Edge based Foreground Background Estimation with Interior/Exterior Classification

Gianni Allebosch, David Van Hamme, Francis Deboeverie, Peter Veelaert and Wilfried Philips
Image Processing and Interpretation Group (IPI), Department of Telecommunications and Information Processing (TELIN), Ghent University - iMinds, St-Pietersnieuwstraat 41, 9000, Ghent, Belgium

Keywords: Foreground Background, Edge Motion, Local Ternary Patterns, Interior.

Abstract: Foreground background estimation is an essential task in many video analysis applications. Considerable improvements are still possible, especially concerning light condition invariance. In this paper, we propose a novel algorithm which attends to this requirement. We use modified Local Ternary Pattern (LTP) descriptors to find likely strong and stable "foreground gradient" locations. The proposed algorithm then classifies pixels as interior or exterior, using a shortest path algorithm, which proves to be robust against contour gaps.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Over the years, many foreground background segmentation methods have been proposed. Several of them are variations on the classical Gaussian Mixture Model (GMM) (Stauffer and Grimson, 1999) concept. It is assumed that the frequency of an appearance level (intensity, RGB ...) occurring at a certain pixel can be modelled statistically as a mixture of Gaussian distributions. The more recent ViBe algorithm (Barnich and Droogenbroeck, 2011) and its successor ViBe+ (Droogenbroeck and Paquot, 2012) are built on similar principles as the GMM, but store the distributions as a collection of samples rather than by the model parameters.

Pixel appearances can change drastically under changing lighting conditions (Cristani et al., 2010). The previously described methods handle light changes poorly. Edge based approaches on the other hand are able to produce stable results under illumination changes (Gruenwedel et al., 2011). However, moving objects are generally contiguous, so additional contour filling strategies are required. Foreground edges are prone to gaps, which classical boundary filling techniques (floodfill, scanline fill ...) are unable to cope with. The robust filling of contours with gaps is still a challenge.

Illumination invariant descriptors provide an alternative solution. In the method of Heikkilä and Pietikäinen (Heikkila and Pietikainen, 2006), Local Binary Patterns (LBPs) are used to construct histograms over a larger region. Different objects usually have different textures, so they give rise to differing LBP histograms. However, objects can have regions with textures similar to the background, which results in incomplete foreground masks. A recent method, coined SuBSENSE (St-Charles et al., 2014) attempts to overcome these issues by using the more advanced Local Binary Similarity Patterns (Bilodeau et al., 2013), additional color information, and a framework for automatic local parameter tuning. Even though these steps significantly improve the overall performance, there are still considerable improvements possible for difficult illumination conditions. For instance, the F-measure of SuBSENSE on the Night Video sequences on the ChangeDetection.NET 2014 dataset is limited to only 53.9%, while still being the highest ranking method to date (Wang et al., 2014).

In this paper, we describe a foreground background segmentation algorithm which attends to the shortcomings described above. We combine the benefits of edge based approaches with the robustness of illumination invariant descriptors. Firstly, we assume that each local image gradient value above noise level is an indication for the possible presence of an edge. Combining these features over a larger area increases the robustness. As we will show in Section 2, this information can be extracted from the Local Ternary Pattern (LTP (Tan and Triggs, 2010)) descriptor.

Secondly, even with increased robustness, gaps can still occur in the contour. We propose a fast ($O(n)$), graph search based method that measures how difficult it is to reach points from the corners of the image, where the edge pixels serve as obstacles.

Allebosch G., Van Hamme D., Deboeverie F., Veelaert P. and Philips W..
Edge based Foreground Background Estimation with Interior/Exterior Classification.
DOI: 10.5220/0005358003690376
In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications* (VISAPP-2015), pages 369-376
ISBN: 978-989-758-091-8
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

Closed contours are considered impossible to enter. A contour with gaps can be entered, but with increasing difficulty when more edge pixels block the way in. Based on this metric, we introduce a novel, shortest path based interior filling mechanism. We will show that these techniques perform especially well under difficult lighting conditions compared to state-of-the-art techniques.

2 LTP BASED GRADIENT DESCRIPTION

Using LTPs, we developed a gradient descriptor, tailored specifically to the needs of robust foreground detection.

2.1 Alternative Local Ternary Patterns Description

Local Ternary Patterns (LTPs) are classes of point descriptors which represent the local variations of appearance levels in a ternary description (Tan and Triggs, 2010). Let us denote a gray scale image I. For each pixel $\mathbf{z}$, a number of surrounding pixels are selected, often symmetrically from a circle around $\mathbf{z}$. For each selected surrounding pixel p_i, the gray level difference d_s with the center pixel $\mathbf{z}$ is coded as

$$d_s[\mathbf{z},i] = \begin{cases} 1 & \text{if } I_s[\mathbf{z},i] - I[\mathbf{z}] > T \\ -1 & \text{if } I_s[\mathbf{z},i] - I[\mathbf{z}] < -T \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

with T a given, typically low threshold and $I_s[\mathbf{z},i]$ the gray scale value of p_i. $d_s[\mathbf{z},i]$ represents whether the selected surrounding pixel p_i has a significantly different intensity value from $\mathbf{z}$. When $d_s[\mathbf{z},i] = 0$, the thresholded difference is considered to be below noise level. An example is given in Figure 1.

We observe that every $d_s[\mathbf{z},i]$ can also be thought of as a binary gradient $\mathbf{g_b}[\mathbf{z},i]$, either with unit or zero length, and with a direction either from $\mathbf{z}$ towards p_i, or from p_i towards $\mathbf{z}$ (Figure 1c). Formally,

$$\mathbf{g_b}[\mathbf{z},i] = d_s[\mathbf{z},i]\mathbf{e_i} \,, \quad (2)$$

where $\mathbf{e_i}$ is the unit vector pointing from $\mathbf{z}$ towards p_i.

If there is an edge at the center pixel $\mathbf{z}$, we expect the individual binary gradients in the LTP descriptor to comply with each other. As a measure for this "compliance", we use the summation $\mathbf{g}[\mathbf{z}]$ of these binary gradients.

$$\mathbf{g}[\mathbf{z}] = \sum_{i=0}^{N-1} \mathbf{g_b}[\mathbf{z},i] \,. \quad (3)$$

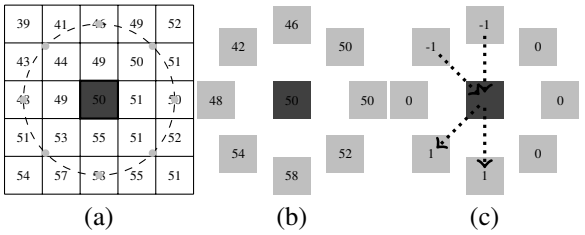


Figure 1: An example of a Local Ternary Pattern with 8 neighborhood points: (a) Intensity levels in a certain image region, with circular neighborhood points denoted. (b) Interpolated neighborhood intensity levels. (c) LTP representation ($T = 3$), with our binary gradients representation superimposed.

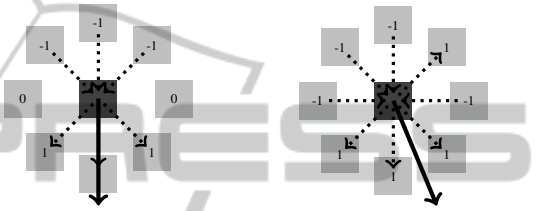


Figure 2: Visualization of the summed binary gradient (black arrow, not to scale). The individual binary gradients are shown as dotted arrows. (a) Example with obvious gradient direction (b) Example with less obvious gradient direction.

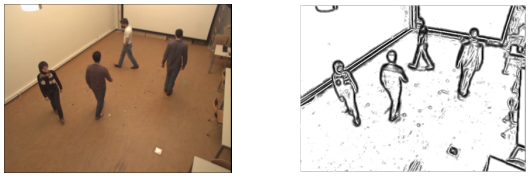
This vector's direction can be regarded as the most likely gradient direction. Its length is a measure for the confidence of the calculated gradient direction. Note that $\mathbf{g}[\mathbf{z}]$ should thus be considered a gradient descriptor as opposed to the original LTP texture descriptor (Tan and Triggs, 2010). Examples are given in Figure 2.

2.2 Orientation Dependent Edge Smoothing

Edges in real images tend to continue in the same direction (perpendicular to the local gradient) over a certain length. To further improve the results for our LTP based gradients, we apply an adaptive smoothing strategy. Assume the gradients at pixel location $\mathbf{z}$ are stored as a 2D vector, $\mathbf{g}[\mathbf{z}] = (x,y)$. The gradients can thus be represented by two images, which we both filter with rotated, anisotropic 2D Gaussian kernels. The Gaussians have significantly different standard deviations along both dimensions, so smoothing is mainly performed along the chosen orientation. The orientation of the kernels is adjusted locally, depending on the summed gradient direction (Section 2.1).

Our global strategy of thresholding the individual intensity differences first and then combining the evi-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS



(a) (b)

Figure 3: (a) Input image. (b) Gradient confidence levels, where darker means more likely. Note that all clear edges in the image are represented by similarly strong gradient confidences.

(a) (b)

Figure 4: (a) Our foreground gradients. (b) Foreground edges from (Gruenwedel et al., 2011). Note that our method trades a bit of localization accuracy for higher robustness against gaps, due to the low LTP threshold and orientation dependent smoothing.

dence over a larger area, is what makes our descriptor robust in different illumination conditions. In our approach, all significant intensity differences are treated as being equally important. All that matters is that the evidence is supported in the surrounding region. This stabilizes the gradient confidence levels and also diminishes the possibility that single (noisy) pixels sharply disturb the descriptor. A result of the gradient calculation and successive smoothing step of our algorithm is shown in Figure 3.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

3 MOVING CONTOURS

In this section, we discuss the construction of our LTP based temporal model. Our model is closely related to the framework developed by Heikkilä and Pietikäinen (Heikkila and Pietikainen, 2006), but our contribution offers a more advanced updating mechanism that is able to handle a wider range of scenarios. The basic model is explained briefly in Section 3.1, while the novel updating mechanism is explained in Section 3.2.

3.1 Temporal Model

At each pixel location $\mathbf{z}$, a fixed number M of our adaptively smoothed gradient vectors $\mathbf{v}_j$ ($1 \leq j \leq M$) are stored. These vectors represent the temporal model for the chosen pixel. In each frame, the new gradient vector is compared to each vector in the temporal model at that location. Only the vector $\mathbf{v}_k$ ($1 \leq k \leq M$) with the lowest squared difference from the input gradient is updated. Let $\mathbf{v}[\mathbf{z},t]$ be the new LTP based gradient from the input image at that location at time index t. The best matching vector is updated as follows:

$$\mathbf{v}_k[\mathbf{z},t] = \alpha[\mathbf{z},t]\mathbf{v}[\mathbf{z},t] + (1-\alpha[\mathbf{z},t])\mathbf{v}_k[\mathbf{z},t-1] \quad (4)$$

with $\alpha[\mathbf{z},t]$ the learning rate, which will be discussed in depth in Section 3.2.

Pixel locations where the gradient in the current image differs substantially from the most likely background vectors are considered to be part of foreground gradient regions. Therefore, for each vector $\mathbf{v}_j$, a weight factor w_j is kept, which is a measure for the likelihood that the vector represents the background. w_j is updated as follows:

$$w_j[\mathbf{z},t] = \begin{cases} \alpha[\mathbf{z},t] + (1-\alpha[\mathbf{z},t])w_j[\mathbf{z},t-1] & \text{if } j = k. \\ (1-\alpha[\mathbf{z},t])w_j[\mathbf{z},t-1] & \text{otherwise} \end{cases} \quad (5)$$

The temporal vectors are sorted according to their weights. The most weighted vector always represents background. Consecutive vectors are added to the background vector set, until their cumulative weight exceeds a chosen threshold T_w. The input gradient vectors are then compared to the vectors in the background set. If the lowest squared error exceeds another threshold T_e, this pixel is considered to be foreground. Furthermore, if the lowest squared error with all vectors in the temporal model exceeds T_e, the temporal vector with the lowest weight is replaced by the current vector $\mathbf{v}[\mathbf{z},t]$. Results are shown in Figure 4.

3.2 Adaptive Learning Rate

In the simplest cases, the background is entirely visible in the first frame, and remains unchanged throughout the rest of the sequence. However, in many realistic scenario's, one cannot make this assumption. Examples include background object displacements or deformations (e.g. waving trees) and (small) camera displacements. Besides that, it is sometimes difficult to capture an 'empty frame', where there is no foreground to begin with.

One approach to overcome these issues, is to slowly adapt the background model, using a learning rate (Eq. 4 and Eq. 5). In many algorithms with similar background models, this learning rate is kept constant for all pixels and throughout the sequence, and has to be tuned manually in advance for each sequence individually (Stauffer and Grimson, 1999) (Heikkila and Pietikainen, 2006). Some methods

adapt the maintenance mechanism at runtime if a large illumination change is detected (Porikli and Tuzel, 2003), but this is not necessary for our method. A more advanced approach is to raise the learning rate locally for rapidly changing ("blinky") pixels (St-Charles et al., 2014). However, this might create issues for fast moving objects, or similarly on videos captured at a low framerate.

Our framework is built on the following principles:

- A minimal amount of learning is necessary throughout the sequence, in order to adapt to minor changes.
- The learning rate must be higher in the beginning of the sequence. This ensures that foreground objects present in the first few frames get replaced quickly by background vectors in the model once they move.
- The learning rate should increase if the background model becomes unreliable.

So, the learning rate should at least be $\geq \alpha_c$, a very small constant. Furthermore, we impose that part of the learning rate α_e decreases exponentially over time:

$$\alpha_e[t] = \frac{1}{M} e^{-\tau t} , \tag{6}$$

where t is the number of frames since the background model was initialized and τ is a user settable parameter, which determines the rate at which the function decreases. The multiplication by $\frac{1}{M}$ avoids that the weights of the current model drop too drastically, once all temporal vectors have been initialized.

Finally, the background model can become unreliable if there are large changes in the image, not due to foreground objects. We denote a background unreliability rate α_u, which is increased in two different scenario's:

1. The background model should be corrected globally, e.g. when there is a large camera displacement.
2. The background model should be corrected locally, e.g. when a gust of wind deforms a tree.

The first scenario can be identified fairly easily. If there is a large disagreement between the current frame and the model (i.e. there are too many foreground pixels), the camera position has likely shifted or a significant change has occurred in the background. The model should adapt to these changes quickly. For this purpose, we impose a 'global unreliability' term U_G, that is sigmoid-shaped (Figure 5):

$$U_G[t] = \frac{0.5}{M} \left(1 + \text{erf}\left(\frac{R_f[t] - \mu}{\sigma}\right)\right) \tag{7}$$

$$\text{with } \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} \text{d}y . \tag{8}$$

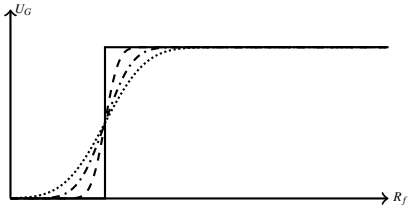
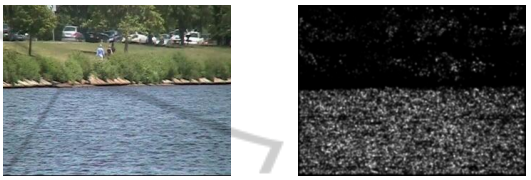


Figure 5: U_G as a function of the foreground (gradient) pixel percentage R_f for different values of σ. The solid line represents the case where $\sigma = 0$, and behaves like a regular (hard) threshold or step function.



(a) (b)

Figure 6: (a) Input frame from the ChangeDetection.NET 2014 dataset with dynamic background below a busy street (Wang et al., 2014). (b) Visualization of U_G for the entire image, lighter means more unreliable. Note that the region corresponding to the water is correctly classified as more unreliable background, while the busy street above has a much smaller unreliability term.

erf is the error function, $R_f[t]$ is the percentage of foreground pixels in frame t and σ and μ are user settable parameters. It behaves like a 'soft threshold' at μ. σ determines how steep the function is. If $\sigma = 0$, U_G behaves like a regular threshold, much like in the method of Wang and Suter (Wang and Suter, 2005). The minimum of $U_G[t]$ is 0 and the maximum is again $\frac{1}{M}$. In this way, the model should be able to handle $M-1$ different camera positions (one remaining for foreground gradient pixels).

The second scenario is less trivial to identify. Simply looking at the amount of foreground pixels is not recommended, since dynamic background can be easily confused with regions that simply contain a lot of foreground (e.g. a busy street). However, dynamic background will often produce many isolated pixels (Figure 6). In our algorithm, this phenomenon is exploited by building a local isolated pixel rate R_i over time:

$$R_i[\mathbf{z},t] = \alpha_i[t] R_{i,curr}[\mathbf{z},t] + (1 - \alpha_i[t]) R_i[\mathbf{z},t-1] , \tag{9}$$

$$\text{where } \alpha_i[t] = C_i + (1 - C_i) M \alpha_e[t] \tag{10}$$

with C_i a small constant. $R_{i,curr}[\mathbf{z},t] = 1$ if $\mathbf{z}$ is the only foreground gradient pixel in his 8-connected neighborhood, otherwise $R_{i,curr}[\mathbf{z},t] = 0$. Note that α_i is higher in the beginning of the sequence, since the accumulated evidence of dynamic background gets more reliable over time. The 'local unreliability' U_L

at $\mathbf{z}$ in frame t is now

$$U_L[\mathbf{z},t] = R_i[\mathbf{z},t] * K_G \ , \tag{11}$$

where K_G is an isotropic Gaussian kernel, used to combine evidence of unreliability in the neighborhood of $\mathbf{z}$.

The background unreliability rate $\alpha_u[\mathbf{z},t]$ is now defined as the maximum of the global and local unreliability:

$$\alpha_u[\mathbf{z},t] = \max[U_G[t], U_L[\mathbf{z},t]] \ . \tag{12}$$

The total learning rate $\alpha[\mathbf{z},t]$, taking into account all of the considerations described above, is

$$\alpha[\mathbf{z},t] = \min\left[\frac{1}{M}, \alpha_c + \alpha_e[t] + \alpha_u[\mathbf{z},t]\right] \ . \tag{13}$$

4 SHORTEST PATH BASED INTERIOR FILLING

In the previous sections, we described our strategy to create a foreground image, consisting of significant gradient differences. Here, we describe how interior points can be filled.

4.1 Interior Points

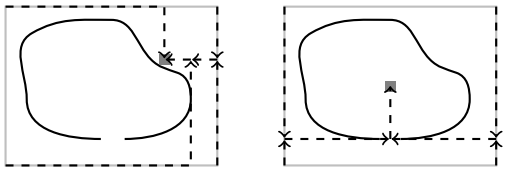
To determine the entire moving object, we must fill its moving contour. Topologically, a point can be classified as interior if it is completely enclosed by contour points. If the foreground gradients would only consist of closed contours, classical contour filling strategies would be sufficient. However, even with the robustness measures described in previous sections, gaps can still occur.

One possible approach to overcome this problem is to group foreground edge pixels, based on connectivity and distance, into curve segments. The convex hull of these segments can then be taken as an approximation of the filled silhouette. This does produce foreground regions which are too large, if the objects themselves are not convex. A more restrictive approach is to use the orthogonal hull (Ottmann et al., 1984). However, even the orthogonal hull is often still too large. The algorithm described in this paper produces a silhouette $S(P)$ that satisfies

$$C(P) \subseteq S(P) \subseteq H_o(P) \subseteq H_c(P) \ , \tag{14}$$

where the set $C(P)$ is obtained by filling the closed contours in the pixel set P (and leaving the other edges intact). $H_o(P)$ and $H_c(P)$ are the orthogonal hull and convex hull of P respectively.

The basic idea is as follows: *The probability of a point being interior, is proportional to how difficult it is to reach that point from the outside, where the strong gradient points act as obstacles.* If this object's contour is closed, every possible "path" to one of its interior points would be blocked by foreground gradient pixels. Alternatively, if there is a single gap in the contour, it would be possible to reach the interior points, but a path to it would likely be longer than in a scenario without edge pixels. An example is given in Figure 7.



(a) (b)

Figure 7: Examples of the path distance calculation from all 4 image corners to a chosen pixel. Black solid lines represent strong edges, dotted lines the individual paths. In (a), the path from 1 image corner is larger than the Manhattan distance. In (b), the paths from 2 image corners are longer.

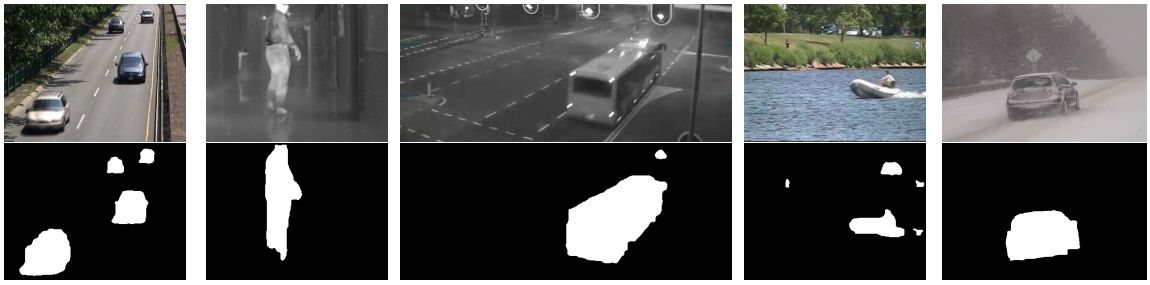
We point out that the foreground gradients consist of contours of foreground objects (external edges), but might also contain other, internal foreground edges. However, since these edges should be part of the entire foreground silhouette as well, this reasoning still holds.

4.2 Path Length Comparison

We consider the foreground gradient image as a 4-connected input graph. Starting from a fully connected graph, vertices coinciding with foreground pixels and their links are removed. The path length D_P from the four image corners $\mathbf{c_i}$ $(1 \leq i \leq 4)$ to each of the image pixels $\mathbf{z}$ is found by using Breadth First Search (BFS) in the graph. We define the "excess" distance D_E as the difference between the actual path length and the minimal length (if there would be no edge pixels). In particular, we have

$$D_E[\mathbf{z},i] = D_P[\mathbf{z},i] - D_M[\mathbf{z},i] \ , \tag{15}$$

where $D_M[\mathbf{z},i]$ is the Manhattan distance from $\mathbf{c_i}$ to $\mathbf{z}$, ignoring edges. If $\mathbf{z}$ cannot be reached, we set $D_E[\mathbf{z},i] = \infty$. It is easy to show that, if the foreground object is orthogonally convex, it is possible to reach any point on the outside of the contour from at least 3 image corners with minimal distance (if there is a free path along the image borders). The distance from the 4th corner often exceeds this minimum substantially. Keeping in mind the subset relations in Eq. 14, we exclude the corner $\mathbf{c_l}$ with the largest excess distance from further analysis in each pixel (Figure 7). Summing over the 3 remaining D_Es, results in the total



(a) (b) (c) (d) (e)

Figure 8: Performance of our proposed method on a few hand picked frames from the ChangeDetection.NET 2014 dataset (Wang et al., 2014). (a) Baseline. (b) Thermal. (c) Night videos. (d) Dynamic background. (e) Bad Weather.

excess distance $D_{E,tot}$ for that pixel position $\mathbf{z}$:

$$D_{E,tot}[\mathbf{z}] = \sum_{i \neq l} D_E[\mathbf{z}, i] \; . \tag{16}$$

If a pixel lies outside of the orthogonal hull of all edge pixels, the excess distance is *always* zero. If a pixel cannot be reached, this distance is infinite. We now classify the interior pixels as foreground by imposing a threshold $T_d \geq 0$. The binary foreground image F is constructed as follows:

$$F[\mathbf{z}] = \begin{cases} 1 & \text{if } D_{E,tot}[\mathbf{z}] > T_d \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

As a final remark, we observe that the resulting foreground blobs are often larger than strictly necessary. This is because the contours themselves were not thinned earlier, to avoid the possible creation of gaps. So, all blobs are thinned using morphological erosion as a final step.

If there is one moving object, and the filled silhouette is constructed as described above, Eq. 14 is always satisfied. Higher thresholds "cut" out more pixels from the silhouette, but never more than contour filling, and never less than the orthogonal hull. If there are multiple objects moving in the scene, Eq. 14 is satisfied for the individual objects as long as their rectangular bounding boxes do not overlap.

The time complexity of BFS in a graph can be expressed as $O(|V| + |E|)$, with $|V|$ the number of vertices and $|E|$ the number of links (edges) (Cormen et al., 2001). So, the time complexity of this filling step can be written as $O(n)$ with n the number of pixels in the input image.

5 RESULTS

We compared our algorithm to the state-of-the-art, through the ChangeDetection.NET 2014 dataset (Wang et al., 2014), which currently ranks 16 recent (e.g. SuBSENSE (St-Charles et al., 2014)) or classical (e.g. GMM (Stauffer and Grimson, 1999) (Zivkovic, 2004)) methods. This dataset contains a total of 53 videos, spread across 11 categories. For all videos, we used the ground truth images and scripts provided by the owners of the website to calculate a total of 7 different measures for each video and category, as well as the overall performance. Note that the same parameter set has to be used for all videos in the dataset, such that optimizing for one particular category is discouraged.

Table 1: Parameters used in the experiments.

Parameter	Value
T	3 (intensity range: $[0,255]$)
T_w	85 (% of total weight)
T_e	12 (% of maximum gradient vector length)
M	5
α_c	0.0005
τ	0.04
μ	0.2
σ	0.05
C_i	0.05
T_d	3

As a preprocessing step, all input images were smoothed with a 3 by 3 Gaussian kernel. Postprocessing was done with a 5 by 5 median filter on the binary foreground images. The parameters were tuned manually, and their values can be found in Table 1. The proposed algorithm was developed using C++ code. It runs at $29fps$ for 320 by 240 pixel videos on a desktop with an Intel® Xeon® E5 Quad Core processor. By way of comparison, the SuBSENSE algorithm runs at approximately $19fps$ on the same hardware configuration, using the C++ libraries provided by the authors (St-Charles et al., 2014).

Some visible results are shown in Figure 8. The calculated measures are given in Table 2. Since the video sequences differ strongly between the categories (e.g. many or few foreground objects, overall degree of difficulty ...), these measures show considerable variance. The overall results are compared with those of other methods in Table 3.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 2: Results on the ChangeDetection.NET 2014 dataset using all 7 measures (Wang et al., 2014).

	Recall	Specificity	FPR	FNR	PBC	Precision	F-Measure
Night Videos :	0.6857	0.9907	0.0093	0.3142	1.6507	0.5973	0.6187
Baseline :	0.8313	0.9962	0.0038	0.1687	1.1425	0.8442	0.8310
Low Framerate :	0.6575	0.9976	0.0024	0.3425	1.0957	0.7409	0.6450
Camera Jitter :	0.7716	0.9801	0.0199	0.2284	2.7949	0.6475	0.6933
Intermittent Object Motion :	0.4429	0.9909	0.0091	0.5571	5.2399	0.7478	0.4755
Dynamic Background :	0.5809	0.9966	0.0034	0.4191	1.0171	0.6426	0.5196
Bad Weather :	0.7782	0.9970	0.0030	0.2218	0.6921	0.8534	0.8003
Shadows :	0.8195	0.9909	0.0091	0.1805	1.8368	0.7992	0.7999
Air Turbulence :	0.6711	0.9986	0.0014	0.3289	0.2865	0.6736	0.6391
Thermal :	0.6613	0.9946	0.0054	0.3387	3.8097	0.8452	0.7118
Pan-Tilt-Zoom :	0.5605	0.9705	0.0295	0.4395	3.2776	0.3030	0.3430
Overall :	0.6782	0.9912	0.0088	0.3218	2.0768	0.6995	0.6434

Table 3: Comparison of results on the ChangeDetection.NET 2014 dataset, Overall. These numbers and references to the other methods can be found in (Wang et al., 2014).

	PBC	F-Measure
FTSG (Flux Tensor, Split Gaussian models)	1.3763	0.7283
SuBSENSE	1.8416	0.7331
CwisarDH	1.5273	0.6812
Spectral-360	2.2722	0.6732
Proposed Method	2.0768	0.6434
Bin Wang Apr 2014	2.9009	0.6577
KNN	3.3200	0.5937
SC_SOBS	5.1498	0.5961
RMoG (Region-based Mixture of Gaussians)	2.9638	0.5735
KDE - ElGammal	5.6262	0.5688
SOBS_CF	6.0709	0.5883
Mahalanobis distance	3.4750	0.2267
GMM, Stauffer and Grimson	3.7667	0.5707
CP3-online	3.4318	0.5805
GMM, Zivkovic	3.9953	0.5566
Multiscale Spatio-Temporal Background Model	5.5456	0.5141
Euclidean distance	6.5423	0.5161

Table 4: Comparison of results on the ChangeDetection.NET 2014 dataset, Night Videos. These numbers and references to the other methods can be found in (Wang et al., 2014).

	PBC	F-Measure
Proposed Method	1.6507	0.6187
SuBSENSE	3.7145	0.5390
FTSG (Flux Tensor, Split Gaussian models)	4.0052	0.5130
Spectral-360	4.4642	0.4832
Mahalanobis distance	3.7362	0.1374
CwisarDH	3.9853	0.3735
SC_SOBS	6.1567	0.4503
RMoG (Region-based Mixture of Gaussians)	5.1606	0.4265
KNN	4.9813	0.4200
SOBS_CF	6.5308	0.4482
KDE - ElGammal	5.2735	0.4365
GMM, Zivkovic	4.7227	0.3960
GMM, Stauffer and Grimson	4.9179	0.4097
Multiscale Spatio-Temporal Background Model	5.8859	0.4164
Euclidean distance	5.5378	0.3859
CP3-online	7.6963	0.3919
Bin Wang Apr 2014	7.8383	0.3802

The proposed method ranks among the better performing methods, with a fourth place on the average PBC measure (Percentage of Bad Classifications) and a sixth place on the F-Measure. The true potential of our algorithm shows when looking at the results for the Night Videos (Table 4). Both the F-Measure and PBC show a vast improvement on the State-of-the-Art. The LTPs, used as edge descriptors, allow the system to detect even very small gradient changes accurately, while the filling method is able to deal with gaps in the contours.

We also note that our method only incorporates gray scale information, which currently limits its performance in some categories. The addition of color information through future work, could allow this method to rank higher in the other categories as well. Furthermore, in order to be able to better handle camera displacements, the proposed method could benefit from the addition of optical flow or geometric image transformation based techniques.

6 CONCLUSION

In this paper, we presented a new foreground background estimation technique. Gradient orientations and confidence measures are calculated by using Local Ternary Patterns. These gradients serve as input in a background modelling framework. Finally, interior points are added to the foreground image, based on Breadth First Search in the foreground gradient image. We have shown our method performs well in the presence of difficult lighting conditions, e.g. at night, compared to state-of-the-art methods.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

ACKNOWLEDGEMENT

We would like to thank the creators of ChangeDetection.NET and all those responsible for providing the means to evaluate our foreground background estimation algorithm.

REFERENCES

Barnich, O. and Droogenbroeck, M. V. (2011). Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724.

Bilodeau, G.-A., Jodoin, J.-P., and Saunier, N. (2013). Change detection in feature space using local binary similarity patterns. In *CRV*, pages 106–112. IEEE.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Cristani, M., Farenzena, M., Bloisi, D., and Murino, V. (2010). Background subtraction for automated multisensor surveillance: A comprehensive review. *EURASIP J. Adv. Sig. Proc.*, 2010.

Droogenbroeck, M. V. and Paquot, O. (2012). Background subtraction: Experiments and improvements for vibe. In *CVPR Workshops*, pages 32–37. IEEE.

Gruenwedel, S., Hese, P. V., and Philips, W. (2011). An edge-based approach for robust foreground detection. In *ACIVS*, pages 554–565.

Heikkila, M. and Pietikainen, M. (2006). A texture-based method for modeling the background and detecting moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):657–662.

Ottmann, T., Soisalon-Soininen, E., and Wood, D. (1984). On the definition and computation of rectilinear convex hulls. *Information Sciences*, 33(3):157 – 171.

Porikli, F. and Tuzel, O. (2003). Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*.

St-Charles, P.-L., Bilodeau, G.-A., and Bergevin, R. (2014). Flexible background subtraction with self-balanced local sensitivity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *CVPR*, pages 2246–2252.

Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650.

Wang, H. and Suter, D. (2005). A re-evaluation of mixture of gaussian background modeling [video signal processing applications]. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 2, pages ii/1017–ii/1020 Vol. 2.

Wang, Y., Jodoin, P.-M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P. (2014). Cdnet 2014: An expanded change detection benchmark dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *ICPR (2)*, pages 28–31.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS