Multimed Tools Appl (2016) 75:2393–2409
DOI 10.1007/s11042-014-2411-6

A real-time visual object tracking system based on Kalman filter and MB-LBP feature matching

Zebin Cai · Zhenghui Gu · Zhu Liang Yu · Hao Liu · Ke Zhang

Received: 24 February 2014 / Revised: 21 October 2014 / Accepted: 3 December 2014 /
Published online: 19 December 2014

Abstract Visual tracking has very important applications in practice. Many proposed visual trackers are not suitable for real-time applications because of their huge computational loads or sensitivities against changing environments such as illumination variation. In this paper, we propose a new tracker which uses modified Multi-scale Block Local Binary Patterns (MB-LBP) like feature to characterize the tracked object. Such feature has low computational load and robustness against illumination variation. An updated appearance model is build based on the modified MB-LBP feature. The model is updated in every frame by replacing the appearance model with the features extracted from the most current detected image patch of target. Moreover, we use the predicted information about the target to constructed a smaller searching area for target in new frame. It greatly reduces computational load for target searching. Numerical experiments show that the drift effect of tracker is greatly avoided and the tracker has very effective and robust performance on various test videos.

Keywords Object tracking · Feature matching · Kalman filter · Multi-scale block local binary patterns

1 Introduction

Visual tracking is a hot research topic in computer vision. The visual tracker predicts locations of certain objects in video sequences and has wide applications in visual

Z. Cai · Z. Gu (✉) · Z. L. Yu
College of Automation Science and Engineering, South China University of Technology, Guangzhou, China
e-mail: zhgu@scut.edu.cn

H. Liu
Beijing Transportation Information Center, Beijing, China

K. Zhang
Beijing Transportation Operation Coordination Center, Beijing, China

surveillance, object counting, navigation, guidance and so on. Although there are many methods proposed for such applications [9, 11, 22, 33], to derive a robust tracker working under a wide range of practical environments needs to consider the following factors, like the lack of prior knowledge about the object, changing environments etc. Moreover, in some applications, the interested object may not have a rigid body, so its appearance and size may change over time. The working environments are complicated, like the illumination variation, partial or full object occlusions etc. All of these factors will make the object tracking to be a very challenging task.

Generally, a visual tracking system consists of three functional models: (1) Appearance model, which is used to characterize the object so that the tracked object can be distinguished from non-objects; (2) Motion model, which describes the object's motion trail. Based on this model, various tracking methods can be applied to predict the future potential location of the target; (3) Searching strategy, which is guided to search a image patch that matches the target object in video sequences. Many methods proposed in literatures [19, 34, 39] are different in the above three aspects. For example, regarding the processing of appearance model, some tracking algorithms use only the first frame to build an appearance model [12, 16, 21]. They are able to track rigid body objects well in some certain situations. However, such methods can not cope with those targets which suffer from serious appearance changing because non-rigid body objects can change their appearances over time and become even completely different from the one in the first frame. In order to overcome this shortcoming, recently, some online model learning algorithms were proposed to deal with the appearance change problem during object tracking [6, 7, 11, 24, 40]. They select discriminative features of the target from video sequences and use these features to search the most probable target patch in new video frames. The appearance model is then updated based on the new founded target patch.

Regarding the processing of motion model, various methods can be applied to track the trajectory of object [38]. With a linear sate-space motion model and assumption on Gaussian noises, Kalman filter [26] is an theoretically optimal tracker. It has been proved to be very successful in practical applications. If the assumptions on model and noise deviate from the above ones, extended Kalman fitler [15], Unscent Kalman filter (UKF) [23] and particle filter [20] are suitable choices depending on different assumptions adopted. Theoretically, particle filter does not need assumptions on linear model and Gaussian noise. It may has higher tracking accuracy in practical applications. However, considering the restrictions on computational resources in practice, Kalman filter achieves a good compromise between performance and computational complexity.

Searching strategy largely depends on appearance model as well as features used to build appearance model. One of the most important factors is the feature selected for object matching. In [6, 24, 40, 41], the Haar-like features or generalized Haar-like features are used in building appearance model and searching for target in frames. The system based on Haar-like features is not so robust because the Haar-like features are sensitive to illumination changes. There are many other robust features to be adopted, like, SIFT [30], SURF [8], histogram of Block Binary Patterns (LBP) [31], etc. Although some of these features, like SIFT and SURF [8, 30] etc., demonstrate high performance in tracking, most of them are not suitable for online processing because of their high computational load.

The target of this paper is to develop a robust real-time object tracking system. To achieve this goal, a suitable kind of feature must be selected which has low computational load and low sensitivity against environment variations like illumination variation. The LBP [31] as a powerful and efficient texture operator is robust against illumination variation. Hence,

we modify the variation of LBP, Multi-scale Block Local Binary Patterns (MB-LBP) [31, 32], as the feature for object tracking. Normally, the MB-LBP is calculated for pixels in an image and the histogram of the target's MB-LBP is used as feature. For efficient processing, we introduce two modifications on MB-LBP. Firstly, the MB-LBP is only calculated at some randomly selected points in the image patch of target. Secondly, we use MB-LBP string directly as feature instead of using its histogram. A new efficient compare operator for two MB-LBP feature vectors is developed. We also adopt the Kalman filtering for object tracking, so that the estimated object location and its variance can be obtained simultaneously [15]. Moreover, the Kalman filter can predict the expectation and variance of the target location. Such information is further used in target searching so that the search area could be greatly narrowed down. Due to the potential appearance changes during tracking, the offline trained detectors [9] are not suitable for online tracking. In some methods [40, 41], though the appearance model evolves during the tracking process, the update scheme relates the current model to the past target features heavily. This confuses the tracker and degrades the tracker's performance as the time goes on. In this paper, we propose a simple yet powerful appearance model updating scheme. It just uses the object's feature vector in the last frame as appearance model for object searching. Experiments on challenging video sequences show that the proposed method is more effective, efficient and robust than many of the state-of-the-art methods [4, 6, 13, 14, 24, 40].

The remainder of this paper is organized as follows. In Section 2, we give a brief introduction on MB-LBP and then propose our new operator for comparing two MB-LBP feature strings. The proposed tracking system is presented in Section 3, including detail implementations of appearance model updating, motion tracking and target searching. In Section 4, we compare the performance of the proposed system with some other state-of-the-art methods on some challenging video sequences. Brief conclusions and discussions on the proposed system can be found in Section 5.

2 Modified MB-LBP feature

The LBP [31] operator is a powerful and efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel with the value of the center pixel and considers the result as a binary number. An illustration of the basic LBP is shown in Fig. 1, where the center pixel is compared with its surrounding pixels. The resulting binary pattern stands for the feature of that point. Commonly, this binary pattern can be transferred to be an integer number if we select a most significant bit and the bit order. The integer number is called LBP label which is always used to derive histogram of LBP [31].

The discriminative LBP features extracted represent the picture's texture. This feature is robust against monotonic gray-scale change caused by illumination variation, which is

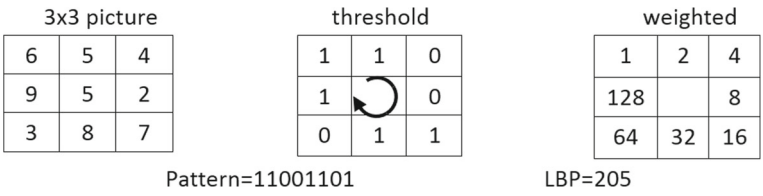


Fig. 1 Illustration of LBP

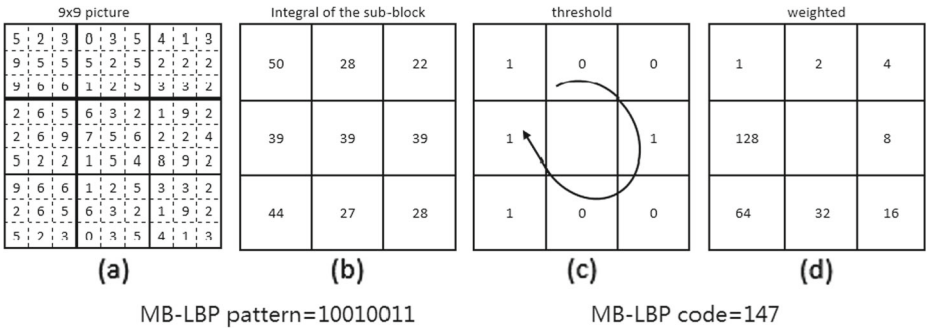


Fig. 2 Illustration of MB-LBP

the character that why the LBP is superior to Haar-like feature. Even LBP has slight more computational cost than the Haar-like feature, it finds a lot of practical applications, especially in face recognition [5, 28]. Due to the computational simplicity[1] and robustness of LBP, it is a good choice of potential features for tracking object in real time.

There are many variants of LBP [32] and in this paper we are interested in one of its variations called MB-LBP [32]. In [28], it has been proved that the basic LBP only encodes the micro-structures and is easy to be affected by noise while MB-LBP encodes both the micro-structures and the macro-structures of image. Therefore, the MB-LBP is more robust to characterize a target than the basic LBP. In this paper, we introduce the concept of MB-LBP using a 9×9 MB-LBP sample in Fig. 2. The MB-LBP can be considered as LBP whose single pixel is replaced by the sum of pixels of 3×3 sub-block. The computational load for one MB-LBP feature point is at most 72 additions and 8 comparisons.

Generally, after obtaining the MB-LBP/LBP, the histogram of the target's MB-LBP/LBP is calculated and used as feature for further processing. For example, in [5], a face image is divided into $k \times k$ subregions and the corresponding LBP code of every pixel are calculated. Histograms of LBP in every subregion are then calculated and concatenated into an enhanced histogram vector. Finally, the face recognition is carried out by comparing the enhanced histogram vector. In [28], it uses a similar way to recognize a face with MB-LBP feature histogram.

In this paper, we don't use histogram of MB-LBP as feature vector. Instead, we use the MB-LBP binary pattern directly. Since each bit in the binary pattern has equal importance in object matching, we define a comparator for two MB-LBP patterns ξ_x and ξ_y. Since the MB-LBP pattern is a binary string, we compare the corresponding bit of binary strings, and the number of different bits is taken as the output result

$$diff(\xi_x, \xi_y) \triangleq \sum_{n=0}^{7} \xi_x^{(n)} \oplus \xi_y^{(n)}, \tag{1}$$

$\oplus$ is the XOR operator; $\xi_x^{(n)}$ stands for the nth bit of ξ_x.

By using the MB-LBP pattern as feature and the comparator defined in (1), we can use the MB-LBP pattern to build appearance model of object in a new way. Details are presented in Section 3.1.

[1] Compared with the other robust features like SIFT and SURF etc., the computation of LBP has much lower computational complexity.

3 Proposed tracking method

The overall diagram of the proposed tracking algorithm is shown in Fig. 3. Firstly, the initial appearance model is constructed using the features extracted from the image patch (represented by a bounding box) of the interested target in the first frame. Then, the tracker processes iteratively as follows. With the estimated location and uncertainty range of the target in tth frame, the Kalman filter is used to predict the location and uncertain range of the target in $(t+1)$th frame. The observed location of the target in the $(t+1)$th frame is obtained by matching the appearance model with the features extracted from the area constructed by the predicted location and uncertain range. The observed location of the target in $(t+1)$th frame is finally used to update the Kalman filter to get the optimal location and uncertainty range. The features extracted from the image patch centered at that optimal location is used to update the appearance model. The whole tracking process works iteratively until stop criteria meets.

3.1 Appearance model

In the application of MB-LBP in face recognition, the MB-LBP pattern is calculated for every pixel of the target. All the MB-LBP patterns are used to construct histogram for distinguishing a face from the others. Such process has a heavy computational cost and is difficult to run in real time. Fortunately, for the target tracking, we can use a simpler method similar to [41] that only extracts feature points randomly from the target patch. Such simple strategy works well if the tracking target and background have different appearances [41].

In this paper, we use $\mathbf{l}_t$ to represent the target's location in tth frame and $\mathbf{l}_t(\mathbf{X})$ to denote the center location of an image patch $\mathbf{X}$ in tth frame. For an image patch $\mathbf{X}$ centered at location $\mathbf{l}_t(\mathbf{X})$ and of size (w, h) (w and h are the width and height of image patch respectively), a set of pixels are randomly selected from the patch with uniform probability. These selected pixels are called feature points in the following context. As shown in Fig. 3, there are k feature points selected and the MB-LBP pattern $\{\xi_{i,t}\}$ of these feature points are used to construct a vector $\mathbf{f}_t = [\xi_{1,t}, \xi_{2,t}, \cdots, \xi_{k,t}]^T$, which represents the appearance model of the target in tth frame.

In image processing, the feature points are normally selected as the corners [35], edge [10] or blob [29], etc, because these kinds of feature points are more robust. However, the shortcoming of selecting such feature points is its expensive computational cost (feature point detection and matching). In this paper, we use the features from randomly selected points. They may not be the optimal ones but they are simple to operate in practice.

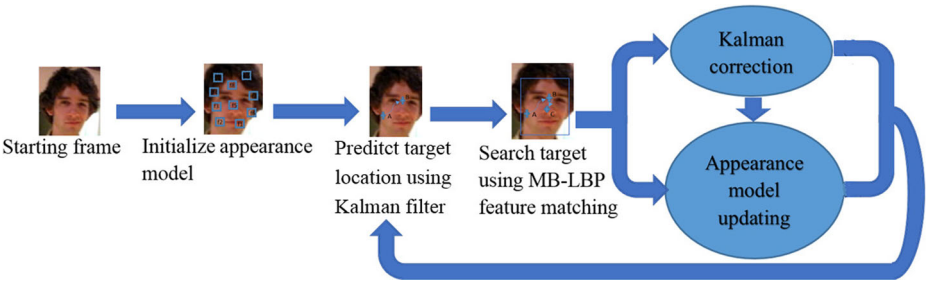


Fig. 3 Diagram of the proposed tracking algorithm

Moreover, numerical experiments demonstrate that if we select sufficient number of feature points, like $k = 40$ in all the experiments, the randomly selected feature points produce high performance in object tracking and demonstrate robustness under various environments. Also, since the feature points are generated randomly, which are not for a specific object so that it could be used to characterize more general objects. Generally, once the feature points are selected randomly in the target image patch in the first frame, these selected points are always used for the whole tracking process. This process can greatly accelerate the appearance model updating.

3.2 Motion model and updating

Without loss of generality, the motion of a target can be approximately represented as a uniform linear motion in a short interval. Hence, the target's location $\mathbf{l}_t = [u_t, v_t]^T$, where u_t, v_t represent the vertical and horizontal coordinates of target in tth frame respectively, can be expressed as

$$\mathbf{l}_t = \mathbf{l}_{t-1} + \Delta\mathbf{l}_{t-1}\tau + \mathbf{w}_t, \tag{2}$$

where $\mathbf{w}_t = [w_{u,t}, w_{v,t}]^T$, $w_{u,t}, w_{v,t}$ are the approximation errors of vertical and horizontal coordinates respectively and assumed to be random zero-mean Gaussian noises with variance σ_w^2. We denote the speed of target as $\Delta \mathrm{l}_t = [du_t/dt, dv_t/dt]^T$ and τ is time interval between frames. In practice, the speed of target may change in tracking, so we can consider that the speed in tth frame is a noise disturbed version of the speed in $(t-1)$th frame

$$\Delta\mathbf{l}_t = \Delta\mathbf{l}_{t-1} + \mathbf{dw}_t, \tag{3}$$

where $\mathbf{dw}_t = [dw_{u,t}, dw_{v,t}]^T$ and $dw_{u,t}, dw_{v,t}$ are the disturb noises of vertical and horizontal speeds respectively. Both of them are zero-mean Gaussian noises with variance $\boldsymbol{\sigma}_{dw}^2$.

Express the state vector as $\mathbf{x}_t = [u_t, v_t, du_t/dt, dv_t/dt]^T$, and the observation of $\mathbf{l}_t$ as $\mathbf{z}_t$, we have the state space model as

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{n}_t \\ \mathbf{z}_t &= \mathbf{H}\mathbf{x}_t + \mathbf{m}_t \end{aligned}, \tag{4}$$

where the random noises $\mathbf{n}_t = [\mathbf{w}_t^T, \mathbf{dw}_t^T]^T$ is called process noise with diagonal covariance matrix $\mathbf{Q} = diag\{\sigma_w^2, \sigma_w^2, \sigma_{dw}^2, \sigma_{dw}^2\}$ and $\mathbf{m}_t = [m_{u,t}, m_{v,t}]^T$ represents zero-mean measurement noise with diagonal covariance matrix $\mathbf{R} = diag\{\sigma_m^2, \sigma_m^2\}$, and $m_{u,t}, m_{v,t}$ are the measurement noises of vertical and horizontal coordinates respectively. The matrix $\mathbf{A}$ is the state transition matrix and $\mathbf{H}$ is the measurement matrix. They are given as follows

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \tau & 0 \\ 0 & 1 & 0 & \tau \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{6}$$

With the state space model given in (4), the task of tracking is to estimate $\mathbf{x}_t$ with given noisy observation $\mathbf{z}_t$. Kalman filter [15] is an optimal estimator if the model is linear and with Gaussian noises. The two key steps "prediction" and "correction" of Kalman filter are introduced as follows. More details on implementation can refer to [36].

Prediction:

$$\begin{aligned} \hat{\mathbf{x}}_t^- &= \mathbf{A}\hat{\mathbf{x}}_{t-1} \\ \mathbf{P}_t^- &= \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q} \end{aligned}, \tag{7}$$

where $\hat{\mathbf{x}}_t^-$ stands for the predict of $\mathbf{x}_t$. The prediction step uses $\hat{\mathbf{x}}_{t-1}$, the optimal estimate of $\mathbf{x}_{t-1}$, to predict the location of target in tth frame. The matrix $\mathbf{P}_t^-$ stands for the covariance matrix of $\hat{\mathbf{x}}_t^-$. The matrix $\mathbf{P}_{t-1}$ is the covariance matrix of estimated vector $\hat{\mathbf{x}}_{t-1}$.

With given observation $\mathbf{z}_t$, the correction step can be applied to estimate the optimal state vector $\hat{\mathbf{x}}_t$ as

Correction:

$$\begin{aligned} \hat{\mathbf{x}}_t &= \hat{\mathbf{x}}_t^- + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^-) \\ \mathbf{P}_t &= (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_t^- \\ \mathbf{K}_t &= \mathbf{P}_t^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_t^-\mathbf{H}^T + \mathbf{R})^{-1} \end{aligned}, \tag{8}$$

where $\mathbf{K}_t$ is the Kalman gain matrix, $\mathbf{P}_t$ is the covariance matrix of estimated vector $\hat{\mathbf{x}}_t$.

From the viewpoint of motion tracking, "Prediction" and "Correction" in Kalman filter are related to "Estimate of Coarse Location" and "Fine Tuning of Location" in tracking. With the information about the tracked target in $(t-1)$th frame, we can use the Prediction process to do a coarse estimation of the location $\hat{\mathbf{x}}_t^-$ of target as well as the uncertainty of this estimate (covariance matrix $\mathbf{P}_t^-$). The location of target follows Gaussian distribution with mean $\hat{\mathbf{x}}_t^-$ and covariance matrix $\mathbf{P}_t^-$. If a noisy observation of target location $\mathbf{z}_t$ is obtained, then we can fine tune the estimate of $\mathbf{x}_t$ using the predicted location as prior information. As shown in Fig. 4, the predicted location is B, then after fine tuning the location with observed data $\mathbf{z}_t$, the optimal estimated location is C.

A problem remains here that how to get an observation of the location of target, i.e. $\mathbf{z}_t$? To answer this question, we need to introduce two techniques: feature matching and target searching. Feature matching is achieved by comparing the difference of two MB-LBP patterns. Target searching looks for a image patch with minimum feature difference to the appearance model of target. Most of the tracking algorithms [6, 24, 40, 41] directly search the patch that best matching the tracked target in new frame around the last tracked location. If the target has large movement between adjacent frames, a large searching radius has to be used. A large searching area finally results in more candidate locations and higher computational cost.

In this paper, we search for the image patch $\mathbf{X}$ which best matches the appearance model of previous detected target at location $\mathbf{l}_{t-1}$. We limit the search area of image patch as $\mathbf{X}^s = \{\mathbf{X} : ||\mathbf{l}_t(\mathbf{X}) - \mathbf{l}_{t-1}|| \leq s\}$, where $s = \alpha\sqrt{trace\{\mathbf{P}_t^-\}}$ and α is a parameter to control

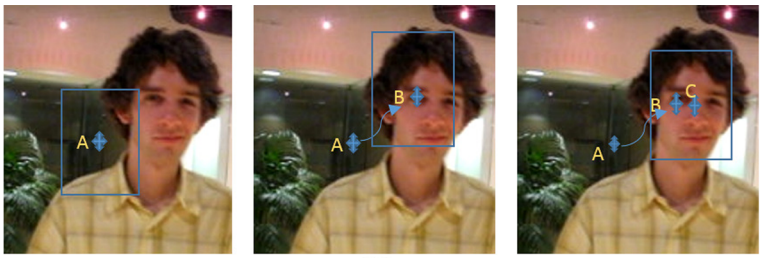


Fig. 4 Illustration of Kalman filter in motion tracking. **a** Point A represents the location in the previous frame. **b** The coarse location B after Kalman prediction. **c** The optimal location C after Kalman correction

the search radius, it can be set as $0 \leq \alpha \leq 3$ for a variable with Gaussian distribution. The observation of target location is found by

$$\mathbf{z}_t = \mathbf{l}_t(\arg\min_{\mathbf{X}\in\mathbf{X}^s} \mathbf{w}^T * Diff(\mathbf{f}_{t-1}, \mathbf{f}(X))), \tag{9}$$

where $\mathbf{f}_{t-1}$ is the appearance model of target in $(t-1)$th frame, and $\mathbf{f}(\mathbf{X})$ stands for the function to construct the appearance model of image patch $\mathbf{X}$. The difference operator for two appearance model vectors $\mathbf{f}_{t-1}, \mathbf{f}_t$ is defined as

$$Diff(\mathbf{f}_{t-1}, \mathbf{f}_t) = [diff(\xi_{1,t-1}, \xi_{1,t}), \cdots, diff(\xi_{k,t-1}, \xi_{k,t})]^T. \tag{10}$$

The vector $\mathbf{w} = [\boldsymbol{\omega}_{\xi_1}, \boldsymbol{\omega}_{\xi_2}, \cdots, \boldsymbol{\omega}_{\xi_k}]^T$ is used to weight the importance of each feature points in matching. A Gaussian weighted model can be selected for $\mathbf{w}$ with

$$\boldsymbol{\omega}_{\xi_i} = e^{-\frac{|i-u_c|+|j-v_c|}{\epsilon*(w+h)}}, \tag{11}$$

where (i, j) is the coordinate of the feature point, (u_c, v_c) is the coordinate of the center point of the patch. (w, h) is width and height of image patch. ϵ is a constant. The reason why we use the above weighting vector $\mathbf{w}$ is given as follows. Since we use a bounding box to crop the target or image patch, the feature points close to the center of the image patch is more probably to be part of the tracked target, the feature points far away from the center of image patch may contain some background information. Hence, the feature points close to the center of patch is emphasized in searching of the target.

Finally, the proposed algorithm is summarized in Algorithm 1.

Algorithm 1 Summary of the Proposed Method

Require: Given the initial location $\mathbf{l}_0$ and appearance model $\mathbf{f}_0 = \{\xi_1, \xi_2, \cdots, \xi_k\}$, initial state vector $\mathbf{x}_0$, noise covariance matrices $\mathbf{R}$ and $\mathbf{Q}$;

1: **for** $t = 1$ to K **do**

2: Predict the state vector $\mathbf{x}_t^-$ and prior covariance matrix $\mathbf{P}_t^-$ using (7);

3: Search the matched target patch $\mathbf{X}_t = \arg\min_{\mathbf{X}\in\mathbf{X}^s} \mathbf{w}^T * Diff(\mathbf{f}_{t-1}, \mathbf{f}(\mathbf{X}))$ and find its location as $\mathbf{z}_t = \mathbf{l}_t(\mathbf{X}_t)$, where $\mathbf{X}^s = \{\mathbf{X} : ||\mathbf{l}_t(\mathbf{X}) - \mathbf{l}_{t-1}|| \leq s\}$;

4: Correct the predicted location with $\mathbf{z}_t$ using (8);

5: Update the target's location $\mathbf{l}_t$ as the first two items of the Kalman filter output $\hat{\mathbf{x}}_t$ in (8), and update appearance model $\mathbf{f}_t = \mathbf{f}(\hat{\mathbf{X}}_t)$, where $\hat{\mathbf{X}}_t$ is the image patch centered at estimated optimal target location $\mathbf{l}_t$;

6: **end for**

In practice, two important problems must be considered, target initialization and miss-tracking of target. Every tracking method requires target initialization mechanism when the object first appears in the video. The target initialization method could be manual or automatic [38]. A common automatic approach for target initialization is to use information in a single frame or make use of the temporal information computed from a sequence of frames. Classical methods include point detectors, background subtraction, segmentation and supervised learning based approaches [38]. In practice, selecting a target initialization approach depends on the application. Since the target of this paper is to propose a new object tracking method, for simplicity, we initialize the target by manually selecting it from the first frame of video. Miss-tracking may be caused by many factors, for example, abrupt object motion, camera motion, object-to-object and object-to-scene occlusions, changing

appearance patterns (target rotating, illumination variation, non-rigid object structures), etc. In the proposed method, we adopt a common approach to handle miss-tracking [38]. The proposed method searches the target around the area centered at the location predicted by Kalman filter. If no matched target found (the matching error is larger than a threshold), we skip the processing step of correction, i.e., use the predicted location as the observed location of the target, and repeat the Kalman filtering until the object reappears. After N repeats, if the target still cannot be found, the target is considered as lost and the tracking process is stoped.

4 Numerical experiments

A benchmark for online object tracking is proposed in [37]. Many state-of-the-art methods are evaluated on some popular databases [37]. Similar as [37], in this section, we also evaluate the performance of proposed method with several state-of-the-art methods on 9 public available challenging video sequences which have been widely used to test video trackers. The selected trackers are Multiple Instance Learning (MIL) [6], the online AdaBoost method (OAB) [13], the online Discriminative Feature Selection (ODFS) [24], Semi-supervised online boosting (SemiBoost) [14], the fragment tracker (Frag) [4], Incremental learning for robust Visual Tracker (IVT) [33],Tracking-Learning-Detection (TLD) [25], Visual Tracking Decomposition (VTD) [27] and Circulant Structure of Tracking-by-Detection with Kernels (CSK) [18]. The source codes or the binary codes provided by the authors are used in performance evaluation. As some trackers randomly generate the features of training samples, so all the tracking results are averaged for 5 runs in the experiments. Since the tracker OAB will stop tracking when the target is lost, we only use the successful results for performance comparison. The 10 video sequences are $Board$ [1], *Car4, David_indoor, trellis* [3], *dollar, faceocc, faceocc2, sufer, sylv* [2] and $skating1$. For video *Board, dollar, faceocc, faceocc2, sufer, skating1* and $sylv$, we use the ground true location of target in the video sequences provided by the author, while for the sequences *Car4, David_indoor* and $trellis$, we compute the ground true location by ourselves.

Why we choose the above 10 video sequences for performance evaluation? The reason is that these video sequences contain some different challenging factors. These 10 video

Table 1 Average center location error

Sequences	MIL	Proposed	OAB	ODFS	SemiBoost	Frag	IVT	TLD	CSK	VTD
Board	*79.3*	**36.1**	116.3	96.3	244.8	90.1	147.8	310.2	86.35	105
Car4	119.2	*6.2*	**5.8**	74.8	22.5	104.4	11.1	165.5	8.232	48.8
David_indooor	46.9	*16.1*	33.7	38.2	49.3	105.5	220.7	144.8	**13.53**	71.1
Dollar	13.0	5.8	*5.0*	17.0	94.7	56.1	18.3	100.8	**3.44**	63.4
Faceocc	15.1	16.9	15.7	28.3	47.2	*5.5*	12.7	12.9	**5.2**	9.1
Faceocc2	15.9	13.7	39.1	22.5	48.4	59.1	15.3	73.7	**7.2**	*11.1*
Sufer	137.2	**9.3**	10.5	213.2	61.2	199.4	152.7	250.3	10.0	*9.89*
Sylv	532.3	**5.6**	13.5	15.6	66.3	11.3	158.1	64.1	*5.8*	11.3
Trellis	88.0	**21.7**	60.3	68.6	107.1	104.7	138.7	135.1	*21.8*	31.5
Skating1	172.5	*12.6*	**11.8**	167.5	fail	114.8	145.4	fail	13.9	100.7

(Bold font indicates the best performance, italic font indicates the second best.)

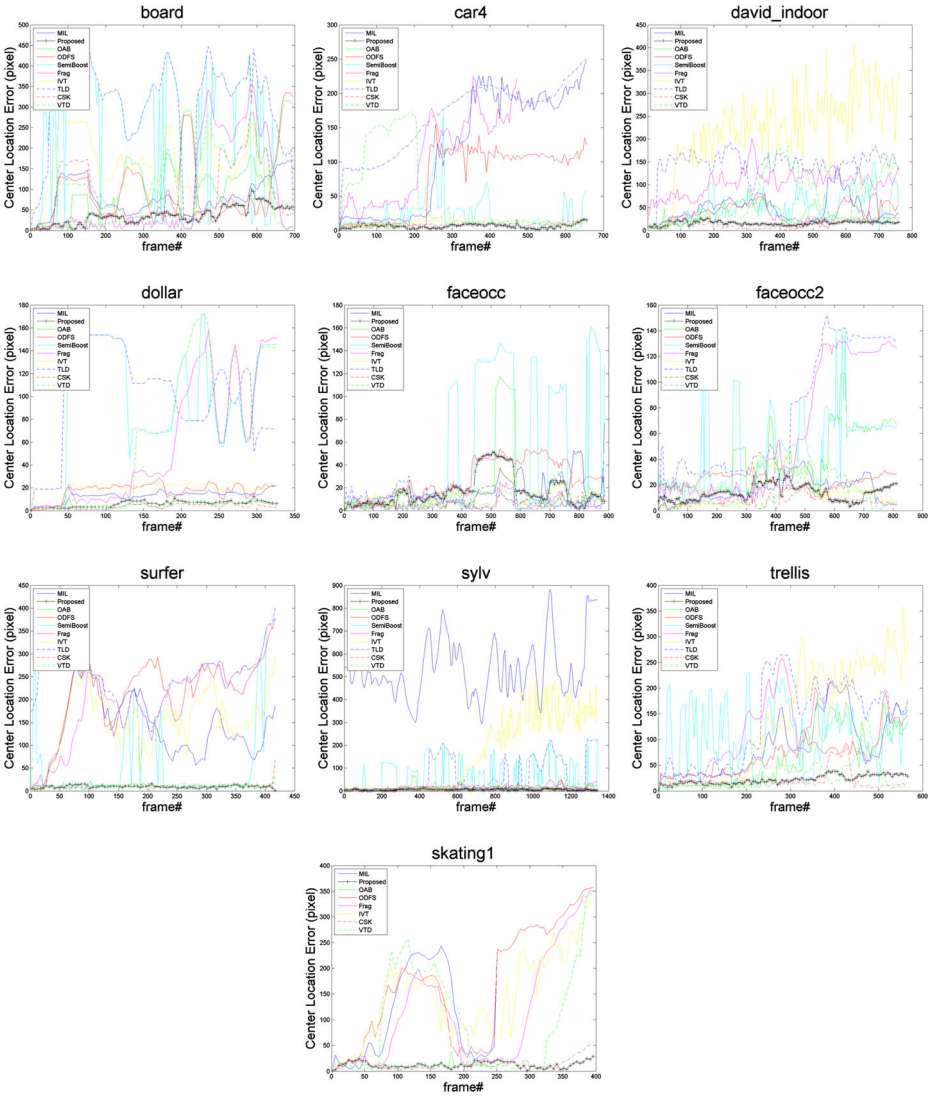
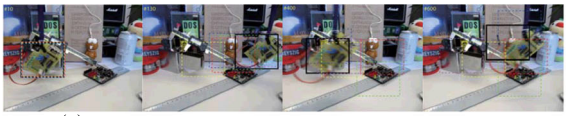


Fig. 5 Error plots of the evaluated methods on different testing video sequences

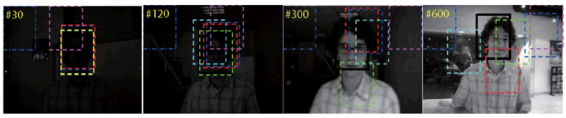
sequences can be separated into four classes. Video $Car4$ (car)[2], *David_indoor* (face), $Sylv$ (toy), $skating1$ (human) and $trellis$ (face) comprise challenging scale, heavily illumination and pose changes. Both video $faceocc$ (face) and $faceocc2$ (face) suffer from seriously partial occlusions and $faceocc2$ presents swing pose. For video $Board$ (PCB board) and $sufer$ (human head) contain scale changes and object rotation. And the last video *dollar* exhibits two similar objects with the similar scale and appearance in same time.

[2]The item in the bracket is the object to track.

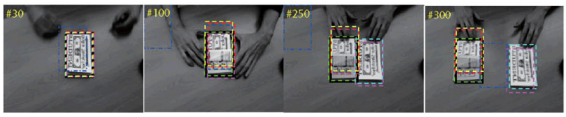


(a) Snapshots of processed board sequence (#10, #130, #400, #600).

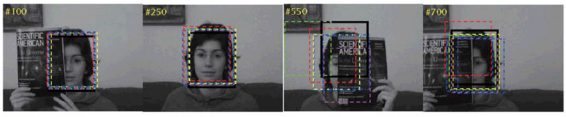
(b) Snapshots of processed car4 sequence (#50, #150, #300, #500).



(c) Snapshots of processed david_indoor sequence (#30, #120, #300, #600).

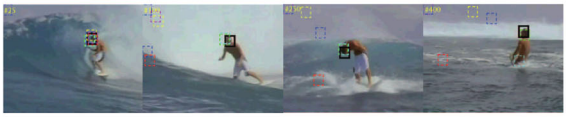


(d) Snapshots of processed dollar sequence (#10, #100, #250, #300).

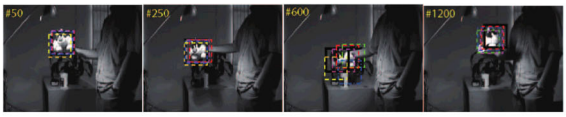


(e) Snapshots of processed faceocc sequence (#100, #250, #550, #700).

(f) Snapshots of processed faceocc2 sequence (#100, #150, #500, #700).



(g) Snapshots of processed surfer sequence (#25, #100, #250, #400).



(h) Snapshots of processed sylv sequence (#50, #250, #600, #1200).

(i) Snapshots of processed trellis sequence (#50, #250, #300, #400).

Fig. 6 Snapshots of the tracking results

For the proposed method, we set the number of feature points $k = 40$ in experiments. Although we test the tracker on many sequences, this parameter works well for all video sequences. The parameter to control the searching range is set as $\alpha = 2$. For calculating MB-LBP, a 9×9 template is used.

In order to illustrate the effectiveness and robustness of the proposed method, we adopt the performance evaluation methodologies [6] in this paper. The error of center location e_c of target is calculated for every frame as

$$e_c = ||\mathbf{l}_t - \mathbf{l}_{ground\ true}||. \tag{12}$$

In Table 1, the center location error of each method on all testing video sequences are shown. From the viewpoint of center location error, the proposed method produces very good results on most of the testing sequence. In Fig. 5, we show the center location error of each method on all testing video sequences. In Fig. 6, we also show some snapshots selected from the tracking results. From these results, it is clear that the proposed tracker always has low center location error for all sequences. We can also find that some trackers lose the tracking of target in the middle of the sequences and produce large center location errors. In the subfigure of skating1, we omit the results of SemiBoost and TLD because they lost the target at the very beginning of processing. This experiment reveals that the proposed method has robustness against various video sequences taken in different environments and for different applications.

The success rate is another index to evaluate the effectiveness of the trackers. It is defined as the percentage of the frames whose center location errors are within a threshold. In our experiment, the threshold is chosen as 20. The results are shown in Table 2. All the results of average center location error and the success rate illustrate that the proposed method produces the top 2 best results in most of the cases. It should be noted that the CSK also produces very good performance in experiments. It is observed that CSK only lost the tracking of target in *Board* experiment. These experiments support that the proposed method is effective and robust compared with many other trackers.

Processing speed is also an key index to evaluate the efficiency of the trackers. In this Section, we evaluated all the trackers on the video sequences. All the trackers run on a PC with *Intel(R)core 3.2 GHz CPU* and *8Gb RAM*. The processing speed on a test video sequence is calculated as

$$P_s = \frac{n_f}{T}, \tag{13}$$

Table 2 Success rate (Bold font indicates the best performance, italic font indicates the second best)

Sequences	MIL	Proposed	OAB	ODFS	SemiBoost	Frag	IVT	TLD	CSK	VTD
Board	0.079	*0.242*	0.207	0.100	*0.242*	**0.514**	0.107	0.021	0.086	0.157
Car4	0.288	**1.000**	**1.000**	0.356	*0.659*	0.042	1.0	0.008	**1.000**	0.606
David_indooor	0.112	**0.815**	0.223	0.223	0.217	0.000	0.099	0.033	*0.810*	0.164
Dollar	*0.939*	**1.000**	**1.000**	0.545	0.151	0.409	0.38	0.136	**1.000**	0.424
Faceocc	0.814	0.764	0.910	0.398	0.651	*0.960*	0.82	0.76	**1.000**	0.899
Faceocc2	0.638	*0.822*	0.355	0.392	0.276	0.398	0.72	0.056	**1.000**	0.81
Sufer	0.071	**1.000**	0.976	0.071	0.702	0.047	0.06	0.0	*0.988*	*0.988*
Sylv	0.000	**1.000**	0.773	0.728	0.479	0.858	0.45	0.51	*0.996*	0.844
Trellis	0.008	*0.464*	0.438	0.008	0.122	0.259	0.021	0.06	**0.825**	0.45
Skating1	0.105	*0.845*	**0.93**	0.095	fail	0.273	0.058	fail	0.843	0.465

Table 3 Comparison of processing speed (f/s:frames per second)

Method	Board	Car4	David	Dollar	Faceocc	Faceocc2	Surfer	Sylv	Trellis	Skating1
Proposed	28.3	25.2	36.5	26.3	34.5	27.8	30.6	29.4	27.9	28
MIL	4.4	5.0	5.0	4.9	4.9	4.9	4.8	4.9	5.0	4.5
OAB	12.5	21.2	16.1	16.1	9.3	12.5	21.2	21.2	15.2	15.8
ODFS	52.1	119.0	113.0	113.1	117.0	114.6	77.7	126.8	121.4	64.7
SemiBoost	9.1	12.8	10.2	10.0	6.2	8.0	15	15.5	10.6	12
Frag	0.08	0.27	0.27	0.27	0.22	0.27	0.15	0.28	0.26	0.11
IVT	13.6	16.5	21.1	16.1	15.7	16.2	13.9	16.9	18.9	12.78
TLD	0.7	0.8	0.9	0.8	0.8	0.8	0.7	0.8	0.9	0.8
STRUCK	0.16	0.14	0.16	0.21	0.14	0.15	0.17	0.18	0.13	0.14
CSK	123.8	273	133.7	171	183.2	206.0	507.7	264.3	208.9	246
VTD	0.6	0.42	0.81	0.78	0.55	0.83	0.65	0.72	0.84	0.9

where the processing speed P_s is characterized by number of processed frames per second, n_f represents the number of frames processed and T is the processing time. From the evaluation results shown in Table 3, we can find that the proposed method has a fast processing speed. It can process at least 27 frames per second. Hence, it is a potential solution for real-time video tracking applications. From Table 3, we can also find that the CSK and $ODFS$ are more efficient than other methods. However, the $ODFS$ has poor performance on average location error and success rate as shown in Table 1–2. CSK also has high performance in tracking. The STRUCK tracker [17] has too high computational load so that it can only process about 0.16 frames per second. Hence, it is not suitable for real-time applications with limited computational resource.

5 Discussions and conclusions

A new method for real time video tracking is proposed in this paper. In the proposed tracker, the MB-LBP like feature with modified feature comparing method is used to characterize the target's appearance model and search for the target in new frame. Kalman filter is adopted to predict the target's location as well as the uncertainty range of estimation. Such information is used to construct a smaller compact search area for finding the target in new frame. This idea not only greatly reduces the search radius but also make the tracker more robust against fast motion. The results of the numerical experiments show the effectiveness and robustness of the proposed method. However, the proposed method still has some drawbacks. For example, the proposed method is developed based on sparse sampling strategy. When it is compared with the recently proposed method, CSK, under dense sampling strategy and exploiting circulant structure of tracking-by-detection with kernels, the CSK has much higher processing speed than the proposed one. If we apply some skills like integral image and input image downsampling (used in CSK) into the proposed method, there still exist some rooms for processing speed improvement. Moreover, since the MB-LBP feature is rotation variant, the proposed tracker cannot track the target with very fast rotation. However, in the experiments, we find that proposed method can still track the rotating target very well while the CSK lost the tracking ($Board$ experiment). In our future work, we will try to merge the benefits of CSK and the modified MB-LBP like feature into a more robust and efficient tracker.

Acknowledgments The authors would like to thank the associate editor and the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was supported in part by the National Natural Science Foundation of China under grants 61105121 and 61175114, the Natural Science Foundation of Guangdong under grants S2012020010945, the High Level Talent Project of Guangdong Province 2013KJCX0009, China Postdoctoral Science Foundation 2014M560060.

References

1. http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/prost.php
2. http://vision.ucsd.edu/bbabenko/project_miltrack.shtml
3. http://www.cs.toronto.edu/dross/ivt/
4. Adam A, Rivlin E, Shimshoni I (2006) Robust fragments-based tracking using the integral histogram. In: Proceedings of the IEEE conference computing vision pattern recognition, vol 1, pp 798–805
5. Ahonen T, Hadid A, Pietik A, Inen M (2004) Face recognition with local binary patterns. In: European conference on computer vision, vol 3021, pp 469–481
6. Babenko B, Ming-Hsuan Y, Belongie S (2009) Visual tracking with online multiple instance learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 983–990
7. Balan AO, Black MJ (2006) An adaptive appearance model approach for model-based articulated object tracking. In: Proceedings of the IEEE Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, pp 758–765
8. Bay H, Tuytelaars T, Van Gool L (2006) Surf: Speeded up robust features. In: European conference on computer vision, vol 3951, pp 404–417
9. Black MJ, Jepson AD (1998) Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. Int'l J Comput Vision 26(1):63–84
10. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
11. Collins RT, Liu Y, Leordeanu M (2005) Online selection of discriminative tracking features. IEEE Trans Pattern Anal Mach Intell 27(10):1631–1643
12. Comaniciu D, Ramesh V, Meer P (2000) Real-time tracking of non-rigid objects using mean shift. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 2, pp 142–149
13. Grabner H, Grabner M, Bischof H (2006) Real-time tracking via on-line boosting. In: Proceedings of the british machines vision conference, vol 1, p 6
14. Grabner H, Leistner C, Bischof H (2008) Semi-supervised on-line boosting for robust tracking. In: European conference on computer vision, vol 5302, pp 234–247
15. Grewal MS, Andrews AP (2001) Kalman filtering: Theory and practice using matlab. John Wiley & Sons, 2011
16. Hager GD, Belhumeur PN (1998) Efficient region tracking with parametric models of geometry and illumination. IEEE Trans Pattern Anal Mach Intell 20(10):1025–1039
17. Hare S, Saffari A, Torr PHS (2011) Struck: Structured output tracking with kernels. In: Proceedings of the international conference on computer vision, pp 263–270
18. Henriques JOF, Caseiro R, Martins P, Batista J (2012) Exploiting the circulant structure of tracking-by-detection with kernels. In: Computer Vision-ECCV 2012. Springer, Berlin Heidelberg, 2012, pp 702–715
19. Isard M, Blake A (1996) Contour tracking by stochastic propagation of conditional density. In: European conference on computer vision, vol 1064, pp 343–356
20. Isard M, Blake A (1998) Condensation conditional density propagation for visual tracking. Int J Comput Vis 29(1):5–28
21. Isard M, MacCormick J (2001) Bramble: A bayesian multiple-blob tracker. In: Proceedings of the international conference on computer vision, vol 2, pp 34–41
22. Jepson AD, Fleet DJ, El-Maraghi TF (2003) Robust online appearance models for visual tracking. IEEE Trans Pattern Anal Mach Intell 25(10):1296–1311
23. Julier SJ, Uhlmann JK (1997) New extension of the kalman filter to nonlinear systems. In: Proceedings of the international symposium aerospace defense sensing, vol 3068, pp 182–193
24. Kaihua Z, Lei Z, Ming-Hsuan Y (2013) Real-time object tracking via online discriminative feature selection. IEEE Trans Image Process 22(12):4664–4677
25. Kalal Z, Mikolajczyk K, Matas J (2012) Tracking-learning-detection. IEEE Trans Pattern Anal Mach Intell 34(7):1409–1422

26. Kalman RE (1960) A new approach to linear filtering and prediction problems. Trans ASME-J Basic Eng 82(1):35–45
27. Kwon J, Kwon J, Lee KM, Lee KM (2010) Visual tracking decomposition. In: Proceedings of the CVPR, pp 1269–1276. IEEE
28. Liao S, Zhu X, Lei Z, Zhang L, Li SZ (2007) Learning multi-scale block local binary patterns for face recognition. In: Proceedings of the international conference biometrics, vol 4642, pp 828–837
29. Lindeberg T (1993) Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. Int J Comput Vision 11(3):283–318
30. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
31. Ojala T, Pietik A, Inen M, Harwood D (1996) A comparative study of texture measures with classification based on featured distributions. Pattern Recogn 29(1):51–59
32. Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24(7):971–987
33. Ross DA, Lim J, Lin RS, Yang MH (2008) Incremental learning for robust visual tracking. Int'l J Comput Vision 77(1-3):125–141
34. Salzmann M, Lepetit V, Fua P (2007) Deformable surface tracking ambiguities. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–8
35. Shi J, Tomasi C (1994) Good features to track. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, pp 593–600
36. Welch G, Bishop G (1995) An introduction to the kalman filter. In: University of North Carolina at Chapel Hill Tech. Rep. TR-95-041
37. Wu Y, Lim J, Yang MH (2013) Online object tracking: A benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2411–2418
38. Yilmaz A, Javed O, Shah M (2006) Object tracking: A survey. ACM Comput Surv 38(4):13
39. Zeisl B, Leistner C, Saffari A, Bischof H (2010) On-line semi-supervised multiple-instance boosting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1879–1879
40. Zhang K, Song H (2012) Real-time visual tracking via online weighted multiple instance learning. Pattern Recogn 46:397–411
41. Zhang K, Zhang L, Yang MH (2012) Real-time compressive tracking. In: European conference on computer vision, vol 7574, pp 864–877

Zebin Cai received the B.S. degree in Automation Science and Engineering from South China University of Technology, Guangzhou, China, in 2012. He is currently working toward the M.S. degree in pattern recognition and intelligent systems, South China University of Technology, Guangzhou, China. His research interests include video and image processing.

Zhenghui Gu received the Ph.D. degree from Nanyang Technological University, Singapore, in 2003. From 2002 to 2008, she was with the Institute for Infocomm Research, Singapore. In 2008, she joined the College of Automation Science and Engineering, South China University of Technology, Guangzhou, as an associate professor. Her current research interests include the fields of signal processing and pattern recognition.

Zhu Liang Yu received his BSEE in 1995 and MSEE in 1998, both in electronic engineering from the Nanjing University of Aeronautics and Astronautics, China. He received his Ph.D. in 2006 from Nanyang Technological University, Singapore. He joined Center for Signal Processing, Nanyang Technological University from 2000 as a research engineer, then as a Group Leader from 2001. In 2008, he joined the College of Automation Science and Engineering, South China University of Technology and was promoted to be a full professor in 2011. His research interests include video and image processing, array signal processing, pattern recognition, machine learning and their applications in communications, biomedical engineering, etc.

Hao Liu received his Bachelor degree in 1998 in Transportation School of Southeast University. Afterwards, he completed his master dissertation and obtained M.Sc. degree in 2002 in Research Institute of Highway (RIOH), Ministry of Transport. He then started to work for Chinese National Intelligent Transport Systems Research Center of Engineering and Technology (ITSC). In November 2003, he was selected by RIOH to start his Ph.D. study at the Transportation and Planning Section of the Delft University of Technology, the Netherlands. He received his Ph.D. degree in 2008. He continued his research work in ITSC until 2011. Then he worked in department of Science and Technology, Ministry of Transport for two year. In 2013, he joined Beijing Transportation Information Center. His research interests include transportation modeling, intelligent transport system applications.

Ke Zhang received his Bachelor degree in 1995 in Mathematics School of Northwest University of China. Afterwards, he completed his master dissertation and obtained M.Sc. degree in 1998 in Mathematics School of Beijing University of Technology. He received his Ph.D. degree in 2001 in Transportation School of Beijing University of Technology. He then started to work for Chinese National Intelligent Transport Systems Research Center of Engineering and Technology (ITSC), Research Institute of Highway (RIOH), Ministry of Transport. In 2011, he joined Beijing Municipal Transportation Operations Coordination Center(TOCC). His research interests include transportation operations surveillance, transportation data analysis, intelligent transport system (ITS) planning and applications.