed. Classical methods for background subtraction consist in unsupervised approaches that use traditional computer vision techniques to address the problem. Nonetheless, more recently some supervised methods have been proposed, which leverage the capabilities of deep neural networks for the background subtraction task.

2.1 Unsupervised Methods

Unsupervised methods for background subtraction are convenient because no learning process nor labelled data is required, and they do not involve complex computations. These algorithms generally follow a three-step workflow, where 1) a reference model of the background is first built and then 2) the difference between the background model and new incoming frames is computed, thus classifying the pixels into background or foreground information. Finally, 3) after evaluating each frame, the observed information is updated into the background model before moving onto the next frame. Each of these steps is explained in detail in the following sub-sections.

2.1.1 Building a Background Reference Model

Building a background model of the scene is used as a reference to compare new frames and determine what is likely to be foreground information. Most methods in the literature attempt to model the background relying on a probability density function or statistical parameters. These are known as parametric methods, and its seminal example is the adaptive Gaussian mixture model (GMM) [20] proposed by Stauffer and Grimson in 1999, in which the authors model each pixel with a number of Gaussians. Other authors proposed improvements of the Stauffer-Grimson method, with the purpose of automatically adapting its parameters [28], improve initialization [7] or reduce the computational load [21]. Some methods in the literature propose alternative approaches to parametric models, known as non-parametric or consensus-based, which generally use a stack of past, observed samples as a representative model of the background. A few non-parametric approaches had been proposed before ViBe, such as SACON [23], which builds a consensus of the background samples and determines whether new samples agree with past samples with a threshold-base mapping. As of today, the consensus-based methods ViBe and SuBSENSE [19], a method based on ViBe that uses linear binary patterns (LBP) [15] features for robustness in dynamic backgrounds, are considered state-of-the-art unsupervised methods for background subtraction.

2.1.2 Foreground Segmentation

In this step, a new observed frame is compared to the previously built background model in order to measure their differences. To discriminate background from foreground information, a suitable error metric is necessary. Error metrics generally use appearance or motion representations to map pixels to either background or foreground clusters and output a binary mask with the two-level information. While traditional approaches [20, 23] use pixel intensity and appearance representations to discriminate changes, they generally show lower performance for complex scenes presenting for example lighting and shade variations. Different methods have been proposed to address these challenges. Lisani et al. [13] measured the significance of an observed change in regards to their false alarm rate, so that only meaningful changes are detected. Others use texture and motion descriptors to achieve representations invariant to illumination changes and slow-motion. These are known as region-based methods, which evaluate the neighborhood of each pixel to extract spatial information. Nguyen and Smeulders proposed a robust tracker by background/foreground texture discrimination [14], and motion prediction using Kalman Filters [8, 16] was used to track rapidly changing objects in [22].

2.1.3 Model Update

Lastly, after a frame has been evaluated and a binary mask has been produced, the new information is used to update the background model. Thus, an updating mechanism needs to be defined to regulate how the information is incorporated into the reference model. The two main options for this are a conservative update policy or a blind update policy. The first one does not include any pixel labelled as foreground information into the background model, thus only the pixels classified as background are used for the update. A conservative policy is effective at providing sharp detection of moving objects, since the background model will consist in a reliable representation of the empty scene. Nevertheless, when a pixel is incorrectly classified as foreground, this approach is sensitive to deadlocks and ghost effects. A background pixel classified as foreground will not be included in the model and indefinitely remain a false positive. On the other hand, a blind update policy incorporates the information of each frame regardless of the segmentation result. This means that the reference model is updated with both foreground and background information observed in the sequence frames. Hence, methods using this approach are not subject to deadlocks and can quickly recover from ghost effects. The main drawback of a blind update policy is that it often provides poor detection of slow-moving objects. Since all information is progressively included into the model, objects that slightly move across frames can become false negatives.

2.2 Supervised Methods

Convolutional Neural Networks (CNNs) have recently shown great performance in multiple computer vision tasks, such as image classification [26], semantic segmentation [10] or image reconstruction [18]. Their capability of learning high-level complex representations has placed them at the forefront of machine learning research. In regards to the background subtraction problem, some CNN-based methods have been proposed in the literature to replace one or more steps of the traditional flow. Braham and Van Droogenbroeck propose in [4] a simplification of the background modelling task by training a CNN to perform the background removal process, trained with scene-specific examples. Babaee et al. proposed in [1] a general CNN-based approach that can be trained with multiple scenes at once and uses as ground truth the binary masks generated by the SuBSENSE algorithm. Sakkos et al. [17] proposed an end-to-end method that exploits temporal information using 3D convolutions. Overall, supervised approaches have proved to yield better results than traditional methods. Nonetheless, they tend to require heavy computational resources and large annotated data. For this reason, real-time background subtraction is still a challenge for these techniques.

3 Method

ViBe was proposed in 2011 by Barnich and Van Droogenbroeck as a general motion detector based on a few innovative ideas at the time. The motivation behind its use for real-time applications resulted in a fast yet efficient method. ViBe is built upon the idea that the most recent samples are not necessarily the most representative of the background, and while a pixel model should include samples from the recent past of the pixel, older samples should not automatically be discarded [2]. The different parts of the algorithm and their fundamental ideas are discussed in the following subsections, including how the background model is built, the classification process, model initialization, and update policy.

3.1 Pixel Background Model and Classification

Most classical approaches model the background with a probability function and compute its pdf or statistical parameters to address the likelihood of a value to belong to the background [11, 20, 7, 28, 21]. Nonetheless, this requires the assumption of a specific shape for the pdf, which is sensitive to outliers. Instead, the authors of ViBe state that already observed values should be more likely to be observed again than values never encountered before. Consequently, ViBe builds a reference background model consisting of a collection of past observed samples, aiming at increasing statistical significance over time. Let $v(x)$ be the value in a given Euclidean color space at pixel x of an image, and v_i a background sample value with index i. Each background pixel x is modelled by a collection of N previously observed background sample values

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

Considering that no particular pdf is assumed and therefore no pdf estimation is performed, new pixel values are compared with their corresponding sample values in the background model $\mathcal{M}$. The authors express 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 [2]. Hence, to assign a new incoming value to the background, it should be close to some of the sample values in the collection instead of the majority or all the values. The comparison of pixel values for classification is computed as follows. A pixel value $v(x)$ is compared to the closest values in its corresponding model $\mathcal{M}(x)$ by defining a sphere $S_R(v(x))$ of radius R centered on $v(x)$. Then, the pixel value $v(x)$ will be classified as background if the cardinality $\#$ of the intersection of the sphere and the collection of samples in $\mathcal{M}(x)$ is larger or equal than a threshold $\#_{min}$. In other terms, $\#_{min}$ is compared to

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

This mechanism is characterized by two main parameters, the radius of the sphere R and the minimum cardinality $\#_{min}$. Furthermore, the sensitivity of the model can be fine-tuned by the following ratio

$$\frac{\#_{min}}{N}, \tag{3}$$

where N is the number of samples per pixel stored in the collection or background model $\mathcal{M}$. Figure 1 illustrates the classification process described in this section.

3.1.1 Code Implementation to Speed Up the Matching Process

The official source code implementation of ViBe includes a strategy to speed up the matching process, namely when a value is evaluated and classified as background information. The higher the number of samples in the background model N, the more computations are needed, as the distance from the observed value to each of the ones in the model have to be determined. Nonetheless, for $\#_{min} = 2$, we can discard the rest of the samples when we reach the minimum cardinality, i.e. two samples are closer to the evaluated value than the radius R. Knowing this, the authors of ViBe swap the positions in memory of the samples in the model to increase the probability of reaching the minimum cardinality with the first checked values. For a collection of samples of $N = 20$ and $\#_{min} = 2$, when two matches are found in positions 8 and 13, the algorithm moves the value at position 8 to position 0, and the value at position 13 to position 1. Therefore, the next frame is more likely to fulfill the minimum cardinality with the first two checked values. Algorithm 1 of Section 3.4 describes the pseudo-code of the ViBe algorithm, where lines 18-35 correspond to the classification process. The change of memory position after a match is depicted in lines 26-30.

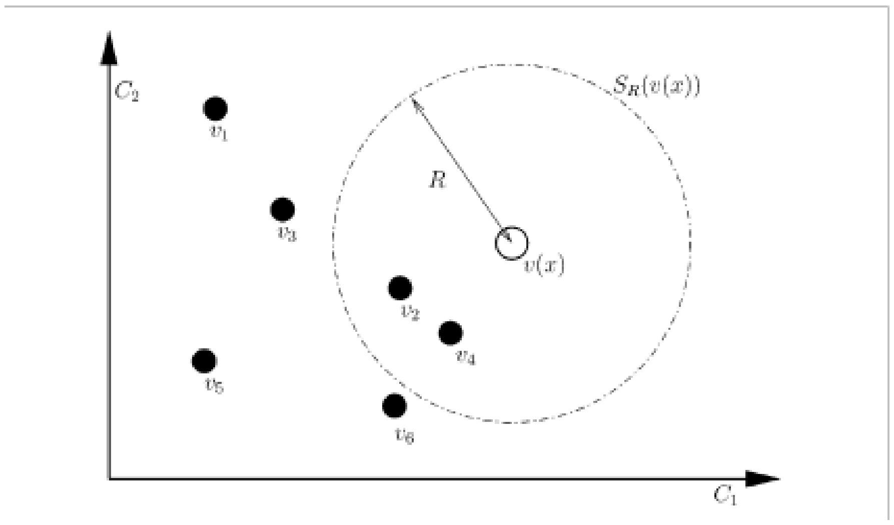


Figure 1: Extracted from the original publication of ViBe [2]. Classification process for a new pixel value. For a minimum cardinality $\#_{min}$ of 2, the evaluated pixel would be clustered as a background pixel.

Finally, ViBe is officially implemented using the RGB color space. Given an observed value, the code determines that the distance with a sample in the background model is close enough when the following inequality is fulfilled

$$|r_1 - r_2| + |g_1 - g_2| + |b_1 - b_2| \leq 4.5R, \tag{4}$$

where r_1 and r_2 are the red color values, g_1 and g_2 are the green color values, b_1 and b_2 are the blue color values and R is the radius parameter.

3.2 Initialization of the Reference Background Model

It is common for traditional statistical methods to require several frames to reach a significant amount of information so that they can build a reliable background model. Nevertheless, one could need a quick response when analyzing short sequences, or videos where objects rapidly incorporate into the scene. In order to account for this, ViBe proposed to initialize the background model with only one frame. This is advantageous in another manner beyond a quick initialization. When a sequence is exposed to sudden illumination variation, background subtraction algorithms tend to produce a high quantity of false positives due to the large variation of pixel values. Initialization from a single frame allows to discard the current model and re-initialize it from the observed frame when sudden illumination changes are observed. This allows the method to rapidly adapt to the new lightning conditions without interruption.

In ViBe's publication, the assumption that neighboring pixels share a similar temporal distribution is adopted to build a background model without temporal information. Thus, the pixel model is built by randomly choosing pixels within the neighborhood of each pixel. While the size of the spatial neighborhood area should be sufficiently large to include enough samples, the correlation of neighbor pixels to the central pixel diminishes as the size increases. The authors of ViBe found it optimal to use the 8-connected neighborhood for images of 640×480 resolution. Thus, let $N_G(x)$ be the spatial neighborhood of a pixel location x, and $t = 0$ indexing the first frame. The background model is then built as

$$\mathcal{M}^0(x) = \{v^0(y|y \in N_G(x))\}, \tag{5}$$

where pixels in the 8-connected neighborhood are selected randomly following a uniform law. Nonetheless, the official source code implementation provided by the authors generates the background model from one frame with a slightly different strategy. Instead of randomly picking values within the neighborhood, they initialize the background model with the value of the first pixel at each location plus some noise. By adding random noise to the first sample, a set of similar values are built, forming the background model. The authors state that such modification is done to improve the speed of the method and does not affect the results of the algorithm. The pseudo-code of the single-frame initialization is depicted in lines 9-17 of Algorithm 1.

3.3 Update of the Reference Background Model

After processing each frame, background subtraction algorithms update the model with the new information. Traditional methods replace old values every certain number of frames, so that new observed values are represented in the model. As briefly mentioned earlier, ViBe is based on the principle that the most recent samples are not necessarily the most representative of the background. While new information should be considered, older samples can be valid as well. Moreover, the authors define a slight variation of a conservative update policy. The next paragraphs describe in depth this update mechanism and detail its properties.

The authors of ViBe deviate from the classical first-in-first-out approach because the background model is then constrained by a rigid temporal window that depends on the frame-rate of the video sequence. For high frame-rates, the representation of the background will span over a short temporal range. Hence, ViBe uses a stochastic approach to include new values into the model, ensuring a smooth exponential decaying lifespan for the sample values. To do so, when adding a pixel value into the model, one of its samples is randomly selected according to a uniform probability density function and replaced by the new one. This allows a few old samples to stay in the model, which will cover a longer temporal window independently of the frame-rate of the sequence. Figure 2 illustrates this simple update strategy.

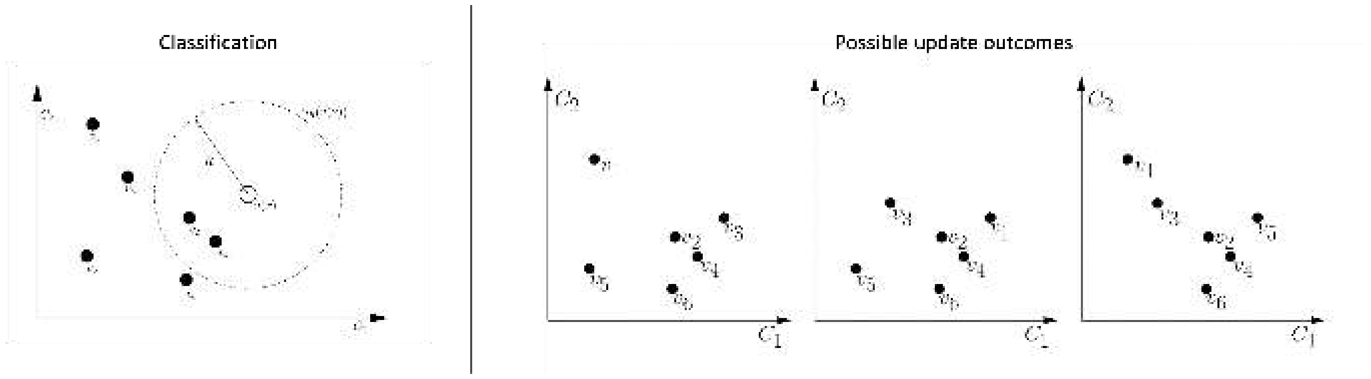


Figure 2: Extracted from the original publication of ViBe [2]. Following the classification example from Figure 1, three different possible update outcomes are shown. On the left, v_3 is randomly chosen and replaced by the new value. In the middle, v_1 is chosen, while v_5 is randomly selected on the right.

Notice that while this allows the model to span over a longer temporal window, it could still not be long enough for high frame-rates, if an update is performed every frame. Consequently, a stochastic time sub-sampling is also used to further enlarge the size of the window. A time sub-sampling factor $\phi \in \mathbb{Z}^+$ is defined to regulate the updating frequency of pixel values. When a pixel value is classified as background, it has one chance in ϕ to be selected to update the pixel model. Barnich and Van Droogenbroeck set $\phi = 16$ in their article.

Lastly, the update mechanism includes a strategy to avoid deadlocks, since this is a potential drawback of a conservative update policy, as discussed in Section 2.1.3. Deadlocks occur when a background pixel is wrongly classified as foreground or when the initial frames of the sequence include foreground objects. To recover from these, a progressive inclusion of foreground samples into the background model is applied. As mentioned earlier, the assumption that neighboring pixels should have similar temporal distributions is taken. Thus, when a pixel is classified as background and updated into the model, it should sometimes update neighboring pixels as well. This policy allows spatial diffusion of background information, which leads to a quicker adaptation to illumination variations and other evolving events in the scene, contrary to a strictly conservative update policy. In practice, when a pixel value is classified as background and selected to update the set of samples $\mathcal{M}(x)$, that same value will be used to update one of the samples in the 4- or 8-connected spatial neighborhood, according to a uniform law. Lines 36-49 from the pseudo-code, shown in Algorithm 1, illustrate the implementation of the update mechanism, where lines 43-46 correspond to the deadlock diffusion strategy.

3.4 Pseudo-code

Pseudo-code shown in Algorithm 1 describes the high-level implementation of ViBe for the parameter values suggested in the original article.

3.5 Dissipating ghosts with frame difference

As mentioned before, the ViBe algorithm is one of the most popular background subtraction methods for its reduced complexity and real-time capabilities. For this reason, other works have explored means to improve its weaknesses [25, 27]. Li et al. [12] summarize the most relevant works in the literature to improve the ViBe. Proposed approaches focus on minimizing the introduction of ghosts in the background model and the classification of shadows as foreground objects. According to Li et al., a high number of improvements are based on multi-frame difference, where adjacent frames are used to determine static and moving object areas. While proposed modifications slightly differ in the way they leverage this information, it seems to generally be an effective strategy to increase robustness and quickly dissipate ghosts. Hence, a version of the 3-frame difference approach proposed in [27] is explained in this section and implemented in the demo (see Section 5).

Computing the 3-frame difference Let $I(x,t)$ be the current observed frame in a video sequence and $I(x,t-1)$ and $I(x,t+1)$ its two adjacent frames, where x corresponds to pixel location. The 3-frame differential image $D(x,\Delta t)$ is then computed as

$$D(x,\Delta t) = \mid I(x,t+1) - I(x,t) \mid \times \mid I(x,t) - I(x,t-1) \mid . \tag{6}$$

From this, a binary image $BW(x)$ is derived using the value of $D(x,\Delta t)$ and a threshold τ, such that

$$BW(x) = \begin{cases} 1, & \text{if } D(x,\Delta t) \geq \tau \\ 0, & \text{if } D(x,\Delta t) < \tau \end{cases}, \tag{7}$$

where values assigned to 0 indicate that there is no relevant movement across the three frames at that position, contrary to the values assigned to 1. Considering that the optimal value of the threshold τ might depend on the specific pixel location, it is adaptively updated to reflect uncommon frame differences at a given position. The steps to select a suitable value of τ are thus:

1. The minimum and maximum pixel values of the image at time t are taken. The value of τ is initially set to the average of those two values.

2. According to Equation 7 and the initial value of τ, the image is divided into two parts: foreground, where $BW(x) = 1$ and background, where $BW(x) = 0$.

Algorithm 1 Pseudo-code of the ViBe algorithm

```
 1: // Parameter definition
 2: $N = 20$; // number of samples
 3: $R = 20$; // radius
 4: $\#_{min} = 2$; // minimum cardinality
 5: $\phi = 16$; // update factor
 6: for every image path do
 7:     Read image file from the path
 8:     if image is the first frame then
 9:         // initialize background model with parameters
10:         instantiate buffer $\mathcal{M}$ of size $N$
11:         for every pixel value do
12:             for i = 1 to N do
13:                 Add random noise to the pixel value (each RGB channel independently)
14:                 Add noisy sample to the buffer $\mathcal{M}$
15:             end for
16:         end for
17:     end if
18:     // Classify frame into binary mask
19:     $currentMatches = 0$;
20:     for every pixel value do
21:         for i = 1 to N do
22:             Compute distance between pixel value and sample $i$
23:             if $distance \leq R$ then
24:                 $currentMatches++$;
25:             end if
26:             if $currentMatches \geq \#_{min}$ then
27:                 Classify pixel as background
28:                 Move matches to the first memory positions of the buffer $\mathcal{M}$
29:                 Break from loop
30:             end if
31:         end for
32:         if $currentMatches < \#_{min}$ then
33:             Classify pixel as foreground
34:         end if
35:     end for
36:     //Update the model
37:     for every pixel classified as background do
38:         // Pick random value from 1 to $\phi$
39:         if $\text{rand}(1, \phi) = 1$ then
40:             Choose a random sample value from the buffer $\mathcal{M}$
41:             Replace value chosen from the buffer $\mathcal{M}$ by pixel value
42:             // Pick random value from 1 to $\phi$
43:             if $\text{rand}(1, \phi) = 1$ then
44:                 Choose a random sample value from the 8-neighbourhood
45:                 Choose a random sample value from the buffer $\mathcal{M}$ (at the location of the selected value
    above)
46:                 Replace value chosen from the buffer $\mathcal{M}$ by pixel value chosen from the 8-neighbourhood
47:             end if
48:         end if
49:     end for
50: end for
```

3. Subsequently, the average pixel value ρ of each of the two parts is computed as
$$\rho_F = \frac{S_F}{N_F}, \rho_B = \frac{S_B}{N_B}, \tag{8}$$
where S_F and S_B are the sum of pixel values of foreground and background portions, respectively. N_F and N_B are the number of pixels in the foreground and background parts, respectively.

4. Lastly, the final value of τ at time t is determined by
$$\tau = \frac{\rho_F + \rho_B}{2}. \tag{9}$$

Updating $\mathcal{M}$ with the 3-frame difference The value of $BW(x)$ indicates where the pixel values have not changed across frames, as well as the locations where the frame difference exhibits a relevant change. In the presence of ghosts, ViBe will wrongly compute background pixels as foreground. Nevertheless, the 3-frame difference in such cases will remain low, since the background structure will not change across frames once the initial moving object is gone. Hence, such information can be used to dissipate ghosts by updating the produced change map $C_m(x)$ for each frame with a 0 at pixel locations where $BW(x) = 0$, i.e.:
$$C_m(x) = \begin{cases} 0, & \text{if } BW(x) = 0 \\ C_m(x), & \text{if } BW(x) = 1 \end{cases}. \tag{10}$$
The proposed strategy will encourage the inclusion of ghost pixels into the background model $\mathcal{M}$, which will quickly dissipate their wrong classification. Furthermore, real moving instances will not be affected, as the areas where they are located will exhibit high frame-difference.

4 Experiments

In this section, ViBe's official source code implementation is used to evaluate its performance and test the algorithm beyond well-known benchmarks. Thus, a set of experiments are carried out with the aim of reviewing the method's results, find its limitations, and illustrate the effect of its parameters. Consequently, all experiments, apart from the ones involving parameter evaluation, are performed with the default parameter values suggested by the authors of the algorithm, i.e. $N = 20$, a radius $R = 20$, a minimum cardinality $\#_{min} = 2$, and an update factor $\phi = 16$.

4.1 Results in Ordinary Conditions

The boundless range of possible scenarios requires background subtraction methods to adapt to both normal cases and extreme scenarios, such as bad weather, dynamic backgrounds or illumination variation. In this section, a set of experiments in controlled conditions are performed, thus observing how ViBe responds to ordinary circumstances. In the next section, more challenging conditions are evaluated to push the algorithm and show its limitations.

In order to understand how the algorithm performs in a well-known benchmark, the first two evaluated sequences correspond to the popular CDNet dataset [6, 24]. Figure 3 shows the results on five frames for each of the two sequences, *subway* (named *PETS2006* in CDNet) and *office*. The first one consists in a surveillance camera shot of a subway station where people walk by. In the office sequence, a person walks in and stands for some time, finally leaving the scene. The results

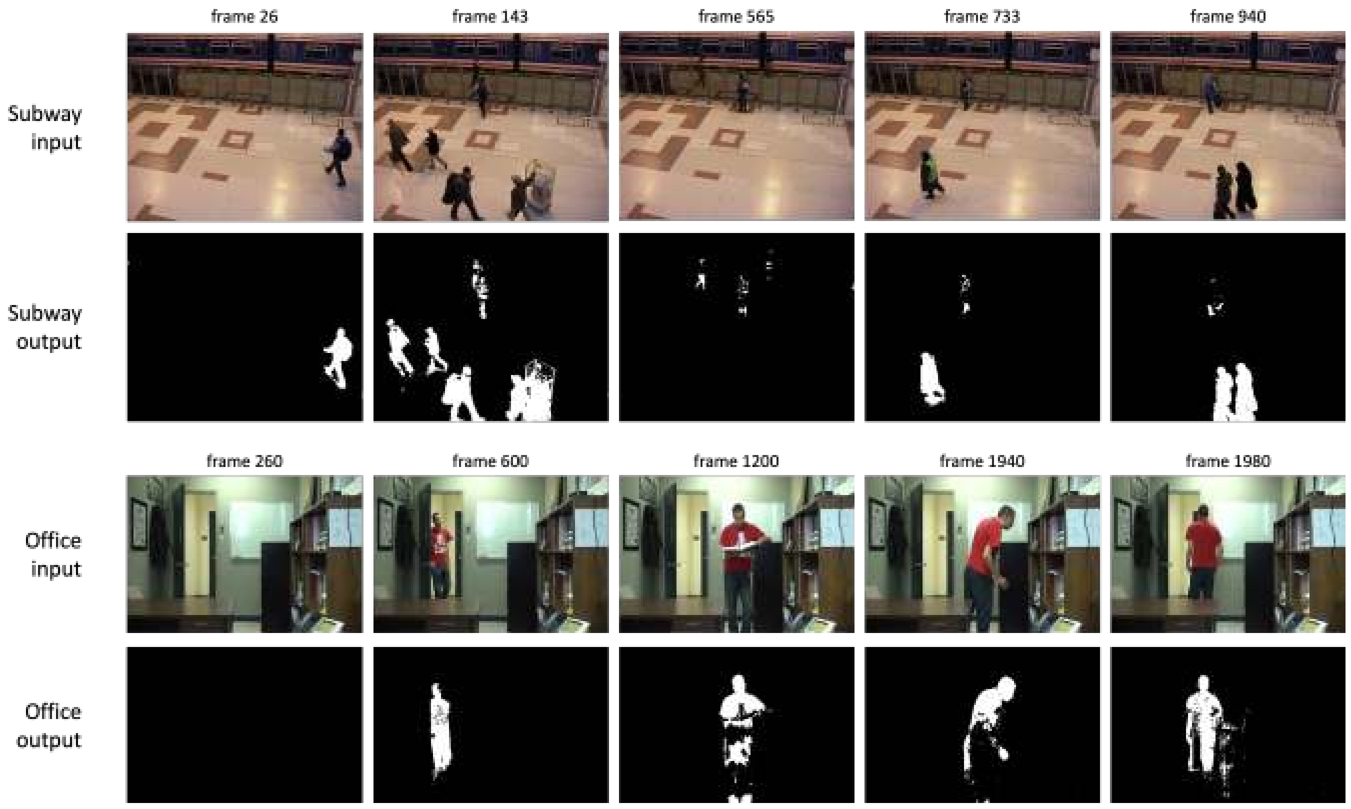


Figure 3: Results of ViBe for a set of frames corresponding to two CDNet Baseline sequences. The two rows on top show the input and output of the *subway* sequence, while the two bottom rows correspond to the input and output of the *office* sequence. The experiments were performed using the default parameters of ViBe, given in lines 1-5 of the pseudo-code, shown in Algorithm 1.

show that ViBe handles well the changes and shows no false positives in both scenes. Nevertheless, a few foreground pixels are classified as background (e.g. the third and fourth frames of the *office* sequence). This appears to be caused by the similarity between the background reference pixels and the foreground entity. ViBe builds a background model using RGB intensity values, thus a high contrast between the background and foreground values will result in smooth detections, while closer values can confuse the algorithm. Furthermore, we can observe how some pixels are not classified as foreground even if they are surrounded by foreground pixels. This characteristic effect is produced because the classification is performed at pixel level.

Next, we evaluate two other sequences that do not contain overly challenging cases, and do not belong to any benchmark. Firstly, we use a house surveillance sequence where a bear is seen dragging the garbage bin. The second sequence corresponds to a person climbing at an indoor gym, where a few people pass by. We name the sequences *hungry bear* and *climbing*, respectively. Figure 4 shows five examples of the results of ViBe for each of the sequences. As observed, the *hungry bear* sequence shows quite impressive results, despite being an outdoor scene. Regardless of the presence of a few false negatives within detected blobs, the overall foreground detections seem to be accurate. The *climbing* sequence appears to be slightly more challenging, as the climber and the pedestrians are well detected, yet a few wrong detections can be observed. In the first frame of Figure 4, some false positives on the floor mattress borders can be seen. This is likely due to the fact that the mattress surface moves to some degree when the climber steps on it. Moreover, there are some pixels corresponding to hard shadows that are classified as foreground. This occurs as the intensity values of these pixels changes further than the threshold.

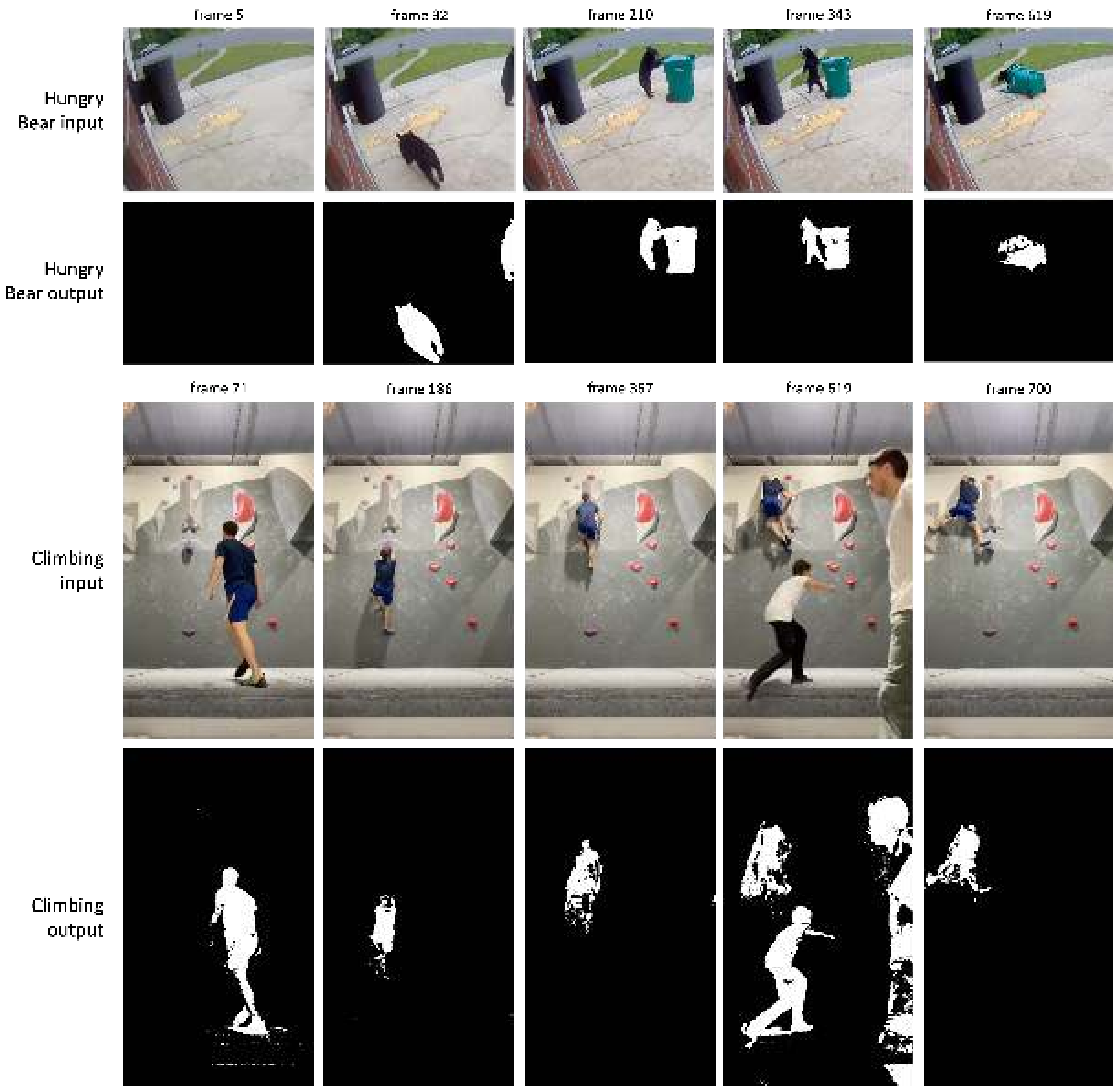


Figure 4: Results of ViBe for a set of frames corresponding to *hungry bear* and *climbing* sequences.

4.2 Limitations

In Subsection 4.1, we observed the results of ViBe in ordinary conditions, where no extreme cases took place. Nonetheless, this will not always be the case, and challenging scenarios such as rain or dynamic backgrounds need to be considered, specially outdoors. For this reason, this section exposes the method to complex scenes to illustrate and comprehend the limitations of ViBe.

4.2.1 Ghosts

In background subtraction, ghost effects are referred to cases where the method wrongly predicts a set of connected pixels as a foreground object. When a foreground object that has been initially attributed to the background moves from its position, the initial area it occupied will be classified as a foreground due to the relevant change in pixel values that will be observed. This is very common when initializing the background model with a frame that includes foreground objects or when a static object considered part of the background, e.g. a parked vehicle, starts moving. Thus, we perform a set of experiments to evaluate the update mechanism of ViBe, which is designed to recover from

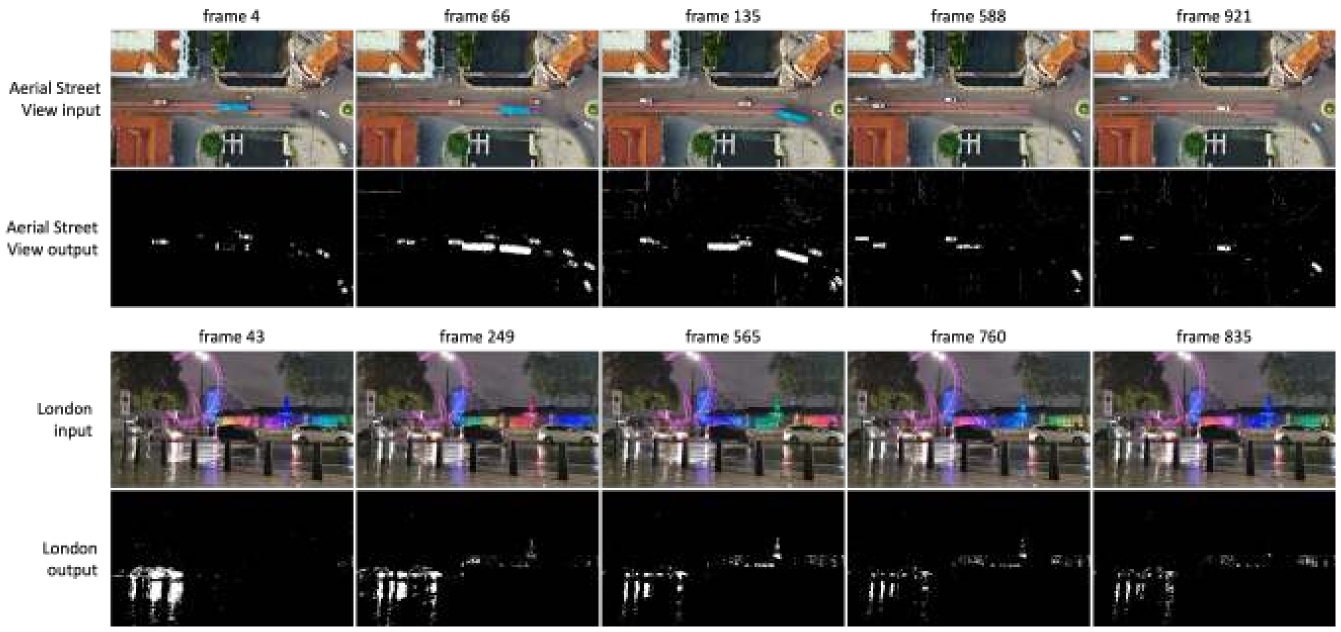


Figure 5: Results of ViBe for a set of frames corresponding to *aerial street view* and *London* sequences.

such ghost effects. The two sequences considered are *aerial street view*, where a street is observed from the air, and *London*, which shows a street with cars during a rainy night in the British capital. The trait that these two sequences have in common is that they both show foreground objects (cars) in the first frame. This leads the algorithm to initialize the background model with these objects, which eventually move and produce ghost effects. This can be seen in Figure 5, which shows some of the results for both sequences. *Aerial street view* initializes the model with a bus and a few cars in the scene, which later become ghosts as they move. As seen in the figure, the second frame of the sequence shows already the moving bus and the region where the bus started (ghost effect), which no longer includes a foreground object. Some other false detections are shown, which seem to be caused by the slight movement of the camera from the air. This phenomenon is addressed in the following section. In regards to the *London* sequence, we can observe quite confusing output masks. This sequence is in fact very challenging for a few reasons. First, the scene corresponds to a rainy night in London, which accentuates the lights of the cars and the street and generate hard shadows. Moreover the first frame includes cars that move later in the sequence. Figure 5 shows how, as the cars slightly move, the ghost effects appear. Moreover, the illumination conditions on the right of the frame (e.g. see the tower at the end) vary throughout the sequence, which results in the algorithm classifying those areas as foreground. As mentioned earlier, the update mechanism of ViBe is designed to alleviate ghost effects with time. This can be seen in the *aerial street view* example, where the ghost effects have dissipated by the end of the sequence. Nonetheless, these false detections stay for quite some time. The effect of the parameters to accelerate ghosts dissipation is discussed later in Section 4.3.

4.2.2 Camera Motion

The background reference model built by ViBe functions at pixel level. When the recording camera moves, the pixel spatial information observed in the new frames does not correspond with the background model. This becomes a problem for the algorithm that results in false positives within the spatially shifted area. The *aerial street view* case shown in Figure 5 illustrates this effect, as some of the pixels around the buildings edges are wrongly classified as foreground pixels. The motion of the camera in this sequence is minimal, but still enough to produce false positives. In the case of

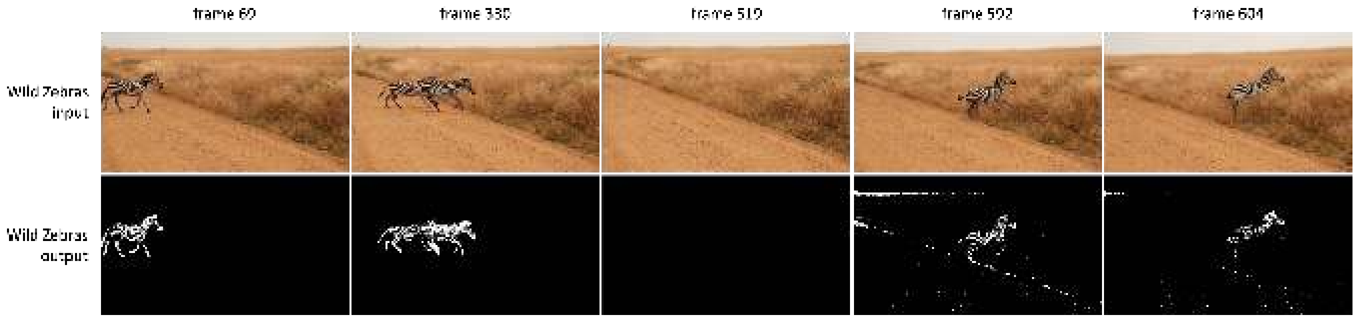


Figure 6: Results of ViBe for a set of frames of the *wild zebras* sequence.

a major motion, the spatial decorrelation between background model and input frames is higher, thus generating a more extreme effect. This can be seen in Figure 6, where the results of a sequence named *wild zebras* is shown. This sequence shows a few zebras crossing a road, recorded by a hand held cellphone. Towards the end of the sequence, the camera moves considerably, and this effect is clearly displayed.

4.2.3 Dynamic Backgrounds and Bad Weather

Finally, we evaluated the response of ViBe to dynamic backgrounds, i.e. when the background of a scene involves moving patterns, which is the case for waving trees or moving water. Furthermore, bad weather produces a similar case where rain or snow continuously fall and challenges the algorithm to model the generated textures. For this reason, two sequences with dynamic backgrounds and challenging weather were tested. These sequences are *snowing river* and *raining street*, and Figure 7 shows the results for both of them. As seen, some false detections are made in both scenes. In *raining street*, the results are more robust and seem to be less prone to false alarms. Nonetheless, some segmentation noise is observed in the regions covered by the street light, where the rain pattern is more visible. Additionally, the light on the upper right hand of the frame turns on throughout the scene, which is considered by the method as a foreground detection. This phenomenon is common in sudden illumination variations. The *snowing river* sequence presents more false positives due to the dynamic background generated by the water. The moving patterns are not captured by the model at the pixel level, which produces these noisy masks. If we look closely, some of the falling snow generates a few false positives as well. This is a well-known challenging case that other methods, such as SuBSENSE, try to solve by leveraging spatial information at each pixel. Nevertheless, ViBe does not incorporate such mechanism and therefore dynamic backgrounds are often problematic.

4.3 Effect of Parameters

In Section 3, we described in detail ViBe's algorithm and introduced its four main parameters, namely the number of samples in the background model N, the radius or matching threshold R, the minimum cardinality or minimum matches $\#_{min}$, and the update factor ϕ. In this section, we conduct a set of experiments aimed to understand the effect of these parameters, and how can they help mitigate some of the pitfalls of the method.

4.3.1 Number of Samples in the Background Model and Minimum Cardinality

The number of samples in the background model N is a key element that determines the quantity of values per pixel stored as a representation of the background. By default, Braham and Van Droogenbroeck proposed $N = 20$. Intuitively, increasing this collection of values should lead to a

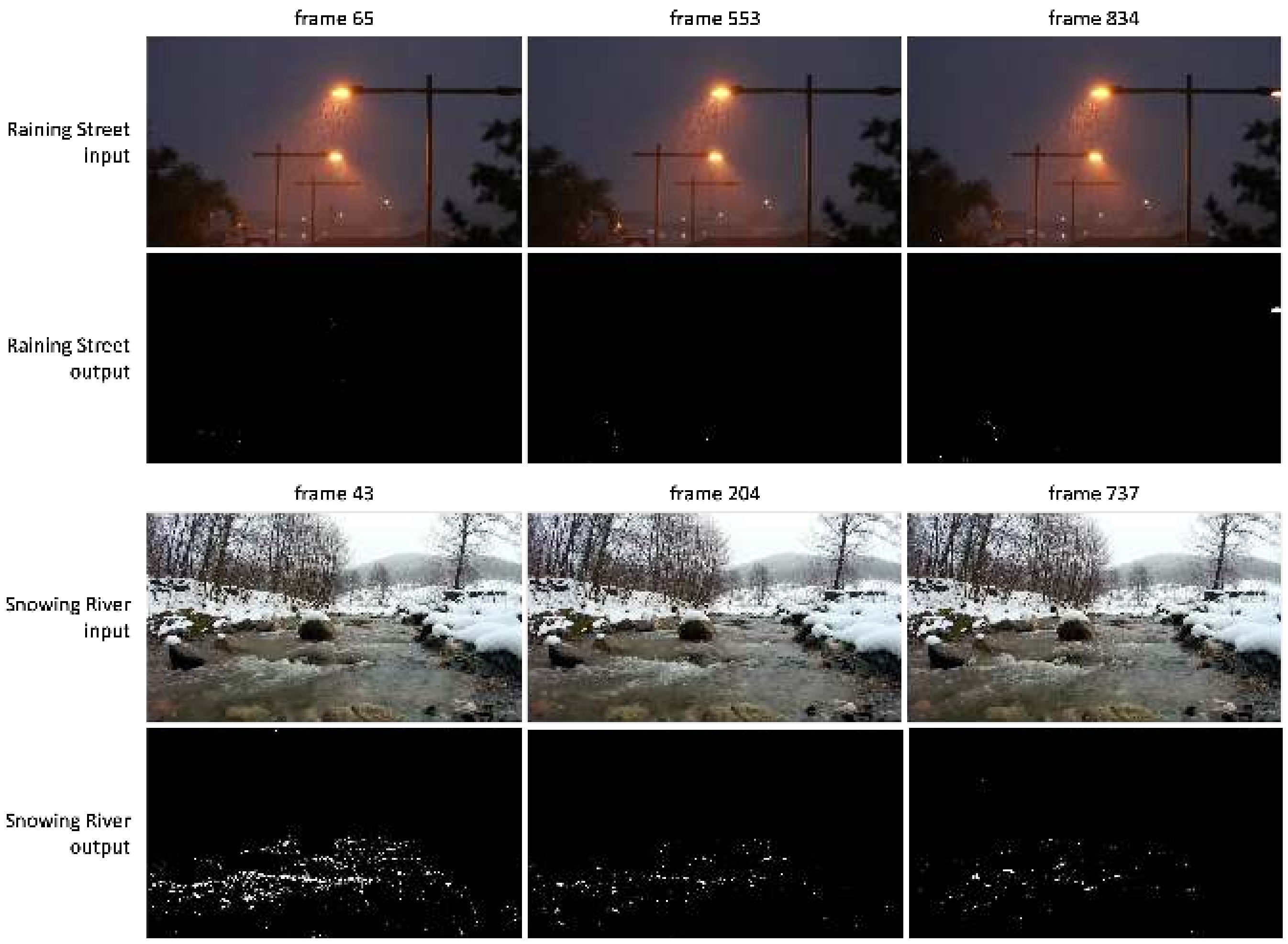


Figure 7: Results of ViBe for a set of frames of the two sequences involving bad weather and dynamic backgrounds.

better representation of the background. We evaluated two of the challenging sequences discussed in Section 4 with different values of N to analyze the effect of the parameter. The results for this experiment are shown in Figure 8. A frame of the sequences *snowing river* and *aerial street view* are displayed for $N = 10$, $N = 20$, $N = 30$ and $N = 40$. As observed, the less samples used to represent the background, the more false positives are observed in the binary masks. The difficulty in *snowing river* resides in modelling the snow and water patterns of the scene, which is much more challenging for a lower N. In the case of *aerial street view*, some false positives appear due to the slight camera movement. These are attenuated as N increases as well. Thus, it seems that a higher N will lead to a better representation of the background. Nonetheless, this also increases the computational complexity. When observing a pixel value, the computation of the distances to the elements in the background model need to be computed. For this reason, we evaluated the effect of different values of N over the execution time of the algorithm. Two phenomena can be observed in the results, shown in Figure 9. On the one hand, the execution time increases for greater values of N, but this occurs at a much lower rate than N. On the other hand, we observe that the execution time of different sequences grow at different rates. Both cases are a consequence of the mechanism explained in Subsection 3.1.1. The official ViBe implementation changes the memory positions of the values in the background model to reach the minimum cardinality faster. This causes that, for $\#_{min} = 2$, the two first values checked have a high probability of being a match for a background pixel, thus discarding the rest of the samples and not computing their distances. However, for a foreground pixel, all the values in the model need to be checked, i.e. the distances for all N values

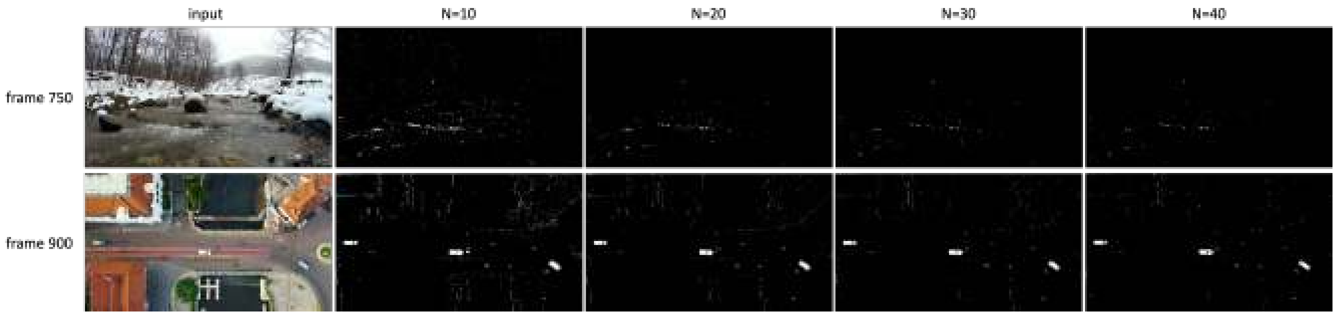


Figure 8: Results of ViBe for different values of the parameter number of samples. The figure shows the results of two sequences, *snowing river* and *aerial street view*, for a number of samples of $N = 10$, $N = 20$, $N = 30$ and $N = 40$.

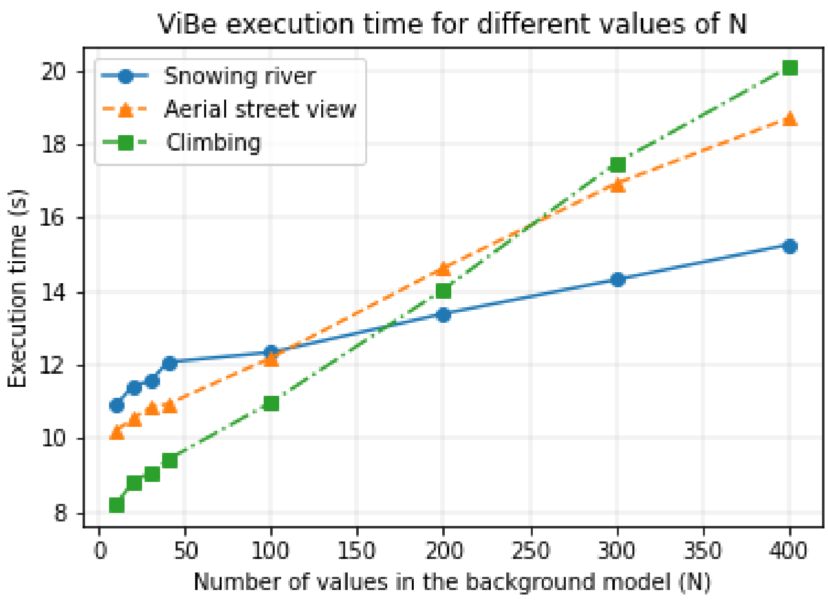


Figure 9: Evaluation of the execution time of ViBe for different values of N.

need to be computed. Therefore, the execution time for sequences that have more foreground pixels will experience a faster increase as the parameter N is raised. Notice in Figure 8 how the *climbing* sequence, which includes constant foreground objects, shows a steeper ascent than the *snowing river* sequence, which includes no real foreground objects.

As mentioned earlier, the ratio between $\#_{min}$ and N can be used to control the sensitivity of the method. Increasing $\#_{min}$ will require more samples in the model to be similar to the observed sample to consider it a background pixel, thus it will be more prone to false positives. Furthermore, due to the implementation of the source code changing memory positions increases the probability of finding a match sooner. For this reason, it is more convenient to regulate the method's sensitivity with N than with $\#_{min}$, as increasing $\#_{min}$ will implicitly increase the required checks every frame, while increasing N will not.

4.3.2 Radius or Matching Threshold

The radius R parameter dictates how different an observed value needs to be from the samples in the model for it to be considered a foreground pixel. Low values of R will only allow very similar values to be considered a match, while larger values make the method more permissive in that sense. Hence, this parameter can be used to regulate the strictness of the method to cluster pixels as

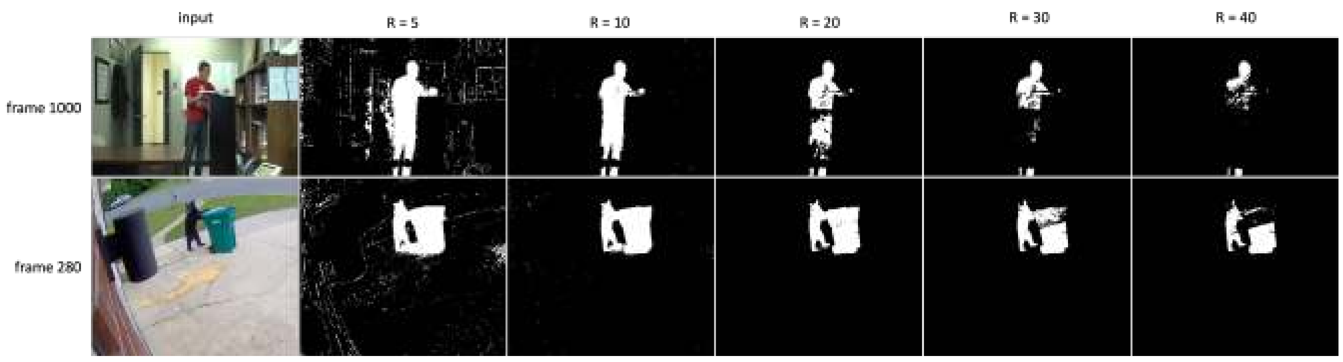


Figure 10: Results of ViBe for different values of the radius. The figure shows the results of two sequences, *office* and *hungry bear*, for a radius $R = 5$, $R = 10$, $R = 20$, $R = 30$ and $R = 40$.

background. Lower values of R will provide a more complete detection of foreground objects, with a low false negative rate. Nonetheless, this will make the method more sensitive to false positives. Moreover, large radii make the method more robust to false positives, but foreground pixels become more likely to be classified as background information. This can be observed in Figure 10, where for $R = 5$ the two observed sequences show precise blobs, but include several false positives. Conversely, for $R = 40$ there are no false positives but the foreground detections appear to be quite poor. Furthermore, notice that in the *office example*, the shirt and the head of the person are successfully detected even for higher values of R, while the pants are not. This is due to the relative intensity of the foreground body compared to the background. The background area corresponding to the upper body of the person is much brighter and therefore highly contrasted with the foreground object. On the other hand, the background behind the lower body of the person is darker and not as contrasted, which makes the pants be considered part of the background for higher values of R. A similar effect occurs with the upper part of the green garbage bin for the *hungry bear* sequence. That area of the bin is superposed to the lawn in the background, which is green as well and the relative intensity change does not exceed the threshold for higher values of R.

4.3.3 Update Factor

As explained in Section 3, the update factor ϕ is a mechanism to regulate the speed at which the background model is updated. Furthermore, it controls the speed at which neighbouring values are injected into the model after updating it, as a mean to dissipate ghost effects and avoid deadlocks. To analyze how this parameter affects ghosts dissipation and foreground detections, we evaluate the sequences *aerial street view* and *office* for different update factor values. The results of the experiment are shown in Figure 11. Looking at *aerial street view*, we observe that the lower the value of the update factor, the faster the ghosts fade out in the binary mask. For $\phi = 16$, the ghost effects at frame 200 are clearly present and, in spite of the fact that they slowly dissipate, they are still visible in frame 700. For an update factor of 1, these effects are removed much faster and we can barely see them at frame 200. One could then think that the most obvious choice is to set a low value for the update parameter. Nevertheless, this can have contradictory effects. In the *office* sequence, a person stays at the same place, reading a book, for several frames. Since the update factor accelerates the rate at which neighboring information is injected to the model when a value is selected for the update, this will impact static foreground detections as well. Thus, low update factors will dissipate valid detections when they become static for several frames, as seen in Figure 11 for $\phi = 1$. Hence, a balance between ghost effect removal and static foreground object dissipation should be set according to the downstream task of the application.

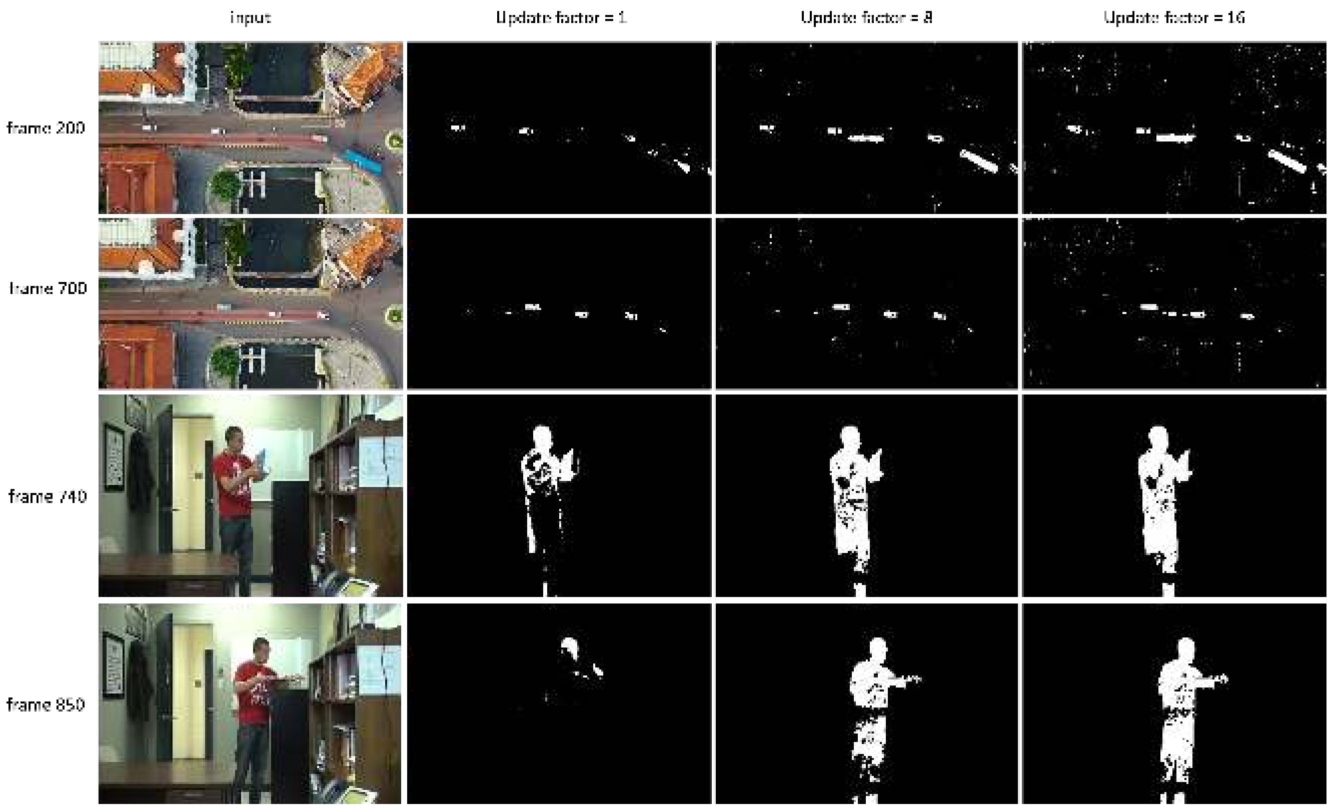


Figure 11: Results of ViBe for different values of the parameter update factor. The figure shows the results of two sequences, *aerial street view* and *office*, for two frames each and update factor values of 1, 8 and 16.

4.3.4 Impact of frame difference variant

A variant of the original ViBe algorithm was detailed in Section 3.5 in order to quickly dissipate ghosts, in cases where the initialization frame contains moving objects. We illustrate its effects compared to the original implementation in Figure 12. Three different frames of the *aerial street view* sequence are shown. This sequence contains several moving vehicles in the initial frame. As observed, the method without the 3-frame difference approach is very slow at dissipating the originated ghost effects. On the contrary, the 3-frame difference variant reduces such phenomenon surprisingly quickly. Consequently, 3-frame difference can be regarded as an effective way to minimize the sensitivity of the ViBe algorithm to ghost effects. Nonetheless, in cases where moving objects stop, these will be quickly introduced into the background model, as the 3-frame difference observed at those points will not be relevant. This should be considered when applying the algorithm to a specific application, as one might be interested in intermittent object motion, or not.

5 Demo

This article is released with an easy-to-use demo to test the performance of ViBe for custom data without the need to write any code. Moreover, the demo allows to fine-tune the parameters of the algorithm with simple sliders and to download the produced output binary masks.

5.1 Running the Demo

The demo can be easily run by selecting an input sequence and defining the desired parameters. Figure 13 shows an image of the input and parameter setting area. The proposed input video

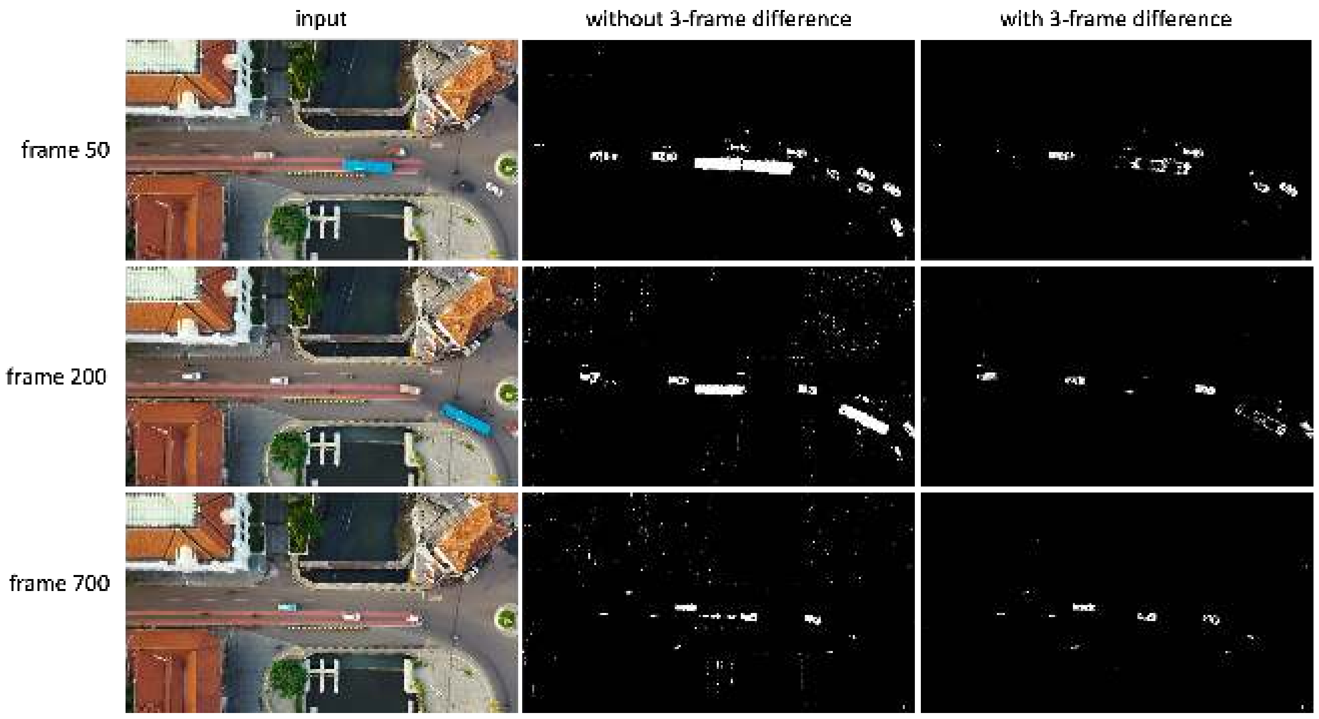
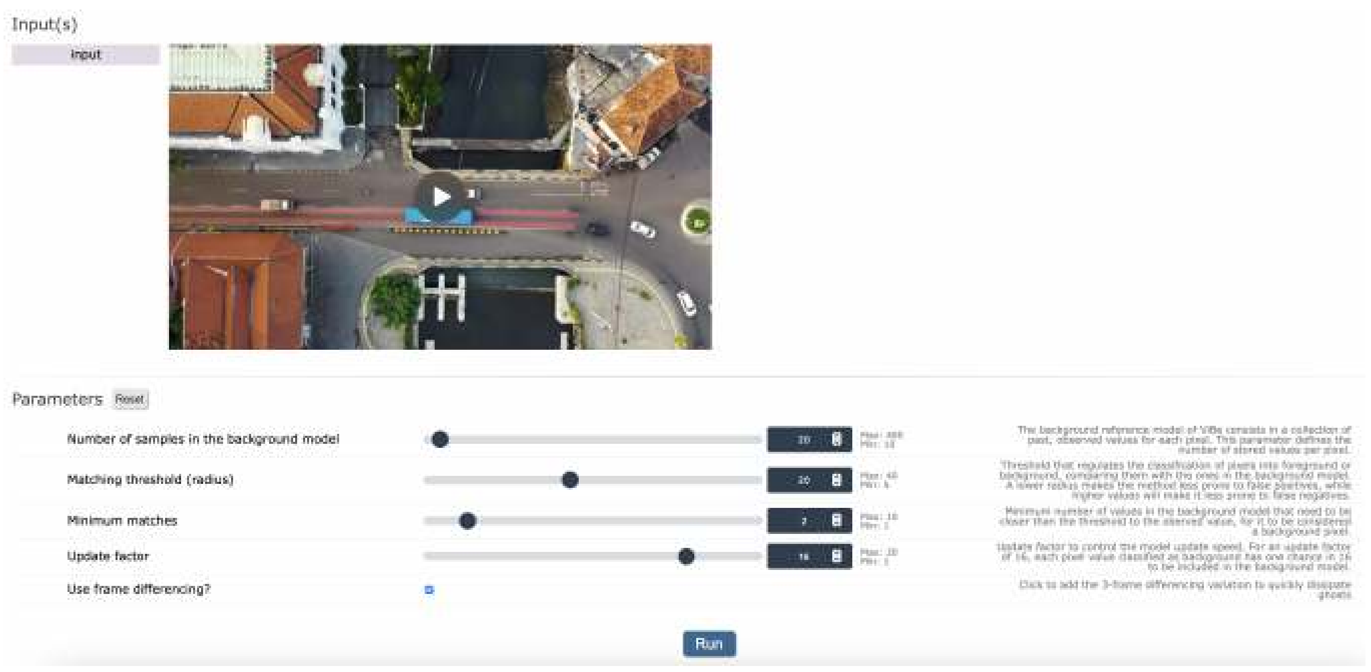


Figure 12: Illustrative example of the effect of the 3-frame difference variant to quickly dissipate ghosts, for three frames of the *aerial street view* sequence.

sequences are those discussed in this article. Clicking on the *upload data* button right above the suggested sequences enables the upload of any (short enough) video. Once an input is selected, the parameters of the algorithm can be modified by moving the displayed sliders. Additionally, the checkbox on the bottom can be used to select the detailed 3-frame difference variant of ViBe. To run the demo, click on the *Run* green button below the parameters area.

5.2 Visualizing the Results

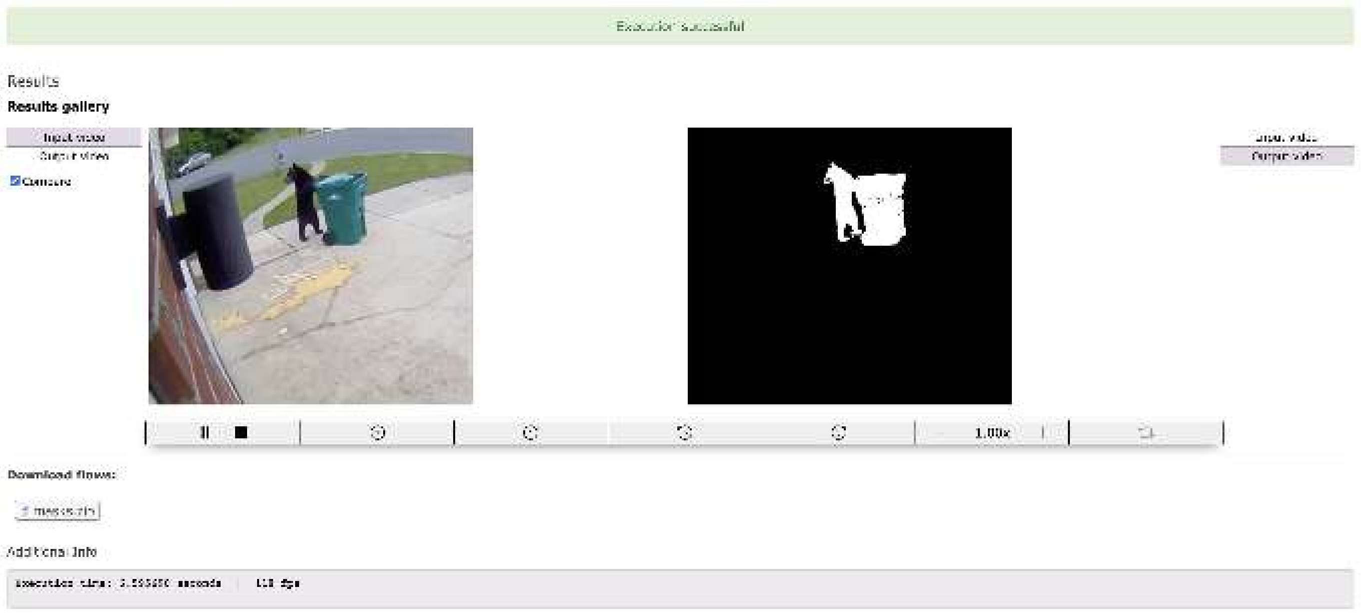
When the demo is finished, a similar layout as the one shown in Figure 14 will appear. To visualize the results, click on the *Compare* box on the upper left of the results section, and select the *Output* option in one of the two displayed videos. Click on the play button on the media bar below both sequences to run them simultaneously, thus observing the output of ViBe for each of the frames of the input sequence. The execution time in seconds and frames per second are also shown at the bottom of the page, within the *additional information* area. Finally, to download the produced binary masks, click on the *masks.zip* button on the bottom left side of the screen.



Parameters

Number of samples in the background model	20	Max: 400 Min: 10	The background reference model of ViBe consists in a collection of past, observed values for each pixel. This parameter defines the number of stored values per pixel.
Matching threshold (radius)	20	Max: 40 Min: 4	Threshold that regulates the classification of pixels into foreground or background, comparing them with the ones in the background model. A lower radius makes the method less prone to false positives, while higher values will make it less prone to false negatives.
Minimum matches	2	Max: 10 Min: 1	Minimum number of values in the background model that need to be closer than the threshold to the observed value, for it to be considered a background pixel.
Update factor	16	Max: 20 Min: 1	Update factor to control the model update speed. For an update factor of 16, each pixel value classified as background has one chance in 16 to be included in the background model.
Use frame differencing?	☐		Click to add the 3-frame differencing variation to quickly dissipate ghosts

Figure 13: Capture of the input field and parameter setting area of the developed demo.



```
Execution time: [illegible] seconds | [illegible] fps
```

Figure 14: Capture of the output view of the developed demo.

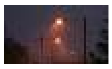
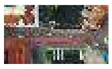
Image Credits

from the ViBe original article [2].

from the CDNet dataset [6, 24]

taken by Xavier Bou.

from Northwest Florida Daily News.

  from Pexels[1].

References

[1] M. Babaee, D.T. Dinh, and G. Rigoll, *A deep convolutional neural network for background subtraction*, CoRR, abs/1702.01731 (2017). https://doi.org/10.48550/arXiv.1702.01731.

[2] O. Barnich and M. Van Droogenbroeck, *ViBe: A Universal Background Subtraction Algorithm for Video Sequences*, IEEE Transactions on Image Processing, 20 (2011), pp. 1709–1724. https://doi.org/10.1109/TIP.2010.2101613.

[3] T. Bouwmans, S. Javed, M. Sultana, and S.K. Jung, *Deep neural network concepts for background subtraction:a systematic review and comparative evaluation*, Neural Networks, 117 (2019), pp. 8–66. https://doi.org/10.1016/j.neunet.2019.04.024.

[4] M. Braham and M. Van Droogenbroeck, *Deep background subtraction with scene-specific convolutional neural networks*, in International Conference on Systems, Signals and Image Processing (IWSSIP), 2016, pp. 1–4. https://doi.org/10.1109/IWSSIP.2016.7502717.

[5] B. Garcia-Garcia, T. Bouwmans, and A.J. Rosales Silva, *Background subtraction in real applications: Challenges, current models and future directions*, Computer Science Review, 35 (2020), p. 100204. https://doi.org/10.1016/j.cosrev.2019.100204.

[6] N. Goyette, P-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, *Changedetection.net: A new change detection benchmark dataset*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2012, pp. 1–8. https://doi.org/10.1109/CVPRW.2012.6238919.

[7] P. KaewTraKulPong and R. Bowden, *An improved adaptive background mixture model for real-time tracking with shadow detection*, in Video-based surveillance systems, Springer, 2002, pp. 135–144. https://doi.org/10.1007/978-1-4615-0913-4_11.

[8] R.E. Kalman, *A new approach to linear filtering and prediction problems*, Journal of Basic Engineering, 82 (1960), pp. 35–45.

[9] R. Kalsotra and S. Arora, *A comprehensive survey of video datasets for background subtraction*, IEEE Access, 7 (2019), pp. 59143–59171. https://doi.org/10.1109/ACCESS.2019.2914961.

[1] https://www.pexels.com/

[10] M. Kampffmeyer, A-B. Salberg, and R. Jenssen, *Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 1–9. https://doi.org/10.1109/CVPRW.2016.90.

[11] D. Koller, J. Weber, and J. Malik, *Robust multiple car tracking with occlusion reasoning*, in European Conference on Computer Vision (ECCV), Springer, 1994, pp. 189–196.

[12] S. Li, Z. Li, and W. Li, *An Overview of Improved ViBe Algorithms*, in International Conference on Computer and Communications (ICCC), 2021, pp. 594–598. https://doi.org/10.1109/ICCC54389.2021.9674734.

[13] J.L. Lisani, L. Rudin, and A. Buades, *Fast video search and indexing for video surveillance applications with optimally controlled false alarm rates*, in IEEE International Conference on Multimedia and Expo, 2011, pp. 1–6. https://doi.org/10.1109/ICME.2011.6012151.

[14] H. Nguyen and A. Smeulders, *Robust tracking using foreground-background texture discrimination*, International Journal of Computer Vision, 69 (2006), pp. 277–293. https://doi.org/10.1007/s11263-006-7067-x.

[15] M. Pietikinen, *Local binary patterns*, Scholarpedia, 5 (2010), p. 9775. https://doi.org/10.4249/scholarpedia.9775.

[16] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*, Artech house, 2003. ISBN 978-1580536318.

[17] D. Sakkos, H. Liu, J. Han, and L. Shao, *End-to-end video background subtraction with 3D convolutional neural networks*, Multimedia Tools and Applications, 77 (2018). https://doi.org/10.1007/s11042-017-5460-9.

[18] J. Schlemper, J. Caballero, J.V. Hajnal, A.N. Price, and D. Rueckert, *A deep cascade of convolutional neural networks for dynamic MR image reconstruction*, IEEE Transactions on Medical Imaging, 37 (2017), pp. 491–503. https://doi.org/10.1109/TMI.2017.2760978.

[19] P-L. St-Charles, G-A. Bilodeau, and R. Bergevin, *SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity*, IEEE Transactions on Image Processing, 24 (2015), pp. 359–373. https://doi.org/10.1109/TIP.2014.2378053.

[20] C. Stauffer and W.E.L. Grimson, *Adaptive background mixture models for real-time tracking*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), vol. 2, 1999, pp. 246–252 Vol. 2. https://doi.org/10.1109/CVPR.1999.784637.

[21] P. Suo and Y. Wang, *An improved adaptive background modeling algorithm based on Gaussian Mixture Model*, in International Conference on Signal Processing, 2008, pp. 1436–1439. https://doi.org/10.1109/ICOSP.2008.4697402.

[22] L. Čehovin, M. Kristan, and A. Leonardis, *Robust visual tracking using an adaptive coupled-layer visual model*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2012). https://doi.org/10.1109/TPAMI.2012.145.

[23] H. Wang and D. Suter, *A consensus-based method for tracking: Modelling background scenario and foreground appearance*, Pattern Recognition, 40 (2007), pp. 1091–1105. https://doi.org/10.1016/j.patcog.2006.05.024.

[24] Y. Wang, P-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, *CDnet 2014: An expanded change detection benchmark dataset*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 387–394. https://doi.org/10.1109/CVPRW.2014.126.

[25] J-Q. Xu, P-P. Jiang, H-B. Zhu, and W. Zuo, *An Improved ViBe Algorithm for Moving Object Detection*, Journal of Northeastern University (Natural Science), 36 (2015), p. 1227.

[26] S.S. Yadav and S.M. Jadhav, *Deep convolutional neural network based medical image classification for disease diagnosis*, Journal of Big Data, 6 (2019), pp. 1–18. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0276-2.

[27] W. Yuanbin and R. Jieying, *An Improved Vibe Based on Gaussian Pyramid*, in International Conference on Control and Robotics Engineering (ICCRE), 2019, pp. 105–109. https://doi.org/10.1109/ICCRE.2019.8724176.

[28] Z. Zivkovic, *Improved adaptive Gaussian mixture model for background subtraction*, in International Conference on Pattern Recognition (ICPR), vol. 2, 2004, pp. 28–31 Vol.2. https://doi.org/10.1109/ICPR.2004.1333992.