J. Vis. Commun. Image R. 24 (2013) 232–243

Contents lists available at SciVerse ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Moving foreground object detection via robust SIFT trajectories

Shih-Wei Sun [a,b], Yu-Chiang Frank Wang [c,d,*], Fay Huang [e], Hong-Yuan Mark Liao [d,f]

[a] Dept. of New Media Art, Taipei National University of the Arts, Taipei, Taiwan
[b] Center for Art and Technology, Taipei National University of the Arts, Taipei, Taiwan
[c] Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan
[d] Inst. of Information Science, Academia Sinica, Taipei, Taiwan
[e] Inst. of Computer Science and Info. Engineering, National Ilan University, Yi-Lan, Taiwan
[f] Dept. of Computer Science and Info. Engineering, National Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Article history:
Received 20 March 2012
Accepted 12 December 2012
Available online 21 December 2012

Keywords:
Template matching
Object tracking
Video object segmentation
Foreground segmentation
Background subtraction

ABSTRACT

In this paper, we present an automatic foreground object detection method for videos captured by freely moving cameras. While we focus on extracting a single foreground object of interest throughout a video sequence, our approach does not require any training data nor the interaction by the users. Based on the SIFT correspondence across video frames, we construct robust SIFT trajectories in terms of the calculated foreground feature point probability. Our foreground feature point probability is able to determine candidate foreground feature points in each frame, without the need of user interaction such as parameter or threshold tuning. Furthermore, we propose a probabilistic consensus foreground object template (CFOT), which is directly applied to the input video for moving object detection via template matching. Our CFOT can be used to detect the foreground object in videos captured by a fast moving camera, even if the contrast between the foreground and background regions is low. Moreover, our proposed method can be generalized to foreground object detection in dynamic backgrounds, and is robust to viewpoint changes across video frames. The contribution of this paper is trifold: (1) we provide a robust decision process to detect the foreground object of interest in videos with contrast and viewpoint variations; (2) our proposed method builds longer SIFT trajectories, and this is shown to be robust and effective for object detection tasks; and (3) the construction of our CFOT is not sensitive to the initial estimation of the foreground region of interest, while its use can achieve excellent foreground object detection results on real-world video data.

1. Introduction

Detecting moving foreground objects from a video taken by a non-stationary camera attracts intensive attention from researchers and engineers in the field of image and video processing. This is of particular interest for applications such as action and event recognition, and automatic annotation of videos. However, moving foreground detection has been a challenging task since the moving foreground object in real-world videos is often highly articulated or even non-rigid. Without prior knowledge (e.g., training data) on the foreground object of interest, it is difficult to model the object information even with user interaction. In practice, the camera is not fixed and thus conventional object detection methods based on frame differences cannot be applied, which makes background modeling approaches fail. In [1], Patwardhan et al. pointed out the three scenarios which make video foreground object detection very difficult. The first scenario is the presence of complex background which contains moving components such as water ripples or swaying trees. The second case is background motion caused by camera motion (e.g., shaky tripod in windy days), which rules out the use of conventional reconstruction-based approaches for object detection. Finally, most existing works for video object detection require training data or user interaction (e.g., at the first frame). This might not be practical and will result in increased processing time.

1.1. Related work

The history of video-based object detection starts from detection of moving objects in videos captured by a stationary camera. Jain and Nagel [2] proposed the frame difference scheme to detect a foreground object. Wren et al. [3] proposed the use of a Gaussian model, Stauffer and Grimson [4] proposed a GMM-based approach, and Elgammal et al. [5] applied kernel density estimation for background modeling. Unfortunately, the above methods cannot serve

* Corresponding author at: Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan.
E-mail addresses: swsun@newmedia.tnua.edu.tw (S.-W. Sun), ycwang@citi.sinica.edu.tw (Y.-C.F. Wang), fay@niu.edu.tw (F. Huang), liao@iis.sinica.edu.tw (H.-Y.M. Liao).

1047-3203/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jvcir.2012.12.003

well for scenarios in which the camera is moving (even with nominal motion). Recent researchers focus more on foreground object detection in videos captured by freely moving cameras. In [6], Sheikh and Shah proposed to build foreground and background models using a joint representation of pixel color and spatial structures between them. In [1], Patwardhan et al. decomposed a scene and used maximum-likelihood estimation to assign pixels into layers. From their experimental results, only moving foreground objects with the average velocity up to 12–15 pixels per frame can be detected. As a result, their approach is only capable of handling videos captured by a camera with mild camera motions.

In this paper, we address automatic video foreground object detection problems under arbitrary camera motion (e.g., panning, tilting, zooming, and translation). Prior methods focusing on this type of problem can be classified into two categories. The first category (e.g., Meng and Chang's method [7]) is to detect moving foreground object as the outliers, and thus to estimate the global motion of the camera [8]. Irani and Anandan [9] proposed a parametric estimation method for detecting the moving objects, and Wang et al. [10] also approached this problem in a similar setting. Furthermore, Bugeau and Perez [11] proposed motion and feature clustering techniques for estimating foreground object regions. The second category of moving object detection algorithms aims to model the reference background image. Felip et al. [12] proposed to estimate the dominant motion from the sampled motion vectors. Zhao et al. [13] proposed to detect objects present or removed from a non-static camera for indoor scenes based on the calculation of SIFT features [14] and homography. While it is possible to model the scene as background information for videos captured from an indoor scene or a closed area scene, modeling outdoor scene or more complicated background remains a very difficult problem.

The correspondences of feature points are widely used for linking the relationship between pairs of video frames. SIFT flow is recently proposed by Liu et al. [15] to determine the dense correspondences between image pairs for the retrieval of similar scenes. On the other hand, Sand and Teller [16] proposed a particle video approach, which is able to construct the long trajectory based on the optical flow correspondences, and thus provides more chances to detect and track foreground objects. We note that, although the method of Liu et al. [15] is able to determine dense corresponding SIFT points, it would be impractical to enforce the trajectory across all video frames, which results in linking SIFT points in dissimilar pairs of video frames. While the approach of Sand and Teller [16] better links corresponding particle points, its high computational cost would prohibit future speed-up or higher-level processing or learning tasks. In [17], a motion-flow based approach was proposed to analyze MPEG bitstreams for moving objects using the associated trajectories. Motivated by the above methods, we advance the context and spatiotemporal information of moving foreground objects, and we advocate the use of the trajectories to provide rich information in detecting foreground objects in videos.

1.2. Our contributions

We present a novel foreground object detection approach in this paper. We focus on detecting and tracking a single and dominant foreground object in uncontrolled videos, i.e., videos captured by freely moving cameras or those downloaded from the Internet. Based on the SIFT matching strategy, we calculate the foreground feature point probability for constructing robust SIFT trajectories, which imply the foreground candidate region across video frames without the need of user interaction such as parameter or threshold tuning. To perform foreground object detection, we propose a consensus foreground object template (CFOT) based on the extraction and association of SIFT points with longer trajectories and higher confidence. We note that our CFOT is not only derived by integrating the information of the candidate foreground regions in a probabilistic way, we also advance an adaptive re-start scheme to handle false object detection results when tracking the foreground object. This makes our CFOT more robust in detecting objects in real-world uncontrolled videos, and thus we are able to extract and track the foreground objects even when the contrast between the foreground and background regions is low (spatially or temporally). This is why our proposed method can be generalized to videos with dynamic backgrounds, and is robust to viewpoint changes across video frames.

The contribution of our proposed method is trifold: (1) we provide a probabilistic self-decision framework to determine the moving foreground objects in videos, while no user interaction or parameter tuning is required; (2) the extracted SIFT points across video frames allow us to associate the candidate foreground interest points and to calculate the foreground feature point probability for robust object detection; and (3) our CFOT results in a compact representation of the foreground object of interest, while the construction of CFOT is not sensitive to the initial estimation of foreground region of interest due to the re-start mechanism when necessary. From our experimental results, it can be verified that the use of our CFOT produces excellent foreground object detection results in real-world video data.

2. Foreground object detection

Fig. 1 shows the proposed framework for video foreground object detection. This framework consists of two steps: the construction of CFOT and the use of CFOT for foreground object detection, which will be discussed in Sections 2.1 and 2.2, respectively.

2.1. Construction of consensus foreground object template (CFOT)

Fig. 2 depicts the process for constructing the consensus foreground object template (CFOT) for detection purposes. We now detail each step in this subsection.

2.1.1. Foreground key point extraction

Scale-invariant feature transform (SIFT) [14] is a popular computer vision algorithm, which can be used to detect local interest points in an image. As an initial stage of our foreground feature point extraction, we apply the SIFT feature detector in each frame of a video sequence. The goal for this step is to obtain a set of foreground key points which most likely belong to the foreground object of interest, which is achieved by identifying the SIFT key points across video frames in a probabilistic point of view, as we discuss below.

As the initialization stage of this step, a new key point p_i^t is detected as a SIFT point for the first time at time t, and its corresponding probability $f_{point}(i,t)$ will be set as $\alpha = 0.5$ since we have no prior knowledge that whether this key point belongs to foreground or background. The foreground point probability function is defined as:

$$f_{point}(i,t) = \begin{cases} f_{point}(i,t-1)\cdot\lambda + 1\cdot(1-\lambda), & \text{if } p_i^t \neq \emptyset; \\ f_{point}(i,t-1)\cdot\lambda, & \text{if } p_i^t = \emptyset, \end{cases} \quad (1)$$

where λ is an update factor and is set to 0.95 as suggested in [18].

The above equation provides a probabilistic way to update the probability of assigning an extracted key point as foreground, depending on its key point matching history. To be more precise, if a SIFT key point is consistently identified across video frames (by SIFT matching), it is more likely to belong to the foreground object and thus a higher probability value will be assigned. Fig. 3a

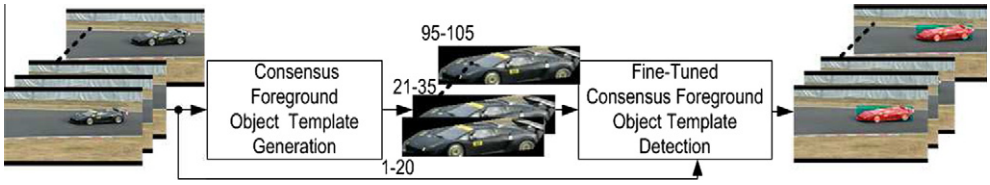

Fig. 1. Our proposed framework for video foreground object detection.

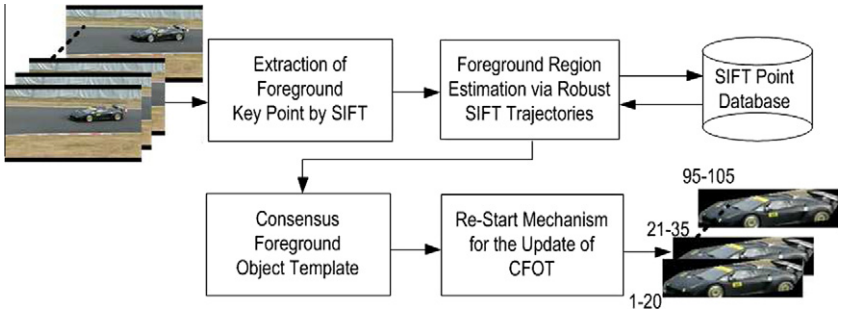

Fig. 2. The flow chart for the construction of the consensus foreground object template (CFOT).

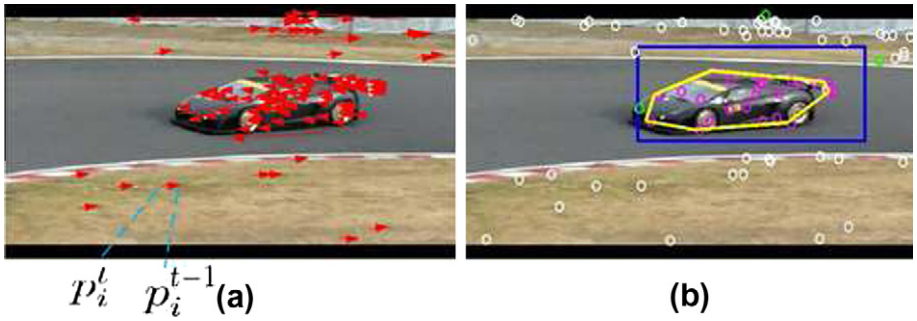

Fig. 3. SIFT points and the foreground region example: (a) corresponding pairs of SIFT feature points between video frames t and $t-1$, and (b) pink circles: foreground SIFT points p_{FG}^t in R_0^t; green circles: foreground SIFT points p_{FG}^t out of R_0^t; white circles: the rest SIFT points having/without correspondence matching; blue rectangle: the rectangular region, R_0^t (bounded by the blue rectangle), defined by its upper-left and lower-right corners, i.e., $(\bar{x}-2\sigma_x, \bar{y}-2\sigma_y)$ and $(\bar{x}+2\sigma_x, \bar{y}+2\sigma_y)$, and yellow polygon: candidate foreground region, R^t, calculated by the convex hull operation from p_{FG}^t locate in the region R_0^t (pink circles). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

shows an example of SIFT point matching correspondences (by red arrows) between two adjacent video frames.

After calculating the probability f_{point} for all SIFT points across video frames, the profile of f_{point} for each SIFT key point can be derived as illustrated in Fig. 4. We note that the x-axis in this figure indicates the frame number t, the y-axis is the index i of SIFT feature points, and the vertical axis denotes the values of the calculated f_{point}. From our observations, a large portion of the key points extracted from the foreground object will be presented in the field of view across video frames, and thus will be associated with higher probability values. This implies that the extracted key points belong to the foreground object of interest.

Once the foreground probability profile of each SIFT keypoint is obtained, we collect a set p_{FG}^t of foreground key points p_i^t at time t, which is defined as follows:

$$p_{FG}^t = \{p_i^t : p_i^t \neq \emptyset \text{ and } f_{point}(i,t) > max((\mu^t - \sigma^t), \alpha)\}, \quad (2)$$

In (2), the threshold value is set to be $max((\mu^t - \sigma^t), \alpha)$, where μ^t and σ^t are the mean and standard deviation of the key point distribution. The use of this data-driven threshold allows us to avoid the case of $(\mu^t - \sigma^t) < \alpha$, i.e., to consider a key point whose

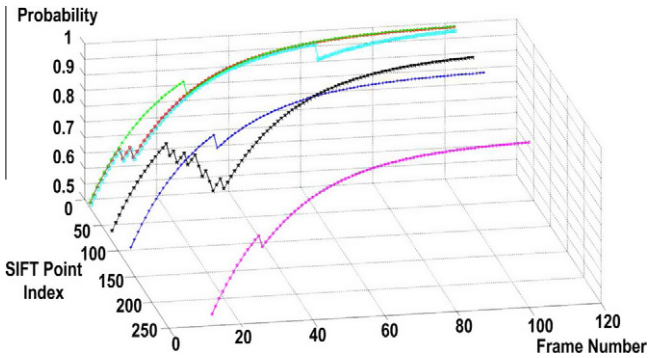

Fig. 4. An example of foreground point probability, the SIFT point correspondence is matched according to [14]. The pink curve is the $f_{point}(i,t)$ with SIFT point index 230. The point occurred as a foreground point at frame 15 and the initial value $\alpha = 0.5$ is given. At frame 30, the corresponding point cannot be obtained to cause the curve value decrease. After that time instant, the corresponding points continuously occurred, keeping the curve increasing. The curve increasing and decreasing situations can also be found from the other curves (different index i) belonging to $f_{point}(i,t)$.

$f_{point}(i,t) < \alpha = 0.5$ and thus is likely to be considered as background.

2.1.2. Foreground region estimation via robust SIFT trajectories

With the above probabilistic way to extract foreground feature points in a video sequence, the set p_{FG}^t is obtained and the location of each foreground key point is denoted as (x_i^t, y_i^t) (i is the index of the key point, and t is the frame number). We note that, when a key point is observed across video frames, its trajectory can be constructed by connecting the key points of interest. Fig. 5a shows example foreground key points (denoted in circles), and Fig. 5b shows the corresponding trajectories, which are generated by connecting corresponding foreground key points with line segments of the same color.

We define the trajectory segment between key points p_i^t and p_i^k between tth and kth frames, noted as $s_i^{t,k} := \overline{p_i^t p_i^k}$, where $k > t$ and both p_i^t and p_i^k belong to the set p_{FG}^t. When piecewise trajectory segments are established for each foreground key point, a SIFT trajectory can be denoted as:

$$s_i = \left\{ s_i^{t,k} : \forall s_i^{t,k}, \text{ where } t, \quad k \in \{1,2,\ldots,T\} \text{ and } k > t. \right\}. \tag{3}$$

Using this technique, we do not need to limit the length of the constructed trajectory for a particular key point. In prior methods, such as SIFT flow [15] and particle video [16], a trajectory will be terminated (or a particle will be eliminated) if a corresponding key point cannot be found across a certain time duration. Unlike these approaches, if the correspondence between the detected key points can be found across any number of frames (using the probabilistic approach discussed earlier), our proposed framework is still able to construct the trajectory segment between them. As a result, we are able to produce long and robust trajectories due to performing SIFT matching in our probabilistic way instead of explicitly key point matching.

Now, to determine the candidate foreground region from the derived trajectories, we calculate their mean location $\overline{p} = (\overline{x}, \overline{y})$ of the key point collection p_{FG}^t at frame t. We further normalize the key points in p_{FG}^t and form a Gaussian distribution with zero mean and standard deviations σ_x and σ_y. From the definition of standard variation, about 95% of the normalized key points are within the range $[-2\sigma_x, 2\sigma_x]$ or $[-2\sigma_y, 2\sigma_y]$. This implies that about 90% of the key points lie within the rectangle defined by its upper-left and lower-right corners, i.e., $(\overline{x} - 2\sigma_x, \overline{y} - 2\sigma_y)$ and $(\overline{x} + 2\sigma_x, \overline{y} + 2\sigma_y)$, denoted by R_0^t as a rectangular region. As a result, we use the foreground points in p_{FG}^t locate within the region R_0^t to generate a candidate foreground region

$$R^t = \left\{ C(p_i^t \in p_{FG}^t, \text{ and } p_i^t \text{ locates in the region } R_0^t) \right\}, \tag{4}$$

where $C(\cdot)$ represents the convex hull operation [19]. The corresponding R_0^t (shown in a blue rectangle) and R^t (the yellow convex hull region) are shown in Fig. 3b.

While we aim at using these foreground regions to perform the foreground object detection, the simple use of appearance (i.e., SIFT) and motion (i.e., trajectory) information along will not be sufficient to practical foreground object detection and tracking problems (see Fig. 6a and b for example). In the following subsection, we will explain how a consensus foreground object template (CFOT) is constructed based on the above candidate foreground regions. We will discuss why it is preferable to prior object detection methods in real-world uncontrolled video data.

2.1.3. Consensus foreground object template

Given the candidate foreground region R^t, we define the foreground object probability, which indicates how likely a pixel within R^t belongs to foreground. Similar to (1), this probability function is calculated as:

$$f_{object}^t(x,y) = \begin{cases} f_{object}^{t-1}(x - \triangle x^t, y - \triangle y^t) \cdot \lambda + 1 \cdot (1-\lambda), & \text{if } (x,y) \text{ in } R^t; \\ f_{object}^{t-1}(x,y) \cdot \lambda, & \text{otherwise}, \end{cases} \tag{5}$$

where (x,y) is the pixel location. $\lambda = 0.95$ is the update factor. At the starting frame of a video sequence, we set $f_{object}^t(x,y) = 0.5$ for all pixels (x,y), since we do not have any prior knowledge of the location of the foreground object. In (5), $\triangle x^t$ and $\triangle y^t$ are the x and y components of the motion vector calculated by sum of absolute difference (SAD) [20] between R^t and R^{t-1}, which will be applied to compensate the shift effects of the foreground object across frames. In practical scenarios, both the foreground object and the camera can move freely, and the location of the foreground candidate region R^t may shift across frames. Thus, the location information will

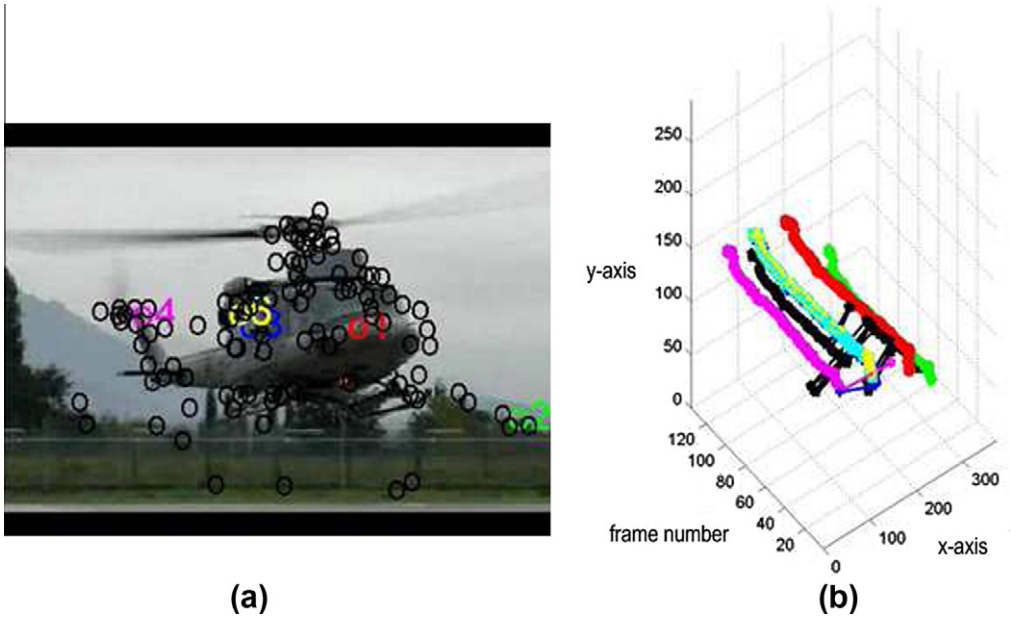

Fig. 5. SIFT trajectory: (a) the detected SIFT feature points at time t, and (b) the SIFT trajectories in a spatio-temporal space.

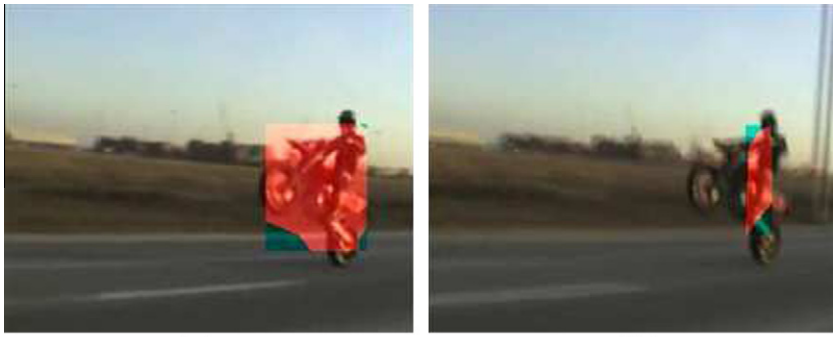

Fig. 6. Foreground region detection examples: (a) a satisfactory foreground region detection result, and (b) an inappropriate foreground region detection result.

Fig. 7. An image example of (a) foreground object probability model $f^t_{object}(x,y)$, (b–d) average foreground image $\overline{I}^t(x,y)$ of frame 82, 84, and 86, respectively, (e) the pixels with $f^t_{object}(x,y) \geqslant \theta^T$, and (f) the associated CFOT.

be calibrated by the calculated foreground region motion vector. An example of our foreground object probability model is shown in Fig. 7a, where we can see that the center part of the image pixels have larger probability values (brighter pixels) and thus are more likely in the foreground object region.

Since our goal is to construct a foreground object template for detection and tracking, we further utilize R^t and construct its averaged foreground image model $\overline{I}^T(x,y) = \left\{\sum_{t=1}^{T} I^t(x,y)\right\}/c^T_{map}(x,y)$. This model can be considered as an average image up to the Tth frame that *accumulates* foreground object pixel information from the starting to the current frame. For normalization purposes, a counter map $c^T_{map}(x,y)$ in the denominator counts the number of frames (out of T) that a pixel belongs to the foreground region. As a result, each foreground object pixel contributes $\frac{1}{c^T_{map}(x,y)}$ of its value to the final average foreground image. An example of the derivation of such an average foreground image model is shown in Fig. 7b–d.

After foreground object probability f^T_{object} calculated from $t = 1$ to $t = T$, and the averaged foreground image $\overline{I}^T$ are produced for a given video sequence, we integrate these two models to construct the CFOT for the foreground object of interest. According to (5), the value of $f^t_{object}(x,y)$ indicates the probability of a pixel at (x,y) belongs to foreground, an *adaptive* threshold $\theta^T = \lambda \cdot \theta^{T-1} + (1-\lambda)\cdot \eta^t$, is introduced to determine whether each pixel should be included to the formulation of our CFOT. To automatically determine the threshold θ^T, we have $\eta^t = 1$ for half of (T-1) frames and $\eta^t = 0$ for the rest (e.g., we set $\eta^t = 1$ for odd frames and $\eta^t = 0$ for even frames). This adaptive threshold θ^T only depends on the length T of the video, and the random assignment for $\eta^t = 1$ with 0 and 1 patterns would not affect the final value of θ^T. As a starting condition, we set the threshold $\theta^1 = 0.5$ for $T = 1$, and λ is the update factor 0.95 (same as (5)). The calculation of θ^T implies that this threshold is dependent on the length of the video sequence and the foreground object present in it.

Finally, we apply the averaged foreground image $\overline{I}^T$ derived earlier and use (5) to construct our CFOT as follows

$$CFOT(x,y) = \left\{\overline{I}^T(x,y) : \forall (x,y) \text{s.t. } f^T_{object}(x,y) \geqslant \theta^T\right\}. \quad (6)$$

From the above equation, we see that our CFOT is refined by the foreground pixel model $\overline{I}^T(x,y)$ with an adaptive filtering threshold θ^T. Once the CFOT is constructed, it can be used for foreground object detection and tracking, as we discuss later in Section 2.2. Fig. 7d–f illustrate an example of $\overline{I}^t(x,y)$, pixels with $f^t_{object}(x,y) \geqslant \theta^T$, and the resulting CFOT of a particular frame of a video sequence.

2.1.4. Re-start mechanism for updating CFOT

In practice, the appearance, scale, and illumination of a moving object can vary significantly throughout the video (see Fig. 8 for example). Under these severe variations, the aforementioned foreground probability models will not be sufficient to describe the object of interest across video frames. In order to address these problems, a re-start scheme of constructing updated CFOTs will be necessary.

As seen in Fig. 8, poor detection result will be obtained using the derived CFOT under significant changes in object motion, scale and appearance, or due to insufficient resolution of the input video. Under these circumstances, the mean location $(\overline{x},\overline{y})$ and variance (σ^2_x,σ^2_y) of foreground pixels p^t_{FG} in R^t (see (4)) at time t can be calculated. The locations of the foreground object may change between adjacent frames, however, (σ^2_x,σ^2_y) should not vary too much unless there is a significant change in scale or appearance. Based on this observation, we use the variations (σ^2_x,σ^2_y) as one of the factors to decide whether the generated CFOT is still suitable for object detection in subsequent frames. Another example is that, with insufficient video resolution, the SAD between consecutive R^t will be large (e.g., the local maximum value shown in Fig. 9). Similarly, if the appearance, scale, and illumination vary significantly across some video frames, this SAD value will also be larger than those from other consecutive frames. Therefore, we use the SAD value as another factor to indicate the quality of our constructed CFOT model at time t.

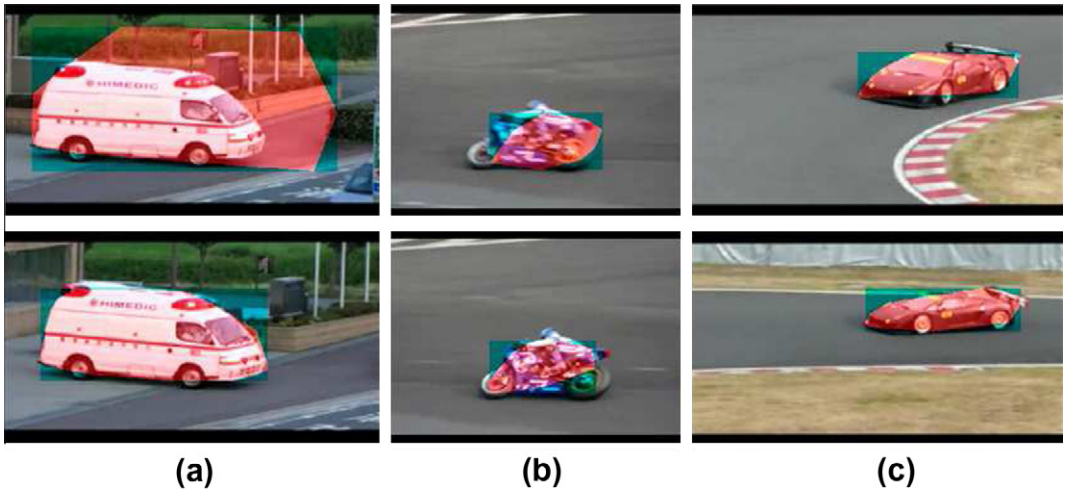

Fig. 8. Examples when the re-construction of CFOT is needed. The comparison from the top row to the bottom row shows the examples of: (a) too much background included in CFOT in the top, (b) blurring effects observed in CFOT in the top, and (c) significant scale and appearance variations within CFOT from the top to the bottom.

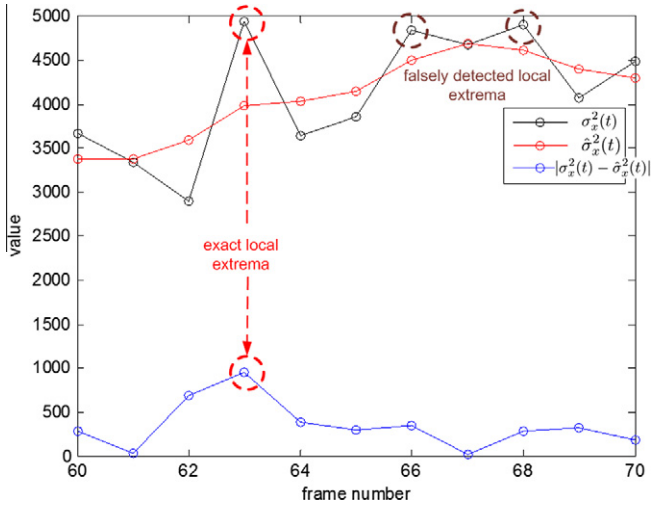

Fig. 9. Re-start frame detection example: $\sigma_x^2(t), \hat{\sigma}_x^2(t)$, and $|\sigma_x^2(t) - \hat{\sigma}_x^2(t)|$ are shown as the black, red, and blue curves, respectively. The falsely detected local extremas of $\sigma_x^2(t)$ are marked as the brown dashed line circles. The exact local extrema position is found by the peak detection in the blue curve $|\sigma_x^2(t) - \hat{\sigma}_x^2(t)|$, and then identify to the black curve $\sigma_x^2(t)$ position (red dashed line circle). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To summarize the above observations, we use the local maximum values observed in σ_x^2, σ_y^2, and SAD to determine whether we should re-start the entire CFOT generation process, including the reset of all probability values in each stage. In other words, at time instant t, the observation for σ_x^2, σ_y^2, and SAD are listed in the following indicating vector:

$$[r_{\sigma_x}, r_{\sigma_y}, r_{SAD}] = local\ \text{argmax}_t\{[|\sigma_x^2(t) - \hat{\sigma}_x^2(t)|, |\sigma_y^2(t) - \hat{\sigma}_y^2(t)|, |SAD(t) - \hat{SAD}(t)|]. \quad (7)$$

In order to avoid the detection of local maximum caused by noise, we further smooth the σ_x^2, σ_y^2, and SAD curves with a Gaussian kernel and obtain the smoothed versions of $\hat{\sigma}_x^2, \hat{\sigma}_y^2$, and $\hat{SAD}$. As a result, the difference between the original curves and the Gaussian smoothed curves can be calculated, and the resulting local maximum values indicate the time instants which we have poor CFOT results and should re-start our process accordingly, as shown in (7).

An example of the curves of $\sigma_x^2(t), \hat{\sigma}_x^2(t)$, and $|\sigma_x^2(t) - \hat{\sigma}_x^2(t)|$ are depicted in Fig. 9. From this figure, we see that $\hat{\sigma}_x^2(t)$ (red curve) is a slowly-varying version of $\sigma_x^2(t)$ (black curve), and the exact local peak (frame 63) can be detected from $|\sigma_x^2(t) - \hat{\sigma}_x^2(t)|$ to avoid the ripple problem (frame 66 and 68 of the black curve $\hat{\sigma}_x^2(t)$) to cause false peak detection. Finally, the re-start points can be determined as the union of the peak indexes ($[r_{\sigma_x}, r_{\sigma_y}, r_{SAD}]$) of the curves. This re-start mechanism will be able to handle practical foreground object detection problems due to obvious appearance variations, including changes in size and resolution.

2.2. Foreground object detection using CFOT

We now discuss how we apply the CFOT to perform object matching in a video for foreground object detection and tracking. Recall the flow chart of our proposed framework shown in Fig. 1, the CFOT is used as a query image over a number of video frames, and this CFOT will look for similar image patterns in each frame within this period of time. In order to determine the most similar image pattern, a similarity test based on SAD is performed to exhaustively search for a region in each frame which best matches the CFOT. We note that SAD is also commonly adopted for block matching in MPEG standards.

The calculation of SAD between a CFOT and a video frame t is defined as follows:

$$(x_c, y_c) = \text{argmin}_{x,y} \left\{ \sum_{w=0}^{W-1} \sum_{h=0}^{H-1} |CFOT(w,h) - I^t(w+x, h+y)| \right\}. \quad (8)$$

where $CFOT$ is the average foreground pixel model calculated from Eq. (6), the size of the CFOT is $W \times H$ (width by height). It is worth noting that we may have multiple CFOTs generated for a given video throughout the entire video sequence (due to the reasons explained in the previous subsection), but there is only one CFOT over a particular period of time. From Eq. (8), it is concluded that the smallest SAD output indicates the best matched foreground object region, and thus the upper-left corner of this region will be recorded by (x_c, y_c). Once this foreground object region is determined, as the completion of the foreground detection process, we also use a red masking polygon to mark the foreground region, as shown in the output stage of Fig. 1.

3. Experimental results

To test the effectiveness of our proposed method, we collect a set of video sequences from YouTube [21] as our video database, which contains videos of six different object classes: *airplane*,

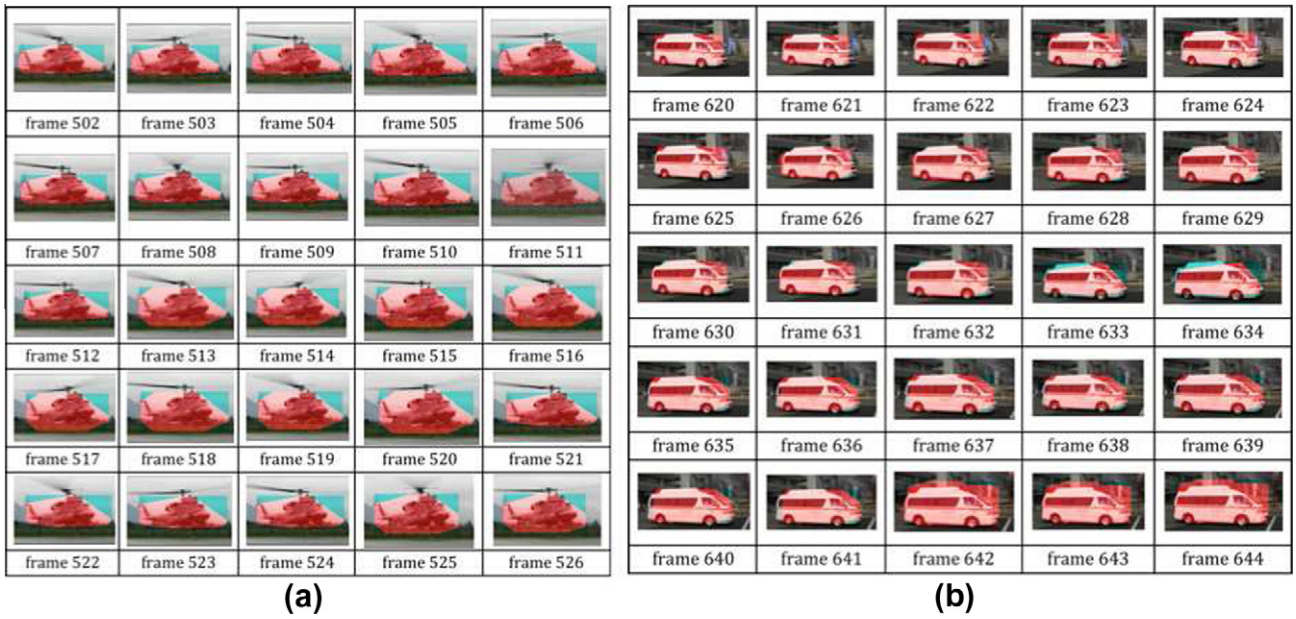

Fig. 10. Example detection results for videos captured by slow-moving cameras: (a) consecutive frames of *helicopter*, frame 502–526, and (b) consecutive frames in *ambulance*, frame 620–644.

ambulance, *car*, *fire engine*, *helicopter*, and *motorbike*. We have eight videos available for each class, and each video contains a moving object captured in a close-up scene and with significant variations in camera motion, background, lighting, etc. In this database, we selected total *5304* frames for evaluation. For the *48* video clips, the average length of the selected video clips is *110.5* frames (only the close-up video clips focusing on a moving object in a representative time period are selected for evaluation).

3.1. Example results of video object detection and tracking

To visualize the results produced by each step of our approach, we choose two video sequences (one from *ambulance* and the other from *fire engine*) for detailed discussions, as shown in Fig. 11. In the first row of Fig. 11, we show the correspondence of SIFT points p_i^t by circles in white, pink, and green.[1] The white circles are the SIFT points presented in the current frame but are not considered as the foreground points due to low probability values. The green ones are the foreground points p_{FG}^t out of R_0^t (blue rectangular region, defined in (4)). The pink circles are the final foreground points in R_0^t. In addition, the yellow polygon is the convex hull generated by the pink circles (i.e., R^t defined in (4)). With the above information, we update the foreground object probability f_{object}^t accordingly (see the second row), accumulate the averaged foreground image model $\bar{I}^t$ (i.e., the third row), and obtain the final CFOT (shown in the fourth row) by (6) for detection purposes. Finally, we apply the CFOT to detect the foreground moving objects (as discussed in Section 2.2), and the results are shown by red regions in the last row of Fig. 11. From Fig. 11b, we see that the foreground object cannot be perfectly detected. The yellow polygon in the first row of Fig. 11b excludes a major part of the fire engine. However, our derived CFOT can recover/compensate the foreground area and provide satisfactory foreground object detection results, as shown by the last row of Fig. 11b.

In our experiments, some video sequences are captured by slow-moving cameras. For example, some helicopter sequences contain the foreground object moving slowing from the ground, and thus both foreground and background exhibit slow motion. Some example frames and the corresponding detection results are shown in Fig. 10a. Other example videos can be observed from the ambulance sequences. As shown in Fig. 10b, the foreground object just leaves the building and thus the camera does not exhibit significant motion when capturing the video. While scale variations (and slight viewpoint changes) can be observed in this case, our method is still able to detect the foreground object and produces satisfactory results.

In addition, the foreground object detection results in consecutive video sequences are shown in Fig. 12, in which only representative frames are shown. It can be observed that both the object of interest and the camera are moving in these test video sequences, and thus it is very challenging to address the tasks of video object detection and tracking. Based on the proposed probabilistic CFOT generation and the re-start mechanism, we see that the foreground object regions (in red) can be properly detected, even under severe orientation and scale variations, or with blurred and fast changing background.

3.2. Performance comparisons

We have also compared our results to state-of-the-art methods in detecting or tracking video objects. Since our foreground object detection method is based on a probabilistic framework using SIFT trajectory, we first considered two trajectory-based approaches, i.e., *SIFT flow* proposed by Liu et al. [15] and *particle video* proposed by Sand and Teller [16]. Since the above methods focused on tracking moving objects and did not address the problem of object detection, we compare our method with the CVEPS framework (a compressed video editing and parsing system) proposed by Meng and Chang [7] and a recent approach of Bugeau and Perez [11].

3.2.1. Detection and tracking of foreground points

We first compare our results with two trajectory-based approaches, i.e., *SIFT flow* and *particle video*. As shown in Fig. 13, our method has the smallest number of foreground points (pink circles) while exhibiting satisfactory representation ability in locating the foreground object. It is worth noting that, although

[1] For interpretation of the references to colour in this figure text, the reader is referred to the web version of this article.

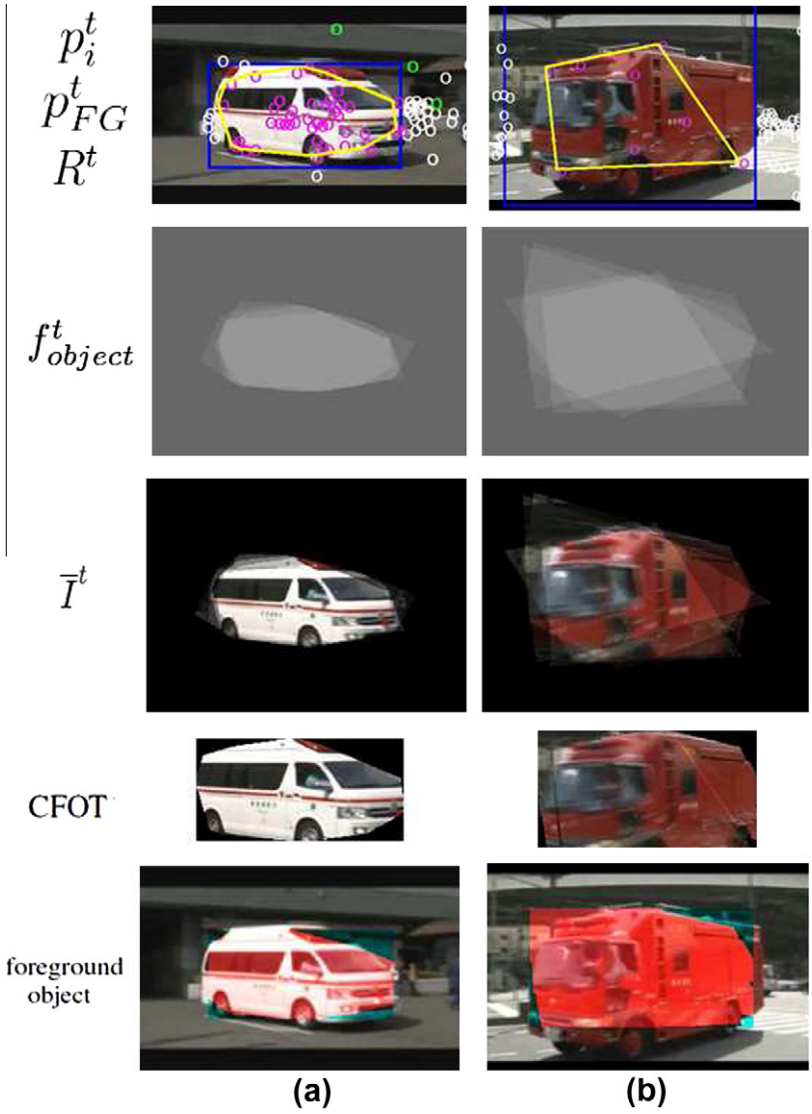

Fig. 11. Example detection results for each step in our proposed framework: (a) frame 590 of *Ambulance* sequence, and (b) frame 2137 of *FireEngine* sequence.

Fig. 12. Example foreground object detection results (denoted in red regions) under different camera view angles. The frame numbers of *Black Car* are: 191, 206, 222, 236, 252; the frame numbers of *Yellow Car* are: 580, 674, 698, 722, 739; and the frame numbers of *Motorbike* are: 250, 293, 310, 334, 352. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

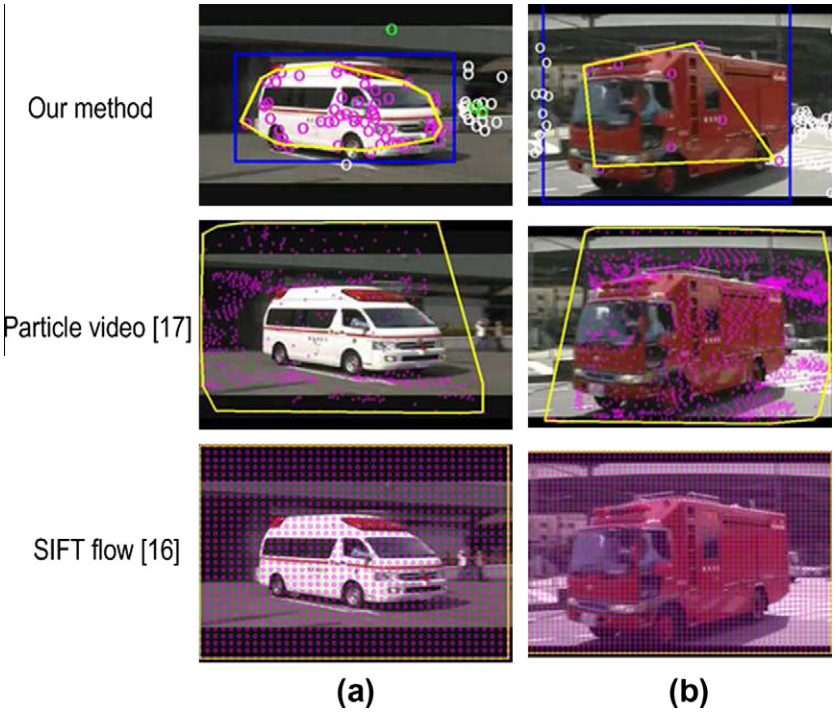

Fig. 13. Examples of keypoint correspondence using *Particle video* [16], *SIFT flow* [15], and our method: (a) *Ambulance*, frame 587, and (b) *Fire Fighting Car*, frame 2137. Note that the pink circles indicate foreground points with probability larger than our adaptive threshold, while our approach has the smallest number of foreground points (i.e., the most representative ones) with the ability to describe the foreground object (depicted by yellow polygons in the first row). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

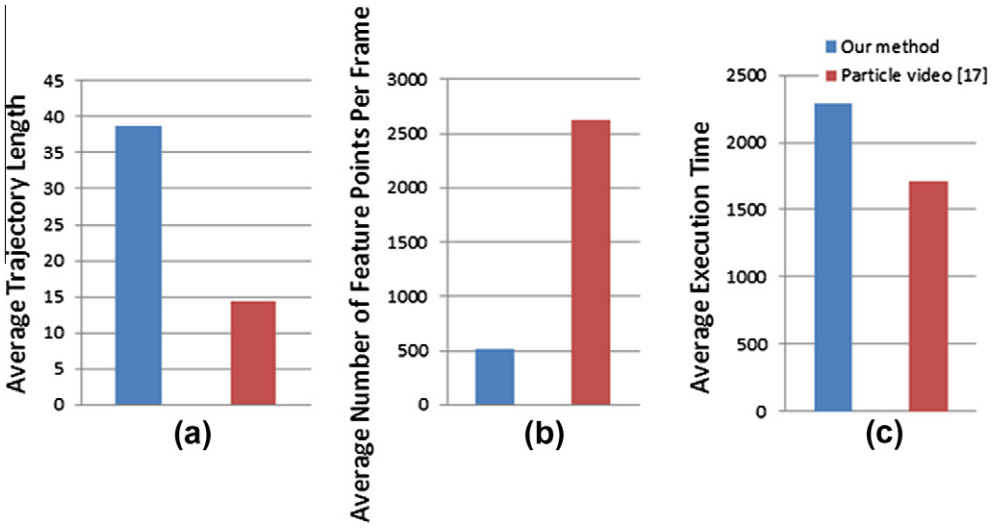

Fig. 14. Comparisons of feature point detection and tracking results: (a) average trajectory length (frame), (b) average number of feature points per frame, and (c) average computation time (in seconds).

the number of the foreground points (according to the definition in this paper) is larger using *SIFT flow* or *particle video*, the foreground object (as shown by the yellow polygons at the second and third rows) cannot be successfully detected.

Fig. 14 compares the average trajectory length, number of feature points per frame, and execution time on the video dataset considered, when using our proposed method and *Particle video*. We should notice that, *SIFT flow* [15] enforces a correspondence for each pixel with dense representation, providing the trajectory length the same as the video frame length, but the found correspondences could match uncorrelated feature points as non-representative trajectories. Therefore, in Fig. 14, the results of *SIFT flow* is not shown. We see that our approach produces a longer average trajectory length as shown in Fig. 14a, which confirms that our framework is more robust in locating the foreground object and thus is able to produce satisfactory detection results. As shown in Fig. 14b, our method has a much smaller number of feature points, and this observation is consistent with the example shown in Fig. 13. From Fig. 14c, it can be seen that our computation time is slightly longer than that of *Particle video*, since we need to extract and collect the SIFT points across video frames in advance for SIFT matching purposes (which allows us to determine SIFT

Table 1
Performance summary and comparison of the proposed method and *PARTICLE VIDEO* [16].

	Our method	*Particle video* [16]
Trajectory length	Medium-long	Short-medium
Complexity	Medium–high	High
Foreground point decision	Yes	No

correspondences under noisy condition, and produce more robust trajectories).

We summarize the comparisons in Table 1. Comparing to this prior method, our proposed framework is able to produce longer feature point trajectories for the construction of the CFOT model for detection purposes. From the above observations and comparisons, we confirm that our proposed method is able to determine the most representative feature points with robust trajectories. This is the key to the success for detecting moving foreground objects in real-world videos.

3.2.2. Foreground object detection

Next, we compare our foreground object detection results with the CVEPS approach proposed by Meng and Chang [7], which suggests the motion produced by the foreground moving object can be considered as an outlier of the camera motion. We also consider the method of Bugeau and Perez [11], who proposed to estimate the sensor motion with mean shift clustering and graph cut segmentation techniques to determine the foreground object region. We implement these two methods, and the foreground object detection is performed based on the video data ground truth labeled by human experts. To evaluate the performance, we considered the true positive rate *tpr* and false positive rate *fpr* in our experiments, which are defined as follows:

$$tpr = \frac{tp}{tp + fn}, \text{ and } fpr = \frac{fp}{fp + tn}, \tag{9}$$

The definitions of tp, tn, fp, and fn are illustrated in Fig. 15. The black rectangle is the original image frame. The purple rectangle indicates the detected foreground area, and the blue rectangle shows the ground truth foreground. Their overlapped region represents the true positive (tp) area. The purple rectangle without the tp area is the false positive fp region, and the blue rectangle without the tp area is the false negative fn area. The tn area is the original image without the blue rectangle.

We calculate *tpr* and *fpr* at each frame, and the results using different parameters for the two prior methods are shown as the receiver operating characteristic (ROC) curve in Fig. 16. We note that the block size b (in pixels) for the two methods considered are set to 8, 16, and 32, are considered as the basic unit for camera motion estimation (also standard settings for motion estimation in MPEG).

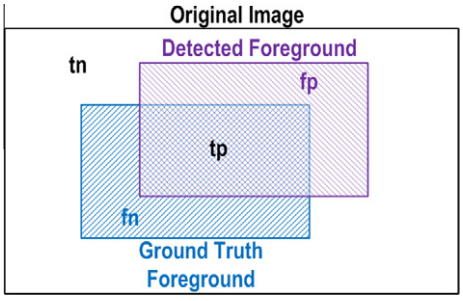

Fig. 15. The definitions of true positive (tp), true negative (tn), false positive (fp), and false negative (fn).

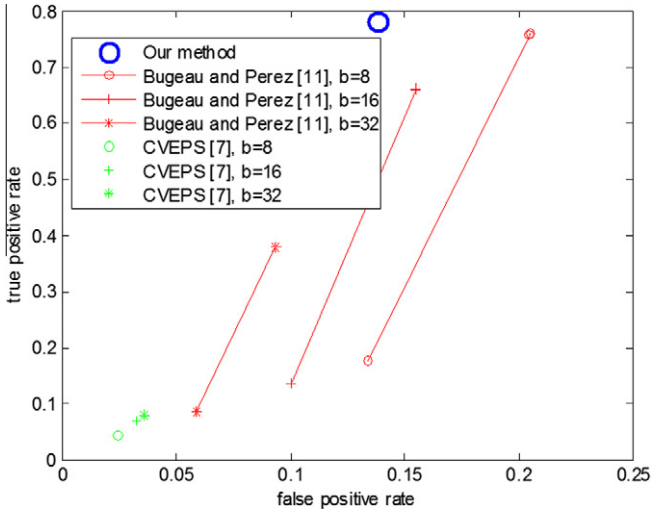

Fig. 16. Receiver operating characteristic comparisons. Note that b denotes the block size.

Since our proposed method automatically selects the data-dependent threshold when formulating the CFOT, only one operating point in ROC is noted. In Fig. 16, we note that the complete ROC curves cannot be obtained due to the existence of a much larger background area (i.e., $tn \gg fp$ and $fpr \to 1$ according to (9)), and thus only representative results are shown. However, it is worth noting that our approach results in the best performance since our operating point is closest to the perfect point $(0,1)$ for (fpr, tpr) in the ROC space.

Fig. 17 shows example foreground object detection results produced by different approaches, while both foreground and background objects are moving. As shown in the first row of Fig. 17, our method achieves satisfactory detection results even the foreground object is occluded (e.g., Fig. 17c). We note that the methods of [11,7] did not achieve comparable performance due to poor motion estimation. While our approach also requires the calculation and linking of SIFT points in consecutive frames, we provide a probabilistic and data-dependent framework, which is able to recover the missing or unstable foreground information.

In order to provide quantitative evaluation, we consider the accuracy of the object motion vector for comparisons. More specifically, the centroid $(\overline{x}^t, \overline{y}^t)$ of the object region at frame t was calculated according to the detected binary map. The centroid difference is determined by the current foreground object and another one at time t' is $(\Delta x, \Delta y) = (\overline{x}^t - \overline{x}^{t'}, \overline{y}^t - \overline{y}^{t'})$. The normalized motion vector is thus calculated based on the size of the image: $(\frac{\Delta x}{width}, \frac{\Delta y}{height})$, whose value is between $(0,1)$. Finally, a stability score can be defined as:

$$S = \frac{\sum_{t=1}^{T}(1 - ||(\frac{\Delta x}{width}, \frac{\Delta y}{height})||)}{T}, \tag{10}$$

where $\|\cdot\|$ is the Euclidean norm that measures the length of a vector, and T is the number of frames in a video clip. The stability scores are evaluated in two ways: S_g with $(\overline{x}^{t'}, \overline{y}^{t'}) = (\overline{x_g}^t, \overline{y_g}^t)$ (centroid of the ground truth at t frame) in the $(\Delta x, \Delta y)$ term. This term denotes the stability score measurement, which quantifies the difference between the centroid of the detection result to that of the ground truth. We also consider S_a with $(\overline{x}^{t'}, \overline{y}^{t'}) = (\overline{x}^{t+1}, \overline{y}^{t+1})$ in the $(\Delta x, \Delta y)$ term, representing the stability score measurement from consecutive frames. Comparisons results of S_g and S_a for each video in each class are shown in Fig. 18, and it is clear that our method achieves the highest stability scores for both cases. Finally,

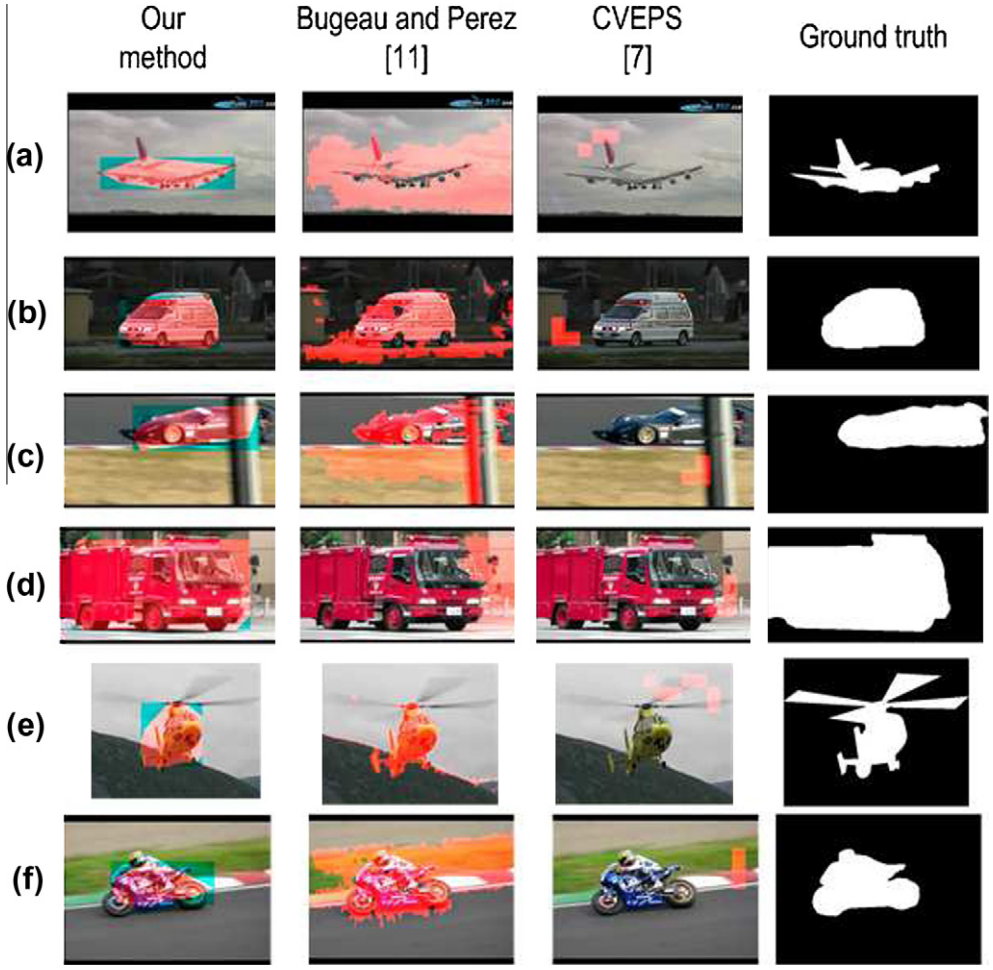

Fig. 17. Example results of foreground object detection (marked as red areas): (a) frame 1140, airplane 8, (b) frame 1009, ambulance 3, (c) frame 3296, car 4, (d) frame 236, fire engine 7, (e) frame 555, helicopter 3, and (f) frame 252, motorbike 5. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

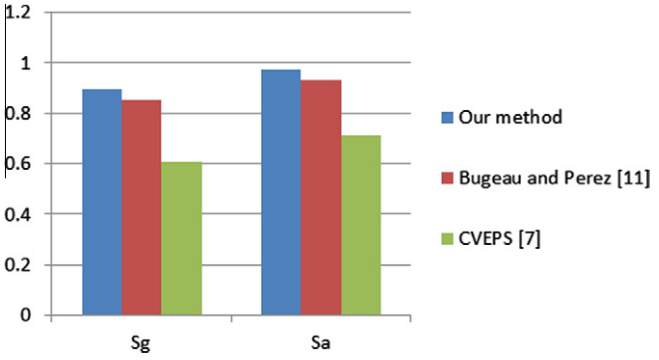

Fig. 18. Comparisons of stability scores: S_g indicating the difference between the detected object centroid to that of the ground truth, and S_a for the detection stability over consecutive frames.

Table 2
Summarization and comparison of detection-based methods: Bugeau and Perez [11], and CVEPS [7], and ours. Note that FG, BG, FGP, MS, and GC represent foreground, background, foreground point, mean-shift, and graph-cut respectively.

	Our method	[11]	[7]
FG/BG separation	FGP probability	GME + MS + GC	GME
Detection accuracy	High	Mid	Low
Reference frames	Multiple	Adjacent	Adjacent
Complexity	High	Mid	Low

the summarization and comparisons of different detection-based methods are shown in Table 2.

4. Conclusions

A novel foreground object detection method was presented in this paper. The proposed approach aims at detecting foreground object in a close-up scene of a video captured by a freely moving camera. By associating the SIFT motion vectors across video frames, we calculated the foreground object probability for the candidate foreground keypoints, and constructed a data-driven CFOT model for foreground object detection. Our detection framework does not require user interaction or parameter tuning as some prior work did. More importantly, we do not assume that the motion of the foreground or background is dominant across video frames. This is why our proposed method is able to handle uncontrolled videos, even under low contrast and viewpoint changing conditions. From our experiments, we verified the effectiveness and robustness of our method on a variety of Web-based videos, and we also confirmed that our method outperforms several state-of-the-art trajectory and detection-based algorithms.

References

[1] K. Patwardhan, G. Sapiro, V. Morellas, Robust foreground detection in video using pixel layers, IEEE Trans. Pattern Anal. Mach. Intell. 30 (4) (2008) 746–751.

[2] R.C. Jain, H.H. Nagel, On the analysis of accumulative difference pictures from image sequences of real world scenes, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 206–213.
[3] C. Wren, A. Azarbayejani, T. Darell, A. Pentland, Pfinder: real-time tracking of human body, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 780–785.
[4] C. Stauffer, W. Grimson, Learning patterns of activity using real time tracking, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 747–767.
[5] A. Elgammal, R. Duraiswami, L.S. Davis, Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (11) (2003) 1499–1504.
[6] Y. Sheikh, M. Shah, Bayesian modeling of dynamic scenes for object detection, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (2005) 1778–1792.
[7] J. Meng, S.F. Chang, CVEPS – a compressed video editing and parsing system, in: Proc. ACM Intl. Conf. Multimedia, 1997, pp. 43–53.
[8] Y. Tan, S.R. Kulkarni, P.J. Ramadge, A new method for camera motion parameter estimation, Proc. IEEE Intl. Conf. Image Process. 1 (1995) 23–26.
[9] M. Irani, P. Anandan, A unified approach to moving object detection in 2D and 3D scenes, IEEE Trans. Pattern Anal. Mach. Intell. 20 (6) (1998) 577–589.
[10] R. Wang, H.J. Zhang, Y.Q. Zhang, A confidence measure based moving object extraction system built for compressed domain, Proc. IEEE Intl. Sympos. Circ. Syst. (5) (2000) 21–24.
[11] A. Bugeau, P. Perez, Detection and segmentation of moving objects in highly dynamic scenes, Proc. IEEE Comput. Vision Pattern Recogn. (2007) 1–8.
[12] R.L. Felip, L. Barcelo, X. Binefa, J.R. Kender, Robust dominant motion estimation using MPEG information sports sequences, IEEE Trans Circ. Syst. Video Technol. 18 (1) (2008) 12–22.
[13] Y. Zhao, M. Casares, S. Velipasalar, Continuous background update and object detection with non-static cameras, in: Proc. IEEE Intl. Conf. Adv. Video Signal Based Surveillance, 2008, pp. 309–316.
[14] D.G. Lowe, Object recognition from local scale-invariant features, in: Proc. IEEE International Conference on Computer Vision, 1999, pp. 1150–1157.
[15] C. Liu, J. Yuen, A. Torralba, SIFT flow: dense correspondence across scenes and its applications, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2010) 978–994.
[16] P. Sand, S. Teller, Particle video: long-range motion estimation using point trajectories, Int. J. Comput. Vision 80 (1) (2008) 72–91.
[17] C.W. Su, H.Y.M. Liao, H.R. Tyan, C.W. Lin, D.Y. Chen, K.C. Fan, Motion flow-based video retrieval, IEEE Trans. Multimedia 9 (6) (2007) 1193–1201.
[18] A. Senior, Tracking people with probabilistic appearance models, Proc. IEEE Perform. Eval. Track. Surveill. (2002) 48–55.
[19] http://www.mathworks.com/matlabcentral/fileexchange/22260-fast-convex-hull-algorithm.
[20] Y.Q. Shi, H. Sun, Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards, CRC Press LLC, 1999. pp. 407–408.
[21] http://www.youtube.com.