Forward–Backward Mean-Shift for Visual Tracking With Local-Background-Weighted Histogram

Lingfeng Wang, Hongping Yan, Huai-Yu Wu, and Chunhong Pan

***Abstract*—Object tracking plays an important role in many intelligent transportation systems. Unfortunately, it remains a challenging task due to factors such as occlusion and target-appearance variation. In this paper, we present a new tracking algorithm to tackle the difficulties caused by these two factors. First, considering the target-appearance variation, we introduce the local-background-weighted histogram (LBWH) to describe the target. In our LBWH, the local background is treated as the context of the target representation. Compared with traditional descriptors, the LBWH is more robust to the variability or the clutter of the potential background. Second, to deal with the occlusion case, a new forward–backward mean-shift (FBMS) algorithm is proposed by incorporating a forward–backward evaluation scheme, in which the tracking result is evaluated by the forward–backward error. Extensive experiments on various scenarios have demonstrated that our tracking algorithm outperforms the state-of-the-art approaches in tracking accuracy.**

***Index Terms*—Forward–backward mean shift (FBMS), local-background-weighted histogram (LBWH), visual tracking.**

I. Introduction

OBJECT TRACKING is a critical component in many intelligent transportation systems (ITS), such as visual surveillance systems [1]–[4] and driver assistance systems [5]–[7]. In visual surveillance systems, object tracking can be used for event detection and target activity analysis. With the help of object tracking, driver assistance systems can automatically detect potentially dangerous situations, so that it can warn the driver or initiate appropriate protective measures in time.

Although many excellent approaches have been proposed for object tracking, it is still very difficult to develop a robust tracking algorithm, because the performance of object tracking is greatly influenced by factors such as the variability of the target appearance and the partial or entire occlusion of the target.

To tackle the problem of appearance variation, many classical algorithms, such as the mean shift [8], [9], focus on constructing an adaptive target model. However, in many tracking applications, such as surveillance, a tracker needs to clearly separate the target from its local background. Thus, it is very restrictive if the tracker only utilizes the target model but ignores its surroundings (background). A better representation of the target should combine both aspects. Based on this idea, we incorporate the local background into the target model by presenting the new local-background-weighted histogram (LBWH).

To handle the difficulty that is caused by occlusion, we improve the traditional mean-shift tracking framework by adding a forward–backward evaluation scheme, in which the tracking result is evaluated in both forward and backward manners. Specifically, our tracking result adaptively fuses two results, i.e., the observation that is calculated by the mean shift and the prediction that is computed by the median filter of the historical trajectory.

The advantages of our tracker as compared with the state-of-the-art approaches can be highlighted in two aspects:

1) The proposed LBWH treats the local background as the context of the target and selects salient features to describe the target. Therefore, our tracking algorithm is more robust to the case in which the target is similar with its local background. Furthermore, different from the corrected background-weighted histogram (CBWH) [10], which assumes that the target and candidate regions share the same background, the LBWH considers that these regions have their own *local* background. By introducing the LBWH, our method better suits general tracking scenarios, particularly when the background varies.
2) The forward–backward scheme makes our tracking process smoother than other traditional methods. Hence, our method can tackle the short-term occlusion better. Moreover, as pointed out in [11], Kalman-filter-based methods [12], [13] adaptively fuse the results of observation obtained from current image and the prediction obtained from previous tracking result. From the view of information fusion, our method is similar to the Kalman filters. The main difference is that, in our method, the fusion weight is determined by the tracking consistence, whereas in the Kalman filters, it is controlled by the observation and prediction covariance. In fact, in object tracking, the tracking consistence is more important than the observation and prediction covariance. The experimental results also indicate that our tracking process is more stable than those of the filtering methods.

Manuscript received June 21, 2012; revised October 22, 2012 and February 7, 2013; accepted May 3, 2013. Date of publication June 19, 2013; date of current version August 28, 2013. This work was supported in part by the National Natural Science Foundation of China under Grant 61175025, Grant 61075016, Grant 61272049, and Grant 61190125, and in part by the Fundamental Research Funds for the Central Universities. The Associate Editor for this paper was N. Papanikolopoulos.

L. Wang, H.-Y. Wu, and C. Pan are with the Department of National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: lfwang@nlpr.ia.ac.cn; hywu@nlpr.ia.ac.cn; chpan@nlpr.ia.ac.cn).

H. Yan is with the College of Information and Engineering, China University of Geosciences, Beijing 100083, China (e-mail: hongpingyan.cn@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2013.2263281

The rest of this paper is organized as follows. An overview of related work is given in Section II. Section III presents our forward–backward mean-shift (FBMS) tracking method based on the LBWH. Section IV gives the experimental results. Discussions and conclusions are presented in the last section.

II. Related Work

Object tracking is widely applied in ITS. Generally, the approaches that are most related to our paper can be grouped into four categories, i.e., appearance-based [14]–[17], dynamical-model-based [13], [18], [19], online-selection-based [20]–[23], and online-learning-based methods [24]–[29].

The template-based method [14], [30] is one of the classical appearance-based methods, as it is robust to nonrigidity and partial occlusion. Tracking with fixed templates can be reliable over short duration, but it copes poorly with appearance changes over longer durations that occur in most applications. Robustness can be enhanced with the use of subspace models of appearance [15], [31]. Another appearance-based method relies on invariant features, such as histogram [8], [9], spatiogram [16], and spatial-color mixture of Gaussians [3], [32]. For example, Jepson *et al.* [17] proposed an online appearance model for visual tracking.

For the dynamical-model-based method, probabilistic modeling and sampling techniques are employed to achieve efficient visual tracking. The dynamical model based method is often composed of two stages. In the first stage, a dynamic model is utilized to predict the next state; in the second stage, the prediction is refined by the observations from the images. The classical Kalman filter algorithm [12], [13], [33] has been used to track objects via the randomness that is generated by a linear dynamic operator with the Gaussian noise. However, it is hard to handle the nonlinear or non-Gaussian conditions. The particle-filter [18], [19], [34], [35] algorithm becomes an effective tool to solve these problems. The critical disadvantage of the particle filter is its high computational cost.

Recently, treating object tracking as an online selection problem has attracted much attention. The core of the scheme is to select appropriate features to classify each pixel as foreground or background. To the best of our knowledge, the first online-selection-based tracking algorithm was proposed by Collins *et al.* [20], by switching the mean-shift tracking algorithm between different linear combinations of three color channels to select the features that can distinguish between the target (foreground) and the background. Motivated by the work of Collins *et al.*, Liang *et al.* [21] solve the online feature selection problem by using the Bayesian error rate instead of the variance ratio. Similarly, Kwolek [36] presents multiple color histograms instead of the linear combination of the three color channels. Improved performances have been reported in these papers. However, these methods perform well only when the object appearance does not drastically change. Background-weighted methods [10], [37], [38] can be treated as a variant of online-selection-based methods. The essential idea behind these methods is to weaken inner-background features or make target features prominent. For example, Ning *et al.* [10] adopted a CBWH to represent the target.

A family of online-learning-based tracking approaches [24]–[29], [39] have been recently proposed. Since these methods are in common with object detection, they have been termed "tracking by detection." Similar with the online-selection-based method, the online-learning-based method also formulates visual tracking as a classification problem, i.e., optimally separating the target from the background in each frame. However, the main difference between the two groups is the target representation. The online-selection-based method represents the target as a pixel set, whereas the online-learning-based method represents it as a whole template. For example, Grabner *et al.* [25] designed an online boosting classifier that selects features to discriminate the target from the background. The online-learning-based method demonstrates that, by using online feature selection, the tracking problem can be considerably simplified, and therefore, the classifier can be quite compact and fast. However, when the object is occluded, these methods may fail because it is hard to be detected.

III. Method

Here, we first introduce the mean-shift-like trackers. Then, we describe the proposed LBWH and the forward–backward scheme in detail. Finally, we present the implementation details about our tracker.

A. Overview of Mean-Shift-Like Trackers

In object tracking, the target is usually described in a rectangle shape. For mean-shift-like trackers, the target region is represented by the color histogram $\mathbf{q} = \{q_u\}_{u=1}^{B}$ with B bins, i.e.,

$$q_u = C_1 \sum_{i=1}^{n} k\left(\|\mathbf{x}_i^\star\|_2^2\right) \delta\left[b\left(\mathbf{x}_i^\star\right) - u\right] \tag{1}$$

where $\{\mathbf{x}_i^\star\}_{i=1}^{n}$ are the locations of n pixel, $b(\mathbf{x}_i^\star)$ is the bin at location $\mathbf{x}_i^\star$, C_1 is a normalization constant, $k(.)$ is the kernel function, and $\delta[.]$ is the Kronecker delta function. Similarly, the candidate region that is centered at $\mathbf{y}$ is represented by the color histogram $\mathbf{p}(\mathbf{y}) = \{p_u(\mathbf{y})\}_{u=1}^{B}$, in which

$$p_u(\mathbf{y}) = C_2 \sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|_2^2\right) \delta\left[b(\mathbf{x}_i) - u\right] \tag{2}$$

where C_2 is a normalization constant, and h is the bandwidth.

The object tracking problem is defined as finding a meaningful position $\mathbf{y}$, in which the Bhattacharyya coefficient between $\mathbf{p}(\mathbf{y})$ and $\mathbf{q}$, namely, $\rho(\mathbf{p}(\mathbf{y}), \mathbf{q})$, reaches a maximum value, i.e.,

$$\mathbf{y}^\star = \max_{\mathbf{y}} \rho\left(\mathbf{p}(\mathbf{y}), \mathbf{q}\right) = \max_{\mathbf{y}} \sum_{u=1}^{B} \sqrt{p_u(\mathbf{y}) q_u}. \tag{3}$$

The mean-shift-like trackers search for the new target location in the current frame starting from location $\mathbf{y}_m$ of the target in the previous frame. By applying the first-order Taylor expansion at $p_u(\mathbf{y}_m)$, we get that

$$\rho(\mathbf{p}(\mathbf{y}), \mathbf{q}) \approx \frac{1}{2}\sum_{u=1}^{B}\sqrt{p_u(\mathbf{y}_m)q_u} + \frac{1}{2}\sum_{u=1}^{B} p_u(\mathbf{y})\sqrt{\frac{q_u}{p_u(\mathbf{y}_m)}} \approx \frac{1}{2}\sum_{u=1}^{B}\sqrt{p_u(\mathbf{y}_m)q_u} + \frac{C_2}{2}\sum_{i=1}^{n} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|_2^2\right) \quad (4)$$

where

$$w_i = \sum_{u=1}^{B}\sqrt{\frac{q_u}{p_u(\mathbf{y}_m)}}\,\delta\left[b(\mathbf{x}_i) - u\right]. \quad (5)$$

To obtain the solution to (3), the second term in (4) has to be maximized. Note that, in (4), the first term is independent of $\mathbf{y}$, and the second one is the kernel density term with a kernel function $k(.)$. As pointed in [40], the maximum value can be obtained by a mean-shift procedure, i.e.,

$$\mathbf{y}_{m+1} = \frac{\sum_{i=1}^{n}\mathbf{x}_i w_i g\left(\left\|\frac{\mathbf{y}_m-\mathbf{x}_i}{h}\right\|_2^2\right)}{\sum_{i=1}^{n} w_i g\left(\left\|\frac{\mathbf{y}_m-\mathbf{x}_i}{h}\right\|_2^2\right)} \quad (6)$$

where $g(.)$ is the shadow of the kernel profile $k(.)$. $\{\mathbf{x}_i\}_{i=1}^{n}$ are the pixel locations in the candidate region, and $\{w_i\}_{i=1}^{n}$ are the corresponding weights of all pixels.

The main difference between all mean-shift-like trackers lies in the calculation of weight w_i. For example, the classical mean-shift algorithm proposed in [41] computes weight w_i by (5). This algorithm performs well when the target appearance is homogeneous and well distinguished from the background. However, it often fails when the target region is roughly initialized. In such case, the target region might contain background features (called inner-background features) as well. Thus, background features cannot be ignored in the representation of the target and candidate regions.

To incorporate background features, Comaniciu *et al.* [41] introduced the background-weighted histogram (BWH) to simultaneously represent the target and candidate regions, i.e.,

$$\widetilde{q}_u = q_u * v_u \quad (7)$$

$$\widetilde{p}_u(\mathbf{y}) = p_u(\mathbf{y}) * v_u. \quad (8)$$

The background weight v_u is obtained by $v_u = \min\{o_u^\star/o_u, 1\}$, where $\mathbf{o} = \{o_u\}_{u=1}^{B}$ is the background histogram and $o_u^\star$ is its smallest nonzero entry. As shown in (7) and (8), the main insight of the BWH is that it can reduce the original histogram entries [both q_u and $p_u(\mathbf{y})$], where target features often appear in the background (where o_u is large). However, in [10], Ning *et al.* proved that the BWH did not introduce any background features, since the background histogram simultaneously transforms the target and candidate histograms. Instead, they proposed the CBWH, which only transforms the target histogram. In [10], weight w_i is calculated as follows:

$$w_i = \sum_{u=1}^{B}\sqrt{\frac{\widetilde{q}_u}{p_u(\mathbf{y}_m)}}\,\delta\left[b(\mathbf{x}_i) - u\right]. \quad (9)$$

From (5) and (9), we see that only the target histogram is weighted by the background histogram, i.e., q_u is replaced by $\widetilde{q}_u$ in the weight calculation.

B. LBWH

The authors in [10] and [41] assume that the target and candidate regions share the same background that is represented by $\mathbf{o}$. However, in practice, these regions have their own *local* backgrounds. That is, the surrounding background of each region is different (or varying frame by frame) and needs to be distinguished. Therefore, the LBWH is proposed to weaken the inner-background features of all these regions by using their own local backgrounds.

The proposed LBWH assigns each bin $b(\mathbf{x}_i^\star)$ weight ν_i and reformulates the histogram calculation as follows:

$$\widehat{q}_u = C_1\sum_{i=1}^{n} k\left(\|\mathbf{x}_i^\star\|_2^2\right)\delta\left[b\left(\mathbf{x}_i^\star\right) - u\right]\nu_i. \quad (10)$$

Weight ν_i denotes the probability of each pixel belonging to the target. The pixel that is located on the target has a higher weight than the pixel that is located on the inner background. Thus, introducing the weight can help weaken inner-background features or make the target features more prominent.

We assume that, in a small local region Ω_i (e.g., 3×3) that is centered at the ith pixel, weight $\nu_{j(i)}$ is a linear transformation of the original input image $\mathbf{I}_{j(i)}$, i.e.,

$$\nu_{j(i)} = w_i\mathbf{I}_{j(i)} + b_i \quad \forall j(i) \in \Omega_i \quad (11)$$

where w_i and b_i are the linear coefficients that are assumed to be constant in the local region Ω_i. Theoretically, the total regression error can be represented as follows (refer to [42]):

$$\mathcal{E}(\nu) = \nu^T L\nu \quad (12)$$

where L is a Laplacian matrix that is constructed on image $\mathbf{I}$.

During the tracking process, the target is represented by a rectangle. Hence, we can obtain a rough weight $\widehat{\nu}$ based on the tracking result that is obtained in the previous frame, i.e.,

$$\widehat{\nu}(i) = \begin{cases} 1, & \text{pixel } i \text{ in the target} \\ 0, & \text{pixel } i \text{ in the local background.} \end{cases} \quad (13)$$

Weight ν should be close to the rough weight $\widehat{\nu}$, i.e., $\|\nu - \widehat{\nu}\|_2^2$ should be small. Combining with (12), weight ν can be calculated by

$$\nu = \arg\min_{\nu}\|\nu - \widehat{\nu}\|_2^2 + \lambda\nu^T L\nu \quad (14)$$

where λ is a weighting constant. By minimizing (14), ν is calculated as $\nu = (\mathrm{Id} + \lambda L)^{-1}\widehat{\nu}$, where Id is an identity matrix. Calculating $(\mathrm{Id} + \lambda L)^{-1}$ is time consuming. Fortunately, the

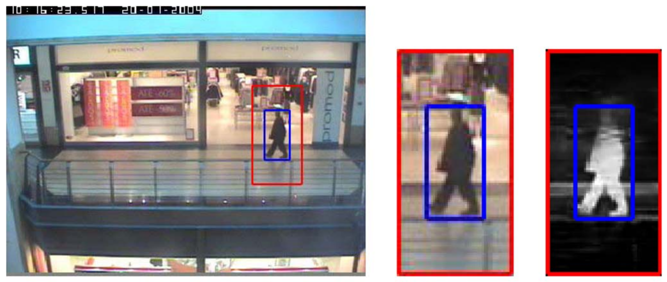

Fig. 1. Example of weight calculation with the guided filter (a) Input frame. (b) Target and its local background rectangles. (c) Corresponding weight map obtained by the guided filter.

filtering method that is proposed in [43] can approximately solve (14). Accordingly, to obtain weight ν_i, we perform the guided filter [43] with the target rectangle and the local-background rectangle as the inputs, as shown in Fig. 1. In this figure, the blue and red rectangles are the target and the local background, respectively. Fig. 1(c) shows the corresponding weight map after performing the guided filter. Combining with (5), weight w_i is calculated as

$$w_i = \sum_{u=1}^{B} \sqrt{\frac{\widehat{q}_u}{\widehat{p}_u(\mathbf{y}_m)}} \delta\left[b(\mathbf{x}_i) - u\right]. \tag{15}$$

The calculation of weights in (5), (9), and (15) is similar. The main difference is the definition of the histograms. In (5), the original histograms are used. In (9), the weighted target histogram is used. In our method, both the target and candidate histograms are used as the weighted versions.

From the aforementioned description, we can see that all the background-weighted methods aim to weaken the inner-background features or to make the target features prominent, but the way to reach the goal is different. Specifically, *in our LBWH, the target and candidate regions use their own local backgrounds to reach this goal, whereas in the BWH and the CBWH, the targetand candidate regions share the same background*. Accordingly, our tracking algorithm is more robust to the scenarios where the background is varying.

C. FBMS

In some scenarios, the target may be occluded by the background. In such case, the mean-shift vector that is calculated by (6) may be incorrect. The direction of the mean-shift vector may be decided by the local background rather than the target. When this occurs, the tracking algorithm often suddenly fails. To prevent this sudden failure, we propose a forward–backward evaluation scheme, as well as the corresponding FBMS tracking algorithm.

The principle of the forward–backward evaluation scheme is based on the forward–backward consistency assumption [44] in which correct tracking should be reversible over the time flow. Algorithmically, the forward–backward evaluation consists of the following two steps (see Fig. 2).

Forward: Starting from the previous location $\mathbf{y}_{t-1}$, we produce a forward mean-shift vector $\hat{\mathbf{y}}_t$ on the current frame [see the red dashed arrow in Fig. 2(b)]. Hence, current location $\mathbf{y}_t^{\circ}$ estimated by mean shift is represented as

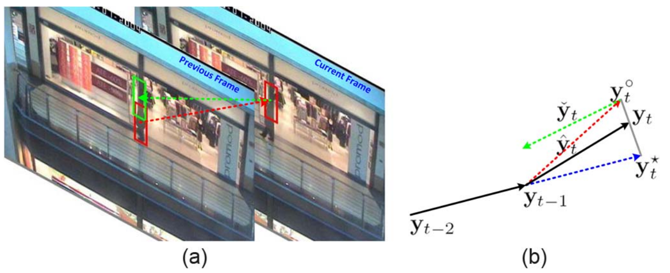

Fig. 2. Illustration of FBMS. (a) Forward mean-shift procedure in the current frame and the backward mean-shift procedure in the previous frame. (Red rectangle in the previous frame) Previous target location $\mathbf{y}_{t-1}$. (Red rectangle in the current frame) $\mathbf{y}_t^{\circ}$ estimated by the forward mean shift. (Green rectangle) $\mathbf{y}_t^{\circ} + \check{\mathbf{y}}_t$ estimated by the backward mean shift. (b) All the flow vectors. (Red dashed arrow) is Forward mean-shift vector $\hat{\mathbf{y}}_t$. (Green dashed arrow) Backward mean-shift vector $\check{\mathbf{y}}_t$. (Gray line) Final location $\mathbf{y}_t$ is a linear combination of the mean-shift result $\mathbf{y}_t^{\circ}$ and the predicted result $\mathbf{y}_t^{\star}$.

$$\mathbf{y}_t^{\circ} = \mathbf{y}_{t-1} + \hat{\mathbf{y}}_t. \tag{16}$$

Backward: Starting from the current location $\mathbf{y}_t^{\circ}$, we compute a backward mean-shift vector $\check{\mathbf{y}}_t$ on the previous frame [see the green dashed arrow in Fig. 2(b)].

If the target is correctly tracked, the opposite of forward mean-shift vector $\hat{\mathbf{y}}_t$ should be equal to the backward mean-shift vector $\check{\mathbf{y}}_t$, namely, $\hat{\mathbf{y}}_t + \check{\mathbf{y}}_t = \mathbf{0}$. Thus, we can use the forward–backward error, i.e.,

$$\varepsilon_t = \|\hat{\mathbf{y}}_t + \check{\mathbf{y}}_t\|_2 \tag{17}$$

to evaluate how well the target is tracked.

To make the tracking process smooth, we also provide the prediction result of the current location $\mathbf{y}_t^{\star}$ based on the historical trajectory $\mathcal{Y}_{t-1} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}\}$, given by $\mathbf{y}_t^{\star} = p(\mathbf{y}_t|\mathcal{Y}_{t-1})$, where $p(\mathbf{y}_t|\mathcal{Y}_{t-1})$ is approximated by the median-flow model, i.e.,

$$\mathbf{y}_t^{\star} = p(\mathbf{y}_t|\mathcal{Y}_{t-1}) = \mathbf{y}_{t-1} + \text{med}_{\tau\in\mathcal{T}}(\Delta\mathbf{y}_\tau) \tag{18}$$

where $\mathcal{T} = \{t-1, t-2, \ldots, t-N\}$ (N is experimentally set to 20) is the historical moments, $\Delta\mathbf{y}_\tau = \mathbf{y}_\tau - \mathbf{y}_{\tau-1}$, and function $\text{med}_{\tau\in\mathcal{T}}(.)$ is used to obtain the median value.

Two components, namely, the mean-shift result $\mathbf{y}_t^{\circ}$ and the predicted result $\mathbf{y}_t^{\star}$, are combined together to estimate the final target location $\mathbf{y}_t$ of time t, i.e.,

$$\mathbf{y}_t = (1 - g(\varepsilon_t))\,\mathbf{y}_t^{\circ} + g(\varepsilon_t)\mathbf{y}_t^{\star} \tag{19}$$

in which $g(\varepsilon_t)$ is the weighting function that is controlled by the forward–backward error ε_t. Here, soft thresholding is used to estimate the weighting function $g(\varepsilon_t)$, i.e.,

$$g(\varepsilon_t) = \begin{cases} \frac{\varepsilon_t}{\xi} & \varepsilon_t \le \xi \\ 1 & \varepsilon_t > \xi \end{cases} \tag{20}$$

where ξ is the threshold. Parameter ξ is mainly determined by the size and the speed of the moving target. Based on plenty of experiments, $\xi = 10$ can provide satisfactory results.

Algorithm Flowchart: The main flow of the FBMS with the LBWH is summarized in Algorithm 1. In the forward and backward mean shifts, the weight is calculated by (15).

Algorithm 1: FBMS with LBWH

```
1. Initialize the target location $\mathbf{y}_0$.
2. Calculate its histogram $\widehat{\mathbf{q}}$ by (10).
3. for $t = 1$ to total time do
4.     Calculate $\hat{\mathbf{y}}_t$, $\mathbf{y}_t^\circ = \hat{\mathbf{y}}_t + \mathbf{y}_{t-1}$, and $\check{\mathbf{y}}_t$.
5.     Calculate the forward–backward error $\varepsilon_t$ by (17).
6.     Calculate the fusion weight $g(\varepsilon_t)$ by (20).
7.     Calculate the predicted result $\mathbf{y}_t^\star$ by (18).
8.     Calculate the fusion result $\mathbf{y}_t$ by (19).
9      if $t\%n == 0$ then
10         Update the target histogram by (21).
11     end
12 end
```

Model Updating: To handle the change of target appearance, the target histogram $\widehat{\mathbf{q}}$ is updated at each frame. To reduce drifting probability, we use a slight model-updating method based on the Bhattacharyya distance, i.e.,

$$\widehat{\mathbf{q}} = \begin{cases} (1-\beta)\widehat{\mathbf{q}} + \beta\widehat{\mathbf{q}}(\mathbf{y}_t) & D\left(\widehat{\mathbf{q}}, \widehat{\mathbf{q}}(\mathbf{y}_t)\right) < \eta \\ \widehat{\mathbf{q}} & \text{otherwise} \end{cases} \tag{21}$$

where $\widehat{\mathbf{q}}(\mathbf{y}_t)$ is the observed histogram at location $\mathbf{y}_t$. Parameter $\beta \in [0, 1]$ represents the contribution of the current histogram $\widehat{\mathbf{q}}(\mathbf{y}_t)$ to the target model $\widehat{\mathbf{q}}$, and threshold $\eta \in [0, 1]$ decides the condition of model updating. To avoid excessively updating the target model, β and η are usually set to small values. After performing plenty of experiments, β and η are set to 0.01 and 0.1, respectively.

IV. Experimental Results

Extensive experiments are conducted to evaluate the performance of the proposed tracking algorithm by comparing with several state-of-the-art approaches, i.e., mean shift, CBWH mean shift [10],[1] online boosting [25],[2] tracking–learning–detection TLD [45], [46],[3] L1 tracker (L1) [14],[4] and incremental visual tracking (IVT).[5] In our method, the RGB color space is used as the feature space, and it was quantized into $16 \times 16 \times 16$ bins. In the guided filter [43], the window size is set to 10, and the regularization parameter is set to 10^{-4}. We provide two versions of our tracking algorithm. The first version uses the mean shift, which is named "our method with MS," while the second one uses FBMS, which is named "our method with FBMS."

In our experiments, we adopt two widely used criteria, i.e., the $\mathbf{F\text{-}score}$ and the position error $\mathbf{Pe}$, to quantitatively evaluate the tracking accuracy. The ground truth is created by manually selecting the image region that best covers the target. $\mathbf{F\text{-}score}$ measures the tracking accuracy by considering both the recall and the precision, i.e.,

$$\mathbf{F\text{-}score} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{22}$$

where TP, FP, and FN are the true positive, the false positive, and the false negative, respectively. In practice, the position error $\mathbf{Pe}$ illustrates the distance between the center position of the tracked target and the ground truth, i.e.,

$$\mathbf{Pe} = \sqrt{\left(p_x^{\mathrm{trk}} - p_x^{\mathrm{gt}}\right)^2 + \left(p_y^{\mathrm{trk}} - p_y^{\mathrm{gt}}\right)^2} \tag{23}$$

where p_x^{trk} and p_y^{trk} are the x- and y-center positions of the tracked target, and p_x^{gt} and p_y^{gt} are those of the ground truth. A good result should provide a high $\mathbf{F\text{-}score}$ value but a low $\mathbf{Pe}$ value.

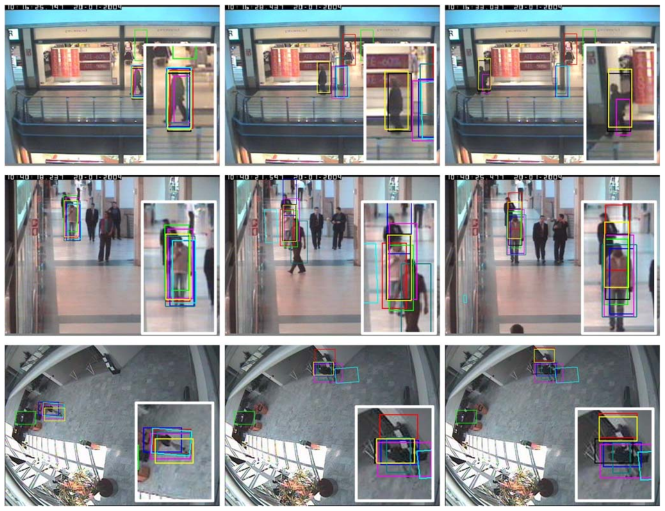
Fig. 3. Comparison on CAVIAR data set. The sequences from top to bottom are **CAVIAR:fosne2, CAVIAR:csa2** and **CAVIAR:bww1**, respectively. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT. Bottom-right patches illustrate the close-ups.

A. *Qualitative Comparisons*

CAVIAR Data Set: In this experiment, we aim at tracking pedestrians in three indoor sequences. The comparative results are shown in Fig. 3. The tested video sequences are downloaded from the CAVIAR Web site.[6]

In the first sequence (**CAVIAR:fosne2**), the challenges for tracking come from two aspects. First, the illumination condition gradually changes. Second, the appearance of a dark pillar is very similar with the target. As illustrated in the first row, most algorithms except TLD and ours fail when the target passed through the pillar. Note that, the tracking rectangles that are provided by TLD do not fit the pedestrian.

From the results of the second sequence (**CAVIAR:csa2**), we can see that the mean shift, IVT, and online boosting algorithms

[1]The code is available at www4.comp.polyu.edu.hk/~cslzhang/code.htm
[2]The code is available at www.vision.ee.ethz.ch/boostingTrackers
[3]The code is available at info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html
[4]The code is available at www.ist.temple.edu/~hbling/code_data.htm
[5]The code is available at www.cs.toronto.edu/~dross/ivt/
[6]http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

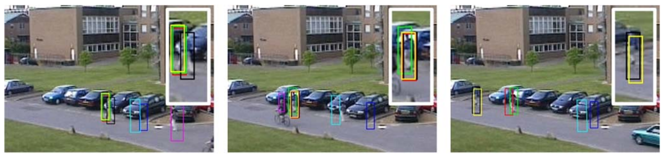
Fig. 4. Comparison on **PET2001**. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT. Top-right patches illustrate the close-ups.

fail when the target person is partially occluded by another person. Among them, only the online boosting algorithm can redetect the target after it reemerges. The results of the CBWH mean shift and TLD flutter a little.

As shown in the third row (the results on **CAVIAR:bww1**), the CBWH mean shift fails to track the target as he moves away from the strong light area. The mean shift, IVT, and our algorithm with MS fail to track the target as he approaches the face-recognition machine. The local backgrounds around the initial target region and the current candidate region are very different, as the pedestrian moves away from the strong light area. Hence, the weight that is generated by the CBWH is inaccurate, which makes the CBWH mean shift fail. Our method with FBMS tracks the target successfully throughout the whole sequence.

There are two main reasons that lead to the good performance of our algorithm. First, the LBWH selects salient features, which helps in dealing with the variations of the target and the background. Second, the proposed forward–backward scheme better models the smoothness of the pedestrian's motion.

PET2001: In the second experiment, we select an outdoor video sequence from the PET 2001 data set,[7] and the visual result is shown in Fig. 4. As illustrated in this figure, the appearance of the target is very similar to its local background. Moreover, the target is partially occluded by a bicycle. From this figure, we can see that the online boosting, TLD, IVT, and L1 algorithms very quickly lose the target when the target moves away from the black car (see the second image). The mean shift and the CBWH mean shift fail to track the pedestrian after the bicycle passes through. It is worth noting that the TLD algorithm is able to recover the failure in some frames (refer to the third image). Unfortunately, the tracked rectangle quickly drifts away, when the target is partially occluded by the bicycle. The ability to recover from failure relies on the utilization of the detection module based on the p/n learning. However, the failure is caused by the use of an optical flow framework, which is proven sensitive to the partial occlusion. On the contrary, our tracker can still track the pedestrian in this difficult scene.

PET2006: A third comparison result is shown in Fig. 5. The tested video sequence is downloaded from the PET 2006 web site.[8] The similarity between the target and the background is very high that, in some frames, it is difficult even for a human observer to recognize the target. Moreover, the target is very small and can be only represented by a 17×25 rectangle. From

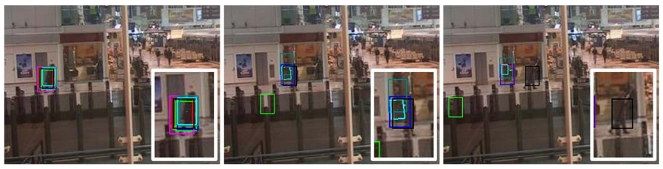
Fig. 5. Comparison on **PET2006**. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT. Bottom-right patches show the close-ups.

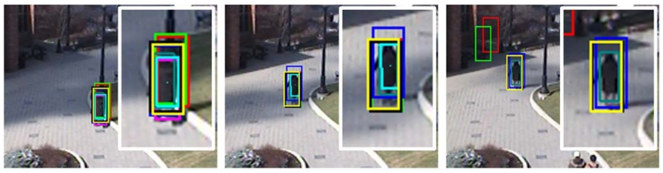
Fig. 6. Comparison on **OSU:1b**. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT. Top-right patches illustrate the close-ups.

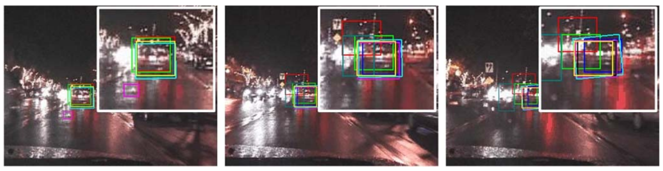
Fig. 7. Comparison on **CAR11**. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT. Top-right patches illustrate the close-ups.

this figure, we can see that all algorithms except ours fail when the target moves to the store entrance.

OSU:1b: The fourth experiment is to track a pedestrian in an outdoor video sequence.[9] The result is shown in Fig. 6. At the beginning, the pedestrian walks in the sun. After a period of time, the target moves into the shadow. The online boosting, L1, and our algorithm track the pedestrian well regardless of these difficulties, whereas the others eventually fail.

CAR11: In the fifth experiment, we aim at tracking a moving car. The video sequence is downloaded from the IVT Web site.[10] Tracking a moving car is frequently used in a driver assistance system. From Fig. 7, it is even hard for a human to distinguish the car from its local background. With the help of the background weighing strategy, our method with FBMS can successfully track the car throughout the whole sequence.

B. Quantitative Comparisons

Frame-by-Frame Comparison: The numerical comparisons of all methods on the aforementioned seven video sequences are shown in Figs. 8 and 9. From these figures, our algorithm holds the highest $\mathbf{F\text{-}score}$ value, as well as the lowest $\mathbf{Pe}$ value. The high $\mathbf{F\text{-}score}$ indicates that our algorithm accurately tracks the targets, while the low $\mathbf{Pe}$ indicates that our tracking results are

[7]http://www.cvg.cs.rdg.ac.uk/PETS2001/pets2001-dataset.html
[8]http://www.cvg.rdg.ac.uk/PETS2006/data.html
[9]http://www.cse.ohio-state.edu/otcbvs-bench/
[10]http://www.cs.toronto.edu/~dross/ivt/

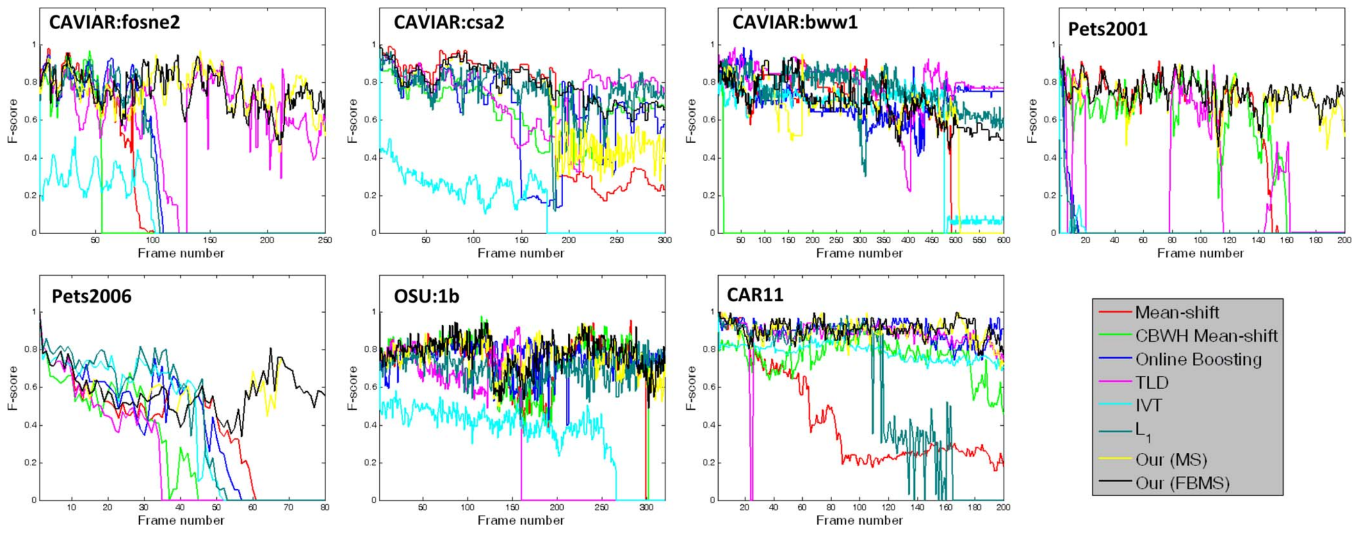

Fig. 8. $\mathbf{F}$**-score** comparisons. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT.

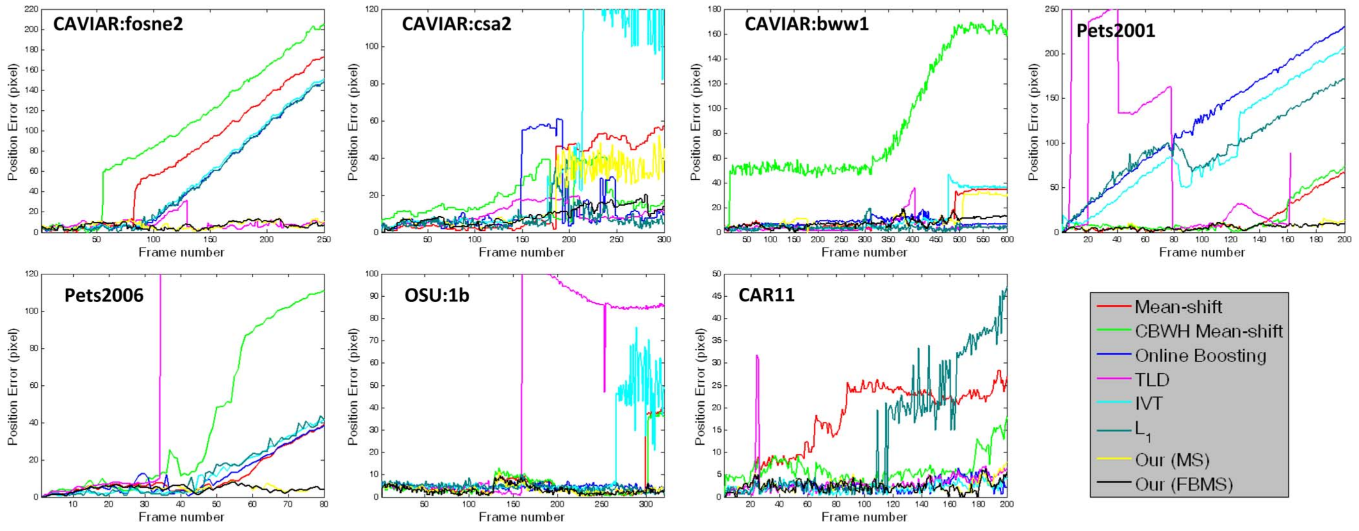

Fig. 9. $\mathbf{Pe}$ comparisons. We compare (yellow) "our method with MS" and (black) "our method with FBMS" with (red) the mean shift, (green) the CBWH mean shift, (blue) online boosting, (pink) TLD, (dark green) L1, and (cyan) IVT.

stable. In particular, our algorithm successfully tracks all video sequences.

Statistical Comparisons: For each video sequence, statistical comparisons are provided from the following two aspects. The first one is the average $\mathbf{F}$**-score** and the average $\mathbf{Pe}$ (see Table I). The second one is the success ratio r (see Table II), which is defined as the ratio between the success frames N_{suc} and the total frames N_{tot}, i.e., $r = (N_{\text{suc}}/N_{\text{tot}})$. The success frames N_{suc} is calculated as follows: $N_{\text{suc}} = \sum_{i=1}^{N_{\text{tot}}} 1_\tau(\mathbf{F}\textbf{-score}(i))$, where τ is the threshold, $\mathbf{F}\textbf{-score}(i)$ is the $\mathbf{F}$**-score** in the ith frame, and $1_\tau(x)$ is a threshold function, which is given by

$$1_\tau(x) = \begin{cases} 1, & x > \tau \\ 0, & \text{otherwise.} \end{cases}$$

Threshold τ is set to 0.5. As shown in Tables I and II, our results are better than the others for most video sequences.

C. Comparison of Computational Cost

The speed of our algorithm is also measured in frames per second (fps). Our tracker is implemented in MATLAB, which runs at 4.48 fps on an Intel Core 2 personal computer with 2.40-GHz central processing unit and 1-GB random access memory. As shown in Table III, our method is faster than L1 and TLD but slower than the mean shift, the CBWH mean shift, online boosting, and IVT. However, the mean shift, the CBWH mean shift, and IVT often fail to track the target, and online boosting is implemented by C++, which is, in general, more efficient than MATLAB.

The 4.48 fps of our method can be almost used for real-time applications. In the future, a large speedup can be expected from code optimization and hardware acceleration. For example, in the near future, we will reimplement our approach in C/C++ and use the graphic processing unit to speed up the guided filter module that is the slowest part of our system.

TABLE I
NUMERICAL COMPARISONS OF THE AVERAGES OF **F-score** AND **Pe**. THE BEST RESULTS ARE INDICATED BY "(A)." THE SECOND BEST RESULTS ARE INDICATED BY "(B)." #: SOME TRACKED RECTANGLES ARE OUT OF THE IMAGE DOMAIN

Videos & Criteria \ Algorithms		Mean-shift	CBWH Mean-shift	Online Boosting	TLD	L_1	IVT	Our (MS)	Our (FBMS)
CAVIAR:fosne2	Pos. Error	73.29	99.79	50.82	8.667	50.84	52.93	7.075(B)	6.672(A)
	F-score	0.257	0.174	0.321	0.637	0.313	0.115	0.753(B)	0.755(A)
CAVIAR:csa2	Pos. Error	20.50	18.54	14.96	9.727	7.190(A)	43.04	17.01	8.652(B)
	F-score	0.647	0.659	0.641	0.735	0.751(B)	0.156	0.691	0.789(A)
CAVIAR:bww1	Pos. Error	11.03	86.08	7.477	4.565(B)	3.906(A)	11.72	10.73	7.523
	F-score	0.613	0.015	0.699	0.793(A)	0.755(B)	0.586	0.589	0.688
Pets2001	Pos. Error	14.68	16.37	123.7	#	94.01	96.77	5.879(B)	5.480(A)
	F-score	0.547	0.530	0.025	0.206	0.018	0.004	0.723(B)	0.741(A)
Pets2006	Pos. Error	11.25	40.65	11.75	144.2	12.64	12.39	4.533(B)	4.433(A)
	F-score	0.383	0.260	0.373	0.223	0.409	0.382	0.569(A)	0.569(A)
OSU:1b	Pos. Error	5.838	4.627	4.035	47.76	4.516	12.53	3.196(B)	3.164(A)
	F-score	0.697	0.703	0.754	0.356	0.697	0.345	0.770(B)	0.777(A)
CAR11	Pos. Error	17.17	6.076	2.532	3.218	13.20	2.127(A)	2.564	2.363(B)
	F-score	0.434	0.770	0.903(B)	0.879	0.589	0.791	0.900	0.910(A)

TABLE II
NUMERICAL COMPARISONS OF THE SUCCESS RATIO. THE BEST RESULTS ARE INDICATED BY "(A)." THE SECOND BEST RESULTS ARE INDICATED BY "(B)"

Videos \ Algorithms	Mean-shift	CBWH Mean-shift	Online Boosting	TLD	L_1	IVT	Our (MS)	Our (FBMS)
CAVIAR:fosne2	0.304	0.212	0.400	0.816	0.388	0.004	0.984(A)	0.984(A)
CAVIAR:csa2	0.613	0.830	0.803	0.886	0.953(B)	0.004	0.713	0.996(A)
CAVIAR:bww1	0.805	0.018	0.975(B)	0.960	0.980(A)	0.790	0.837	0.967
Pets2001	0.705	0.685	0.015	0.235	0.015	0.005	0.980(B)	0.990(A)
Pets2006	0.438	0.325	0.488	0.200	0.550	0.538	0.775(A)	0.775(A)
OSU:1b	0.872	0.859	0.988	0.984	0.988	0.106	0.993(A)	0.993(A)
CAR11	0.325	0.985	1.000(A)	0.990	0.580	1.000(A)	1.000(A)	1.000(A)

TABLE III
COMPARISON OF COMPUTATIONAL COST OR SPEED BY FRAMES PER SECOND

Videos \ Algorithms	Mean-shift	CBWH Mean-shift	Online Boosting	TLD	L_1	IVT	Our
Language	Matlab	Matlab	C++	Matlab/C	Matlab	Matlab	Matlab
Speed	10.1	11.2	12.9	4.03	0.32	8.42	4.48

V. DISCUSSIONS AND CONCLUSION

A. Discussions

The intuitions behind the proposed tracker are mainly from the following two aspects. First, the features of the target should be as different as possible from those of the background. Second, the target should smoothly move in the whole scene. Hence, our tracker can work well in the scenarios where the target and the background are homogeneous and distinguishable. We illustrate three major limitations of our method and their potential solutions as follows.

The first difficulty is that the target may be occluded (either partially or entirely). If the target is partially occluded for a short-term, such as in the **CAVIAR:csa2** and **PET2001** sequences in our experiments, our tracker can track the target well because the visible part can represent the target. However, if the target is occluded for a long time, our tracker may fail. The second difficulty is that the target may move very fast. In such a situation, the smooth process that is implemented by the forward–backward evaluation will be invalid. Therefore, our algorithm can only track the slowly moving targets. To solve the entire occlusion and the fast moving problems, we can add a detection module to reinitialize the tracker whenever it fails. The third difficulty is that the tracked target is seriously deformable. In such case, it is hard to use a fixed rectangle to fit the target. For example, in the **CAVIAR:bww1** sequence, our result is worse than TLD and L1. To solve this problem, we can use a deformable template, such as the deformable rectangle used in L1, to represent the target.

B. Conclusion

In this paper, we have presented the FBMS tracking algorithm with the LBWH. The main contributions of this paper are from two aspects. First, we have proposed a new LBWH, which can effectively separate the target from a very similar background. Second, we have proposed a forward–backward evaluation scheme to make the mean-shift algorithm robust to short-term occlusion. Extensive experimental results have shown that the proposed method is robust and efficient.

In the future, we will incorporate a dynamic model, such as particle filter, into our tracking algorithm to further improve its performance. Moreover, we desire to add other features, such as gradient, contour, and motion features, to enhance target representation, so that our algorithm can be more robust.

References

[1] Y. K. Jung, K. W. Lee, and Y. S. Ho, "Content-based event retrieval using semantic scene interpretation for automated traffic surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 3, pp. 151–163, Sep. 2001.

[2] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 175–187, Jun. 2006.

[3] C. Y. Liu and N. H. C. Yung, "Scale-adaptive spatial appearance feature density approximation for object tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 284–290, Mar. 2011.

[4] B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 425–437, Sep. 2008.

[5] J. Ge, Y. Luo, and G. Tei, "Real-time pedestrian detection and tracking at nighttime for driver-assistance systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 283–298, Jun. 2009.

[6] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 560–571, Dec. 2009.

[7] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, "Vehicle detection and tracking in car video based on motion model," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 583–595, Jun. 2011.

[8] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2000, pp. 142–149.

[9] R. T. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2003, pp. 234–240.

[10] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Robust mean shift tracking with corrected background-weighted histogram," *IET Comput. Vis.*, vol. 6, no. 1, pp. 62–69, Jan. 2011.

[11] X. S. Zhou, A. Gupta, and D. Comaniciu, "An information fusion framework for robust shape tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 1, pp. 115–129, Jan. 2005.

[12] A. Plant, J. Chan, and Y. Hu, "A Kalman filter based tracking scheme with input estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-15, no. 2, pp. 237–244, Mar. 1979.

[13] A. Gelb, *Applied Optimal Estimation*. Cambridge, MA, USA: MIT Press, 1974.

[14] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 1436–1443.

[15] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1–3, pp. 125–141, May 2008.

[16] C. O. Conaire, N. O. Connor, and A. Smeaton, "Thermo-visual feature fusion for object tracking using multiple spatiogram trackers," *Mach. Vis. Appl.*, vol. 19, no. 5/6, pp. 483–494, Sep. 2008.

[17] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1296–1311, Oct. 2003.

[18] M. Isard and A. Blake, "Condensation—Conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, Aug. 1998.

[19] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[20] R. Collins, Y. Liu, and M. Leordeanu, "On-line selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1631–1643, Oct. 2005.

[21] D. Liang, Q. Huang, W. Gao, and H. Yao, "Online selection of discriminative features using Bayes error rate for visual tracking," in *Proc. Pacific-Rim Conf. Multimedia*, 2006, pp. 547–555.

[22] H. T. Nguyen and A. W. M. Smeulders, "Robust tracking using foreground–background texture discrimination," *Int. J. Comput. Vis.*, vol. 69, no. 3, pp. 277–293, Sep. 2006.

[23] M. Wang, H. Qiao, and B. Zhang, "A new algorithm for robust pedestrian tracking based on manifold learning and feature selection," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1195–1208, Dec. 2011.

[24] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.

[25] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. British Mach. Vis. Conf.*, 2006, pp. 47–56.

[26] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 234–247.

[27] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 983–990.

[28] X. Liu and T. Yu, "Gradient feature selection for online boosting," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[29] S. Avidan, "Ensemble tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 494–501.

[30] N. Jojic and B. J. Frey, "Learning flexible sprites in video layers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2001, pp. 199–206.

[31] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vis.*, vol. 26, no. 1, pp. 63–84, Jan. 1998.

[32] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1661–1667, Sep. 2007.

[33] G. Welch and G. Bishop, "An introduction to the Kalman filter," Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, NC, USA, Tech. Rep. 95-041, 1995.

[34] K. Okuma, A. Taleghani, N. D. Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 28–39.

[35] H. T. Niknejad, A. Takeuchi, S. Mita, and D. A. McAllester, "On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 748–758, Jun. 2012.

[36] B. Kwolek, "Object tracking using discriminative feature selection," in *Proc. 8th Int. Conf. Adv. Concepts Intell. Vis. Syst.*, 2006, pp. 287–298.

[37] M. Tang, X. Peng, and D. Chen, "Robust tracking with discriminative rank lists," in *Proc. 10th Asia Conf. Comput. Vis.*, 2010, pp. 283–295.

[38] H. Oike, H. Wu, and T. Wada, "Adaptive selection of non-target cluster centers for k-means tracker," in *Proc. Int. Conf. Pattern Recog.*, 2008, pp. 1–4.

[39] X. Ren and J. Malik, "Tracking as repeated figure/ground segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–8.

[40] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[41] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[42] A. Levin, D. Lischinski, and Y. Weiss, "A closed form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 228–242, Feb. 2008.

[43] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 1–14.

[44] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-backward error: Automatic detection of tracking failures,” in *Proc. 20th Int. Conf. Pattern Recog.*, 2010, pp. 2756–2759.
[45] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-N learning: Bootstrapping binary classifiers by structural constraints,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 49–56.
[46] Z. Kalal, K. Mikolajczyk, and J. Matas, “Face-TLD: Tracking–learning–detection applied to faces,” in *Proc. Int. Conf. Image Process.*, 2010, pp. 3789–3792.

Lingfeng Wang received the B.S. degree in computer science from Wuhan University, Wuhan, China, in 2007. He is currently working toward the Ph.D. degree with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His research interests include computer vision and image processing.

Hongping Yan received the Ph.D. degree from the Chinese Academy of Sciences, Beijing, China, in 2002.

In 2002, she went to Japan and France for her postdoctoral research. In 2004, she returned to China. She is an currently an Associate Professor with China University of Geosciences, Beijing. She was invited to the Institute of Automation and Technology in France as a Visiting Scholar for cooperative research. She is currently responsible for several national research projects. Her research interests are in computer graphics, image processing, and 3-D reconstruction.

Huai-Yu Wu received the B.E. and M.E. degrees from Beijing University of Aeronautics and Astronautics, Beijing, China, in 2000 and 2003, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from Chinese Academy of Sciences, Beijing, in 2008.

From July 2008 to August 2011, he was a Postdoctoral Lecturer with the School of EECS, Peking University, Beijing. He is currently an Associate Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests are in the fields of 3-D computer vision, visual shape perception and analysis, and interactive computer graphics.

Chunhong Pan received the B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1987 and the M.S. degree and the Ph.D. degree in pattern recognition and intelligent system from the Chinese Academy of Sciences, Beijing, in 1990 and 2000, respectively.

He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, image processing, computer graphics, and remote sensing.