Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

Robust object tracking using linear neighborhood propagation

Chen Gong
Keren Fu
Enmei Tu
Jie Yang
Xiangjian He

Robust object tracking using linear neighborhood propagation

Chen Gong
Keren Fu
Enmei Tu
Jie Yang
Shanghai Jiao Tong University
School of Electronic Information and Electrical Engineering
Department of Automation
800 DongChuan Road
200240, China
E-mail: goodgongchen@sjtu.edu.cn

Xiangjian He
University of Technology Sydney
School of Computing and Communications
Department of Computer Science
Broadway NSW 2007 2000, Australia

Abstract. *Object tracking is widely used in many applications such as intelligent surveillance, scene understanding, and behavior analysis. Graph-based semisupervised learning has been introduced to deal with specific tracking problems. However, existing algorithms following this idea solely focus on the pairwise relationship between samples and hence could decrease the classification accuracy for unlabeled samples. On the contrary, we regard tracking as a one-class classification issue and present a novel graph-based semi-supervised tracker. The proposed tracker uses linear neighborhood propagation, which aims to exploit the local information around each data point. Moreover, the manifold structure embedded in the whole sample set is discovered to allow the tracker to better model the target appearance, which is crucial to resisting the appearance variations of the object. Experiments on some public-domain sequences show that the proposed tracker can exhibit reliable tracking performance in the presence of partial occlusions, complicated background, and appearance changes, etc.* © *2013 SPIE and IS&T [DOI: 10.1117/1.JEI.22.1.013015]*

1 Introduction

As a traditional research topic, numerous researchers have extensively studied object tracking through various methods. Tracking aims to precisely associate the target in consecutive video frames while maintaining its identity. However, this goal is far from being reached because the tracking process is often affected by many unexpected disturbances, such as occlusion, the target's appearance variation, a sudden background or illumination change, etc. Therefore, making tracking more robust and adaptive still represents an important research subject.

Paper 12223 received Jun. 9, 2012; revised manuscript received Dec. 21, 2012; accepted for publication Jan. 10, 2013; published online Jan. 25, 2013.

In existing tracking algorithms, the region of target is usually represented as a point,[1] contour,[2] skeleton,[3] or some primitive geometric shapes such as a rectangle[4] and an ellipse. Point representation is suitable for small objects. A contour or skeleton is a good choice for tracking a nonrigid target with significant appearance variations. In our case, the image region of the object is represented as a rectangle because this representation is more efficient and is not overly sensitive to the slight appearance changes of the target between adjacent frames.

Feature selection is critical for tracking and is closely related to object representation. Color feature is commonly adopted for histogram-based representations,[5] while edge is often used as a feature when the target is represented by a contour. However, these features are not robust to illumination changes. Thus, the Gabor wavelet, which is a kind of texture feature, is implemented over the rectangular region of the target in our algorithm. The Gabor wavelets can capture the energy information in different directions and scales and hence provide better adaptability in the presence of illumination changes.

The mechanisms of traditional trackers can be roughly classified into two types: motion based and matching based. The motion-based approach associates the object in consecutive frames by exploiting the motion information. For example, Kanede-Lucas tracker (KLT)[6] adopted optical flow constraint to describe the relationship between the gray value of the image and the structural variation of the target. Matching-based tracker aims to identify the target that is the most similar to the historical records. Mean-shift tracker[5] used kernel technique to represent the similarity between the candidate regions and template. Kalman filter tracker[7] and particle filter[8] tracker searched the candidate regions for matching by estimating the position of object. Various

tracking algorithms are summarized in the work of Yilmaz et al.[9]

Machine learning has been introduced to the tracking domain in recent years. One or more discriminative classifiers are often used to enable a tracker to learn the specificity and variations of a target's appearance. Avidan proposed the support vector tracker[10] by using a support vector machine to discriminate the target and the background. Grabner et al. used the framework of an on-line AdaBoost to select "good" features for tracking.[4,11] However, these trackers are easily confused under a complicated background, and their re-detection ability after losing the target is weak. To enhance stability, multiple instance learning was applied to tracking.[12,13] Instead of classifying a single instance, this learning strategy classifies the bags consisting of several instances. The bag is positive as long as it contains one positive instance. Therefore, severe drifting can be avoided when the target is located with slight inaccuracy.

Moreover, some graph-based semisupervised learning algorithms, such as the harmonic function,[14] have also been introduced to address the tracking problem. A large number of unlabeled samples are adopted in this method to increase the discriminative ability of the tracker. However, this method has two main limitations. First, the graph is constructed simply using the pairwise relationship of the sample measured by Euclidean distance and hence cannot effectively discover the property of a large complex dataset with high-dimensional samples. Second, this algorithm regards tracking as a binary classification problem and attempts to use very few negative training samples to describe overly complex backgrounds.

To overcome these two defects, the proposed algorithm treats tracking as a one-class semisupervised classification problem and establishes the graphical model using local information around each sample. When the target moves around in the scenario, the background varies continually while the target itself substantially remains unchanged. Therefore, the collected positive samples can sufficiently represent the appearance of the target. On the contrary, the negative samples, which are needed for binary classification methods, cannot properly capture the variations of the background. Thus we only focus on the positive samples that represent the target areas.

Moreover, the notation of a manifold has been verified in many scientific works[15,16,17] that postulate that the variable high-dimensional observations of the target can be embedded in a low-dimensional geometry. Hence, finding the manifold hidden in the different appearances of the target during the tracking process will help the tracker learn the object essentially, which will be a step forward to a robust performance (Fig. 1). Given the aforementioned concepts, a graph-based semisupervised learning method called linear neighborhood propagation (LNP)[18] is suitable for tracking because it uses local information of the samples to discover the low-dimensional structure when the graph is established. Our tracker is thus named the LNP tracker. However, unlike in the standard LNP algorithm, target location is regarded as a one-class classification problem in our tracking case to meet the necessary criteria of "specificity versus variability."

The structure of this paper is as follows. In Sec. 2, the framework of the proposed method is introduced. Details of our tracking algorithm are explained in Sec. 3. Section 4 illustrates the experimental results on the public sequences and provides corresponding analysis. Finally, a conclusion is drawn in Sec. 5.

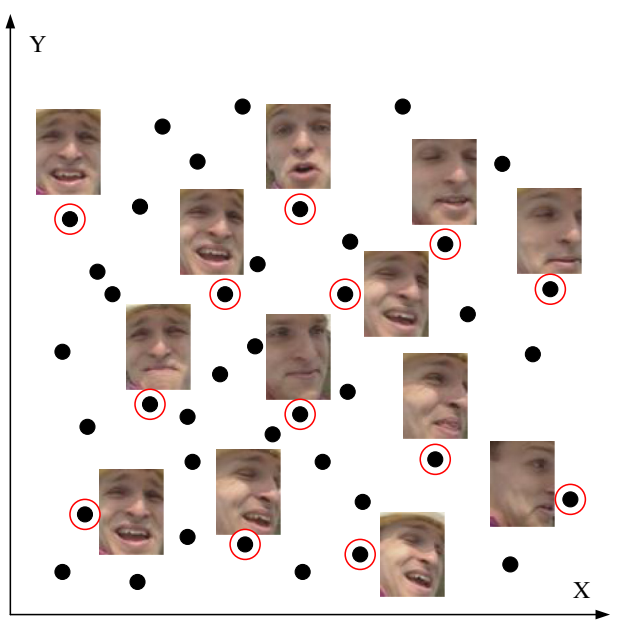


Fig. 1 Illustration of a manifold. The face images are extracted from the *Foreman* sequence used in the experimental section. The appearances of the face differ in many aspects, such as in observation angle, illumination, and expression. However, these face images can be finely projected onto a low-dimensional embedding space.

2 Framework

The flow chart of our approach is shown in Fig. 2. First, a target $TG^{(1)}$ (the number in superscript is used to denote the frame number) is chosen manually in the first frame. Then a

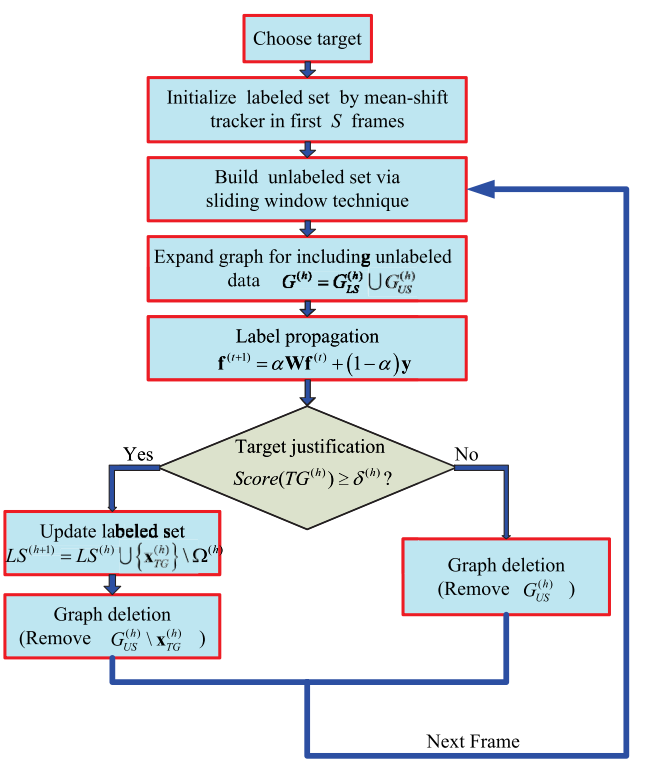


Fig. 2 Flow chart of the LNP tracker.

temporary tracker (e.g., mean-shift tracker) is utilized in the subsequent $S-1$ frames to collect several positive samples (i.e., $\mathrm{TG}^{(2)}, \mathrm{TG}^{(3)}, \cdots, \mathrm{TG}^{(S)}$ tracked by mean-shift in every frame) to establish the labeled dataset $\mathrm{LS}^{(h)} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{l^{(h)}}\}$ ($l^{(h)} = S$ at present) and the corresponding graph $G_{\mathrm{LS}}^{(h)}$. Subsequently, the LNP tracker replaces the mean-shift tracker. When the h-th ($h > S$) frame arrives, $u^{(h)}$ unlabeled samples are collected by the sliding window technique to form the unlabeled set $\mathrm{US}^{(h)} = \{\mathbf{x}_{l^{(h)}+1}, \mathbf{x}_{l^{(h)}+2}, \cdots, \mathbf{x}_{l^{(h)}+u^{(h)}}\}$. The samples are combined with $G_{\mathrm{LS}}^{(h)}$ to construct $G^{(h)} = G_{\mathrm{LS}}^{(h)} \cup G_{\mathrm{US}}^{(h)}$ ($G_{\mathrm{US}}^{(h)}$ is the subgraph in $G^{(h)}$ corresponding to $\mathrm{US}^{(h)}$), of which the weight matrix is denoted as $\mathbf{W}$. Afterward, labels of positive samples are propagated to unlabeled ones through edges. The label prediction vector $\mathbf{f}^* = (\mathbf{f}_{\mathrm{LS}}^T \vdots \mathbf{f}_{\mathrm{US}}^T)^T$ can then be obtained. The sample that has the largest value in $\mathbf{f}_{\mathrm{US}}^T$ is taken as $\mathrm{TG}^{(h)}$. Finally, $\mathrm{TG}^{(h)}$ and the samples gathered in the previous $M-1$ frames are preserved in $G^{(h)}$, while the remaining samples in $\mathbf{f}_{\mathrm{US}}^T$ are deleted. This procedure is iterated until the end of the sequence.

3 LNP Tracker in Detail

3.1 *Temporary Tracker*

After the target is annotated in the first frame, an existing simple tracker begins to work in the subsequent $S-1$ frames to obtain a small number of labeled samples. Here S should not be set very large, and any tracker can locate the target precisely in such a short period. Hence we call this tracker a temporary tracker. In this paper, the mean-shift tracker is employed, and S is set to 20.

3.2 *Graph Construction*

In a graph-based semisupervised learning algorithm, a graph is represented as $G = \langle V, E \rangle$, where V and E stand for the vertex and the edge sets, respectively. Samples are represented by vertices, and their similarity is described by the edge weights. Once the first S frames of the sequence have been processed, the LNP tracker begins to deal with the many unlabeled samples. At this time, an initial graph $G_{\mathrm{LS}}^{(h)}$ ($h = S$ currently) is established using $\mathrm{LS}^{(S)}$. Existing methods often use the Gaussian kernel to define the similarity between samples $\mathbf{x}_1$ and $\mathbf{x}_2$ (Refs. 14 and 19) such as in

$$\omega_{12} = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right). \tag{1}$$

However, Euclidean distance cannot honestly reflect the similarity of the high-dimensional samples in the complex dataset.[17] Therefore, this approach for graph construction is not applicable in the present study. Instead of considering the pairwise relationship between instances as Eq. (1), this paper focuses on the local neighborhood information of samples. The benefit of our method to graph construction is discussed in Sec. 4.1.

Initially, K nearest neighbors of $\mathbf{x}_i (i = 1, 2, \cdots, S)$ are calculated by the Euclidean distance because in a small local area, this similarity measurement is reliable. Subsequently, in graph $G_{\mathrm{LS}}^{(S)}$, $\mathbf{x}_i$ can be optimally expressed as the linear combination of its K neighbors. The objective function of each $\mathbf{x}_i$ is defined as

$$\varepsilon_i = \left\| \mathbf{x}_i - \sum_{j:\mathbf{x}_{ij} \in N(\mathbf{x}_i)} \omega_{ij} \mathbf{x}_{ij} \right\|^2, \tag{2}$$

where $N(\mathbf{x}_i)$ denotes the neighbors of $\mathbf{x}_i$, and coefficient ω_{ij} is the contribution of $\mathbf{x}_{ij}$ to $\mathbf{x}_i$. This contribution describes the similarity between $\mathbf{x}_{ij}$ and $\mathbf{x}_i$. The main idea behind this method is borrowed from a well-known manifold learning algorithm called locally linear embedding (LLE).[15] The LLE and our method pay attention to the local information around a data point. Therefore in graph $G_{\mathrm{LS}}^{(S)}$, the similarities between samples are determined by considering the manifold structure hidden in a dataset. Furthermore, the sum of the weights of the neighbors should be restricted to 1. Thus the additional constraints are formulated as

$$\sum_j \omega_{ij} = 1, \qquad \omega_{ij} \geq 0. \tag{3}$$

Unlike the graph established in Ref. 14, the weight matrix $(\mathbf{W})_{ij} = \omega_{ij}$ here is no longer symmetrical because it is a sparse matrix in which $\omega_{ij} > 0$ if $\mathbf{x}_j \in N(\mathbf{x}_i)$ and $\omega_{ij} = 0$ if $\mathbf{x}_j \notin N(\mathbf{x}_i)$.

Reference 18 also pointed out that Eqs. (2) and (3) can be written in the matrix formation as follows:

$$\begin{aligned} &\min_{\boldsymbol{\omega}_{ij}} \sum_{j,k:\mathbf{x}_j,\mathbf{x}_k \in N(\mathbf{x}_i)} \omega_{ij} \mathbf{H}_{jk} \omega_{ik} \\ &s.t. \sum_j \omega_{ij} = 1, \omega_{ij} \geq 0 \end{aligned}, \tag{4}$$

where $\mathbf{H}_{jk}$ is the (j,k)'th element of matrix $\mathbf{H}$, and $\mathbf{H}_{jk}$ equals to $(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_k)$ at the point $\mathbf{x}_i$. Equation is a standard quadratic programming problem and can easily be solved by existing methods. $G_{\mathrm{LS}}^{(S)}$ is then found by solving S optimization problems such as in Eq. (4). As the tracking process continues, the initial graph $G_{\mathrm{LS}}^{(S)}$ is updated incrementally through graph expansion and graph deletion operations, which are introduced in Secs. 3.3 and 3.5, respectively.

3.3 *Graph Expansion*

In the frame h ($h \geq S+1$), a search window slides in the region near the $\mathrm{TG}^{(h-1)}$ to collect $u^{(h)}$ unlabeled samples $\mathbf{x}_{l^{(h)}+1}, \mathbf{x}_{l^{(h)}+2} \cdots \mathbf{x}_{l^{(h)}+u^{(h)}}$. As this paper does not consider the scale change of the target, the size of the search window in all frames remains the same. Next, $G_{\mathrm{LS}}^{(h)}$ should be expanded to a larger graph $G^{(h)}$ to include these unlabeled samples, namely, $G^{(h)} = G_{\mathrm{LS}}^{(h)} \cup G_{\mathrm{US}}^{(h)}$, as mentioned in Sec. 2 (Fig. 3). Reconstructing the graph here is unnecessary and inefficient because this process needs to calculate K neighbors of all the instances and seek the solutions to Eq. (4). However, only a small fraction of samples require researching neighbors actually. Thus, an incremental graph expansion strategy comprising two steps is adopted.[20] Unlabeled samples $\mathbf{x}_{l^{(h)}+1}$ to $\mathbf{x}_{l^{(h)}+u^{(h)}}$ are inserted into $G^{(h)}$ one by one. When we search K neighbors of a query point $\mathbf{x}_q \in \mathrm{US}^{(h)}$, $q = l^{(h)}+1, l^{(h)}+2, \ldots, l^{(h)}+u^{(h)}$, a hyper sphere Θ centered on $\mathbf{x}_q$ (Fig. 3) is first defined by the radius given as

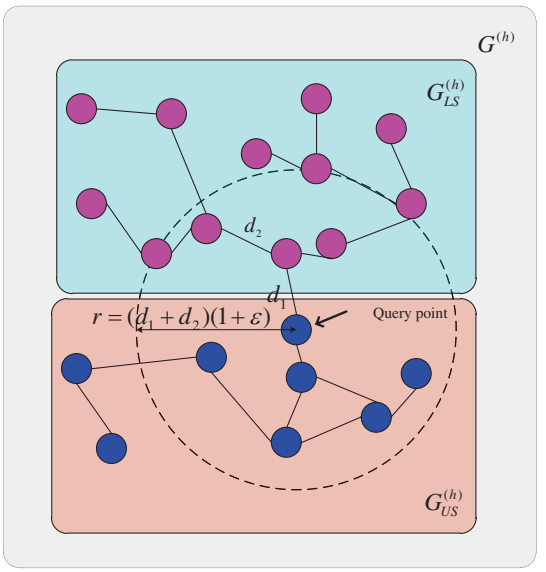


Fig. 3 Incremental graph expansion. The pink and blue dots represent the labeled samples and the unlabeled samples that are added to the original graph, respectively. The dashed circle indicates the hyper sphere in which the samples should update their neighbors.

$$r=(d_1+d_2)(1+\varepsilon). \tag{5}$$

In Eq. (5), ε is a constant that is set to 0.5. d_1 is the Euclidean distance between $\mathbf{x}_q$ and its 1-NN $\mathbf{x}_{q1}$, and d_2 is the Euclidean distance between $\mathbf{x}_{q1}$ and its furthest neighbor $\mathbf{x}_{q1K}$. Next, K neighbors of all the points in Θ are recalculated. After $G^{(h)}$ is expanded, its weight matrix $\mathbf{W}$ is modified accordingly using the same method explained in Sec. 3.2. The complexity of inserting one $\mathbf{x}_q$ following this method is $O[2(q-1)+n_\Theta^3]$, where n_Θ is the number of samples in the Θ, and $n_\Theta \ll q-1$. However, if the graph is rebuilt from scratch when a new point $\mathbf{x}_q$ is inserted, the complexity is as high as $O(q^3)$.

3.4 *Label Propagation*

Label propagation aims to transmit the label information from labeled samples to unlabeled ones through graph G. Suppose that function f can assign a real value f_i to the corresponding sample $\mathbf{x}_i$. Then the soft label of samples is given by $f_i=f(\mathbf{x}_i)$, where $i=1,2\ldots l^{(h)}, l^{(h)}+1, l^{(h)}+2,\ldots, l^{(h)}+u^{(h)}$. The initial state in frame h is defined as

$$\mathbf{y}=(\mathbf{y}_{\mathrm{LS}}^T \vdots \mathbf{y}_{\mathrm{US}}^T)^T, \tag{6}$$

where

$$\mathbf{y}_{\mathrm{LS}}=(y_1\, y_2\ldots y_{l^{(h)}})^T$$
$$\mathbf{y}_{\mathrm{US}}=(y_{l^{(h)}+1}, y_{l^{(h)}+2}\ldots y_{l^{(h)}+u^{(h)}})^T$$

with elements

$$y_i=\begin{cases}1, 1\le i\le l^{(h)}\\ 0, l^{(h)}+1\le i\le l^{(h)}+u^{(h)}\end{cases}.$$

Then the label prediction vector $\mathbf{f}^*$ can be derived iteratively as

$$\mathbf{f}^{(t+1)}=\alpha\mathbf{W}\mathbf{f}^{(t)}+(1-\alpha)\mathbf{y}. \tag{7}$$

In Eq. (7), $\mathbf{W}$ is the weight matrix obtained in Sec. 3.2. α is the trade-off between the initial state and the information from the neighbors of samples. $\mathbf{f}^{(t)}=[f_1^{(t)}\ldots f_{l^{(h)}}^{(t)} f_{l^{(h)}+1}^{(t)}\ \cdots\ f_{l^{(h)}+u^{(h)}}^{(t)}]^T$ is the column vector at iteration t, and $\mathbf{f}^{(0)}=\mathbf{y}$. The equation is proven to converge to[18]

$$\mathbf{f}^*=\lim_{t\to\infty}\mathbf{f}^{(t)}\triangleq(\mathbf{f}_{\mathrm{LS}}^T \vdots \mathbf{f}_{\mathrm{US}}^T)^T, \tag{8}$$

where $\mathbf{f}_{LS}^T$ is the subvector corresponding to $\mathrm{LS}^{(h)}$, and $\mathbf{f}_{\mathrm{US}}^T$ is for $\mathrm{US}^{(h)}$. Here the convergence indicates that the ordering of the magnitude of the elements in $\mathbf{f}^*$ does not change for several successive iterations. Every element in $\mathbf{f}^*$ is above zero because tracking is regarded as a one-class classification task. Therefore the target in frame h is the sample with the largest component in $\mathbf{f}_{\mathrm{US}}$, as indicated in the equation below:

$$\mathrm{TG}^{(h)}=[\mathbf{x}_i | i=l^{(h)}+\arg\max_r(\mathbf{f}_{\mathrm{US}})_r, r=1,2\cdots u^{(h)}]. \tag{9}$$

The confidence score of the target location in frame h is defined as

$$\mathrm{Score}(\mathrm{TG}^{(h)})=\max_r(\mathbf{f}_{\mathrm{US}})_r. \tag{10}$$

3.5 *Labeled Set Updating and Graph Deletion*

$\mathrm{TG}^{(h)}$ can be determined as the final tracking target only if $\mathrm{Score}(\mathrm{TG}^{(h)})$ exceeds a certain threshold $\delta^{(h)}$, because extremely low $\mathrm{Score}(\mathrm{TG}^{(h)})$ indicates that the LNP tracker is not confident on the result. This uncertainty is usually caused by large appearance changes or significant occlusions. These uncertain or inaccurate samples should not be added to the labeled set because they may cause drifting problems. $\delta^{(h)}$ is determined through an adaptive way[21] such that if $\mathrm{Score}(\mathrm{TG}^{(h)})$ is $\lambda\sigma^{(h)}$ lower than the mean level from $\mathrm{Score}(\mathrm{TG}^{(S+1)})$ to $\mathrm{Score}(\mathrm{TG}^{(h-1)})$, $\mathrm{TG}^{(h)}$ being a final target is invalid. Therefore

$$\delta^{(h)}=\mu^{(h)}-\lambda\sigma^{(h)} \tag{11}$$

with the mean $\mu^{(h)}$ and the variance $\sigma^{(h)}$ formulated as

$$\begin{cases}u^{(h)}=\frac{1}{h-S-2}\sum_{j=S+1}^{h-1}\mathrm{Score}(\mathrm{TG}^{(j)})\\ \sigma^{(h)}=\sqrt{\frac{1}{h-S-2}\sum_{j=S+1}^{h-1}[\mathrm{Score}(\mathrm{TG}^{(j)})-u^{(h)}]^2}\end{cases}, \quad h>S+2. \tag{12}$$

We use $\mathbf{x}_{\mathrm{TG}}^{(h)}$ to denote the sample that is finally selected as the target in frame h. Then $\mathbf{x}_{\mathrm{TG}}^{(h)}$ is added to $\mathrm{LS}^{(h)}$ to establish $\mathrm{LS}^{(h+1)}$ for building graph $G^{(h+1)}$, as explained in Sec. 3.3. Moreover, positive samples collected before frames $h-T(h\ge T)$ are eliminated from $\mathrm{LS}^{(h)}$ beforehand, thereby allowing the LNP tracker to learn the latest appearance of the target and to prevent the size of $\mathrm{LS}^{(h)}$ from growing constantly. If we use the notation $\Omega^{(h)}$ to denote the set of eliminated samples, then

$$\mathrm{LS}^{(h+1)} = \mathrm{LS}^{(h)} \cup \{\mathbf{x}_{\mathrm{TG}}^{(h)}\}/\Omega^{(h)}. \quad (13)$$

Similarly, vertices in $G^{(h)}$ with respect to the samples in $\Omega^{(h)}$ should likewise be excluded from the graph. Additionally, only $\mathbf{x}_{\mathrm{TG}}^{(h)}$ is retained while vertices related to $G_{\mathrm{US}}^{(h)}/\mathbf{x}_{\mathrm{TG}}^{(h)}$ are removed. The aforementioned graph deletion is fulfilled using the similar method in Sec. 3.3, that is, by defining Θ and then updating the neighbor information within it.

4 Analysis and Experiments

4.1 *Analysis of the LNP Tracker*

By analyzing the working mechanism of the LNP tracker and by providing corresponding examples, reasons of the improved performance achieved by the proposed tracker are explained. Compared with existing methods, we hold that two points are critical:

1. A one-class classification-based tracker is more accurate than that based on binary classification, especially when the background contains different image structures from the target.
2. Exploiting the local information is better than simply using the pairwise relationship of samples.

The advantage of a one-class classification listed is demonstrated in Fig. 4. In Fig. 4(a), the unlabeled sample is difficult to classify because no record in the positive or negative set is similar to it, even though it should be classified as a background. An unlabeled sample such as this will probably decrease the accuracy of the tracker and limit the tracking performance. Nevertheless, if a one-class classification is utilized, as shown in Fig. 4(b), this problem can easily be resolved. The reason is that as long as the new sample is unlike the object, the sample will not be chosen as the final target.

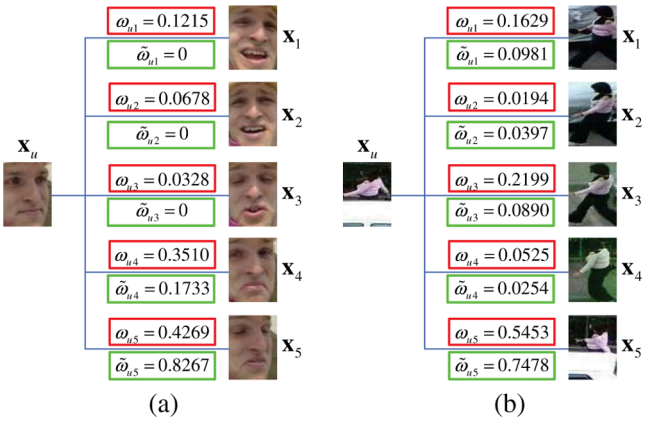


Fig. 5 Comparison of methods for graph construction: (a) indicates the view change and (b) shows the effect of occlusion on edge weights. $\mathbf{x}_u$ is the unlabeled sample and is linked to the labeled samples $\mathbf{x}_1 \sim \mathbf{x}_5$ by two graph construction methods; i.e., pairwise and local methods, respectively. $\omega_{u1} \sim \omega_{u5}$ in red boxes are edge weights generated using the local method, while $\tilde{\omega}_{u1} \sim \tilde{\omega}_{u5}$ in green boxes are those generated using the pairwise method.

The LNP tracker is not easily confused by adopting the local information mentioned in number 2 even when the appearance of the target changes significantly. In Fig. 5, $\mathbf{x}_1 \sim \mathbf{x}_5$ are positive samples, and $\mathbf{x}_u$ is the unlabeled sample. The target of graph construction is to connect the unlabeled samples to the similar labeled ones with edges. Figure 5(a) illustrates that although the observation angle of $\mathbf{x}_u$ looks very different from $\mathbf{x}_1 \sim \mathbf{x}_5$, using Eq. (4) to establish a graph still produces strong ties between them (see edge weights in red boxes). Comparatively, if Eq. (1) is adopted to present the pairwise relationship between $\mathbf{x}_u$ and the five labeled samples, the connections between $\mathbf{x}_u$ and $\mathbf{x}_1 \sim \mathbf{x}_3$ are relatively very weak (edge weights in green boxes approximately equal to 0). Similarly, Fig. 5(b) indicates that when partial occlusion occurs, bigger edge weights can be obtained using the local method (red boxes) compared with those

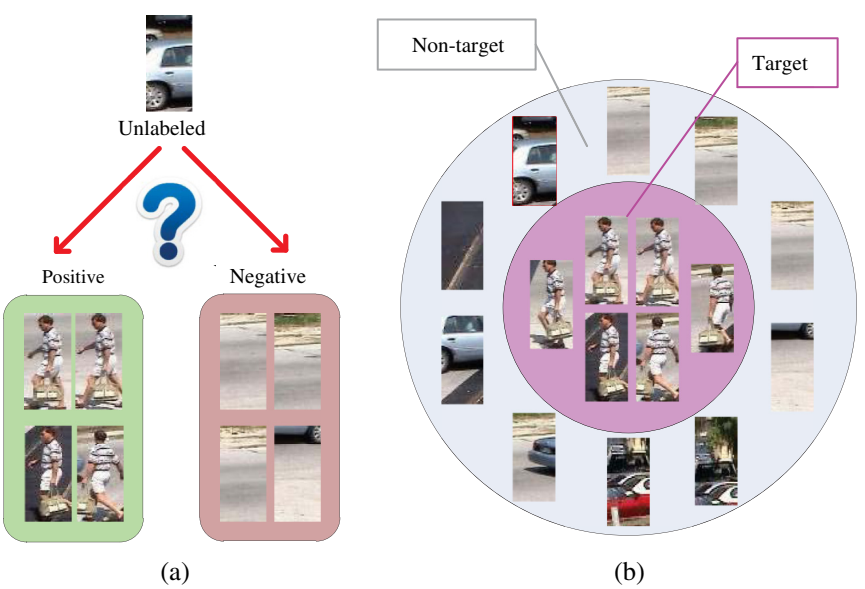


Fig. 4 One-class classification versus binary classification: (a) shows the process of binary classification; (b) indicates the mechanism of the one-class classification. The unlabeled sample that is difficult to be classified in (a) will easily be regarded as the nontarget in (b) (red box).

obtained based on the pairwise idea (green boxes). In fact, a good strategy for graph construction is to assign large weights between $\mathbf{x}_u$ and $\mathbf{x}_1 \sim \mathbf{x}_5$ to let the tracker know that the two are the same. Experimental results suggest that by using our graph construction strategy, the LNP tracker will regard $\mathbf{x}_u$ and $\mathbf{x}_1 \sim \mathbf{x}_5$ as the same thing. Therefore local information around samples can offer more clues for the LNP tracker to recognize the object. Note that in Fig. 5(a) and 5(b), $\mathbf{x}_5$ is actually more similar to $\mathbf{x}_u$ than any other labeled samples. Thus Eqs. (1) and (4) assign the biggest weight (ω_{u5} and $\tilde{\omega}_{u5}$) to it.

4.2 *Comparison with Other Trackers*

The LNP tracker is tested on several challenging public sequences and is compared with other popular trackers, such as mean-shift (MS),[5] semiboosting (SB),[11] FragTracker (FT),[22] and harmonic function tracker (HF).[14] Note that the HF is a binary classification-based tracker wherein the graph is constructed using pairwise information. In all the experiments below, the parameters of the LNP tracker are $K = 5$, $\alpha = 0.99$, and $T = 150$. The Gabor feature[23] is adopted because it discovers the local feature of the target perfectly from different directions and scales.

4.2.1 *Handling appearance change*

The appearance change of a target is a noteworthy obstacle for reaching a robust tracking because such change will most likely make the tracker unable to identify the previous target. In a *Hockey* (available at http://www.vision.ee.ethz.ch/~hegrabne/) sequence, a player skates quickly to defend the attacker, and his body posture would vary dramatically all the time. The performance of five trackers is compared in Fig. 6(a). The HF (green box) fails to track the player (the target) when he waves his arms and makes a turn to catch the attacker (frame 75). The SB (yellow box) cannot find the target in frames 39 and 52 because the defender bends down to accelerate during this period. Nevertheless the SB can redetect and continue to track the target successfully afterward. Generally, the MS (blue box), FT (magenta box), and the LNP tracker (red box) perform better in this sequence. Comparatively, slight drifting appears in frames 25 and 39 in the blue box, while the LNP tracker is able to track the target accurately from beginning to end.

4.2.2 *Face tracking*

Face tracking is also considered as a difficult task because the trackers are easily confused in the presence of facial views or expression variations. The *Foreman* sequence (available at http://media.xiph.org/video/derf/) contains the situations that often occur during the face tracking process, such as significant expression changes, intense head movements, hand occlusions, and view conversions. The frames presented in Fig. 6(b) indicate that the LNP tracker can resist the aforementioned disturbances and locate the face precisely throughout the whole sequence. The HF loses the target in frames 27, 86, and 149 to 174. The MS works well in the majority of the frames except in 159 and 174. In frame 159, the face rotates to a profile view, and when it recovers, the tracker cannot locate the frontal face as it did before. Moreover the SB fails to find the target after an occlusion by hand occurs. Thus no yellow boxes appear in frames 155 to 174.

Moreover, $\mathrm{Score}(\mathrm{TG}^{(h)})$, which is interpreted as tracking confidence level in Sec. 3.4, is also plotted in Fig. 7(a). This figure reveals that in frames 75 to 100 and 150 to 170, the confidence curve drops to a low level. Frames 75 to 100 correspond to the intense head movements, which include the shaking of the head and the action of looking up. Thus in such cases the tracker's confidence level is lower. Frames 150 to 170 correspond to the occlusion and observation view changes. These changes decrease the confidence level of the tracker. Therefore, the confidence curve adopted in our algorithm can honestly reflect the whole tracking process, and it helps exclude many disturbances that may lead to drifting.

4.2.3 *Dealing with occlusion*

Partial or complete occlusion is another critical issue for robust tracking. The *Car* (available at http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html) and *Woman* (available at http://ice.dlut.edu.cn/lu/Project/cvpr12_jia_project/cvpr12_jia_project.htm) sequences are used to test the occlusion handling ability of the LNP tracker.

The target is totally occluded by a tall tree in the *Car* sequence [Fig. 6(c)]. Furthermore, the car being tracked is similar to another one in the scenario, thereby requiring the tracker to make a clear distinction between the two. The second car is chosen as the target, and all trackers successfully locate the target in the initial phase (frames 22 and 25). However, when the car is totally occluded by a tall tree (frame 78) and appears again (frame 87), the MS, HF, and FT cannot locate the target accurately. The SB mistakenly tracks the first car until the end. Only the LNP tracker redetects and tracks the preset target (the second car) successfully after the occlusion. Note that in frames 78 and 257, the LNP tracker judges the target to be out of the scenario (no red box is available) because the target is occluded in frame 78 and part of the object is clipped by the edge of image in frame 257. These two periods are also illustrated in the confidence curve [Fig. 7(b)]. This curve indicates that during frames 60 to 80 and 250 to 260, the confidence level drops significantly.

In the *Woman* sequence [Fig. 6(d)], the upper body of the target is occluded by three cars successively. In frame 94, 134, 216, and 254, the SB, MS, FT, and HF all begin to locate the object inaccurately. Only the LNP tracker is not influenced by the occlusions.

Moreover, if the position error in the i'th frame is defined as

$$\mathrm{Error}_i = \|\mathrm{Target\ Center}_i(x, y) - \mathrm{Ground\ Truth}_i(x, y)\|_2,$$

then the position errors over all frames of the *Car*, *Woman*, and more sequences (*CarInDark* and *Girl*, available at http://ice.dlut.edu.cn/lu/Project/cvpr12_jia_project/cvpr12_jia_project.htm and http://www.dabi.temple.edu/~hbling/code_data.htm, respectively) are plotted in Fig. 8. The figure illustrates that the LNP tracker is closer to the ground truth in general. The tracking results of the LNP tracker for the *CarInDark* and *Girl* sequences are also illustrated in Fig. 9.

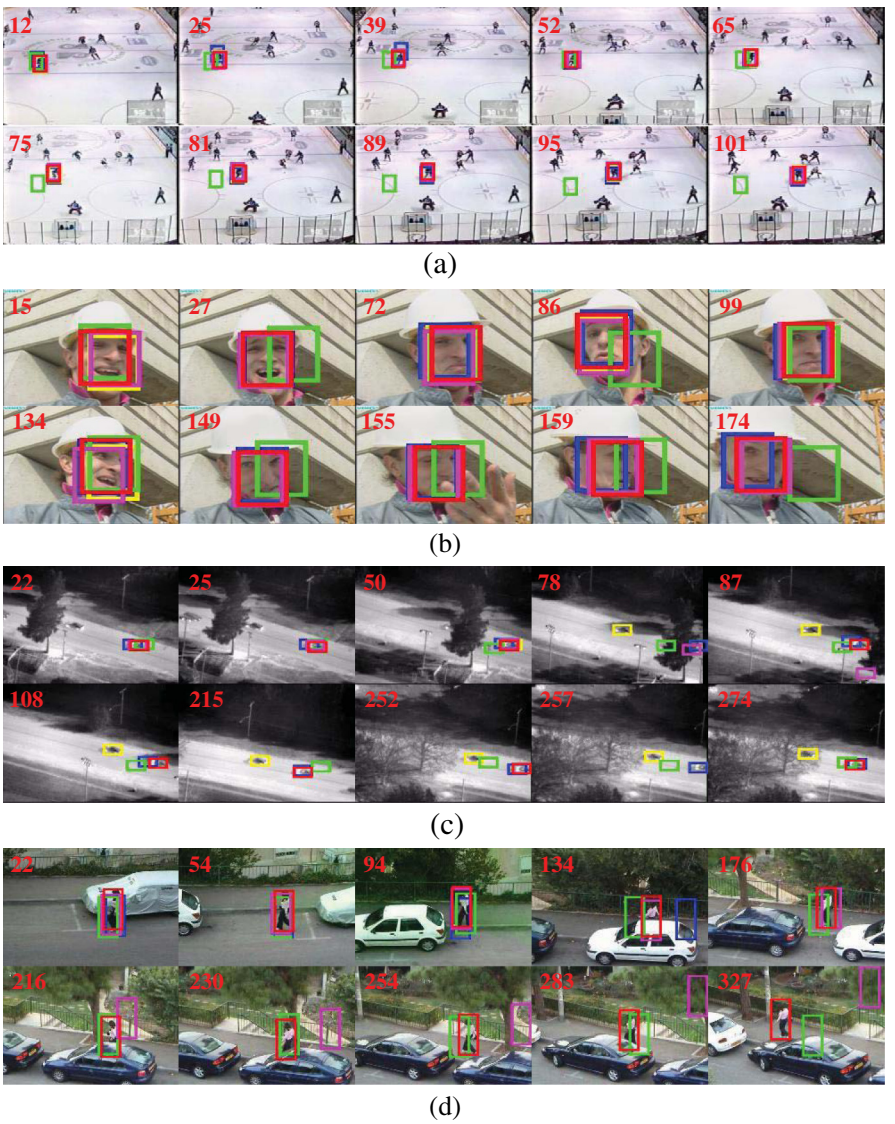


Fig. 6 Comparison of the MS, SB, HF, FT, and the LNP in the *Hockey*, *Foreman*, *Car*, and *Woman* sequences. Red box: LNP, blue box: MS, yellow box: SB, green box: HF, magenta box: FT.

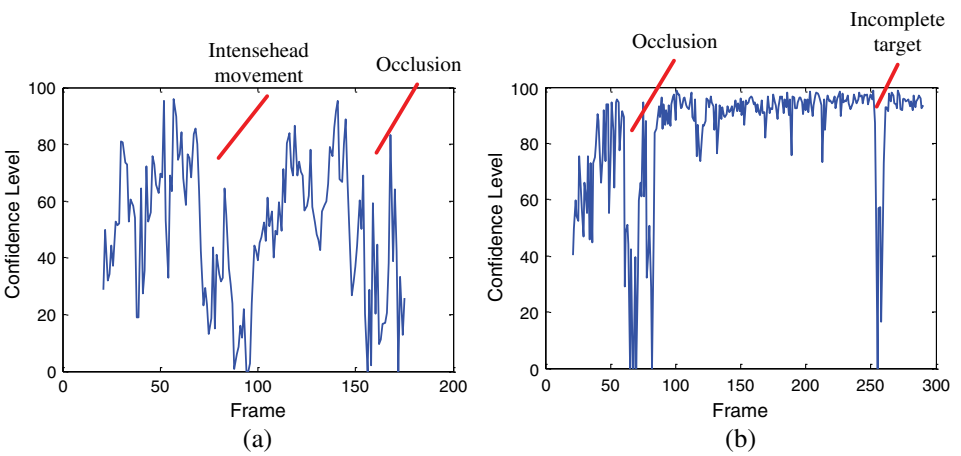


Fig. 7 Confidence curve for the LNP tracker: (a) is the *Foreman* sequence, and (b) denotes the *Car* sequence.

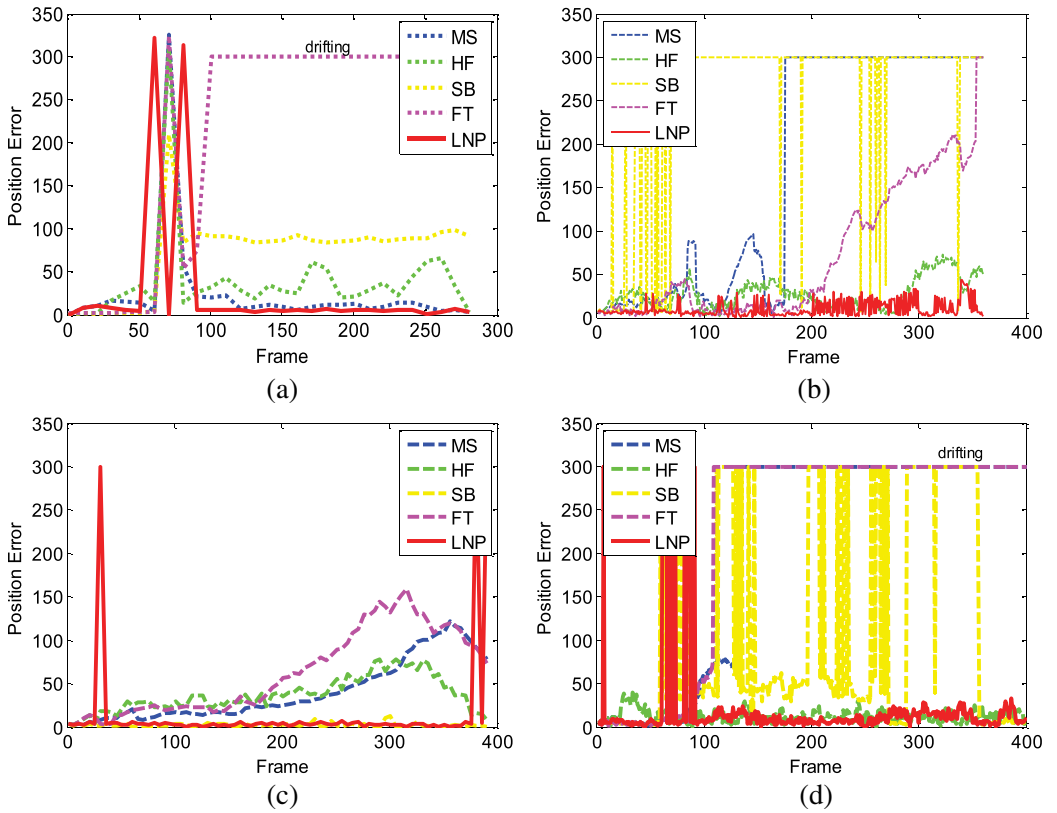


Fig. 8 Position error of the five trackers: (a) *Car* sequence; (b) *Woman* sequence; (c) *CarInDark* sequence; (d) *Girl* sequence.

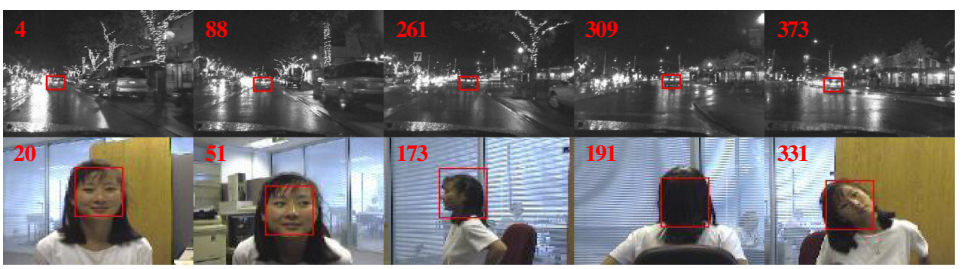


Fig. 9 Tracking results of the LNP tracker: (a) is the *CarInDark* sequence, and (b) is the *Girl* sequence.

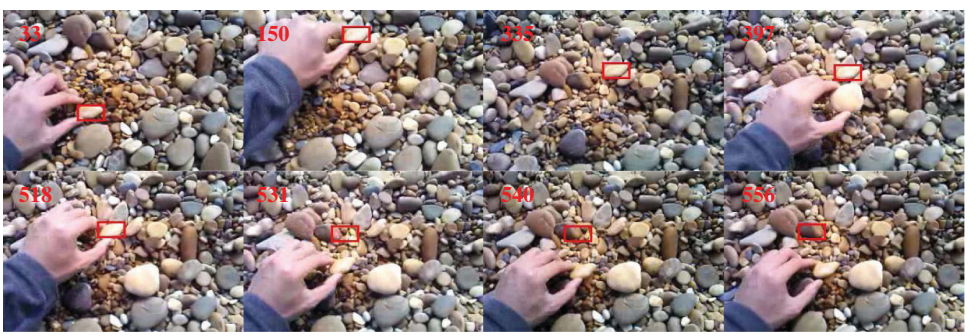


Fig. 10 Illustration of the failure case.

4.3 *Failure Case*

Although the proposed tracker achieves satisfying results in most instances, it is expected to fail in some extreme situations. For example, in Fig. 10, a stone is tracked accurately from the beginning up to frame 397. When the stone is moved in frames 518 to 556, the tracker gets confused and drifts to track the wrong object. This failure may be attributed to two reasons. First, the background contains many objects whose structures are very similar to that of the target. This scenario causes great difficulty for the one-class tracker to make a distinction between the target and the background. Hence the tracker is likely to mistakenly treat other stones as the target. Second, the target in this case is too small and has no notable texture feature. Therefore the Gabor feature adopted is no longer discriminative. In such circumstances, the traditional trackers based on binary classification will probably perform better than the LNP tracker when features that are more proper are fused into the tracking framework.

5 Conclusion and Future Work

This paper regards tracking as a one-class classification task and proposes a graph-based semisupervised tracker called the LNP tracker. The graphical model is constructed using the neighborhood information of the samples, and the label is transmitted through the edges. The proposed LNP tracker discovers the manifold structure in the dataset, thus achieving more robust and adaptive performance. Experiments demonstrate the rationality of our method and suggest that the LNP tracker can effectively deal with some internal or external disturbances during the tracking process, i.e., appearance changes, observation view changes, occlusions, etc.

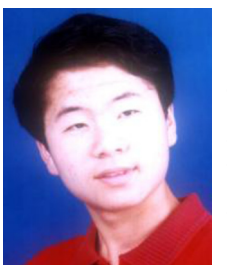
However, the computational complexity is still very high (5.14 fps for a 20×32 target) despite the incremental graph construction is adopted. Future work should therefore focus more on this point. Building a suitable graph for object tracking is also an interesting trend to explore.

Acknowledgments

This research is partly supported by National Science Foundation, China (No: 61273258).

References

1. D. Serby, K. Meier, and L. V. Gool, "Probabilistic object tracking using multiple features," in *IEEE International Conference of Pattern Recognition*, pp. 184–187, IEEE, Cambridge, England (2004).
2. A. Yilmaz, X. Li, and M. Shah, "Contour based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(11), 1531–1536 (2004).
3. A. Ali and J. Aggarwal, "Segmentation and recognition of continuous human activity," in *IEEE Workshop on Detection and Recognition of Events in Video*, pp. 28–35, IEEE, Vancouver, Canada (2001).
4. H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Vol. **1**, pp. 260–267, IEEE, New York (2006).
5. D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(5), 564–575 (2003).
6. C. Tomasi and T. Kanade, "Detection and tracking of point features," Technical Report CMU-CS-91-132, Carnegie Mellon University (April 1991).
7. T. Broida and R. Chellappa, "Estimation of object motion parameters from noisy images," *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8** (1), 90–99 (1986).
8. M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," *Int. J. Comput. Vis.* **29**(1), 5–28 (1998).
9. A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Comput. Surv.* **38**(4), 13–58 (2006).
10. S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(8), 1064–1072 (2004).
11. H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. European Conference on Computer Vision*, Vol. **5302**, pp. 234–247, Springer, Marseille, France (2008).
12. B. Babenko, M. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, pp. 983–990, IEEE, Miami (2009).
13. B. Babenko, M. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(8), 1619–1632 (2011).
14. Y. Zha, Y. Yang, and D. Bi, "Graph-based transductive learning for robust visual tracking," *Pattern Recogn.* **43**(1), 187–196 (2010).
15. S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding LLE," *Science* **290**(5500), 2323–2326 (2000).
16. Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *SIAM J. Sci. Comput.* **26**(1), 313–338 (2004).
17. J. Tenenbaum, V. Silva, and J. Langford, "A Global geometric framework for nonlinear dimensionality reduction," *Science* **290**(5500), pp. 2319–2322 (2000).
18. F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *23rd International Conference on Machine Learning* Vol. 20, pp. 55–67, ACM, Pittsburgh (2006).
19. Y. Jia and C. Zhang, "Instance-level semi-supervised multiple instance learning," in *Proc. Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 640–645, ACM, Chicago (2008).
20. H. Hacid and T. Yoshida, "Incremental neighborhood graphs construction for multidimensional databases indexing," in *Proc. 20th conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pp. 405–416, Springer, Montreal, Canada (2007).
21. Z. Yin and R. Collins, "Object tracking and detection after occlusion via numerical hybrid local and global mode-seeking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, Anchorage (2008).
22. A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 798–805, IEEE, New York (2006).
23. T. Lee, "Image representing using 2D Gabor wavelets," *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(6), 959–971 (1996).



Chen Gong received a BSc from East China University of Science and Technology (ECUST) in 2010. Currently he is a PhD candidate at Shanghai Jiao Tong University (SJTU) at the Institute of Image Processing and Pattern Recognition under the supervision of professor Jie Yang. His research interests mainly include machine learning and object detection and tracking.

Keren Fu received a BSc in the major of Automation from Huazhong University of Science and Technology (HUST) in 2011. Currently, he is a PhD candidate at Shanghai Jiao Tong University (SJTU) at the Institute of Image Processing and Pattern Recognition under the supervision of professor Jie Yang. His research interests include object detection, saliency detection, visual tracking and machine learning.

Enmei Tu received BSc and MSc degrees from University of Electronic Science and Technology of China (UESTC) in 2007 and 2010, respectively. Currently, he is a PhD candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China under the supervision of professor Jie Yang. His research interests are machine learning, computer vision and remote sensing data processing.

Jie Yang received his PhD from the Department of Computer Science, Hamburg University, Germany, in 1994. Currently, he is a professor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China. He has led many research projects (e.g., National Science Foundation, 863 National High Tech. Plan), had one book published in Germany, and authored more than 200 journal papers. His major research interests are object detection and recognition, data fusion and data mining, and medical image processing.

Xiangjian He received a BS in mathematics from Xiamen University in 1982, an MS in applied mathematics from Fuzhou University in 1986, and a PhD in computing sciences from the University of Technology, Sydney, Australia, in 1999. From 1982 to 1985, he was with Fuzhou University. From 1991 to 1996, he was with the University of New England. Since 1999, he has been with the University of Technology, Sydney. He is the director of Computer Vision and Recognition Laboratory and deputy director of the Research Centre for Innovation in IT Services and Applications at the University of Technology, Sydney.