applied sciences

MDPI

Article

Robust Background Subtraction via the Local Similarity Statistical Descriptor

Dongdong Zeng [1,2,*], Ming Zhu [1], Tongxue Zhou [1,2], Fang Xu [1,2] and Hang Yang [1]

1 Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China; zhu_mingca@163.com (M.Z.); zhoutongxue1992@163.com (T.Z.); xufang59@126.com (F.X.); yanghang09@mails.jlu.edu.cn (H.Y.)
2 University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: zengdongdong13@mails.ucas.edu.cn; Tel.: +86-13514499061

Received: 5 September 2017; Accepted: 21 September 2017; Published: 25 September 2017

Abstract: Background subtraction based on change detection is the first step in many computer vision systems. Many background subtraction methods have been proposed to detect foreground objects through background modeling. However, most of these methods are pixel-based, which only use pixel-by-pixel comparisons, and a few others are spatial-based, which take the neighborhood of each analyzed pixel into consideration. In this paper, inspired by a illumination- invariant feature based on locality-sensitive histograms proposed for object tracking, we first develop a novel texture descriptor named the Local Similarity Statistical Descriptor (LSSD), which calculates the similarity between the current pixel and its neighbors. The LSSD descriptor shows good performance in illumination variation and dynamic background scenes. Then, we model each background pixel representation with a combination of color features and LSSD features. These features are then embedded in a low-cost and highly efficient background modeling framework. The color and texture features have their own merits and demerits; they can compensate each other, resulting in better performance. Both quantitative and qualitative evaluations carried out on the change detection dataset are provided to demonstrate the effectiveness of our method.

Keywords: background subtraction; locality-sensitive histograms; local similarity statistical descriptor; video surveillance

1. Introduction

Foreground detection based on change detection is the first step in numerous computer vision applications. Output of the background subtraction is usually an input to a post-higher level process, such as video surveillance, object tracking or activity recognition. Therefore, its performance has a huge effect on the performance of higher level tasks. Needless to say, the quality of many computer vision applications directly depends on the quality of the background subtraction method used.

The general idea of background subtraction is to automatically generate a binary mask which classifies the set of pixels into foreground and background. In the simplest case, disparities between the current frame and a background reference frame are usually indicative of foreground objects; this might work in certain specialized scenarios. However, finding a good empty background reference frame in order to perform actual "background subtraction" is always impossible due to the complexity of real-world scenes, due to, for example, illumination changes and dynamic backgrounds. Thus, a multitude of more sophisticated methods have been proposed in the recent past [1,2]. Their efforts mainly focus on two aspects: the first takes on more sophisticated learning modes, while employs more powerful feature representations.

In the early research stage, researchers assumed that the history of the pixel's intensity could be modeled by some distributions. Following this idea, Wren et al. [3] proposed using a single-Gaussian model to model the distribution of intensities at each pixel location. However, a single model cannot handle dynamic scenes when there is rippling of water or waving trees. Then, a Gaussian mixture model [4] was proposed to solve this problem, which models the color intensities at each pixel location using a mixture of Gaussian probability density functions. Many improvements were developed to make it more adaptive and robust to critical situations. For example, in [5], the authors extended this idea by allowing dynamic Gaussian numbers to model each pixel as well as to improve their convergence rate. In another work, Allili et al. [6] proposed a mixture of general Gaussian to alleviate the constraint of strict Gaussian. However, the Gaussian assumption for pixel intensity distribution is not always true in practical applications. Hence, a nonparametric approach based on Kernel Density Estimation (KDE) was proposed in [7], which builds a statistical representation of the scene background by estimating the probability density function directly from the data without any priori assumptions. In [8], to reduce the burden of image storage, Jeisung et al. modified the KDE method by using an adaptive learning rate according to different situations, which allows the model to automatically adapt to various environments. However, the KDEs are time-consuming, and most of them update their models in a first-in-first-out (FIFO) strategy. Thus, they are unable to model both short-term and long-term periodic events.

The authors of [9] presented an alternative approach to solve the above problem. For each pixel, a codebook is constructed and consists of one or more codewords; history samples at each pixel location are clustered into a set of codewords based on a color distortion metric together with brightness bounds. The number of codewords in each codebook is different following the pixel's activities. During the detection phase, if the current pixel is similar to one of the codewords, it is classified as a background pixel; otherwise, it will be considered as a foreground pixel. The codebook representation is efficient in speed and memory compared with other traditional models, and the original algorithm has also been improved in several ways. For examples, Sigari et al. [10] proposed a two-layer codebook model. The first layer in the main codebook models the current background images, while the second layer is the cache which models new background images. Wu et al. [11] proposed an improved codebook by incorporating the spatial-temporal context of each pixel.

Unprecedented background subtraction methods based on neural networks have been proposed in [12,13] and achieve good results on various scenarios. The Self-Organizing Background Subtraction (SOBS) algorithm models each pixel with a neural map of weight vectors. Moving objects are detected through a map of motion and stationary patterns. The background model update at each pixel location is influenced by the labeling decision of its neighbors. As can be seen, more and more recent methods tend to account for neighboring pixels to add robustness to noise. For examples, superpixels and Markov Random Fields [14], as well as the connected components [15] focus on improving label coherence using advanced regularization techniques. Some other methods rely on the region level [16,17], frame level [18] or hybrid frame–region level [19].

The first non-deterministic background subtraction method, called ViBe, was proposed in [20] and has been shown to outperform many existing methods. Instead of building the probability distribution of the background for each pixel using a Parzen window, ViBe uses a stochastic maintenance strategy to integrate new information into the model. If the pixel in the new frame matches some of the background samples, it is classified as background and has a probability of being inserted into the sample model at the corresponding pixel location. The authors show that the stochastic strategy ensures a smooth, exponentially decaying lifespan for the samples that constitute the pixel models. In order to maintain spatial consistency, a spatial information propagation strategy randomly diffuses pixel values across neighboring pixels, even the ones marked as foreground. Due to its simplicity and effectiveness, we use a similar model update strategy in our background subtraction framework.

As more effective models appear, more powerful feature representations are also developed to better adapt to challenging situations. These features include color features, edge features, stereo features, motion features and texture features. The Local Binary Pattern (LBP) feature [21] is the first texture feature proposed for background subtraction. Each pixel is modeled as a group of LBP histograms calculated over its neighborhoods. This method was demonstrated to be tolerant to illumination variations and robust against multimodal background regions, but at the expense of sensitivity to subtle local texture changes. An improved version of LBP called the Scale-Invariant Local Ternary Pattern (SILTP) was proposed in [22], which exceeds LBP in computational efficiency and tolerance to noises. Recently, Local Binary Similarity Patterns were proposed in [23], based on absolute difference, and were demonstrated to surpass traditional color comparisons via Hamming distance thresholding. Despite the fact that both of them are robust to illumination variations, they perform poorly in flat areas and result in "holes" in objects. Then, some researchers began to combine different features to benefit from eachother. For example, Yao et al. [24] proposed a multi-layer background model based on color features and texture features. Han et al. [25] proposed a background subtraction method using a Support Vector Machine over background likelihood vectors for a set of features which consist of color, gradient, and Haar-like features. More recently, St-Charles et al. [26] performed the background subtraction by integrating the color features and the Local Binary Similarity Pattern (LBSP) features, and showed state-of-the-art performance.

In this paper, we present a robust background subtraction method which combines color features and texture features to characterize pixel representations. Our contributions lie in three aspects. First, inspired by an illumination invariant feature based on locality-sensitive histograms proposed for object tracking [27], we develop a novel texture feature named the Local Similarity Statistical Descriptor (LSSD). The LSSD calculates the similarity between the current pixel and its neighborhood pixels. Second, the color features and LSSD features have their own merits and demerits, they can compensate each other for better performance, so a combination of color features and LSSD features are embedded in a low-cost and highly efficient background modelling framework. Third, using the change detection dataset [28], we evaluate our method against numerous surveillance scenes and the results show that the proposed method outperforms most state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 introduces the proposed Local Similarity Statistical Descriptor (LSSD). Section 3 describes the framework for background subtraction. Experimental results on the change detection dataset [28] are reported in Section 4. Finally conclusions are given in Section 5.

2. Local Similarity Statistical Descriptor

In this section, the novel texture feature the Local Similarity Statistical Descriptor (LSSD) will be described in detail. Color features are the most widely-used features in background subtraction, but they present several limitations like shadows, camouflage, and illumination variations. In our background subtraction method, color and texture features are extracted as the primitive representation for background model.

2.1. Illumination-Invariant Feature

Before introducing the Local Similarity Statistical Descriptor (LSSD) feature, we make a brief review of the illumination-invariant feature (IIF). The conventional image histogram is a 1D array. Each of its values indicates the frequency of occurrence of the intensity value. Let $\boldsymbol{I}$ denote an image, and then the image histogram can be calculated as:

$$\boldsymbol{H}(b) = \sum_{q=1}^{W} Q(\boldsymbol{I}_q, b), \qquad b = 1, \ldots, B \tag{1}$$

where W is the number of pixels, B is the number of bins, and $Q(\boldsymbol{I}_q, b)$ is zero except when the pixel value of $\boldsymbol{I}_q$ belongs to bin b.

In [27], He et al. proposed a novel locality-sensitive histogram (LSH) which computes a local histogram at each pixel location. Instead of counting the frequency of occurrences of each intensity value by adding ones to the corresponding bin, a floating-point value is added to the bin for each occurrence of the intensity value. The LSH at pixel p can be computed by:

$$\boldsymbol{H}_p^E(b) = \sum_{q=1}^{W} \alpha^{|p-q|} \cdot Q(\boldsymbol{I}_q, b), \quad b = 1, \dots, B \tag{2}$$

where $\alpha \in (0,1)$ is a constant parameter controlling the deceasing weight as a neighboring pixel moves away from p, and E is the window centered at pixel p.

Let $\boldsymbol{I}_p$ and $\boldsymbol{I}'_p$ denote the intensity values of p before and after an affine illumination change. Then, we have:

$$\boldsymbol{I}'_p = \mathcal{A}_p(\boldsymbol{I}_p) = a_{1,p}\boldsymbol{I}_p + a_{2,p} \tag{3}$$

where $a_{1,p}$ and $a_{2,p}$ are affine parameters.

Now, the number of pixels in the window E with intensity values falling in the interval $[b_p - r_p, b_p + r_p]$ is:

$$\mathcal{I}_p = \sum_{b=b_p-r_p}^{b_p+r_p} \boldsymbol{H}_p^E(b) \tag{4}$$

where b_p denotes the bin corresponding to the intensity value $\boldsymbol{I}_p$, and r_p controls the size of the interval. If r_p scales linearly with the illumination change:

$$r'_p = a_{1,p} r_p \tag{5}$$

After the affine illumination change, the value of Equation (4) can be calculated as follows:

$$\mathcal{I}'_p = \sum_{b=b'_p-r'_p}^{b'_p+r'_p} \boldsymbol{H}_p^E(b) \tag{6}$$

Using Equations (3) and (5), then $[b'_p - r'_p, b'_p + r'_p] = [a_{1,p}b_p + a_{2,p} - a_{1,p}r_p, a_{1,p}b_p + a_{2,p} + a_{1,p}r_p = [a_{1,p}(b_p - r_p) + a_{2,p}, a_{1,p}(b_p + r_p) + a_{2,p}] = [\mathcal{A}_p(b_p - r_p), \mathcal{A}_p(b_p + r_p)]$, if we ignore the quantization error $a_{2,p}$, the value of $\mathcal{I}'_p$ is equal to $\mathcal{I}_p$. Thus, we can notice that $\mathcal{I}_p$ is independent of illumination changes and can be used as an illumination invariant feature. In Figure 1, two gray images with different illuminations and their corresponding illumination invariant feature images are presented. We can see that although the input images are different, their illumination-invariant feature images remain almost the same.

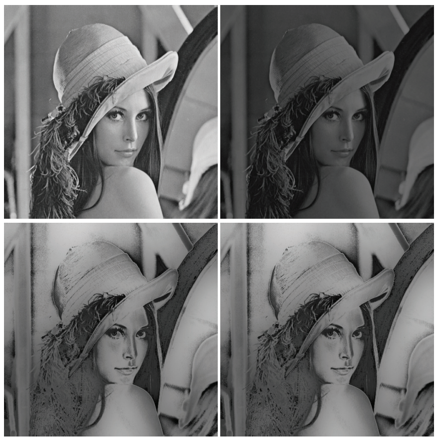

Figure 1. Illumination-invariant features (IIF). First row: input images with different illuminations. Second row: the corresponding illumination invariant feature images.

Now, we may consider taking the illumination invariant feature images for background subtraction. Unfortunately, the experimental results fall short of satisfaction (see Figure 2). On one hand, the illumination-invariant features are too sensitive to image noises, while the background subtraction method must be robust enough to deal with the noises and dynamic changes in the background. On the other hand, background substraction is always considered as a preprocessing step in many computer vision applications. Computational efficiency is very important; obtaining getting the illumination- invariant feature images is a time-consuming process. In the next subsection, we will propose a texture feature named the Local Similarity Statistical Descriptor (LSSD) which derives from an illumination-invariant feature (IIF) and demonstrates the efficiency of the LSSD feature.

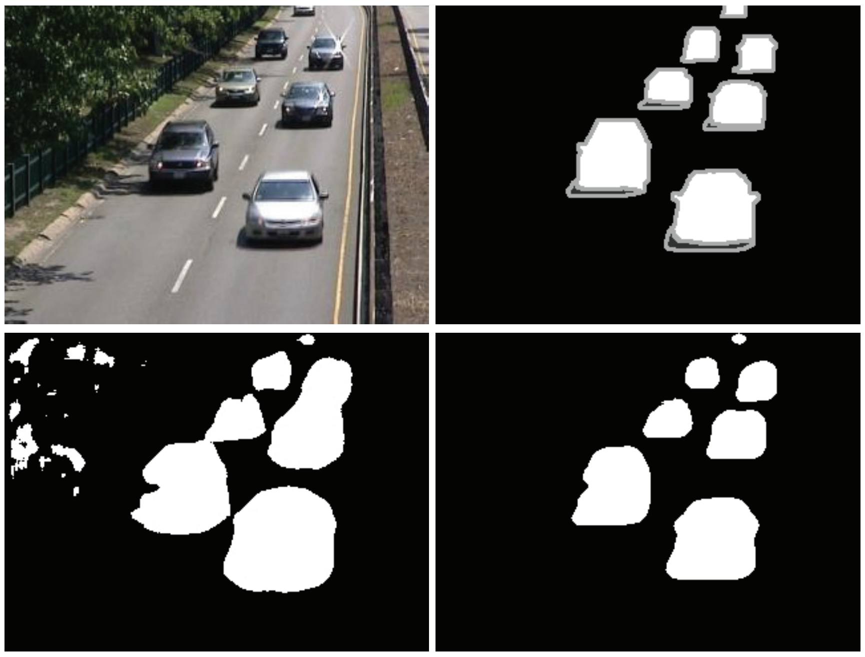

Figure 2. Foreground detection results with IIF and Local Similarity Statistical Descriptor (LSSD) on the sequences of highway. First row: input image and the corresponding ground-truth. Second row: foreground detection results with the original illumination invariant feature image and the segmentation results of the proposed method.

2.2. Local Similarity Statistical Descriptor

Similar to the illumination invariant feature proposed in Equation (4), the Local Similarity Statistical Descriptor (LSSD) is a texture feature which calculates the number of pixels in the neighborhood window E with intensity values fall in a similarity interval with the center pixel. Let a pixel p be in a certain location of image $\boldsymbol{I}$, the coordinate of the pixel is (x_p, y_p), and the size of the neighborhood window E is measured in c columns and r rows. The LSSD operator applied to $p(x_p, y_p)$ can be expressed as:

$$LSSD_{c,r}(p) = \sum_{q \in E_{c,r}} S(\boldsymbol{I}_q, \boldsymbol{I}_p) \tag{7}$$

where $\boldsymbol{I}_p$ is the intensity value of center pixel p, $\boldsymbol{I}_q$ is the intensity value of the pixel q which is in the neighborhood window E, and S is a thresholding function which is defined as:

$$S(\boldsymbol{I}_q, \boldsymbol{I}_p) = \begin{cases} 1, & \text{if } (1-\tau)\boldsymbol{I}_p \leq \boldsymbol{I}_q \leq (1+\tau)\boldsymbol{I}_p; \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where τ is a interval factor affecting the similarity.

An LSSD encoding example is presented in Figure 3. This operator is calculated on a neighborhood window of size 5×5 and the interval factor τ is set to 0.14. The parameter values set here are for demonstration purposes only, and are not the final values used in our experiment.

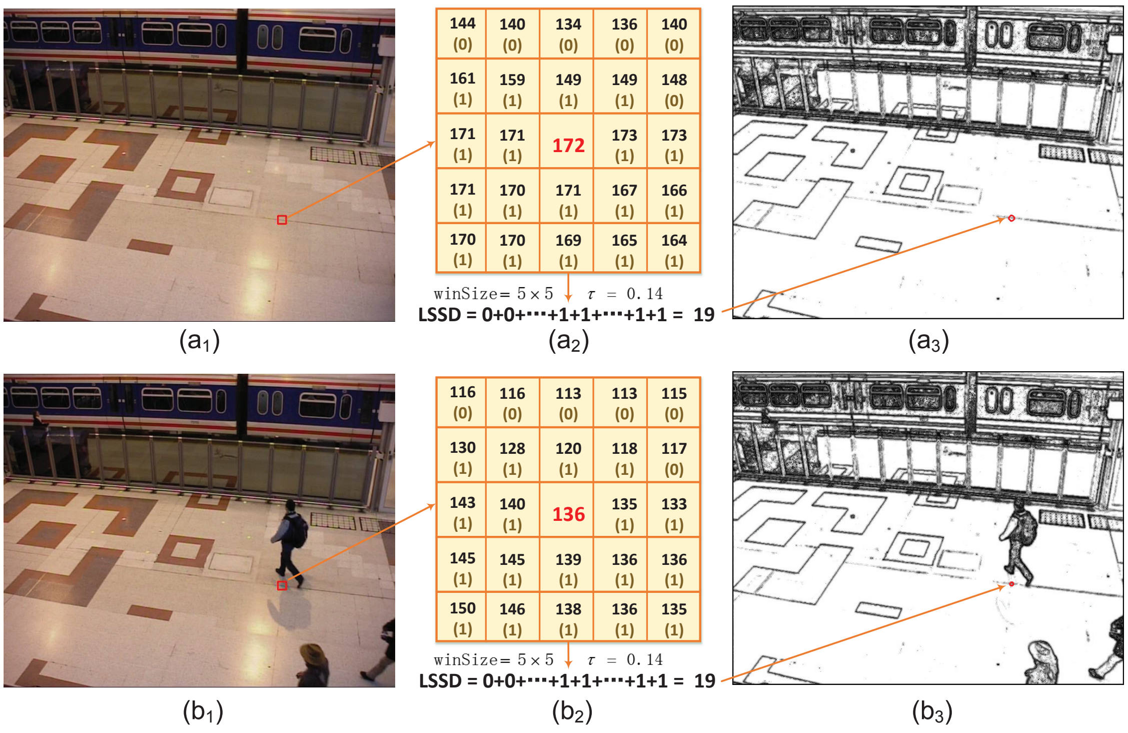

Figure 3. An LSSD encoding example under illumination changes. First column: input images. Second column: the intensity values of the small red box in the input images. Third column: the corresponding LSSD feature images.

As shown in Figure 3, (a_1) and (b_1) are the first and 56th frames selected from the PETS2006 sequence, and (a_3) and (b_3) are the corresponding LSSD feature images (for display purposes, the feature values are rescaled to [0, 255]). In order to simplify the calculation process and make it easier to understand, we convert the color images to gray images and choose a special pixel p with the coordinate of (512, 391) to demonstrate how to get its LSSD feature. First, a image patch with the size of 5×5 centered at p is cropped out from the input image (the small red box in (a_1) and (b_1)); the patch intensity values are shown in (a_2) and (b_2). Then, all of the neighborhood pixels are compared with the central pixel p. If the intensity difference falls within a similarity interval $[-\tau \boldsymbol{I}_p, \tau \boldsymbol{I}_p]$, the neighboring pixel is considered similar to the central pixel, (see Equation (8)). As we can see in (a_2), the central pixel value is 172, the similarity threshold $\tau \boldsymbol{I}_p$ equals 24, and finally we get the LSSD feature of 19, which is the value of $p(512, 391)$ in (a_3). In the same way, the LSSD feature value of (b_2) is obtained as 19.

One of the most important properties of the LSSD feature is its tolerance against illumination variations. As we can see, in most cases when the illumination changes, the pixel values in a localized region decrease or increase proportionally. thus, there will be little change in the difference between the central pixel and its neighboring pixels. In Figure 3(b_1), the pixel values near p are decreased by the shadow of the moving person; the intensity value of p dropped from 172 to 136, it is a drastic changes. However, due to the robustness of LSSD against illumination changes, the feature values have no changes.

One of the other most important properties of the LSSD feature is its flexibility in dealing with dynamic backgrounds, as shown in Figure 4. Two frames contain moving tree branches are selected from the fall sequence, (a$_1$) is the first frame and (b$_1$) is the thirteenth frame; (a$_2$) and (b$_2$) show the intensity values of $p(212, 69)$ and its neighboring pixels. Comparing (a$_2$) and (b$_2$), we can find that the pixel values near p change dynamically due to the swinging leaves, thus, the foreground detection based on pixel values will make a false segmentation. However, as the LSSD feature describes local similarity, its value keeps unchanged during this process, which made it more suitable for coping with dynamically changing scenes.

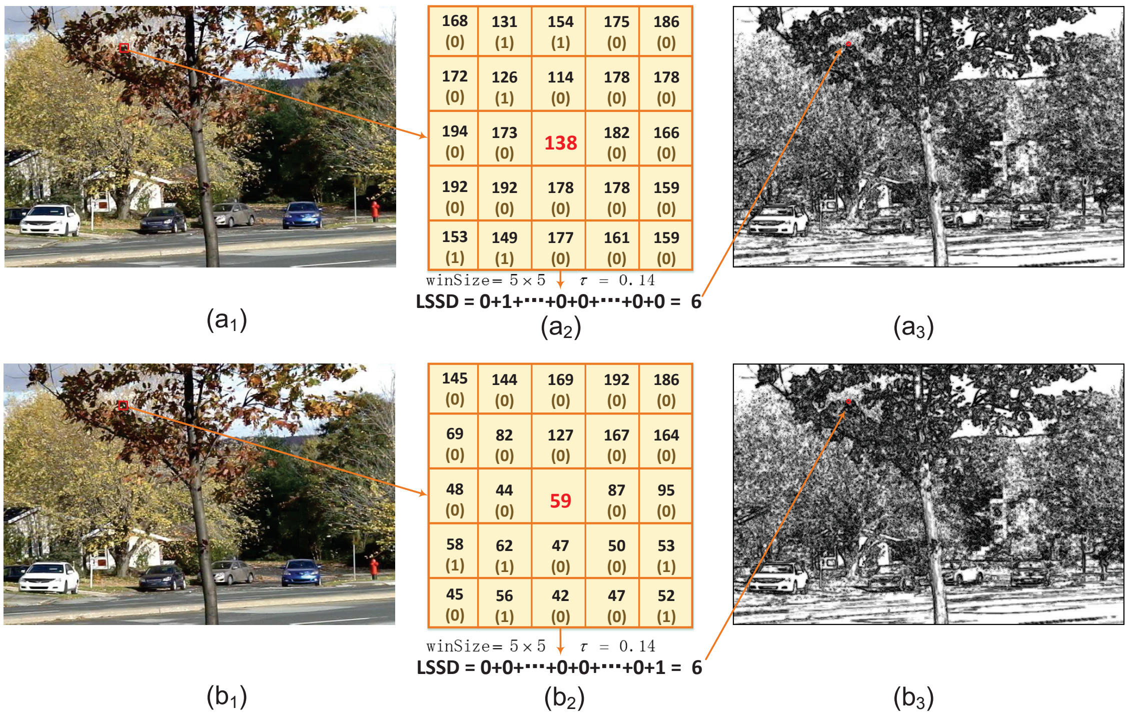

Figure 4. An LSSD encoding example under dynamic backgrounds. First column: input images. Second column: the intensity values of the small red box in the input images. Third column: the corresponding LSSD feature images.

Compared with other texture features like LBP [21], SILTP [22] and LBSP [26], the proposed texture feature LSSD not only has the merit of dealing with illumination variations, but it also has strong robustness to complex background motions. For example, if we concatenate all the compared results of Equation (8) into a binary string like other texture features do, then the new LSSD feature of Figure 3(a$_2$) can be represented as: 00000 11110 11111 11111 11111 and Figure 3(b$_2$) can be represented as: 00000 11110 11111 11111 11111. Taking the bitwise XOR operation to get the Hamming distance, we can see that the result is the same with the LSSD distance. However, in the case of dynamic background motions shown in Figure 4, the new LSSD feature of (a$_2$) can be represented as: 01100 01000 00100 00000 11000, and (b$_2$) can be represented as: 00000 00000 00100 11001 01001. The Hamming distance is 8, while the LSSD distance is 0, which is what we expect. In Figure 5, we also show how the color and LSSD feature values change between consecutive frames. Firstly, in the top subgraph, we give the feature variation plot of the pixel $p(512, 391)$ from the first frame to the 300th frame of the PETS2006 sequence with illumination changes. Then, in the bottom subgraph, we give the feature variation plot of the pixel $p(212, 69)$ from the first frame to the 300th frame of the fall sequence with dynamic background motions. The blue line represents the color feature variation and the red line represents the LSSD feature variation. We can see that the variation of the LSSD feature is much smaller, which indicates the robustness of the LSSD feature against different challenges. In Section 4.4, we will give a more detailed comparison between LSSD and other texture features.

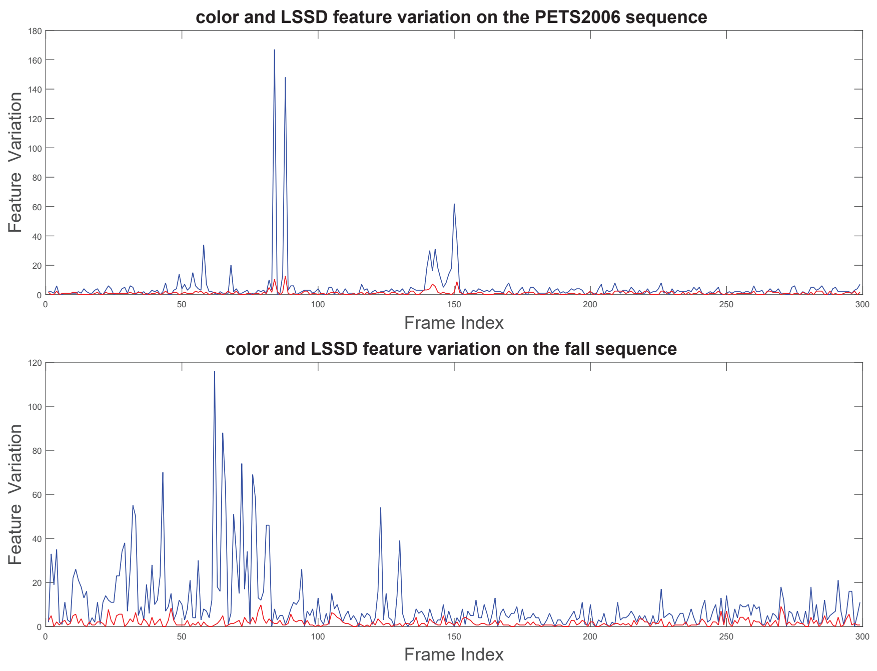

Figure 5. Color and LSSD (Local Similarity Statistical Descriptor) feature values variation of a pixel (the center of the small red box in Figures 3 and 4) from the first frame to the 300th frame of the PETS2006 and fall sequence. The blue line represents the color feature and the red line represents the LSSD feature.

3. Background Modeling

In this section, we will give a detailed description of the framework for the proposed background subtraction method, including background model representation, background model initialization, foreground detection, and a background model update. Figure 6 provides a flow chart that captures the entire proposed method.

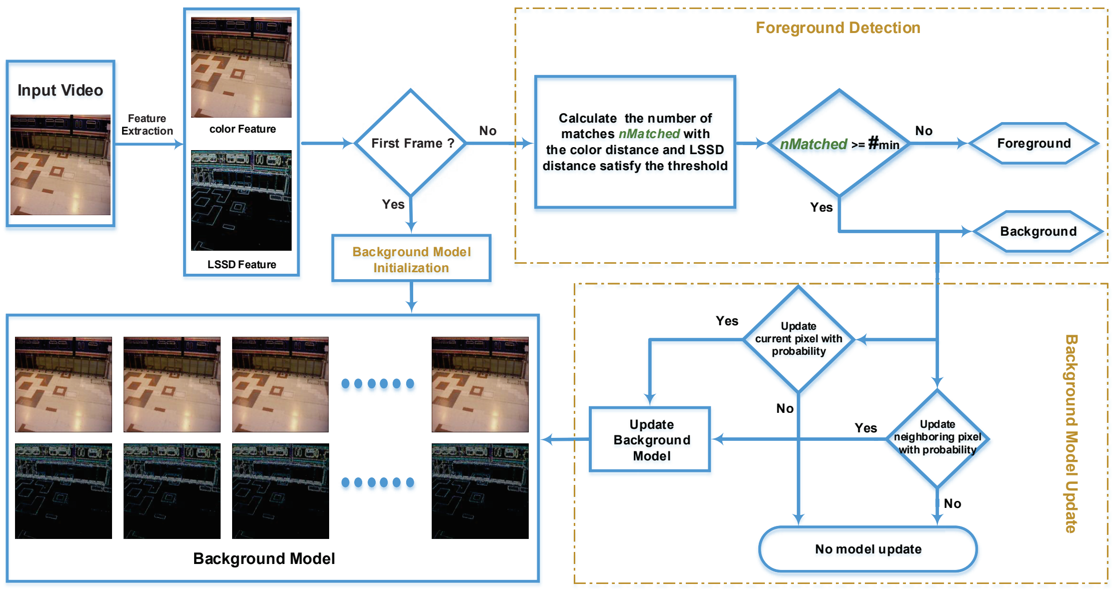

Figure 6. Flow chart of the proposed background subtraction method.

3.1. Background Model Representation

Most of the background subtraction methods rely on probability density functions [4,6] or statistical parameters [7,29]. However, these assumptions inevitably introduce a bias in the real-world scenarios. In ViBe [20], the authors proposed the idea that the observed pixel samples in history would have a higher probability of appearing again. Relying on the collection and maintenance of background model samples with a random approach, a sample consensus background modeling method was proposed and has shown excellent performance in background subtraction.

ViBe is a pixel-based flexible and lightweight background subtraction method. Each pixel $p(x,y)$ in the background is modeled by a set of N recent background samples:

$$\mathcal{B}(x,y) = \{v_1, v_2, \ldots, v_N\} \tag{9}$$

where v_i is the pixel color of the ith background sample. To classify an input pixel $p^t(x,y)$ at time t into foreground or background, it will be compared with the corresponding background model $\mathcal{B}(x,y)$. Denoting the distance between the input pixel $p^t(x,y)$ and background sample v_i as $dist_i$, a match is defined as:

$$\{v_i | dist_i < R, i \in [1, N]\} \tag{10}$$

where R is a fixed maximum distance threshold. If the number of matches is larger than or equal to a given threshold $\#_{min}$, then the input pixel $p^t(x,y)$ is classified as background, otherwise it will be considered as foreground.

The framework of the method presented in this paper is based on ViBe. However, the original algorithm only takes the color feature into consideration; in our background model representation, we integrate the LSSD features and color features to characterize sample representations. That is to say, each background sample v_i in Equation (9) consists of color and texture features. For a pixel $p(x,y)$, its background samples in our method is modeled as:

$$\mathcal{B}(x,y) = \{F_1(x,y), F_2(x,y), \cdots, F_N(x,y)\} \tag{11}$$

where each background sample $F_i(x,y)$ contains the color feature and LSSD feature, $F_i(x,y) = \{I_i(x,y), LSSD_i(x,y)\}$.

3.2. Background Model Initialization

Many popular background subtraction methods need a sequence of frames to initialize the background model [12,30]. However, in some application scenarios, we hope to segment the foreground in few initialization sequences or even from the second frame on. Furthermore, many applications require the algorithm to refresh or re-initialize the background model in the presence of sudden scene changes. Hence, in this paper, we use a single frame (the first frame) to initialize the background model.

Under the assumption that neighboring pixels share a similar temporal distribution at a given time, the background model of pixel $p(x,y)$ is initialized by randomly taking the sample features from its neighborhood pixels for N times as follows:

$$\mathcal{B}(x,y) = \{F(\bar{x},\bar{y}) | (\bar{x},\bar{y}) \in \mathcal{N}(x,y)\} \tag{12}$$

where $\mathcal{N}(x,y)$ is the neighboring pixel of $p(x,y)$ and the probability of chosing $(\bar{x},\bar{y})$ follows a 2D Gaussian distribution. In our experiments, a 7×7 neighborhood region has been demonstrated to be a good choice.

The number of background samples per pixel was recommended to have a value of N = 20 in [20]. In fact, N is used to balance the precision and sensitively of the model. A larger N value leads to great precision but lower sensitivity, and vice-versa. Due to the larger representation space induced by multiple features, we consider increasing the number of background samples and determine the value of N based on the experiment results performed on the CDnet2012 dataset [28]. As can be seen in Figure 9 (the first subgraph), in some categories, like baseline, camera jitter and shadow, their F-Measure scores tend to be saturated when N reaches the value of 65. Meanwhile, in some other categories, like thermal and intermittent object motion, with the value of N increased continuously, the F-Measure score decreases. We can thus find that the value of N depends on the complexity of the scenarios. In this paper, we set $N = 45$ for all categories. Although the overall F-Measure score tends to reach a maximum when N reaches the value of 65, larger N values increase memory and computational complexity, while there is little performance improvement.

As shown in Figure 6 , the background model is initialized as follows. First, the color feature map I and the LSSD feature map $LSSD$ are extracted from the first input frame. Then, for each pixel $p(x, y)$ in the input frame, we randomly select a position $p(\bar{x}, \bar{y})$ in its 7×7 neighborhood region and get the color feature $I(\bar{x}, \bar{y})$ from its color feature map I and the LSSD feature $LSSD(\bar{x}, \bar{y})$ from its LSSD feature map $LSSD$. Finally, the combination of the color feature and the LSSD feature $\{I(\bar{x}, \bar{y}), LSSD(\bar{x}, \bar{y})\}$ becomes a background sample of the pixel $p(x, y)$. Repeating this process N times, the background model of $p(x, y)$ is established. After all the pixels have been traversed, we obtain the final initialized background model. To make it easier to understand, the left bottom of Figure 6 which contains N color feature maps and N LSSD feature maps represents the initialized background model. The color feature and LSSD feature from each column represents a background sample of the corresponding pixel.

3.3. Foreground Detection

The ViBe algorithm relies on the collection and maintenance of history background samples with a random approach, and determine whether the input pixels fit its background model by counting the number of matches within an intersection threshold. Since our background model integrates multiple features, we proposed a few tweaks to the original algorithm to globally improve our results.

Denoting the input frame at time t as $\boldsymbol{I}^t$, to classify a pixel $p^t(x, y)$ as foreground or background, we will first calculate the number of matches between the input pixel and its background sample model $\mathcal{B}(x, y)$. This procedure can be formulated as follows:

$$M(x, y) = \#\{i | dist(F^t(x, y), F_i(x, y)) < R, i \in [1, N]\} \tag{13}$$

where $M(x, y)$ is the number of matches, $F^t(x, y)$ is the input pixel feature, $F_i(x, y)$ is the background sample feature, $dist(F^t(x, y), F_i(x, y))$ obtains the distance between the input sample and the given background model sample, and R is a fixed maximum distance threshold. However, as we know, the input pixel $p^t(x, y)$ contains two features: the color feature and the LSSD feature $F^t(x, y) = \{I^t(x, y), LSSD^t(x, y)\}$. Hence, we should calculate the distances in two different ways.

First, to calculate the similarity between two color features, L1 or L2 distance are the most commonly used metrics due to their simplicity and efficiency [20,31]. However, based on our experimental results, L2 distance is not only an expensive operation, but is also no better than a simpler L1 distance. Hence, we decided to use the L1 distance to calculate the similarity between color features. If the color distance satisfies:

$$dist(I^t(x, y), I_i(x, y)) < R_c \tag{14}$$

then a color feature match is found. The color threshold R_c controls the robustness of the background model. A small R_c leads to sensitive foreground detection result, while a larger R_c has better resistance against relevant change, but makes it more difficult to detect foreground objects that are very similar to the background. We also determine the value of R_c based on the experiment results performed on the CDnet2012 dataset [28]. As it can be seen in Figure 9 (the second subgraph), the overall F-Measure score tends to arrive at a maximum when R_c sets the value of 15.

Second, to calculate the similarity between two LSSD features, we use a similar strategy with the color features. As we defined in Equation (7), we know that the LSSD feature values distribute in the range of $[0, rc]$. rc is the number of pixels contained in neighborhood window E, so we define the texture threshold as: $R_t = \delta rc$, where $\delta \in (0, 1)$. If the texture distance satisfies:

$$dist(LSSD^t(x,y), LSSD_i(x,y)) < R_t \tag{15}$$

then a LSSD feature match is found. According to the experiment results shown in Figure 9 (third and fourth subgraph), we can see that $\delta = 0.2$ and the window size $r \times c$ set to be 5×5 can achieve optimal performance.

In order to integrate the color feature and LSSD feature into the consideration, in the match-calculating procedure, we first calculate the color similarity. If the color distance is less than the pre-defined threshold R_c, then LSSD similarity is calculated. That is to say, to find a match through Equation (13), both the color and LSSD feature must be successfully matched. Figure 7 displays the matching process.

After obtaining the number of matches, we take the label of pixel $p^t(x,y)$ as follows:

$$S^t(x,y) = \begin{cases} 1, & \text{if } M(x,y) < \#_{min} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

where 1 means foreground and 0 means background. $\#_{min}$ is the minimum number of matches required for pixel classification. If the number of matches is larger than or equal to $\#_{min}$, the current pixel is classified as background, and vice-versa. In this paper, we set $\#_{min}$ = 2 to get a reasonable trade-off between computational complexity and noise resistance.

The pseudocode of foreground detection procedure is shown in Algorithm 1:

Algorithm 1: Foreground Detection

```
Input:   current input pixel $p^t(x,y)$
Output:  the FG/BG label of $p^t(x,y)$
 1: $nMatches = 0, i = 0$
 2: while $nMathes < \#_{min}$ && $i < N$
 3:     $colorDist = dist(I^t(x,y), I_i(x,y))$
 4:     if $colorDist \geq R_c$
 5:         goto failedMatch;
 6:     $lssdDist = dist(LSSD^t(x,y), LSSD_i(x,y))$
 7:     if $lssdDist \geq R_t$
 8:         goto failedMatch;
 9:     $nMatches$++;
10:     failedMatch:
11:         $i$++;
12: if $nMatches < \#_{min}$
13:     $p^t(x,y)$ is foreground;
14: else
15:     $p^t(x,y)$ is background;
```

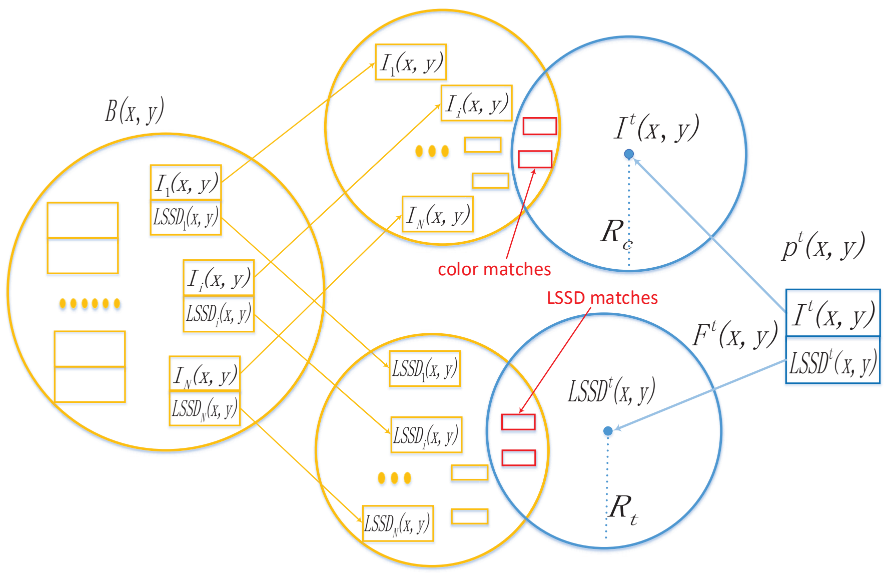

Figure 7. Calculation of the number of matches between the input pixel and its background model. To find a match, both the color feature and LSSD texture feature must be successfully matched.

3.4. Background Model Update

Many background model update strategies have been summarized in [2]. Most of them use the first-in-first-out (FIFO) strategy to update their models. However, there is no evidence to show that this is optimal. In this paper, a conservative, stochastic update strategy is adopted. It contains two steps:

First, if the input pixel $p^t(x,y)$ is classified as background, whether it will be used to update its background model $\mathcal{B}(x,y)$ is determined by a random probability $1/\phi$. We call ϕ a time subsampling factor which controls the adaptation speed of the background model. A small value of ϕ leads to high update probability and makes a rapid evolution background model, and vice-versa. If $p^t(x,y)$ is determined to update its background model, a background sample feature $F_i(x,y)$ randomly picked from its background model $\mathcal{B}(x,y)$ will be replaced by $F^t(x,y)$.

Second, if the input pixel $p^t(x,y)$ is classified as background, it also has the same probability $(1/\phi)$ of updating one of its neighborhood background models $\mathcal{B}(\bar{x},\bar{y})$, where $(\bar{x},\bar{y})$ the position chosen randomly from its neighborhood $\mathcal{N}(x,y)$. Then, a randomly selected background model sample in $\mathcal{B}(\bar{x},\bar{y})$ will be replaced by $F^t(x,y)$.

The pseudocode of the update procedure is shown in Algorithm 2:

Algorithm 2: Background Model Update

```
Input:  the FG/BG label of pixel $p^t(x,y)$
1:  if $p^t(x,y)$ is background
2:      if $rand() \% \phi == 0$
3:          update $\mathcal{B}(x,y)$ with $F^t(x,y)$;
4:      if $rand() \% \phi == 0$
5:          update $\mathcal{B}(\bar{x},\bar{y})$ with $F^t(x,y)$;
6:  else
7:      return;
```

In the first step, the background samples are replaced randomly instead of replacing the oldest one, guaranteeing a smooth, exponentially decaying lifespan for the background samples. This update strategy cancels the time window concept and new samples can be incorporated into the background model only if they are classified as background, thus prevent static foreground objects from being absorbed into the background model too fast. However, this conservative updating strategy may cause a "ghosting" effect, which is the result of falsely classified pixel regions caused by the removal of scene objects, like static objects suddenly starting to move away. A popular method of dealing with this situation is through the "detection support map [32]" which saves the number of times that a pixel has been consecutively classified as foreground. If the value exceeds a given threshold, then the pixel is classified into the background model, however, this strategy will add parameters and increase the computational complexity.

Fortunately, the second step in our model update procedure allows "ghosting" regions to be automatic absorbed into the background model as time goes by. As neighboring pixels share similar spatial distribution, according to the neighborhood diffusion update strategy, background models hidden by the removed object will be updated with neighboring pixel samples from time to time. Moreover, the neighborhood diffusion step also enhances the spatial coherence and prevents the spread of background samples across boundaries. Even if a input sample is wrongfully diffused from one background model to another, the odds that it might be matched are much lower due to the use of the LSSD texture feature.

4. Experimental Results

4.1. Evaluation Datasets

To evaluate the performance of our method and compare it with other state-of-the-art methods, a standard, publicly available dataset, CDnet2012, is considered [28]. This dataset consists of 31 videos from realistic scenarios with nearly 90,000 frames. These videos are grouped into six categories, namely: baseline, camera jitter, dynamic background, intermittent object motion, shadow, and thermal. Accurate human constructed ground-truths are available for all sequences, so exhaustive competitive comparison is possible with different methods. Figure 8 shows some sample images and their corresponding ground-truths. To our knowledge, this is one of the most complete datasets for background subtraction; a complete overview of this dataset is depicted in Table 1.

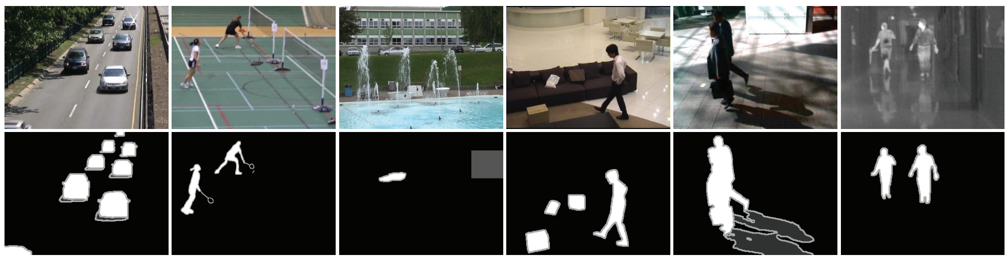

Figure 8. The CDnet2012 dataset [28]: The first row shows an original image from each category and the second row shows its corresponding ground-truth. From left to right: baseline, camera jitter, dynamic background, intermittent object motion, shadow, and thermal. Reproduced with permission from [28], Copyright Mitsubishi Electric Research Laboratories, Inc., 2012.

Table 1. Overview of the CDnet2012 dataset.

Category	Videos	Total Pictures	Evaluation Pictures
baseline	4	6049	4413
cameraJ	4	6420	3134
dynamic	6	18,871	13,276
intermittent	6	18,650	12,111
shadow	6	16,949	14,105
thermal	5	21,100	18,055
Total	31	88,039	65,094

4.2. Evaluation Metrics

In order to compare the methods, a total of seven different metrics have been defined to evaluate different quality characteristics. Let TP stand for the true positives which hold the number of pixels correctly labeled as foreground, TN stand for the true negatives which hold the number of pixels correctly labeled as background, FP stand for the false positives which hold the number of pixels incorrectly labeled as foreground, and FN stand for the false negatives which hold the number of pixels incorrectly labeled as background. According to [28], these metrics are defined as follows:

- Recall (Re) = $\frac{TP}{TP+FN}$
- Specificity (Sp) = $\frac{TN}{TN+FP}$
- False positive rate (FPR) = $\frac{FP}{FP+TN}$
- False negative rate (FNR) = $\frac{FN}{TP+FN}$
- Percentage of wrong classifications (PWC) = 100 · $\frac{FN+FP}{TP+FN+FP+TN}$
- Precision (Pr) = $\frac{TP}{TP+FP}$
- F-Measure (FM) = 2 · $\frac{Re \cdot Pr}{Re+Pr}$

The sums of all pixels in each category are used to calculate these metrics, and an overall category is defined based on the mean of each category. For PWC, FNR and FPR metrics, lower values indicate higher accuracy, but for Re, Sp, Pr and FM, higher values indicate better performance.

During these metrics, we are especially interested in the F-Measure score, which is the most common metric used for background subtraction methods comparison in the literature. As the F-Measure metric is calculated by a combination of multiple evaluation metrics, the overall performance of a background subtraction method is highly correlated with its F-Measure performance. Most state-of-the-art background subtraction methods typically exhibit higher F-Measure scores than worse-performing background subtraction methods [28].

4.3. Parameters Setting

Our method consists of a few parameters which can be adjusted for optimal performance. Since we evaluated our algorithm on the CDnet2012 dataset [28], we used a universal parameter set for all videos to respect the competition rules. Nevertheless, for some applications, parameters can be fine-tuned for some specific needs. Overall, the six parameters, detailed below, were tuned in the dataset. The performance with different parameter settings is shown in Figure 9.

- $N = 45$: the number of samples stored in the background model for each pixel.
- $\#_{min} = 2$: minimum number of sample matches to label an input pixel as background. An optimum is found at $\#_{min} = 2$.
- $R_c = 15$: color distance threshold to determine whether an input pixel matches the background sample.
- $(r, c) = (5 \times 5)$: neighborhood window size to calculate LSSD features in Equation (7).
- $\tau = 0.18$: interval factor used to calculate LSSD features in Equation (8).
- $\delta = 0.20$: parameter factor used to calculate the LSSD distance threshold R_t in Equation (15).

In our proposed method, the classification decision is made independently for each pixel. The foreground detection result can benefit from a regularization step, which combines information from neighboring pixels and assigns homogeneous labels on uniform regions. In preliminary experiments, simple median filtering provided superior results to morphological operations. Thus, we decided to use a median filter for post-processing. In this paper, we use a uniform 7×7 median filter for all evaluated methods. In practice, the input image is a three-channel color image; we process each channel independently and run them in three parallel threads. The final segmentation result is the bitwise OR operation of the three segmentation results from RGB channels.

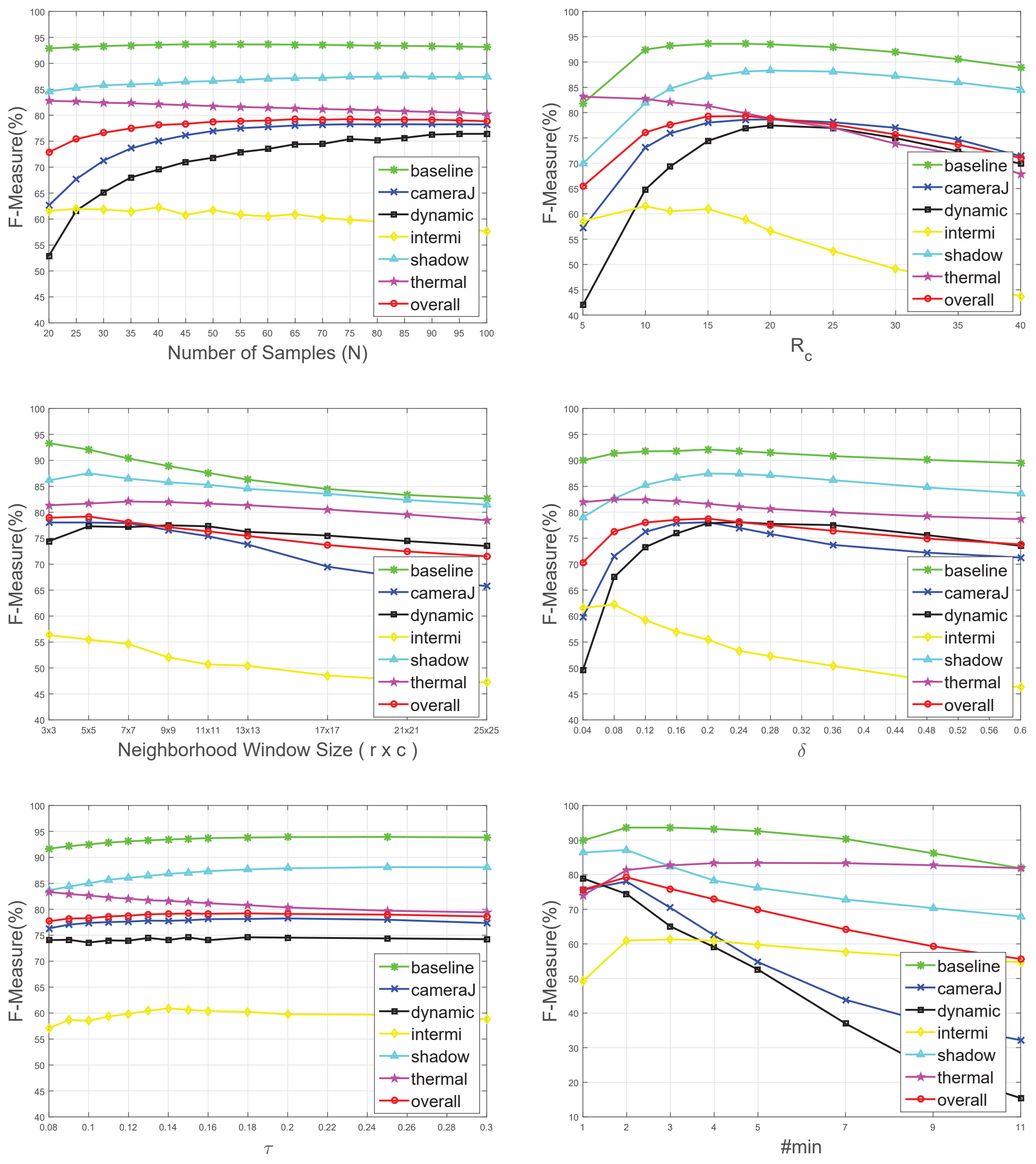

Figure 9. F-Measure of our method on each category as well as overall performance with changing parameter settings. (CDnet2012 dataset).

4.4. Performance Evaluation

Firstly, to demonstrate our key contribution, the LSSD texture feature is shown to be preferable to other texture features (LBP [21], SILTP [22], and LBSP [26]). In Table 2 , we present the performance comparison of different features with respect to the the CDnet2012 dataset. Here, we can see that the LSSD feature obtains a much higher F-Measure score than LBP and SILTP. This may explain that although LBP and SILTP can detect texture variation easily, they are too sensitive to the noise and dynamic background in some scenarios, resulting in poor overall performance. We also see that the LBSP feature obtains much higher F-Measure score than the other texture features. This is the benefit obtained from combining temporal and spatial information to describe texture changes. Others only take spatial information to calculate the features. Although LBSP obtains the highest F-Measure score when only taking the texture features, we will demonstrate that when integrated with the color features, our method performs much better than all others.

Table 2. Average performance comparison of different features on the CDnet2012 dataset. LBP: Local Binary Pattern; SILTP: Scale-Invariant Local Ternary Pattern; LSSD: Local Similarity Statistical Descriptor; LBSP: Local Binary Similarity Pattern.

Features	Recall	Precision	F-Measure
LBP	0.5462	0.3179	0.3748
SILTP	0.6140	0.4121	0.4232
LSSD	0.6673	0.5224	0.5545
LBSP	0.6262	0.8579	0.6841
color	0.6681	0.8426	0.7074
color + LBP	0.7820	0.7470	0.7238
color + SILTP	0.8010	0.7521	0.7352
color + LBSP	0.7658	0.8016	0.7481
color + LSSD	0.8609	0.7771	0.7924

Secondly, to demonstrate the effectiveness of the combination of color features and texture features, we select the two typical sequences office and boat. The clothes in the office include flat texture, which is very similar to the texture of the wall; the boat sequence contains the complex background of a rippled water surface. Figure 10 shows the experimental results when only using color features (the second column), only using texture features (the third column) and using color features and texture features (the fourth column). For the office sequences, the color information can well distinguish the person and the wall, but only using the texture features will result in missed detection, resulting in "holes" in objects. For the boat sequences, which contain the movement of water, only using the color features will generate false detection. However, using the LSSD texture features will suppress these false detections. The experimental results in Figure 10 demonstrate that the color features and the LSSD features have their own merits and demerits. They can compensate for each other to obtain a better segmentation result. In Table 2 , we also present the results obtained when combining the color features and the LSSD features. According to the F-Measure scores, we can see that all texture features obtain performance improvement when combined with the color features; our method surpasses the LBSP and achieves the highest improvement. This is due to the robustness of the LSSD feature as compared with other intrinsic noise-sensitive texture features.

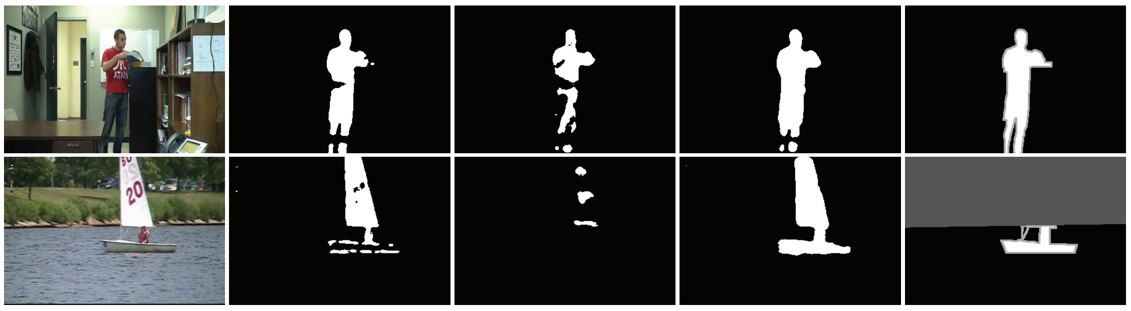

Figure 10. Experimental results with different features on the sequences of office and boat. First column: original images. Second column: only using color features. Third column: only using LSSD features. Fourth column: combined color and LSSD features. Fifth column: the corresponding ground-truth. Gray pixels in the ground truth indicate pixels which are not of interest.

Thirdly, we present the complete results of our method using the evaluation framework of the CDnet2012 dataset [28]. As shown in Table 3, we can see that our method performs well on the dataset with a overall F-Measure of 0.7924. In the baseline and shadow categories, one of the F-Measure scores is 0.9361 and the other is 0.8714. Both of the recall metric scores exceed 0.9. These two categories mainly consist of sequences where cars and pedestrians are the main focus with challenges like illumination variation and camouflage. We also see that the camera jitter and dynamic background categories are well handled by our method; the F-Measure and precision scores are all above 0.7. The same can be said for the thermal sequences, as the precision metric gets a high score of 0.8776. However, we also notice that the intermittent object motion category poses the greatest challenge. This category mainly consists of sequences with abandoned objects and parked cars that suddenly start moving. The main challenge involves static object detection, but it is not what most background subtraction methods are good at.

Table 3. Complete results obtained with the proposed method on the CDnet2012 dataset. FPR: false positive rate; FNR: false negative rate; PWC: percentage of wrong classifications

Category	Recall	Specificity	FPR	FNR	PWC	Precision	F-Measure
baseline	0.9512	0.9975	0.0025	0.0488	0.4199	0.9217	0.9361
cameraJ	0.8760	0.9776	0.0224	0.1240	2.6496	0.7303	0.7803
dynamic	0.8905	0.9960	0.0040	0.1095	0.5141	0.7126	0.7442
intermittent	0.7202	0.9242	0.0758	0.2798	8.8154	0.6053	0.6094
shadow	0.9398	0.9903	0.0097	0.0602	1.1778	0.8148	0.8714
thermal	0.7878	0.9945	0.0054	0.2122	1.3138	0.8776	0.8134
Overall	0.8609	0.9800	0.0199	0.1390	2.4817	0.7771	0.7924

Fourthly, in Table 4 we show how our method compared with some of the state-of-the-art methods. Due to a lack of space, we only chose seven classic methods: GMM [4], KDE [7], ViBe [20], SOBS [12], PSR-MRF [14], PBAS [33] and LOBSTER [26]. Among them, GMM, KDE, SOBS, PBAS and ViBe are pixel-level methods. PSR-MRF is region-level method. LOBSTER and our method are hybrid methods. The results of other methods are from the website www.changedetection.net. For a specific metric, if the method obtains a best score on it, the corresponding value is highlighted in bold. From Table 4 we can see that the ViBe has the best precision performance, LOBSTER has the best specificity performance, PBAS has the best performance on FPR and PWC metrics, and our method is the best in three out of seven metrics: recall, FNR and F-Measure. Of note, the F-Measure score is 0.7924, which is much higher than for all other methods. The LOBSTER method is very similar to our method. Both of them are hybrid methods, which combine pixel-level and region-level analyses. The LOBSTER presents a modified spatio-temporal binary texture descriptor derived from the Local Binary Similarity Pattern (LBSP) and results in a dramatic performance increase. However, as we concluded previously, the local binary pattern texture feature is noise-sensitive. In Table 5, we give a detail per-category average F-Measure comparisons between our method and the LOBSTER method. In the camera jitter and dynamic background categories, the F-Measure scores of our method are much higher than for LOBSTER. In particular, in the dynamic background category, our method shows an amazing 31.4% increase. Again, this demonstrates that the LSSD texture feature not only has the advantage of dealing with illumination variation and camouflage problems as most other texture features do, but also overcomes the disadvantage of noise sensitivity, which makes our method more robust in realistic difficult environmental conditions, like complex background motion and camera vibration.

Finally, we present some qualitative comparisons between LOBSTER, ViBe, GMM, and our method with respect to the CDnet2012 dataset. As shown in Figure 11, we chose six sequences from each category. Each row exhibits a comparison among these methods; from top to bottom they are highway (baseline), traffic (camera jitter), fall (dynamic background), sofa (intermittent object motion), copyMachine (shadow) and library (thermal). The first column is the input image from difference sequences, the second column represents the corresponding ground-truth, the third column shows the segmentation results of our method, the fourth column represents the segmentation results of the LOBSTER [26], the fifth column is the segmentation results of the ViBe [20] and the last column shows the output of the GMM [4] segmentation. Several visual conclusions can be obtained from observing these images. In most cases, our LSSD methodachieves better segmentation results than the other alternatives. In the highway sequence of baseline, the foreground segmentation results of our algorithm are perfect, almost the same as with the ground-truth. In the traffic and fall sequences with camera vibration and dynamic background, unlike the other methods, repetitive movements of the background objects are implicitly avoided in our background model, especially in the traffic sequence, where the highway fence is segmented as foreground in the remaining methods. In the fall sequence, few tree leaves are considered as foreground in our method, and thus the robustness of the LSSD texture feature may be of benefit. A camouflage problem, in which there are significant similarities between the background colors and foreground colors causing holes in the segmentation results, is also shown in several sequences, such as sofa and copyMachine. In the sofa sequence, the color of the man's trousers is similar to the color of the sofa. Even humans find it hard to segment them accurately, while our method obtains a concatenate foreground mask. Meanwhile, holes are detected in the results for LOBSTER and ViBe, resulting in the foreground object being divided into several parts. It is also amazing that the GMM achieved blank results. This may be because, as time goes by, the foreground object is fully absorbed into the background model. For the box left on the floor, which should considered as foreground and never be absorbed into the background model, we observed that in LOBSTER, ViBe and GMM, the box eroded as time went on, creating false negatives, while our method maintained a lower absorption rate. In the copyMachine sequence, we can see a similar phenomenon to the sofa sequence. The segmentation results of other methods are not as good as ours. Lastly, in the library sequence from thermal, most of the methods obtain a perfect segmentation result except GMM.

Table 4. Comparison of the results on the CDnet2012 dataset by different methods.

Method	Recall	Specificity	FPR	FNR	PWC	Precision	F-Measure
GMM [4]	0.6964	0.9845	0.0155	0.3036	3.1504	0.7079	0.6596
KDE [7]	0.7442	0.9757	0.0243	0.2558	3.4602	0.6843	0.6719
ViBe [20]	0.6681	0.9869	0.0130	0.3318	2.7265	**0.8426**	0.7074
SOBS [12]	0.8017	0.9831	0.0169	0.1983	2.4081	0.7315	0.7283
PSP-MRF [14]	0.8037	0.9830	0.0170	0.1963	2.3937	0.7512	0.7372
LOBSTER [26]	0.7658	**0.9904**	0.0125	0.2341	2.3409	0.8016	0.7481
PBAS [33]	0.7840	0.9898	**0.0102**	0.2160	**1.7693**	0.8160	0.7532
LSSD	**0.8609**	0.9800	0.0199	**0.1390**	2.4817	0.7771	**0.7924**

Table 5. Per-category average F-Measure comparisons between LOBSTER and LSSD with respect to the CDnet2012 dataset.

Method	Baseline	CameraJ	Dynamic	Intermittent	Shadow	Thermal		Overall	
	FM	FM	FM	FM	FM	FM	Re	Pr	FM
LOBSTER	0.9320	0.7462	0.5664	0.5940	0.8696	0.7803	0.7658	**0.8016**	0.7481
LSSD	**0.9361**	**0.7803**	**0.7442**	**0.6094**	**0.8714**	**0.8134**	**0.8609**	0.7771	**0.7924**

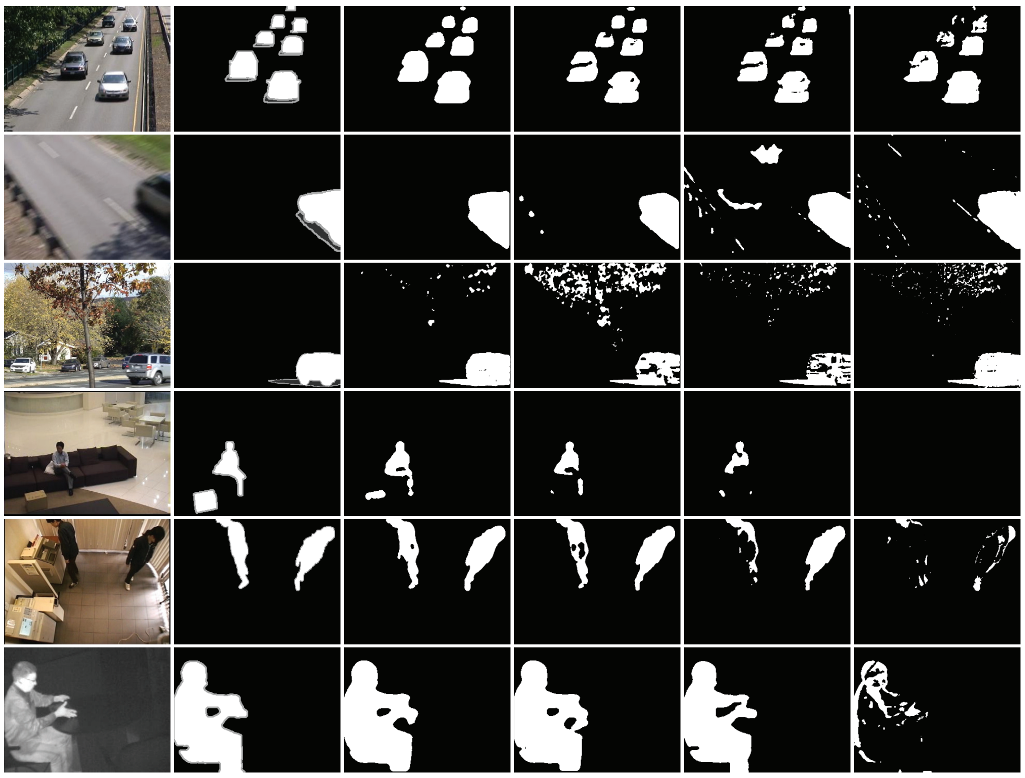

Figure 11. Qualitative performance comparison for various sequences (highway, traffic, fall, sofa, copyMachine and library, from top to bottom), reproduced with permission from [28], Copyright Mitsubishi Electric Research Laboratories, Inc., 2012. The first column is the input image, the second column is the corresponding ground-truth, gray pixels in the ground-truth indicate pixels which are not of interest, the third column shows the segmentation results of our method, the fourth column represents the segmentation results of the LOBSTER [26], the fifth column is the segmentation results of the ViBe [20], and the last column shows the output of the GMM [4] segmentation.

4.5. Processing Speed and Memory Usage

Background subtraction is often the first step in many vision applications. Processing speed and memory usage are critical items of information for researchers to consider before choosing which method to use. Thus, we give a detailed analysis of the time and space complexity of our method in this section.

The computational speed of our method is investigated with different size sequences coming from the CDnet2012 dataset [28]. Our method has been implemented in C++ and uses the OpenCV [34] image processing library. All the experiments are carried out on a 4.2-Ghz Intel Core-i7 7700 K with 32 GB RAM and a Windows 10 operating system. The results are reported in Table 6. Although the combination of color features and texture features will increase the computational complexity, we can see that our algorithm also achieves the real-time performance. Since our method operates at the pixel level, it has the potential for hardware implementation or high-speed parallel implementation.

Table 6. LSSD and ViBe computational speed comparison in terms of frame per seconds (fps).

Sequence	**Frame Size**	**LSSD (fps)**	**ViBe (fps)**
highway	320 × 240	127	323
park	352 × 288	101	273
pedestrians	360 × 240	116	301
badminton	720 × 480	33	76
PETS2006	720 × 576	26	63

As for the memory usage of our method, considering the size of the input image is $W \times H$, the background model sample of each pixel is N, and we find that the space complexity of our method is $\mathcal{O}(NWH)$. For a pixel p, each background sample requires one byte of memory to store the intensity information and one byte of memory to store the LSSD information per channel. According to the parameters set in our experiment described above, each pixel background model contains $N = 45$ background samples. Then, for a color sequence with frame size of 720 × 576 (e.g., PETS2006), the memory requirement of our method would be about 300 MB. This is consistent with the results obtained by the Visual Studio 2015 performance analysis tool. For an embedded platform, decreasing the number of background samples can dramatically reduce the memory usage.

5. Conclusions

In this paper, we first propose a novel texture feature named the Local Similarity Statistical Descriptor (LSSD), which shows good performance in illumination variation and dynamic background scenes. Then, we show that the color features and the texture features have their own merits and demerits. A combination of color features and LSSD features results in a compensation of defects, resulting in a dramatic performance increase. Experiments on the CDnet2012 dataset have shown that in the metric of F-Measure, our method outperforms many recent state-of-the-art background subtraction methods.

A number of improvements can also be considered for our method. In future works, we will integrate our framework with more a complex feedback model update strategy (such as [31,33]). Region-level analyses could also be used to improve the foreground object consistency, and more sophisticated post-processing operations like the Markov Random Field could also help refine our results.

Acknowledgments: This research is supported by the National Natural Science Foundation of China under Grant No.61401425.

Author Contributions: D.Z. and T.Z. performed the experiments and wrote the paper, F.X. helped to perform the experiments, M.Z. and H.Y. helped to wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bouwmans, T. Traditional and recent approaches in background modeling for foreground detection: An overview. *Comput. Sci. Rev.* **2014**, *11*, 31–66.
2. Sobral, A.; Vacavant, A. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Comput. Vis. Image Underst.* **2014**, *122*, 4–21.
3. Wren, C.R.; Azarbayejani, A.; Darrell, T.; Pentland, A.P. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 780–785.
4. Stauffer, C.; Grimson, W.E.L. Adaptive background mixture models for real-time tracking. In Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 23–25 June 1999; IEEE: Piscataway, NJ, USA, 1999; Volume 2, pp. 246–252.
5. Zivkovic, Z.; Van Der Heijden, F. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognit. Lett.* **2006**, *27*, 773–780.

6. Allili, M.S.; Bouguila, N.; Ziou, D. A robust video foreground segmentation by using generalized gaussian mixture modeling. In Proceedings of the 2007 CRV'07 Fourth Canadian Conference on Computer and Robot Vision, Montreal, QC, Canada, 28–30 May 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 503–509.
7. Elgammal, A.; Duraiswami, R.; Harwood, D.; Davis, L.S. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proc. IEEE* **2002**, *90*, 1151–1163.
8. Lee, J.; Park, M. An adaptive background subtraction method based on kernel density estimation. *Sensors* **2012**, *12*, 12279–12300.
9. Kim, K.; Chalidabhongse, T.H.; Harwood, D.; Davis, L. Real-time foreground–background segmentation using codebook model. *Real-Time Imaging* **2005**, *11*, 172–185.
10. Sigari, M.H.; Fathy, M. Real-time background modeling/subtraction using two-layer codebook model. In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, China, 9–21 March 2008; Citeseer: Centre County, PA, USA, 2008; Volume 1.
11. Wu, M.; Peng, X. Spatio-temporal context for codebook-based dynamic background subtraction. *AEU-Int. J. Electron. Commun.* **2010**, *64*, 739–747.
12. Maddalena, L.; Petrosino, A. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Trans. Image Process.* **2008**, *17*, 1168–1177.
13. Maddalena, L.; Petrosino, A. The SOBS algorithm: What are the limits. In Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 21–26.
14. Schick, A.; Bäuml, M.; Stiefelhagen, R. Improving foreground segmentations with probabilistic superpixel markov random fields. In Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 27–31.
15. Abbott, R.G.; Williams, L.R. Multiple target tracking with lazy background subtraction and connected components analysis. *Mach. Vis. Appl.* **2009**, *20*, 93–101.
16. Reddy, V.; Sanderson, C.; Lovell, B.C. A low-complexity algorithm for static background estimation from cluttered image sequences in surveillance contexts. *EURASIP J. Image Video Process.* **2011**, *2011*, 1–14.
17. Hsiao, H.H.; Leou, J.J. Background initialization and foreground segmentation for bootstrapping video sequences. *EURASIP J. Image Video Process.* **2013**, *2013*, 1–19.
18. Tsai, D.M.; Lai, S.C. Independent component analysis-based background subtraction for indoor surveillance. *IEEE Trans. Image Process.* **2009**, *18*, 158–167.
19. Wang, R.; Bunyak, F.; Seetharaman, G.; Palaniappan, K. Static and moving object detection using flux tensor with split gaussian models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 414–418.
20. Barnich, O.; Van Droogenbroeck, M. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Trans. Image Process.* **2011**, *20*, 1709–1724.
21. Heikkila, M.; Pietikainen, M. A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 657–662.
22. Liao, S.; Zhao, G.; Kellokumpu, V.; Pietikäinen, M.; Li, S.Z. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1301–1306.
23. Bilodeau, G.A.; Jodoin, J.P.; Saunier, N. Change detection in feature space using local binary similarity patterns. In Proceedings of the 2013 International Conference on Computer and Robot Vision (CRV), Regina, SK, Canada, 28–31 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 106–112.
24. Yao, J.; Odobez, J.M. Multi-layer background subtraction based on color and texture. In Proceedings of the 2007 CVPR'07 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 1–8.
25. Han, B.; Davis, L.S. Density-based multifeature background subtraction with support vector machine. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1017–1023.
26. St-Charles, P.L.; Bilodeau, G.A. Improving background subtraction using local binary similarity patterns. In Proceedings of the 2014 IEEE Winter Conference on Applications of Computer Vision (WACV), Steamboat Springs, CO, USA, 24–26 March 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 509–515.

27. He, S.; Lau, R.W.; Yang, Q.; Wang, J.; Yang, M.H. Robust object tracking via locality sensitive histograms. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *27*, 1006–1017.
28. Goyette, N.; Jodoin, P.M.; Porikli, F.; Konrad, J.; Ishwar, P. Changedetection. net: A new change detection benchmark dataset. In Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012, pp. 1–8.
29. Sheikh, Y.; Shah, M. Bayesian modeling of dynamic scenes for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1778–1792.
30. Zhou, X.; Liu, X.; Jiang, A.; Yan, B.; Yang, C. Improving Video Segmentation by Fusing Depth Cues and the Visual Background Extractor (ViBe) Algorithm. *Sensors* **2017**, *17*, 1177.
31. St-Charles, P.L.; Bilodeau, G.A.; Bergevin, R. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Trans. Image Process.* **2015**, *24*, 359–373.
32. Haritaoglu, I.; Harwood, D.; Davis, L.S. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 809–830.
33. Hofmann, M.; Tiefenbacher, P.; Rigoll, G. Background segmentation with feedback: The pixel-based adaptive segmenter. In Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 38–43.
34. Bradski, G. Newly changed information, please confirm. The opencv library. *Dr. Dobbs J.* **2000**, *25*, 120–126.