Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 574041, 11 pages
http://dx.doi.org/10.1155/2014/574041

Research Article

Automatic Moving Object Segmentation for Freely Moving Cameras

Yanli Wan,[1,2] Xifu Wang,[1] and Hongpu Hu[2]

[1] *School of Traffic and Transportation, Institute of System Engineering and Control, Bejing Jiaotong University, Beijing 100044, China*
[2] *Institute of Medical Information, Chinese Academy of Medical Sciences, Beijing 100020, China*

Correspondence should be addressed to Yanli Wan; wanyanli0201@gmail.com

Received 13 April 2014; Revised 30 June 2014; Accepted 1 July 2014; Published 12 August 2014

Academic Editor: Bin Jiang

This paper proposes a new moving object segmentation algorithm for freely moving cameras which is very common for the outdoor surveillance system, the car build-in surveillance system, and the robot navigation system. A two-layer based affine transformation model optimization method is proposed for camera compensation purpose, where the outer layer iteration is used to filter the non-background feature points, and the inner layer iteration is used to estimate a refined affine model based on the RANSAC method. Then the feature points are classified into foreground and background according to the detected motion information. A geodesic based graph cut algorithm is then employed to extract the moving foreground based on the classified features. Unlike the existing global optimization or the long term feature point tracking based method, our algorithm only performs on two successive frames to segment the moving foreground, which makes it suitable for the online video processing applications. The experiment results demonstrate the effectiveness of our algorithm in both of the high accuracy and the fast speed.

1. Introduction

Moving object detection and segmentation is a basic technique for many applications such as intelligent video surveillance, intelligent transportation system, video content analysis, video event detection, and video semantic annotation. In all these applications, the cameras capturing the videos may not be static. For example, the camera of an outdoor surveillance system may be slightly shaking because of strong winds, and the video used for content analysis or event detection may be captured by a hand-held camera. Thus a moving object detection and segmentation algorithm that can handle the freely moving cameras is necessary for these cases. However, on one hand most of the existing moving object detection and segmentation algorithms are only designed for the static cameras, such as Gaussian Mixture Models proposed by Stauffer and Grimson [1], Kernel density estimation (KDE) used in [2]. Although many methods have been proposed to improve these kinds of algorithms, such as Sun et al. [3] who proposed to employ graph cut [4] algorithm to improve the accuracy of the segmentation results and Patwardhan et al. [5] who constructed a layer model for the scene to improve the robustness of foreground detection and segmentation, none of these methods can be directly extended for the freely moving cameras.

In recent years, several moving object detection and segmentation algorithms for freely moving cameras have been proposed [6–12]. Liu and Gleicher [6] proposed to learn a moving object model by collecting the sparse and insufficient motion information throughout the video. They first detect the moving patches of the foreground object, and then combine the moving patches of many frames to learn a color model of the foreground object which is used for segmentation. However, this kind of method can only be used to process video sequences offline and cannot be applied for the online cameras. Kundu et al. [7] proposed a motion detection framework based on multiview geometric constraints such as the epipolar constraints. However, this method needs to calibrate the robot-camera with a chess board and can only detect rough moving regions instead of accurate object segmentation. This restricts the application of this algorithm. Zhang et al. [8] proposed to use structure

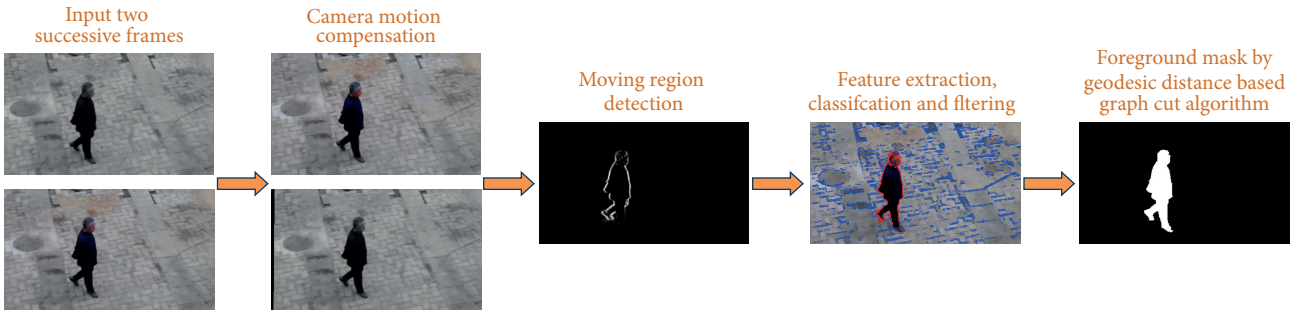


Figure 1: The flow chart of our algorithm.

from motion method to detect and segment the foreground moving object. This method needs to first estimate the dense depth map for each frame, and then in the segmentation step, a global optimization is applied to multiframes to extract the moving object. The depth map estimation and object segmentation step will be run iteratively for several times in order to obtain accurate results. This method is very time consuming and can only be used for offline video sequences. Several algorithms [9–11, 13] employing point trajectories to segment the moving objects are proposed in recent years. The intuition of these kinds of methods is that the motion caused by the camera movement is restricted by some geometric constraints, while the motion caused by the object movement is not. Thus the moving object can be detected and segmented by analyzing the long term trajectories of the key points. However these methods usually need to calculate the dense optical flow over long time frames, which may be too time consuming to run in real time. Once again, these methods cannot be used in online scenario, because they are not processing the video frame by frame. Elqursh and Elgammal [14] improve point trajectories based method by adding Bayesian filtering framework to estimate the motion and appearance models. And it also updates the point trajectories and motion/appearance models online, so that this algorithm can be used for the online video segmentation scenario. However, the high computational cost is still a problem.

In this paper, we propose a novel moving object detection and segmentation algorithm for the freely moving cameras.

Compared to the existing moving object segmentation algorithms for freely moving cameras, our algorithm has the following characteristics.

(1) Unlike most of the existing algorithms, our algorithm does not employ the global optimization or long term feature point tracking. It only uses two successive frames to extract the moving object, which makes it suitable for the online video processing task.

(2) A two-layer iteration based camera motion compensation method is proposed, where the outer layer iteration is used to update the foreground and background feature sets according to the current parameters of the camera motion compensation models, and the inner layer iteration employs a RANSAC method to estimate the parameters of the camera motion compensation model based on the current background feature set. This two-layer iteration based method makes the camera motion compensation more robust and accurate.

(3) A feature classification and filtering algorithm based on GMM color model is proposed, and the classified feature points are used as the input of the geodesic distance based graph cut algorithm, which can return a very accurate segmentation result.

The rest of the paper is arranged as follows. Section 2 is an overview of our algorithm, and Section 3 describes the details of our algorithm. After the experiments and discussions in Section 4, the conclusions are presented in Section 5.

2. Algorithm Overview

Figure 1 shows a flow chart of our algorithm. As we described before, our algorithm is just based on two successive frames, so the input of our algorithm is the former and current frames of one video. The algorithm has 3 steps.

(1) Camera Motion Compensation. Since the camera movement between two successive frames is very small in most cases, we can simply assume that the background between the former frame and the current frame only has the translation and the rotation movement. Thus an affine transformation model can be employed to simulate the movement of the background. When estimating the affine transformation parameters, the corresponding feature points are first found by a forward and backward optical flow algorithm, and then a two-iteration based method is proposed to estimate the parameters.

(2) Feature Extraction and Classification. The edge and the corner features [15] are extracted and then classified into the moving foreground features (denoted as red points) and the background features (denoted as blue points) according to the detected motion regions. The foreground and background feature sets are then filtered by GMM color models.

(3) Foreground Extraction with Geodesic Distance Based Graph Cut. After the foreground and background feature sets are obtained, the geodesic distance from other pixels to the feature points are calculated, and a geodesic confidence map is generated. By incorporating the geodesic distance and

the geodesic confidence map with the graph cut algorithm, accurate foreground object can be segmented.

3. Details of Our Algorithm

3.1. Camera Motion Compensation. For most of the videos, the camera only has a very small movement between two successive frames; thus it is assumed that the camera only has the translation and rotation movement in such a short interval, which can be modeled by the affine transformation. In this model, it is just assumed that the displacement vector $\mathbf{u} = (u, v)$ of pixel (x, y) can be written as an affine function of the coordinate (x, y):

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{T} = \begin{bmatrix} a_1 x + a_2 y + t_1 \\ a_3 x + a_4 y + t_2 \end{bmatrix}, \quad (1)$$

where $\mathbf{R}$ is the rotation matrix with parameters $\mathbf{R} = \left[\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}\right]$, $\mathbf{T}$ is a translation matrix with $\mathbf{T} = \left[\begin{smallmatrix} t_1 \\ t_2 \end{smallmatrix}\right]$. Here the rotation matrix $\mathbf{R}$ also contained the scale change parameters a_1 and a_4; thus this model can handle the scale changes of the background scene, such as the video captured by a forward or backward moving camera.

Since the camera motion and the foreground motion are distinct, this means that the foreground motion is not appropriate to be modeled by the affine transformation model. Thus in ideal, the pixels used to estimate the affine parameters should only contain the background pixels. This can be achieved by our two-layer iteration based method as shown in Figure 2. The outer layer iteration is used to update the fore- and background feature points according to the motion regions detected by the current affine parameters. The RANSAC process is used to estimate the affine parameters based on the updated background features.

The feature points used in our paper are the edge and corner points which can be detected using the method described in [15]. In order to estimate the affine model parameters, the corresponding feature points of the two successive frames should be detected. We employ the forward and backward optical flow estimation to achieve this goal. For the current frame I^t, we first extract its feature points (denoted as $\mathcal{F}_t = \{f_i^t, i = 1 \cdots N\}$, where N is the number of the feature points) and then use the pyramid Lucas Kanade optical flow [16] to track these features to the next frame I^{t+1}. Thus we obtain a set of the feature points on frame I^{t+1} by this forward optical flow, which are denoted as $\mathcal{F}_{t+1} = \{f_i^{t+1}, i = 1 \cdots N\}$. Then we track the features f_i^{t+1} back from I^{t+1} to I^t using the backward optical flow and obtain a new set of features on I^t and denote it as $\mathcal{F}_t' = \{f_{i'}^t, i = 1 \cdots N\}$. In the ideal case, $\mathcal{F}_t$ and $\mathcal{F}_t'$ should be the same. However, due to the errors of the optical flow estimation, they are not identical. By comparing $\mathcal{F}_t$, $\mathcal{F}_t'$, and $\mathcal{F}_{t+1}$, we can remove the feature points that have erroneous optical flow, so as to find the correct corresponding feature points between the two successive frames. We use two criteria to filter the optical flow errors. The first is to employ the ZNCC (zero-mean normalized cross correlation), which is defined as

$$\begin{aligned} \mathrm{ZNCC}(\mathbf{x}, \mathbf{x}') = & \left(\sum_i \left(I^t(\mathbf{x} + i) - \bar{I}^t(\mathbf{x}) \right) \right. \\ & \left. \times \left(I^{t+1}(\mathbf{x}' + i) - \bar{I}^{t+1}(\mathbf{x}') \right) \right) \\ & \times \left(\sum_i \left(I^t(\mathbf{x} + i) - \bar{I}^t(\mathbf{x}) \right)^2 \right. \\ & \left. \times \sum_i \left(I^{t+1}(\mathbf{x}' + i) - \bar{I}^{t+1}(\mathbf{x}') \right)^2 \right)^{-1/2}, \end{aligned} \quad (2)$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the coordinates of the corresponding feature points in $\mathcal{F}_t$ and $\mathcal{F}_{t+1}$, respectively, $\bar{I}^t(\mathbf{x})$ and $\bar{I}^{t+1}(\mathbf{x}')$ are the mean values of the pixel intensity for the given $t \times t$ ($t = 11$ in our experiment) windows centered at $\mathbf{x}$ and $\mathbf{x}'$, respectively. The ZNCC score for each pair of feature points in $\mathcal{F}_t$ and $\mathcal{F}_{t+1}$ should be calculated, and then a part of the feature points with erroneous optical flows can be filter out by setting a threshold η_1; that is, if $\mathrm{ZNCC}(\mathbf{x}, \mathbf{x}') < \eta_1$, then the optical flow from $\mathbf{x}$ to $\mathbf{x}'$ is considered as error. In our experiment, we find that setting η_1 as the median value of the ZNCC scores can obtain good enough results.

Another criterion to filter the erroneous optical flows is to use the displacements of the corresponding pixels between $\mathcal{F}_t$, $\mathcal{F}_t'$, which is defined as the Euclidian Distance between the coordinates of the corresponding points and denoted as $\mathrm{Dis}(\mathbf{x}, \mathbf{x}'')$. Similarly, if $\mathrm{Dis}(\mathbf{x}, \mathbf{x}'') > \eta_2$, the forward optical flow from $\mathbf{x}$ to $\mathbf{x}'$, and the backward optical flow from $\mathbf{x}'$ to $\mathbf{x}''$ are considered as errors. η_2 is also set as the median value of the displacements of the corresponding features points. After filtering the erroneous optical flow, we obtain the feature point matching results as shown in Figure 2. The matching feature points are denoted as a feature set $\mathcal{S}_m$.

Once obtaining the matching feature points, the two-layer iteration is performed. The detail is described as Algorithm 1. In the inner-layer iteration, the RANSAC algorithm requires 3 pairs of corresponding feature points to estimate the affine parameters $(a_1, a_2, a_3, a_4, t_1, t_2)$. Since we use the 6 parameters to estimate the global motion of the whole image, the 3 pairs of feature points sampled from $\mathcal{S}_b$ should be distributed over the whole image instead of a local area. For the moving region detection, we use the estimated affine parameter to compensate the camera motions, and calculate the frame difference to find the moving regions. Then $\mathcal{S}_b$ can be updated by classifying the features set $\mathcal{S}_m$ into foreground and background according to the frame difference:

$$\mathcal{S}_m(x, y) \in \begin{cases} \mathcal{S}_f & \text{if } d(x, y) > \lambda, \\ \mathcal{S}_b & \text{if } d(x, y) \le \lambda, \end{cases} \quad (3)$$

where $\mathcal{S}_m(x, y)$ denotes the feature points at location (x, y), $\mathcal{S}_f$ and $\mathcal{S}_b$ are two sets of foreground and background feature

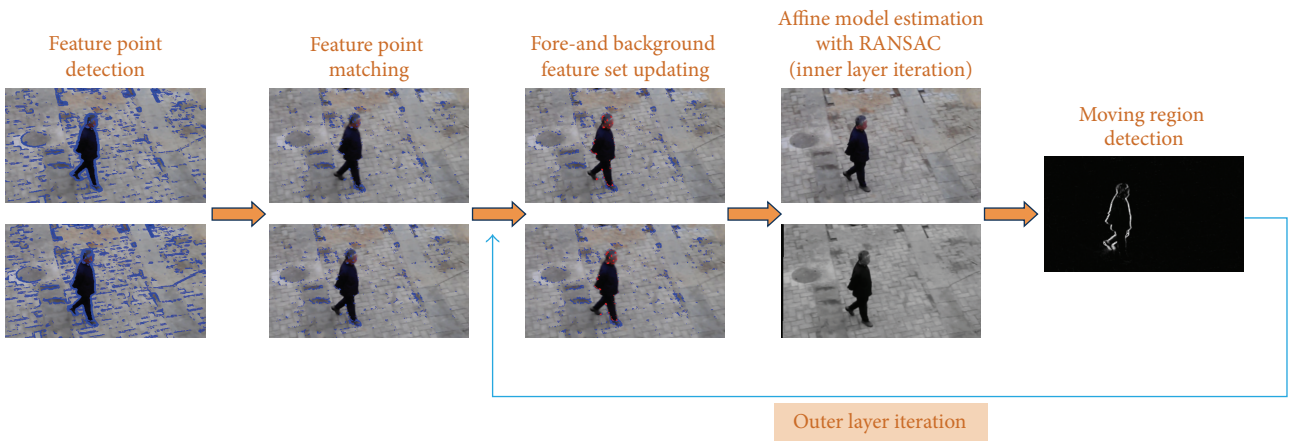


Figure 2: Camera motion compensation based on two-layer iteration.

Initialization: The background feature point set is initialized as $\mathcal{S}_b = \mathcal{S}_m$;
Step 1. Inner-layer iteration, employs the RANSAC algorithm to estimate the affine parameters based on the current feature set $\mathcal{S}_b$;
Step 2. Moving region detection, finds the moving regions based on the current affine parameters;
Step 3. Update the background feature set according to the detected moving regions;
Step 4. Jump to Step 1 to start a new outer-layer iteration until it converges. That is, the feature points in $\mathcal{S}_b$ are stable.

Algorithm 1: Two-layer iteration based camera motion compensation.

points, respectively, and λ is a threshold value. $d(x, y)$ is the frame difference value at pixel (x, y), and is calculated as

$$d_{(x,y)} = \left\| I(x, y, t-1) - I'(x, y, t) \right\|^2, \tag{4}$$

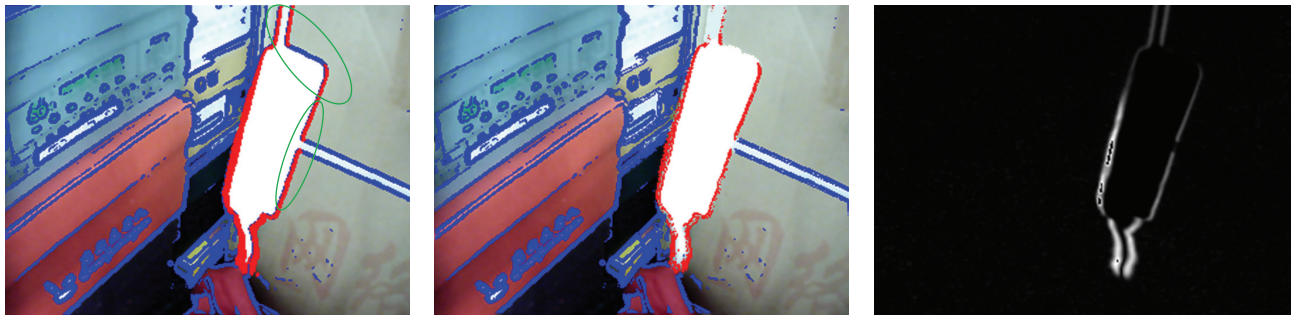
where $I'(x, y, t)$ is the affine warped current frame.

3.2. Feature Extraction and Classification. After obtaining the final affine parameters, we can obtain the frame difference using (4) and then classify the feature points $\mathcal{F}_{t+1}$ of the current frame into the foreground and background feature sets $\mathcal{F}_f$ and $\mathcal{F}_b$ using (3) as shown in Figure 1.

$\mathcal{F}_f$ and $\mathcal{F}_b$ cannot be directly used for graph cut algorithm in the following step to extract the foreground object, because there usually exist some classification errors. As pointed out by the green ellipses in Figure 3(a), some foreground feature points are misclassified into background. This is because the moving regions detected by the frame difference as shown in Figure 3(c) are composed of both the real foreground regions and the false foreground regions. These false foreground regions are actually background regions occluded by the moving object. In order to eliminate these misclassifications, we further perform a refining process in our algorithm. Since we already have an initial classification of the feature points, we can build two Gaussian mixture models (GMMs) for $\mathcal{F}_f$ and $\mathcal{F}_b$ and then use these two models to reestimate the probability of each feature point belonging to the foreground. The feature points in $\mathcal{F}_f$ or $\mathcal{F}_b$ are first classified into K clusters, respectively, by a farthest-point clustering algorithm [17], and then the mean and variance (μ, σ) for each cluster are calculated to construct GMMs. The probability of feature points belonging to the foreground can be estimated as

$$\begin{aligned} p_f(y) &= \sum_{i=1}^{K} P_f(k) \, \Pi_{d=1}^{3} \Phi\left(y_d, \mu_d^f, \sigma_d^f\right), \\ p_b(y) &= \sum_{i=1}^{K} P_b(k) \, \Pi_{d=1}^{3} \Phi\left(y_d, \mu_d^b, \sigma_d^b\right), \\ P_f(y) &= \frac{p_f(y)}{p_b(y) + p_f(y)}, \qquad P_b(y) = 1 - P_f(y), \end{aligned} \tag{5}$$

where y is the color vector of one feature point to be estimated, $P_f(k)$ and $P_b(k)$ are the prior probability of the feature points in this Gaussian component and can be calculated as the ratio between the number of feature points in this component and the number of feature points in the whole



(a) (b) (c)

Figure 3: Feature classification errors and elimination.

GMM, and d denotes the color channel and Φ denotes the Gaussian kernel. Then for the feature points in $\mathcal{F}_f$, if $P_f \leq P_b$, this point will be removed from $\mathcal{F}_f$. Similarly, for the feature points in $\mathcal{F}_b$, if $P_f \geq P_b$, then this feature point will be removed from $\mathcal{F}_b$. It should be noted that the feature points removed out from the $\mathcal{F}_f$ (or $\mathcal{F}_b$) are not added into $\mathcal{F}_b$ (or $\mathcal{F}_f$); they are all denoted as unknowns and will be assigned a label by the graph cut algorithm. The feature classification after eliminating the error becomes much better as shown in Figure 3(b).

3.3. Foreground Extraction with Geodesic Graph Cut. Till now, we have obtained the foreground and background key points. This means we have labeled partial pixels as foreground and background. Starting from the initial labeling, we can obtain a complete foreground segmentation by employing a geodesic graph cut algorithm [18], where we use the geodesic distance and color models to calculate the energy function of the graph cut algorithm, which is defined as

$$E(\mathcal{L}) = \sum_{x_i \in P} R(\mathcal{L}_i) + \lambda \sum_{(x_i, x_j) \in N} B(x_i, x_j) \left| \mathcal{L}_i - \mathcal{L}_j \right|, \quad (6)$$

where $\mathcal{L} = (\mathcal{L}_i)$ is a binary vector and $\mathcal{L}_i$ is the label $\mathcal{F}$ or $\mathcal{B}$ for pixel x_i. $R(\mathcal{L}_i)$ is a unary term and $B(x_i, x_j)|\mathcal{L}_i - \mathcal{L}_j|$ is the pairwise term of the energy function. λ is a weight to balance the unary and pairwise term. The unary term is defined as follows:

$$R_{\mathcal{L}_i}(x_i) = S_{\mathcal{L}_i}(x_i) + u(x_i) G_{\mathcal{L}_i}(x_i), \quad (7)$$

where $S_{\mathcal{L}_i}$ is a constraint for the foreground and background feature points:

$$S_{\mathcal{L}_i}(x_i) = \begin{cases} \infty, & \text{if } x_i \in \Omega_{\overline{\mathcal{L}}_i}, \\ 0, & \text{otherwise}, \end{cases} \quad (8)$$

where $\Omega_{\overline{\mathcal{L}}_i}$ indicates the foreground and background features and $\overline{\mathcal{L}}_i$ denotes the label opposite $\mathcal{L}_i$ (i.e., if $\mathcal{L}_i = \mathcal{F}$, then $\overline{\mathcal{L}}_i = \mathcal{B}$).

$G_{\mathcal{L}_i}(x_i)$ is computed by normalizing the relative foreground/background geodesic distances:

$$G_{\mathcal{L}_i}(x_i) = \frac{D_{\mathcal{L}_i}(x_i)}{D_{\mathcal{F}_i}(x_i) + D_{\mathcal{B}_i}(x_i)}, \quad (9)$$

where the geodesic distances from each pixel to the foreground and background feature points are computed efficiently by the method proposed by [19]. $u(x_i)$ is the geodesic confidence which is defined as

$$u(x_i) = \left| \frac{D_{\mathcal{F}}(x_i) - D_{\mathcal{B}}(x_i)}{D_{\mathcal{F}}(x_i) + D_{\mathcal{B}}(x_i)} \right|^2. \quad (10)$$

The pairwise term $B(x_i, x_j)$ is defined as

$$B(x_i, x_j) = \frac{1 + \left(u(x_i) + u(x_j)\right)/2}{1 + \left\| C(x_i) - C(x_j) \right\|^2}, \quad (11)$$

where $C(x_i)$ and $C(x_j)$ are pixel colors.

4. Experiment and Discussion

4.1. Test of Our Algorithm. We test our algorithm with many videos that were captured by freely moving cameras. Some results are shown in Figure 4.

It can be seen that the image alignment algorithm employed in our algorithm is very efficient, so that the moving object regions can be well detected by the frame difference algorithm as shown in Figure 4(b), and the feature points can be classified accurately as shown in Figure 4(c). From Figure 4(d), we can see that although we only constrain a small part of pixels (feature points), the labels can be correctly propagated to other nonfeature pixels, and the accurate segmentation can be obtained.

We test our algorithm on a laptop with four cores, 2.1 GHz CPU, and 8 G RAM. The image alignment step costs most of the computational time. However, we speed up this algorithm by using the affine transformation parameters obtained from the former pair of frames to initialize parameters of the current pair of frames. Thus the whole system can run in about 15 fps for videos with 320 × 240 size. For image alignment step, we suggest to downsample the image to a relatively small scale to estimate the affine transformation parameters. This not only can improve the speed, but also can help improve the accuracy. This is because for the large scale image, the background displacement may be very large, which may not be well estimated by the affine transformation models, though the affine estimation algorithm has already

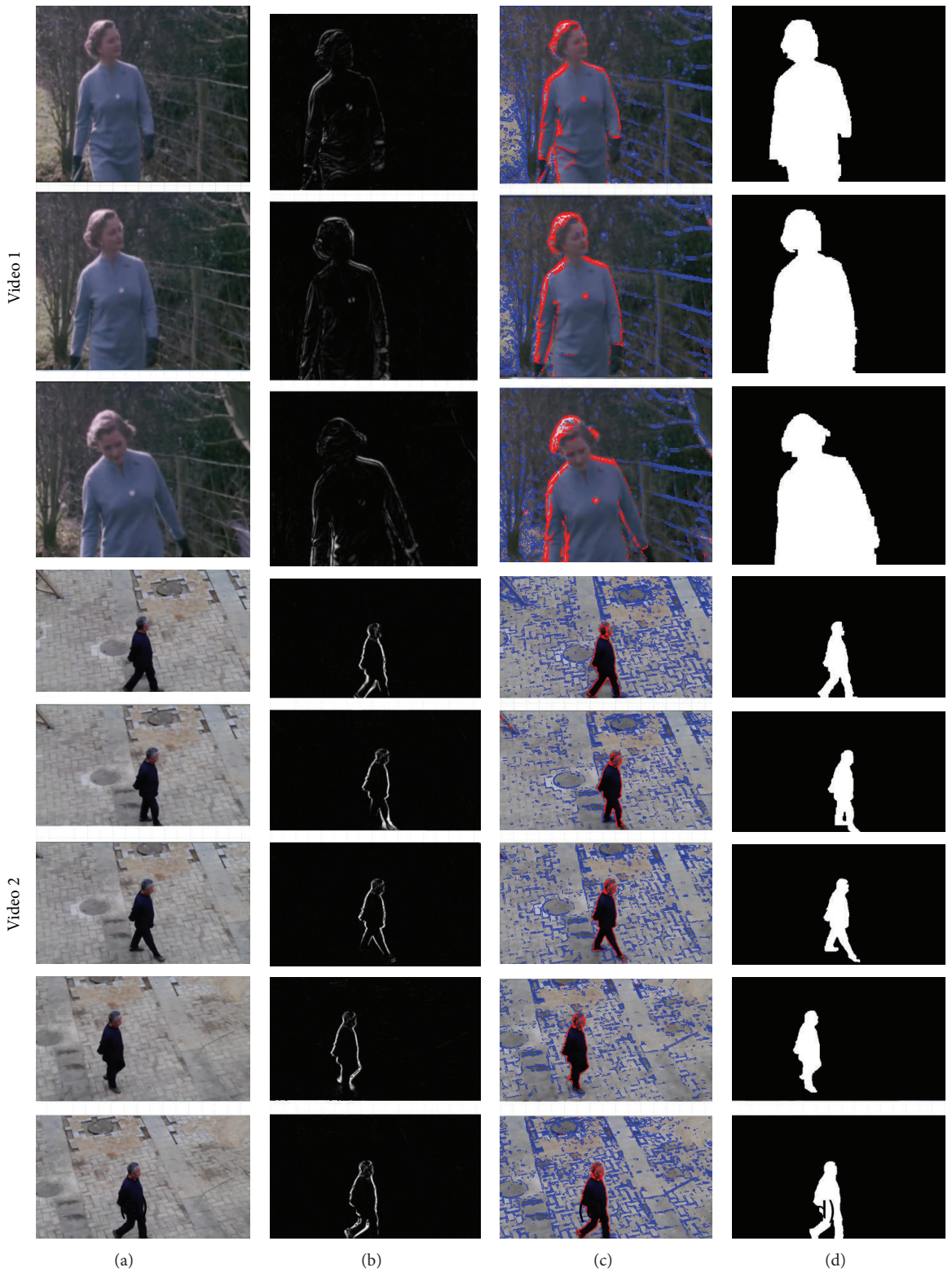
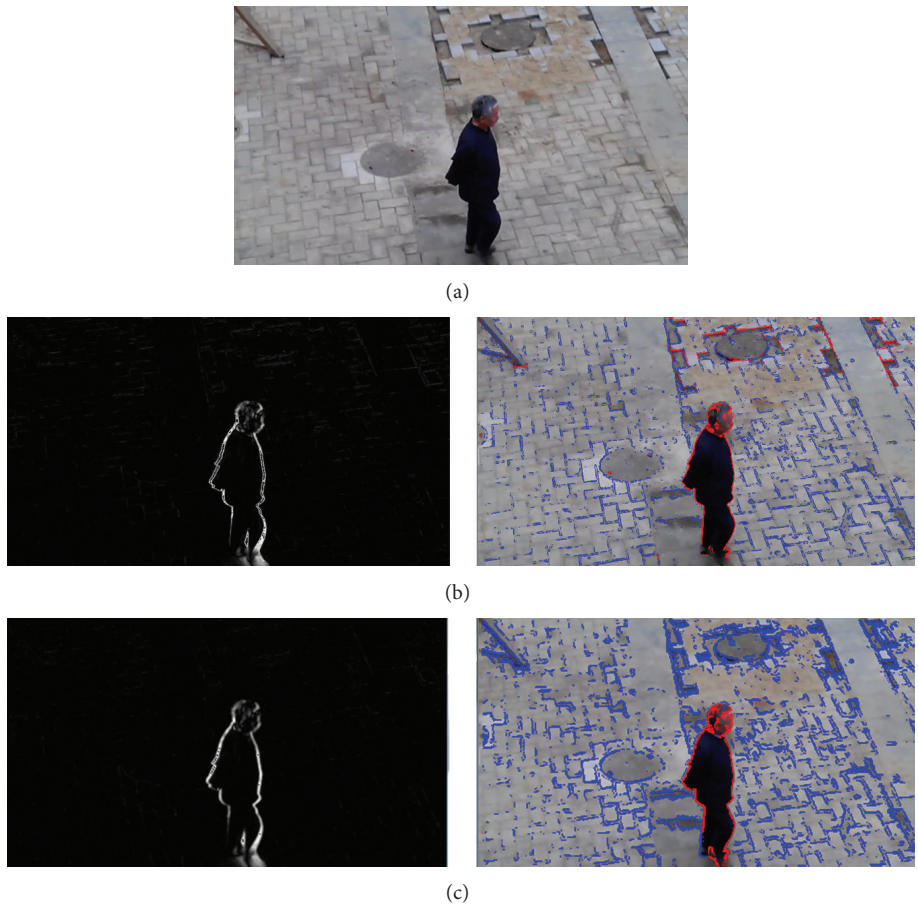


Figure 4: Foreground extraction results for videos captured by moving cameras. (a) shows the original frames, (b) shows the frame difference after the image alignment, (c) shows the feature extraction and classification, and (d) shows the binary mask of the moving object.



(a)

(b)

(c)

Figure 5: Image alignment results at different scales. (a) is the original current image, (b) is obtained at 640 × 480 scale, and (c) is obtained at 320 × 240 scale.

dealt with this problem by employing a pyramid model. Figure 5 shows a comparison of the image alignment results for the same pair of frames at different scales. As you can see, when aligning the images at 640 × 480 scale, the frame difference for the background regions is a little high, so that some background feature points are misclassified into foreground as shown in Figure 5(b). While as a comparison shown in Figure 5(c), aligning the images in 320 × 240 scale obtains much better results.

Our algorithm can also work for the static camera videos and can usually run in very high speed because the image alignment step can be removed. According to our test, our system can run in more than 25 fps on the laptop mentioned above. Figure 6 shows some result of applying our algorithm on the videos captured by static cameras.

4.2. Comparison with Existing Algorithms. We compare our algorithm with the three algorithms proposed recently [20–22]. The comparison results are shown in Figure 7. As can be seen, the results obtained by our method are clearly more accurate than the results obtained by Zhang et al. [20] and Zhou et al. [21]. Both Zhang's and Zhou's methods may mistakenly label the large area of foreground or background area. More concretely, our algorithm is more robust to the topology changes of the object, while Zhang's and Zhou's methods tend to erroneously segment the object when the topology of the object changes greatly. Our algorithm obtains comparable or even better segmentation results than the Papazoglou and Ferrari [22] method as shown in the last rows of Figure 7, which is reported outperforming the state-of-the-art algorithms [22]. The comparison results can be observed more clearly by the *Error Rate* evaluation of these algorithms. *Error Rate* evaluation is criteria commonly used for the accuracy comparison of the object segmentation algorithms, and it is defined as

$$\text{Error Rate} = \frac{\#\text{ mis-labeled pixels in frame } t}{\#\text{ all the pixels in frame } t}, \quad (12)$$

where "#" means "the number of". The error rate comparison results for each frame of the two test image sequences are shown in Figure 8. It can be seen that our results are better than the results obtained by [20] and Zhou et al. [21]. Although the results obtained by our algorithm are comparable with Papazoglou and Ferrari [22], our algorithm runs much faster than Papazoglou and Ferrari [22] as discussed in the following section.

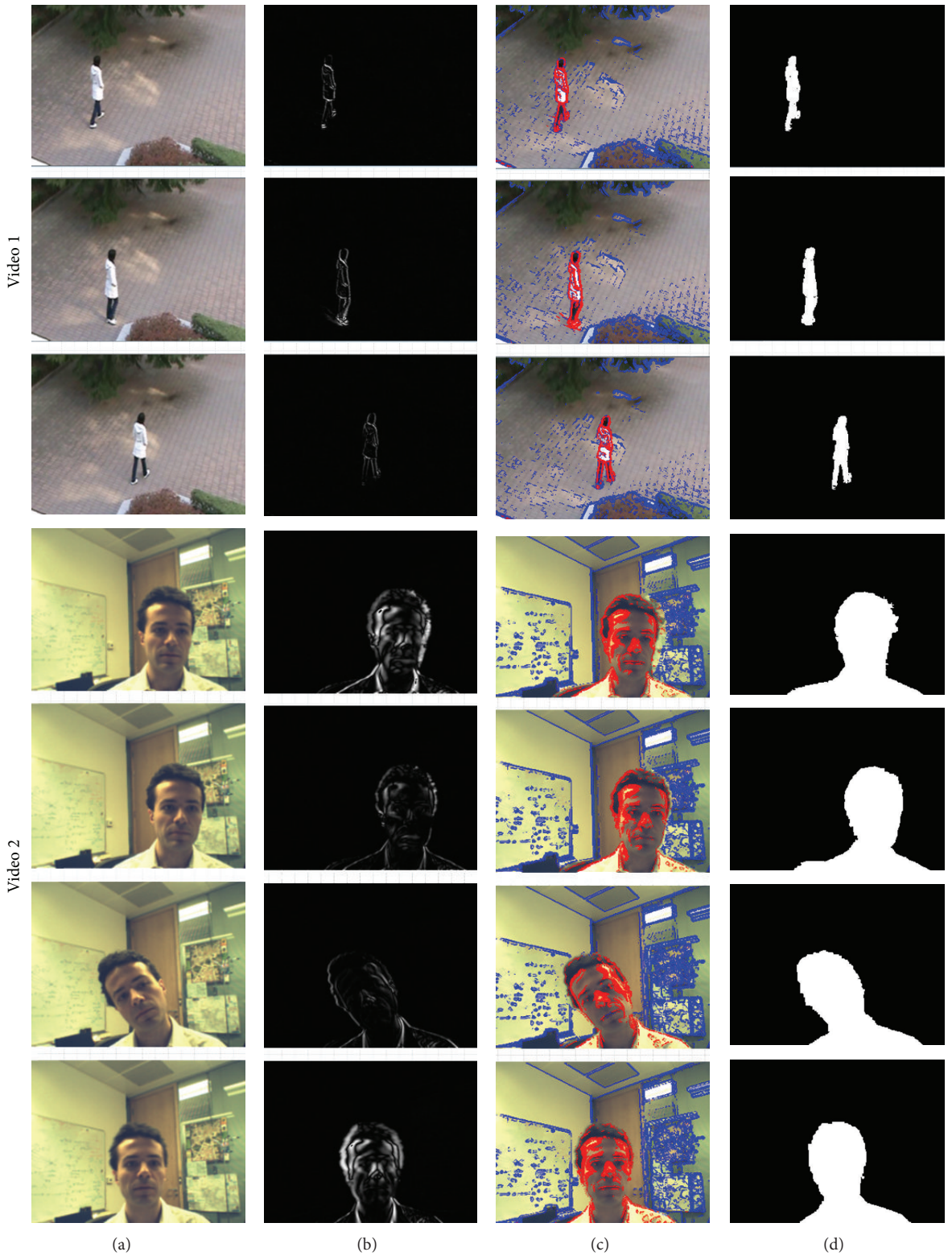


Figure 6: Foreground extraction for videos captured by static cameras.

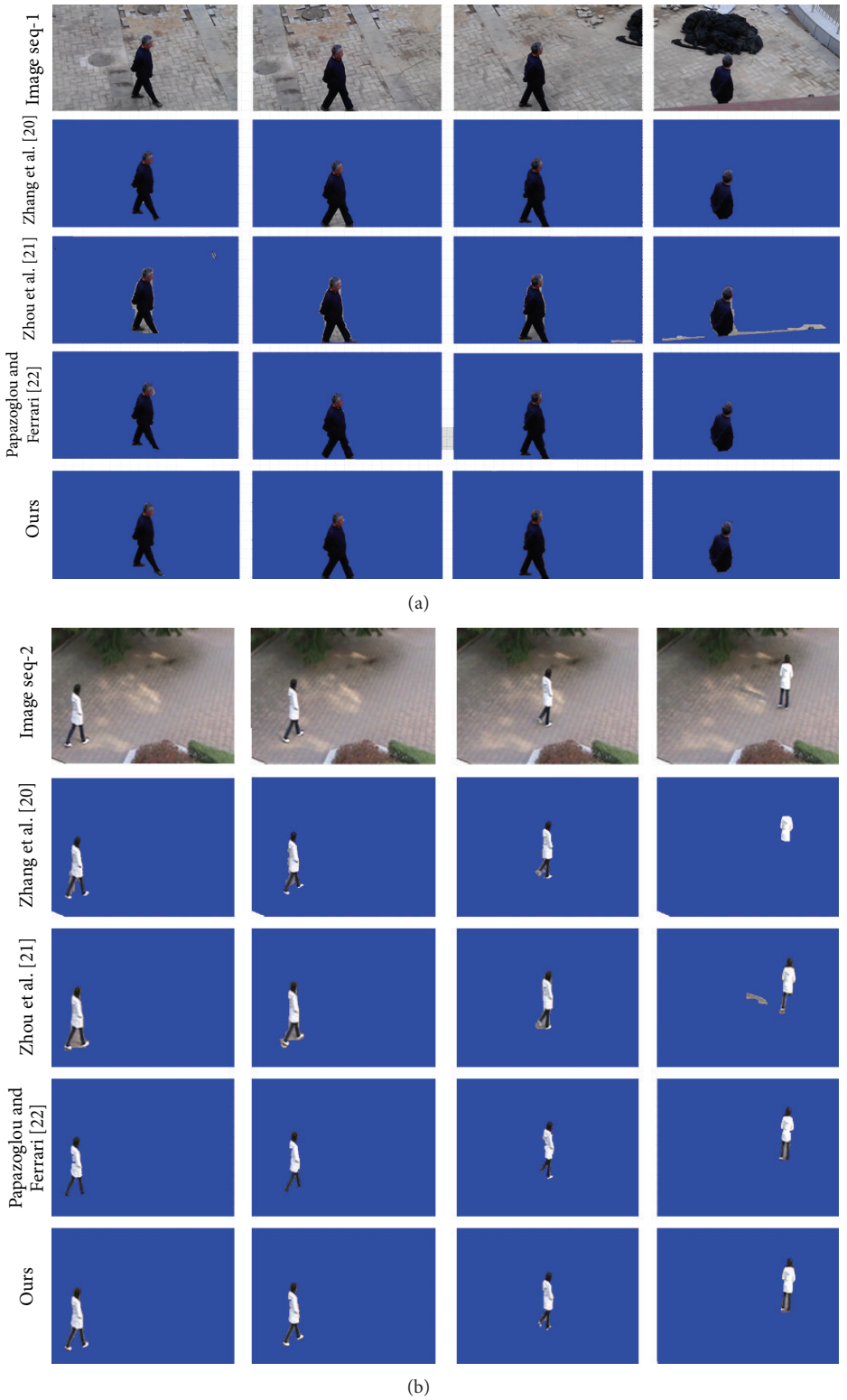
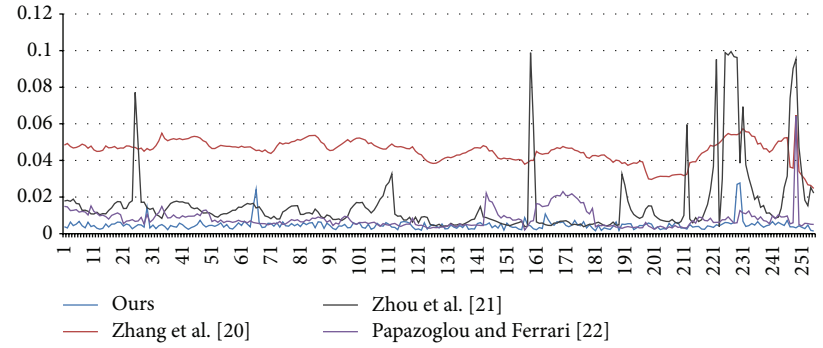
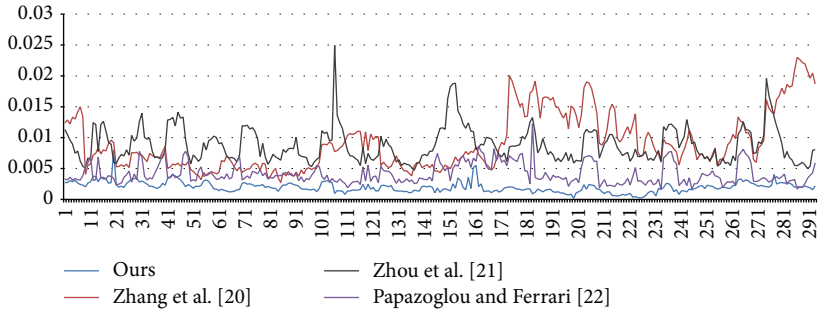


FIGURE 7: Comparison with the algorithm proposed recently by Zhang et al. [20], Zhou et al. [21] and Papazoglou and Ferrari [22]. For both (a) and (b), the top row shows some images of image sequence-1 and sequence-2, the following three rows show the foreground segmentation results obtained by the three existing algorithms, and the last row shows the results obtained by our algorithm.



(a) Error rate comparison on image sequence-1



(b) Error rate comparison on image sequence-2

FIGURE 8: *Error rate* comparison of our algorithm with Zhang et al. [20], Zhou et al. [21] and Papazoglou and Ferrari [22] on two tested image sequences.

4.3. Run Time. As described before, our algorithm can perform in a high speed, about 15 fps for the videos captured by nonstatic cameras and 25 fps for the videos captured by static cameras. The three existing algorithms [20–22] are all performed in a relatively low speed compared to our method. According to our test, Zhou's algorithm [21] takes more than 2 s in average to process a 320×240 image, which may take at lease 30 times longer time to process the same sequence compared to our method. Papazoglou and Ferrari [22] have reported that given optical flow and superpixels, their fast segmentation method only takes 0.5 s/frame, which is still slower than our algorithm. Furthermore, the optical flow and superpixels should also be calculated and will cost a long time. For example, in Papazoglou and Ferrari [22], they employ Brox's optical flow estimator [23], and it will take more than 7 s which is a very time consuming process. Zhang et al. [20] employ an even more complex method including optical flow estimation, GMM and EM algorithm, and a graph cut optimization algorithm. This costs even longer time, that is, more than 8.5 s/frame in our test.

5. Conclusion

This paper proposed a real time online moving object detection and segmentation algorithm for the video captured by freely moving cameras which only use two successive frames to segment the moving object. A two-layer iteration algorithm is proposed to accurately estimate the affine transformation parameters between two successive frames. A feature point detection and filtering algorithm is proposed to remove the error foreground and background feature points. The object is finally extracted by a geodesic graph cut algorithm. This algorithm is demonstrated to be very efficient for many videos. Compared to the existing long term key point trajectory based algorithm, our algorithm not only can perform in online processing mode, but also can run in high speed. This makes our algorithm very practical in many applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the China Postdoctoral Science Foundation (no. 2013M530020).

References

[1] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pp. 246–252, June 1999.

[2] A. Elgammal, D. Hanvood, and L. S. Davis, "Nonparametric model for background subtraction," in *European Conference on Computer Vision*, pp. 751–767, 2000.

[3] Y. Sun, B. Li, B. Yuan, Z. Miao, and C. Wan, "Better foreground segmentation for static cameras via new energy form and dynamic graph-cut," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, pp. 49–52, August 2006.

[4] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[5] K. Patwardhan, G. Sapiro, and V. Morellas, "Robust foreground detection in video using pixel layers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 746–751, 2008.

[6] F. Liu and M. Gleicher, "Learning color and locality cues for moving object detection and segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR '09)*, pp. 320–327, June 2009.

[7] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, pp. 4306–4312, October 2009.

[8] G. Zhang, J. Jia, W. Xiong, T. Wong, P. Heng, and H. Bao, "Moving object extraction with a hand-held camera," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, pp. 1–8, October 2007.

[9] P. Ochs and T. Brox, "Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 1583–1590, November 2011.

[10] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 1219–1225, October 2009.

[11] P. Sand and S. Teller, "Particle video: long-range motion estimation using point trajectories," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 72–91, 2008.

[12] M. J. Black and P. Anandan, "The robust estimation of multiple motions: parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[13] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *European Conference on Computer Vision*, pp. 282–295, 2011.

[14] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *European Conference on Computer Vision*, pp. 228–241, 2012.

[15] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994.

[16] J. Y. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description*, Technical Report for Intel Corporation Microprocessor Research Lab, Santa Clara, Calif, USA, 2000.

[17] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.

[18] B. L. Price, B. Morse, and S. Cohen, "Geodesic graph cut for interactive image segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 3161–3168, June 2010.

[19] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, October 2007.

[20] D. Zhang, O. Javed, and S. Mubarak, "Video object segmentation through spatially accurate and temporally dense extraction of primary object regions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 628–635, June 2013.

[21] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 597–610, 2013.

[22] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '13)*, pp. 1777–1784, June 2013.

[23] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011.