EFIC: Edge Based Foreground Background Segmentation and Interior Classification for Dynamic Camera Viewpoints

Gianni Allebosch(✉), Francis Deboeverie, Peter Veelaert, and Wilfried Philips

Department of Telecommunications and Information Processing,
Image Processing and Interpretation, Ghent University - iMinds, Ghent, Belgium
gianni.allebosch@telin.ugent.be

Abstract. Foreground background segmentation algorithms attempt to separate interesting or changing regions from the background in video sequences. Foreground detection is obviously more difficult when the camera viewpoint changes dynamically, such as when the camera undergoes a panning or tilting motion. In this paper, we propose an edge based foreground background estimation method, which can automatically detect and compensate for camera viewpoint changes. We will show that this method significantly outperforms state-of-the-art algorithms for the panning sequences in the ChangeDetection.NET 2014 dataset, while still performing well in the other categories.

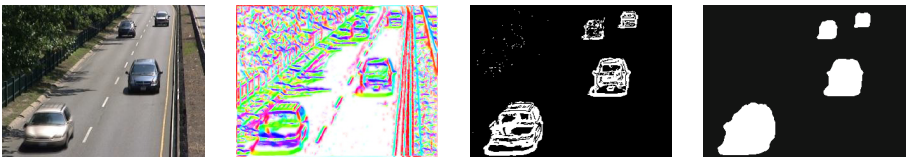
Keywords: Foreground background segmentation · Moving edges · Camera motion · Optical flow

1 Introduction

Foreground background segmentation is frequently used as a first step in many computer vision algorithms [3] [4]. Usually, foreground objects coincide with "moving" pixels, so the terms foreground background segmentation and motion detection are often used interchangeably. However, for some applications, static objects could also be of interest, for example if an object is added to the environment (e.g. a backpack left at a crowded station could contain explosives). On the other hand, the motion of some particular objects, such as waving trees, should often not be analyzed in further steps.

State-of-the-art foreground background segmentation methods consider these remarks, such that they can be employed in a wide variety of situations. The method described in [17] e.g. explicitly models both the foreground and background seperately, while the authors of [14] suggest an automatic and dynamic local parameter tuning mechanism. However, one issue which has not received proper attention in foreground background segmentation literature, is that of camera viewpoint changes. Most algorithms assume a fixed camera position. One algorithm that does not make this assumption, by allowing multiple pixel-wise background models, shows to outperform other methods when the camera does move with respect to the static scene [13].

S. Battiato et al. (Eds.): ACIVS 2015, LNCS 9386, pp. 130–141, 2015.
DOI: 10.1007/978-3-319-25903-1_12



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

Fig. 1. Overview of the algorithm in static camera environments. (a) Input image, (b) LTP based edge descriptors, (c) foreground edges (d) filled foreground image.

In this paper, we present a novel mechanism to overcome issues caused by camera viewpoint changes, for example panning, tilting or jitter. When compared to our previous work [1], the main contribution is the detection of camera viewpoint changes and the selection of appropriate image transformations. In the following section, a brief overview of the original method and some extensions in the detection step are treated. In Sec. 3 and 4 the main novelties, i.e. camera viewpoint change detection and compensation mechanisms, are explained in depth. Finally, the complete algorithm is tested on the comprehensive ChangeDetection.NET 2014 dataset [18]. We will show that our method EFIC (Edge based Foreground background segmentation with Interior Classification) achieves the best F-measure on 4 out of the 11 video categories, with an especially notable improvement on the Pan-Tilt-Zoom sequences.

2 Edge Based Foreground Background Segmentation

The proposed algorithm is an extension of previous work on foreground background segmentation [1]. This method was shown to be very stable in difficult illumination conditions. In this section, the basic methodology to arrive at a foreground image is explained. This section also covers two small extensions on the original method, while the major contributions are covered in Sec. 3 and 4.

2.1 Previous Work

The original algorithm consists of 3 major steps. First, a stable edge descriptor is calculated, using 8-bit Local Ternary Patters (LTPs) [16]. It was shown in [1] that this descriptor can also be regarded as a two dimensional vector, of which the direction represents the edge orientation. The vector length can be regarded as a confidence measure of the angle. Both inter- and intra-pattern information sources are combined to further increase the robustness of the descriptor.

Secondly, the edge descriptor vectors are compared with a multimodal temporal model. Input vectors which significantly differ from the appropriate background vectors are classified as foreground. Since the LTP-features represent image edges, the determined foreground in this step consists of foreground edges. The temporal model is similar to both a Gaussian Mixture Model [15] and the

model proposed by Heikkila et al. [11]. However, the learning rate, which determines how quickly the temporal model is adapted, is adjusted dynamically. More specifically, the learning rate α is comprised of three parts:

1. a base rate α_b
2. an exponential rate α_e
3. an unreliability rate α_u

The base rate, which remains fixed, is typically low, and makes sure that the model is constantly updated slightly in background locations, in order to adapt to slow changes. The exponential rate ensures that the temporal model is updated faster at the beginning of a sequence, as the model generally becomes more reliable over time. If the model does become globally or locally unreliable (e.g. due to a sudden large change in the background or lasting dynamic background), the unreliability rate makes sure that the model is still updated accordingly.

Finally, the algorithm provides a robust contour filling mechanism, which is able to deal with gaps in the object contours. This mechanism treats the foreground edge image as a 4-connected graph, where the edge pixels represent disconnected nodes. The classification of a pixel $\mathbf{p}$ as interior (foreground) or exterior (background) is made with regards to the total "excessive" distance $d_{E,tot}[\mathbf{p}]$ of the shortest paths from the image corners. The distance $d_{E,tot}$ represents the deviation of the shortest paths in the graph, compared to the Manhattan distance d_M between the pixel and the image corners. Formally, given a constant threshold T_E, the edge based foreground image F is defined as follows:

$$F[\mathbf{p}] = \begin{cases} 1 & \text{if } d_{E,tot}[\mathbf{p}] > T_E \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{1}$$

One can show that the resulting foreground shapes are never larger than the orthogonal hull of the objects, and never smaller than a silhouette obtained by filling only the closed contours. One shortcoming of the previous method is that the proposed filling mechanism is sometimes too strong, in particular when objects contain concavities. Therefore, a novel shape correction method is described below.

2.2 Foreground Shape Correction

In our new method we combine pure edge based features with grayscale information. Besides the LTPs, also the grayscale value is fed into a similar multimodal temporal model as described in the previous section. The grayscale based foreground mask is denoted F_G. The eventual foreground mask F' is now determined through a combination of $d_{E,tot}$ (see Sec. 2.1) and F_G as follows:

$$F'[\mathbf{p}] = \begin{cases} 1 & \text{if } d_{E,tot}[\mathbf{p}] > T_E, h \text{ or } (d_{E,tot}[\mathbf{p}] > T_E, l \text{ and } F_G[\mathbf{p}] > 0) \\ 0 & \text{otherwise} \ , \end{cases} \tag{2}$$

where T_E, h and T_E, l are high and low thresholds respectively. The proposed mechanism still ensures the same minimal and maximal size of the silhouettes,

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

Fig. 2. Example of successive foreground shape correction steps. (a) Input image, (b) Foreground mask using single filling threshold, (c) Foreground mask using double threshold and intensity mask (d) Foreground mask after segmentation refinement of (c).

and now uses F_G as a tiebreaker in ambiguous situations. This partially avoids the unwanted filling of concavities in the silhouette (e.g. between arms or legs).

The shapes of the foreground objects are polished further using a variant of the watershed algorithm, described in [12]. The resulting segments are each investigated individually. If, for a certain segment S, at least half of its corresponding pixels in F' represent foreground, then the entire segment S is considered to be a foreground segment. Conversely, if less than half of the corresponding pixels in F' are foreground pixels, S is regarded as a background segment. An example of the shape correction method is shown in Fig. 2.

2.3 Ghost Removal

An object which is static when the temporal model is being built, but later moves to another location, can leave behind a foreground blob in the foreground image, since, at its original location, the image now appears different from the modelled background. This remaining blob is often called a "ghost". In the proposed algorithm, an adaptation of the ghost removal methodology described in [5] is added.

The ghost removal algorithm is executed only in static regions, i.e., regions where the optical flow vectors are small (explained in detail in Sec. 3.1). First, an edge image is constructed from the LTP features described in 2.1. For every static foreground object, the Chamfer distance between the contour of the foreground object and the edge image is calculated and normalized with respect to the contour length. If the Chamfer distance exceeds a threshold T_C, the contour of the foreground object likely does not coincide with a real object in the image. Such objects are removed from the foreground image. After ghost identification, the temporal model has to be updated locally, which is done by setting a high learning rate for all removed pixels. An example of the ghost removal mechanism is shown in Fig. 3.

(a) (b) (c)

Fig. 3. Example of ghost removal. (a) First image in the sequence, (b) input image from further in the sequence, (c) foreground image, where the removed ghost is denoted in red.

3 Camera Motion Detection

If the camera viewpoint is static, reasoning about foreground estimation is generally straightforward. However, as soon as the camera viewpoint changes, the appearance of all pixels might change, even though the background itself does not. In this section we describe how camera motion can be detected. We propose a camera motion detection framework based on optical flow.

3.1 Optical Flow

Optical flow estimation is a widely studied problem in computer vision [10]. It is an image feature which essentially represents the motion of individual pixels between subsequent frames. By assuming a constant brightness, optical flow calculation boils down to the estimation of the displacement $(\Delta x, \Delta y)$ of a pixel at $\mathbf{p} = (x, y)$ [8]. This vector can only be obtained by introducing additional constraints, which is where optical flow methods described in literature differ from one another. These methods can be divided into two main groups: sparse and dense optical flow. Sparse optical flow methods first detect interesting points in the image, and estimate only their motion between successive frames. Dense optical flow methods estimate the motion vectors for all image points.

3.2 Flow Based Camera Motion Detection

If the camera viewpoint is static, it can be expected that the only pixels that appear to move between successive frames come from either foreground objects, or dynamic background otherwise. In most video sequences, the amount of dynamically moving pixels is relatively small compared to the static ones. However, if the camera viewpoint does change, most pixels in the image will also. So, camera motion can be detected by the occurrence of significant optical flow vectors in a large part of the image. In the detection step, dense optical flow is preferred. In sequences with generally smooth backgrounds, not many feature points will be found by sparse flow methods, except on the foreground objects themselves, which could lead to many false positive camera motion detections.

In the proposed method, the dense optical flow vectors are calculated by using the efficient algorithm described in [6]. Let $\mathbf{V}[\mathbf{p}]$ be the flow vector image at pixel $\mathbf{p}$. Now, we define the optical flow mask F_f as follows:

$$F_f[\mathbf{p}] = \begin{cases} 1 & \text{if } ||\mathbf{V}[\mathbf{p}]|| > T_f \\ 0 & \text{otherwise} , \end{cases} \tag{3}$$

where T_f is a typically low threshold (e.g. 1 pixel). So, F_f represents a significance classification of all flow vectors. If the ratio of significant flow vectors is larger than a second threshold T_n, camera motion is detected.

4 Camera Motion Compensation

In order to compensate for the camera motion, the effect of this motion should be mitigated at every pixel location, while moving foreground objects should still be detected. The next section will explain how the necessary image transformations are executed. Afterwards, we will distinguish two scenarios: one where the camera is undergoing a fairly constant motion away from the original position (e.g pan-tilt-zoom) and one where the camera keeps moving around the same position (jitter).

4.1 Affine Image Transformation

In order to compare a new image with a background model, the image should first be transformed such that coinciding pixels also represent the same objects in the model. However, since the distances to the objects in the scene are not known a priori in most applications, the effects of potential perspective changes on the image formation are difficult to model. Luckily, when a scene's relief is small, relative to the average distance from the objects to the camera, the weak-perspective image formation model can be used to describe the image formation [9]. This model assumes that all objects are at a similar distance from the camera. It is proven in [7] that, assuming a weak-perspective model, all arbitrary projection transformation matrices can be written in the form of an affine matrix.

In this work, the affine transformation between two images is estimated by making use of the Pyramidal implementation of the Lucas Kanade Feature Tracker [2]. The algorithm first detects interest points and then calculates sparse optical flow vectors to detect their individual motion. In the second phase of the algorithm, the robust affine transformation matrix M_{tf} is selected through a RANSAC framework. From this matrix, the expected flow $\mathbf{V_e}$ can now be directly determined. For a certain pixel $\mathbf{p} = (x, y, 1)^T$ in homogeneous coordinates:

$$\mathbf{V_e}[\mathbf{p}] = M_{tf}\mathbf{p} , \tag{4}$$

4.2 Distinction Between Panning/Tilting and Jitter

If the center pixel is also the origin of the camera's coordinate system, the expected flow at the center pixel $\mathbf{V_e}[\mathbf{0}]$ can be represented by the third column of M_{tf}, also known as a translation vector $\mathbf{t_r} = (t_{r,x}, t_{r,y})$. If the camera viewpoint changes between successive frames, $\mathbf{t_r}$ will be a nonzero vector whose orientation $(\arctan \frac{t_{r,y}}{t_{r,x}})$ represents the direction of the camera shift. If this direction is more or less constant for a longer period, the camera viewpoint obviously moves away from its original position, and a panning (or tilting) camera motion can be detected. Conversely, if the direction of $\mathbf{t_r}$ changes a lot, it is more likely that the camera is jittery, but not necessarily moving away from the original position.

In the proposed method, the distinction between panning and jittery camera's is derived from the reasoning above. Let us define two accumulators: acc_p for panning and acc_j for jitter, both initialized to 0 in the beginning of the sequence. Every time a camera viewpoint change is detected, the current direction of $\mathbf{t_r}$ is compared to the previous one. If the angles differ by more than 90, acc_j is incremented. Otherwise, acc_p is incremented. The camera motion compensation is then executed with regard to the highest corresponding accumulator value.

4.3 Jitter Compensation

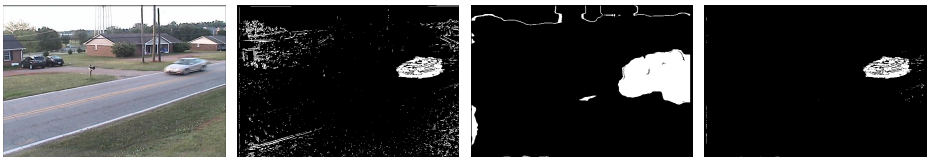
If the camera is moving around the same position, the original background model can still be used. By transforming the current image with regard to the most likely background image as described in Sec. 4.1, a pixelwise evaluation of the difference between the current image and background model is feasible, as in the standard case when no camera motion is detected.

There are however some subtle considerations which should be taken into account. Note that the affine transformation estimation can be erroneous, for example due to comparison with interest points coinciding with foreground objects. Furthermore, depending on the camera settings, a moving camera viewpoint might introduce motion blur, which distorts some image structures, especially around edges. In the proposed algorithm, the constant learning rate α_c is therefore raised when jitter is detected, and applied to the entire image, instead of to the background regions.

4.4 Panning/Tilting Compensation

If a panning or tilting camera change is detected, the original background image is no longer usable if the new camera viewpoint has deviated too far from the original one, since pixel-wise comparisons to the original model have become impossible.

However, if the camera motion is relatively slow, such that the spatial relation between successive frames can be established, it becomes possible to build a short-term edge background model $\mathbf{B_s}$. A new frame is compared to this model after using the affine transformation as discussed in the previous sections. Here, $\mathbf{B_s}$ is also transformed after every frame as long as the panning motion continues.



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

Fig. 4. Creation of the foreground edges in a sequence with panning camera. (a) Input image, (b) Short-term foreground mask, (c) compensated flow mask, (d) logical AND of short-term foreground mask and compensated flow mask.

Note however, that $\mathbf{B_s}$ in the proposed algorithm is unimodal. Experiments show that using a more complex multimodal background model, as described in Sec. 2.1, does not improve the algorithms performance, likely due to the fact that it is difficult to build a complex model quickly enough. Comparing the LTP feature image $\mathbf{I_{LTP}}$ determined from the input with $\mathbf{B_s}$ results in the short-term foreground mask F_s:

$$F_s[\mathbf{p}] = \begin{cases} 1 & \text{if } ||\mathbf{I_{LTP}}[\mathbf{p}] - \mathbf{B_s}[\mathbf{p}]|| > T_s \\ 0 & \text{otherwise .} \end{cases} \tag{5}$$

Still, the requirement of rapid model construction makes it more likely that parts of the model are unreliable. To overcome the potential decrease in accuracy, the flow vector image $\mathbf{V}$ is used as a secondary decision mechanism. The compensated flow image $\mathbf{V_c}$ is now defined as follows:

$$\mathbf{V_c}[\mathbf{p}] = \mathbf{V}[\mathbf{p}] - \mathbf{V_e}[\mathbf{p}] \ . \tag{6}$$

Thus, the original flow vectors are compensated with regard to the expected affine image transformation, resulting from the camera viewpoint change. It is expected that for static objects, the corresponding compensated flow vectors should be close to $\mathbf{0}$. However, if an object is moving in the scene, the optical flow vectors will differ from the globally calculated transformation and will locally coincide with nonzero compensated flow vectors. Thus, utilizing a final flow compensation threshold $T_{f,c}$, the compensated foreground flow mask $F_{f,c}$ is now defined as

$$F_{f,c}[\mathbf{p}] = \begin{cases} 1 & \text{if } ||\mathbf{V_c}[\mathbf{p}]|| > T_{f,c} \\ 0 & \text{otherwise .} \end{cases} \tag{7}$$

The resulting foreground (edges) mask now consists of the pixelwise logical AND operation between $F_{f,c}$ and F_s. Fig. 4 shows a visual example of foreground detection in a sequence with a panning camera. Note that the contour filling, shape correction and ghost removal steps described in Sec. 2 are still executed after the foreground edge detection step. Finally, once the panning motion has stopped, the exponential learning rate (Sec. 2.1) is reset, such that the temporal model is quickly rebuilt at the new position.

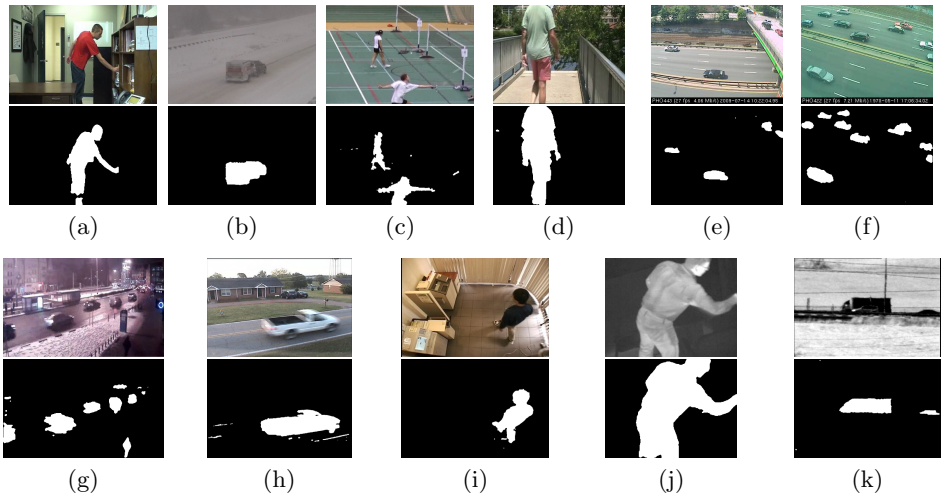


Fig. 5. Performance of our proposed method on a few hand picked frames from the ChangeDetection.NET 2014 dataset [18]. (a) Baseline, (b) Bad Weather, (c) Camera Jitter, (d) Dynamic background, (e) Intermittent Object Motion, (f) Low Framerate, (g) Night Videos, (h) Pan-Tilt-Zoom, (i) Shadows, (j) Thermal, (k) Turbulence.

5 Experiments

The proposed method was tested on the rigorous ChangeDetection.NET 2014 dataset [18]. This dataset comprises of 11 categories of 4 to 6 videos each. The categories are Bad Weather (BW), Low Framerate (LF), Night Videos (NV), Pan-Tilt-Zoom (PTZ), Turbulence (TB), Baseline (BL), Camera Jitter (CJ), Intermittent Object Motion (IOM), Shadows (SH) and Thermal (TH).

The creators of the website also provide ground truth for the majority of the frames, such that 7 performance measures in total can be calculated: Recall, specificity, false positive rate, false negative rate, precision, percentage of wrong classifications and F-Measure. However, as explained in [14] and [13], the F-measure is the most unbiased representation of the performance of a foreground background segmentation algorithm. So, we will focus on this performance measure to compare our proposed method to the state of the art (see Table 1). Note that it is required to use the same parameters and thresholds for all videos, such that optimizing for a particular video or category is discouraged, and only truly versatile methods achieve high scores on this dataset. The values of the thresholds discussed in this paper are given in Table 2.

Smoothing by a 3 by 3 Gaussian kernel was added as a preprocessing step, while the foreground mask were postprocessed by a 5 by 5 median filter. The algorithm runs at about 16 frames per second on a desktop pc with an Intel® Xeon® E5 Quad Core processor for 320 by 240 pixel videos. The proposed method currently achieves the highest F-measure in 4 of the 11 categories.

Table 1. Comparison of the F-Scores of all methods applied to the ChangeDetection.NET 2014 database per category and overall. The highest score is denoted in bold faced numbers. In the categories where the proposed method achieves the highest score, the second highest score is denoted in blue. References to the other methods can be found on the ChangeDetection.NET website [18].

Method	BW	LF	NV	PTZ	TB	BL	DB	CJ	IOM	SH	TH	Over.
EFIC (Proposed)	77.86	**66.32**	**65.48**	**58.42**	67.13	91.72	57.79	71.25	57.83	82.02	**83.88**	70.88
SuBSENSE	86.19	64.45	55.99	34.76	**77.92**	**95.03**	81.77	81.52	65.69	89.86	81.71	**74.08**
FTSG	82.28	62.59	51.30	32.41	71.27	93.30	**87.92**	75.13	**78.91**	88.32	77.68	72.83
SaliencySubsense	85.93	65.15	53.48	33.99	75.12	94.83	81.57	80.71	60.12	**89.94**	68.57	71.76
MBS V0	77.30	62.79	51.58	51.18	56.98	92.87	79.04	**83.67**	70.92	77.84	81.15	71.39
CwisarDH	68.37	64.06	37.35	32.18	72.27	91.45	82.74	78.86	57.53	85.81	78.66	68.12
Spectral-360	75.69	64.37	48.32	36.53	54.29	93.30	77.66	71.42	56.09	85.19	77.64	67.32
Bin Wang Apr 2014	76.73	46.89	38.02	13.48	75.45	88.13	84.36	71.07	72.11	81.28	75.97	65.77
AAPSA	77.42	49.42	41.61	33.02	46.43	91.83	67.06	72.07	50.98	79.53	70.30	61.79
SC_ SOBS	66.20	54.63	45.03	4.09	48.80	93.33	66.86	70.51	59.18	77.86	69.23	59.61
KNN	75.87	54.91	42.00	21.26	51.98	84.11	68.65	68.94	50.26	74.68	60.46	59.37
SOBS_ CF	63.70	51.48	44.82	21.26	47.02	92.99	65.19	71.50	58.10	77.21	71.40	58.83
CP3-online	74.85	47.42	39.19	26.60	37.43	88.56	61.11	52.07	61.77	70.37	79.17	58.05
RMoG	68.26	53.12	42.65	24.70	45.78	78.48	73.52	70.10	54.31	72.12	47.88	57.35
GMM - Stauffer and Grimson	73.80	53.73	40.97	15.22	46.63	82.45	63.30	59.69	52.07	73.70	66.21	57.07
KDE - ElGammal	75.71	54.78	43.65	3.65	44.78	90.92	59.61	57.20	40.88	76.60	74.23	56.88
GraphCutDiff	**87.87**	51.27	46.88	37.23	51.43	71.47	53.91	54.89	40.19	72.28	57.86	56.84
GMM - Zivkovic	74.06	50.65	39.60	10.46	41.69	83.82	63.28	56.70	53.25	73.22	65.48	55.66
Euclidean dist.	67.01	50.15	38.59	3.95	41.35	87.20	50.81	48.74	48.92	67.86	63.13	51.61
Multiscale Spatio-Temporal BG	63.71	33.65	41.64	3.64	52.91	84.50	59.53	50.73	44.97	79.18	51.03	51.41
Mahalanobis dist.	22.12	7.97	13.74	3.74	33.59	46.42	17.98	33.58	22.90	33.53	13.83	22.67

For Night Videos and Pan-Tilt-Zoom, the gaps with the second best scoring methods are especially significant. This can be attributed to the fact that the method is based on the illumination-invariant features described in [1] and that the proposed method is able to cope with slow camera viewpoint changes. However, note that in our experiments the performance on 1 video with a zooming camera was unacceptable. Here, the proposed algorithm failed to detect the changing camera parameters and thus did not compensate for this. A refined zoom detection mechanism could even further improve the performance in this category.

EFIC is also able to provide top performance in the Low Framerate and Thermal videos. For Low Framerate the updating mechanism can quickly learn the correct background, and is not disturbed by rapidly changing foreground objects. Since the Thermal images all consist of single color channel frames and the proposed method does not rely on on chromacity information, the good performance in this category can also be explained.

In the Dynamic Background and Intermittent Object Motion sequences, the proposed method is not yet able to achieve the desired performance. In the other categories, the performance is closer to the state-of-the-art, but not at the very top. As opposed to the Thermal videos, the lack of chromacity information is a limiting factor in videos where the intensity of the foreground objects is similar

Table 2. Threshold values discussed in this paper. Unless stated otherwise, all values are expressed in pixel widths

Threshold	Value
T_E: Single excessive distance	3
$T_{E,l}$: Low excessive distance	1
$T_{E,h}$: High excessive distance	13
T_f: Significant flow	1
T_n: Significant flow ratio	70 (% of total number of pixels)
$T_{f,c}$: Compensated flow	2
T_s: Short-term background model	40 (% of max. LTP vector length)

to that of the background. This effect is especially noticeable in the Dynamic Background videos, where the intensities of the boats are similar to the water intensity, even though the chromacity differs significantly. Utilizing chromacity only when useful through a decision tree approach, could improve the overall results while not harming the performance in categories where EFIC scores best.

It can also be noted that some of these categories present multiple kinds of difficulties. E.g., Bad Weather videos incorporate both difficult illumination conditions, but also dynamic background and thus address both the strengths and weaknesses of the proposed method. We also note that the described ghost removal method did improve the overall results, but still failed in certain sequences with textured background. Improvements could arise from the extension of this method, e.g. through utilization of gradient directional information.

6 Conclusions

In this paper, we propose an edge based foreground background segmentation algorithm, which is able to handle non-static camera viewpoints, using a combination of optical flow and affine image transformations. It was shown that the method achieves good overall performance on the ChangeDetection.NET 2014, while even reaching the best F-measure in 4 out of 11 categories in total. Notably, the addition of color information as an extra feature should help in improving the performance of this method even further.

Acknowledgment. We would like to thank the creators of ChangeDetection.NET, for providing a comprehensive evaluation dataset and all necessary tools to analyze and validate the algorithm's performance.

References

1. Allebosch, G., Van Hamme, D., Deboeverie, F., Veelaert, P., Philips, W.: Edge based foreground background estimation with interior/exterior classification. In: Proceedings of the 10th International Conference on Computer Vision Theory and Applications, vol. 3, pp. 369–375. SCITEPRESS (2015)

2. Bouguet, J.Y.: Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Intel Corporation Microprocessor Research Labs, Tech. rep. (2000)
3. Bouwmans, T., Baf, F.E., Vachon, B.: Background modeling using mixture of gaussians for foreground detection a survey. Recent Patents on Computer Science, 219–237 (2008)
4. Cristani, M., Farenzena, M., Bloisi, D., Murino, V.: Background subtraction for automated multisensor surveillance: A comprehensive review. EURASIP J. Adv. Sig. Proc. (2010)
5. Evangelio, R., Sikora, T.: Complementary background models for the detection of static and moving objects in crowded environments. In: 2011 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), pp. 71–76, August 2011
6. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) SCIA 2003. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003)
7. Faugeras, O.D., Luong, Q.T., Papadopoulo, T.: The geometry of multiple images - the laws that govern the formation of multiple images of a scene and some of their applications. MIT Press (2001)
8. Fleet, D.J., Weiss, Y.: Optical flow estimation. In: Handbook of Mathematical Models in Computer Vision, pp. 237–257. Springer US (2006)
9. Forsyth, D.A., Ponce, J.: Geometric camera models. In: Computer Vision: A Modern Approach, 2nd edn., pp. 33–61. Pearson, international edn. (2012)
10. Fortun, D., Bouthemy, P., Kervrann, C.: Optical flow modeling and computation: A survey. Computer Vision and Image Understanding 134(0), 1–21 (2015). image Understanding for Real-world Distributed Video Networks
11. Heikkila, M., Pietikainen, M.: A texture-based method for modeling the background and detecting moving objects. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(4), 657–662 (2006)
12. Meyer, F.: Color image segmentation. In: International Conference on Image Processing and its Applications, pp. 303–306 (1992)
13. Sajid, H., Cheung, S.C.S.: Background subtraction for static and moving camera. In: IEEE International Conference on Image Processing (2015)
14. St-Charles, P.L., Bilodeau, G.A., Bergevin, R.: Subsense: A universal change detection method with local adaptive sensitivity. IEEE Transactions on Image Processing **24**(1), 359–373 (2015)
15. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: CVPR, pp. 2246–2252 (1999)
16. Tan, X., Triggs, B.: Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Transactions on Image Processing **19**(6), 1635–1650 (2010)
17. Wang, R., Bunyak, F., Seetharaman, G., Palaniappan, K.: Static and moving object detection using flux tensor with split gaussian models. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2014
18. Wang, Y., Jodoin, P.M., Porikli, F., Konrad, J., Benezeth, Y., Ishwar, P.: Cdnet 2014: An expanded change detection benchmark dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2014