Discriminative Object Tracking via Sparse Representation and Online Dictionary Learning

Yuan Xie, Wensheng Zhang, Cuihua Li, Shuyang Lin, Yanyun Qu, *Member IEEE*, and Yinghua Zhang

***Abstract*—We propose a robust tracking algorithm based on local sparse coding with discriminative dictionary learning and new keypoint matching schema. This algorithm consists of two parts: the local sparse coding with online updated discriminative dictionary for tracking (SOD part), and the keypoint matching refinement for enhancing the tracking performance (KP part). In the SOD part, the local image patches of the target object and background are represented by their sparse codes using an over-complete discriminative dictionary. Such discriminative dictionary, which encodes the information of both the foreground and the background, may provide more discriminative power. Furthermore, in order to adapt the dictionary to the variation of the foreground and background during the tracking, an online learning method is employed to update the dictionary. The KP part utilizes refined keypoint matching schema to improve the performance of the SOD. With the help of sparse representation and online updated discriminative dictionary, the KP part are more robust than the traditional method to reject the incorrect matches and eliminate the outliers. The proposed method is embedded into a Bayesian inference framework for visual tracking. Experimental results on several challenging video sequences demonstrate the effectiveness and robustness of our approach.**

***Index Terms*—Dictionary learning, object tracking, robust keypoints matching, sparse representation.**

I. Introduction

VISUAL TRACKING is a challenging problem in computer vision, due to the appearance changes of the target caused by noise, occlusion, background clutter, different viewpoint, and illumination conditions. Although many tracking methods employ static appearance model, such as [19], [20], [21], [22], these methods can only handle the modest changes and tend to fail when the appearance of the object changes significantly. As a result, there is a need for learning appearance model on-the-fly.

A variety of online tracking algorithms have been proposed to overcome these difficulties during tracking. These methods can be formulated in two different ways: generative model and discriminative model. To deal with the challenges mentioned above, the state-of-the-art algorithms focus on robust object representation schemes with generative appearance models and sophisticated classifiers.

Generative methods represent objects with models that have minimum reconstruction errors, and then the tracking is expressed as finding the most similar candidate to the target. Therefore, generative trackers only aim at encoding the target appearance. The works [1], [2], and [18] belong to the generative model. The recent development of sparse representation has attracted considerable interest in object tracking [5], [7]–[11] due to its robustness to occlusion and image noise. However, the traditional generative tracking methods mentioned above are only trained based on object appearance without utilizing the information from the background. Instead of only focusing on appearance model for the target itself, discriminative trackers aim to find a decision boundary that can best separate the target from the background, such as [23]–[27]. Such a way refers to treat object tracking as a binary classification problem to distinguish between the positive sample (target) and negative samples (background). For using the background information, these methods demonstrate strong robustness to avoid distracters in the background.

In this paper, we propose an online tracking algorithm based on the sparse coding and the discriminative dictionary learning. The tracker combines the discriminative model and the generative model. The discriminative part applies the local sparse representation with the online discriminative dictionary to encode the appearance information of both the object target and the background. Then using the sparse codes of the samples to train a linear discriminative appearance model to best separate the target from the background. Different from recent sparse representation based trackers that use the dictionary sampled from the target object, the proposed tracker constructs a discriminative dictionary which contains both the foreground dictionary and the background dictionary. The generative part employs a SIFT feature point matching schema to model the target appearance in order to enhance the performance of the tracker. It utilizes sparse representation and discriminative dictionary to reject the incorrect matches and eliminate the

Manuscript received August 19, 2012; revised March 22, 2013; accepted April 8, 2013. Date of publication May 31, 2013; date of current version March 13, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant U1135005 and Grant 90924026, the Project for the National Basic Research 12th Five Program under Grant 0101050302, National Defense Basic Scientific Research program of China under Grant B1420110155, the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20110121110020, and the Fundamental Research Funds for the Central Universities under Grant 2010121067. This paper was recommended by Associate Editor M. Shin

Y. Xie, W. Zhang, and Y. Zhang are with the State Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: yuan.xie@ia.ac.cn; wensheng.zhang@ia.ac.cn; yinghua.zhang@ia.ac.cn).

C. Li, S. Lin, and Y. Qu are with the Video and Image Laboratory, Department of Computer Science, Xiamen University, Xiamen 361005, China (e-mail: chli@xmu.edu.cn; yyqu@xmu.edu.cn; sylin@xmu.edu.cn).

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

Digital Object Identifier 10.1109/TCYB.2013.2259230

background keypoints. Both the generative part and the discriminative part will collaborate under the Bayesian inference framework. In other words, the information from different parts would be integrated into the observation model and the motion model. Experimental results on several challenging video sequences demonstrate the effectiveness and robustness of our approach.

The contributions of this paper are as follows.

1) A sparsely represented discriminative model based on the discriminative dictionary, which includes both the foreground dictionary and the background dictionary, is proposed to separate the object target from its surrounding.
2) An online learning method is employed to update the discriminative dictionary, which could adapt to the variation of the foreground and the background during the tracking.
3) A keypoint matching schema is used to enhance the performance of the tracker under the Bayesian inference framework.
4) The keypoint matching can be refined by the sparse representation with the online discriminative dictionary.

This paper is structured as follows. We begin by reviewing the related work in the next section. Section III firstly shows how to learn object appearance model using local sparse coding and the discriminative dictionary, then presents the discriminative dictionary that can be updated in the online manner. The robust keypoint matching refinement and background keypoint elimination are described in the Section V. Then, the detail of the proposed tracking algorithm is shown in Section VI. Experimental results and some discussion are shown in Section VII. Finally, Section VIII is devoted to conclusions and future work.

II. Related Work

There is a rich literature in appearance modeling and representation that aim at tackling nonstationary appearance tracking problems. In this section, we review some most relevant topics that motivate this paper: sparse representation, dictionary learning, and the keypoints matching.

Recent advancements in sparse representation indicate another path for us to model the appearance of an object by means of sparsity. Unlike to major sparse representation based trackers [5]–[7], [13] that are posed within the generative framework and use reconstruction errors to determine the location of the target object, [12] describes the object by local sparse coding using the dictionary sampled from the object and trivial bases [4], [5]. Inspired by the work [12], we propose a discriminative appearance model that is based on the local sparse representation with the online discriminative dictionary to encode the appearance information of both the object target and the background. Then using the sparse codes of the samples to train a linear discriminative appearance model to best separate the target from the background. The proposed method overcomes the drawbacks of [12] in the two aspects. Firstly, instead of using the dictionary sampled from the target object, we construct a discriminative dictionary that contains both the foreground dictionary and the background dictionary. Such discriminative dictionary, which encodes the information of both the foreground and the background, may provide more discriminative power than the dictionary used in [12]. Secondly, thanks to the online dictionary learning proposed by J. Mairal *et al.* [16], the discriminative dictionary can be updated in the online manner, while [12] takes a pseudo-online schema actually. The online learned dictionary will adapt to the variation of the foreground and background during the tracking. Moreover, the learned dictionary can memorize the past (the last few frames) and current information of the object target and background, leading to a robust tracker that can handle the drastic appearance change.

When referring to the keypoint based tracking, we will emphasize the keypoint matching. There are incorrect matches due to the ambiguous features or confusing background information as object features. Thus, we need a method to refine the matching. Given the matched pairs, a straightforward way is to use the RANSAC algorithm [28] to fit a fundamental matrix model and reject incorrect matches. Ref. [29] eliminates the mismatches between binary features by computing a homograph using RANSAC, [30] uses RANSAC to filter out the outliers from the matched SURF feature [31] pairs to improve the motion estimation. However, RANSAC will perform poorly when the percent of inliers falls much below 50% [32]. This case usually happens in object tracking, where the target is in the clutter background, thereby the number of the background keypoints is more than that of the foreground keypoints. To address this issue, we propose a novel keypoints pair refinement method based on the local sparse coding. It utilizes the context information of the candidate keypoint pair to improve the result of the keypoint matching.

Another important issue of keypoint-based tracking is to eliminate the background keypoints inside the object target bounding box. Here, we refer to the keypoints inside the target contour as the foreground keypoints, the keypoints outside the contour are called the background keypoints accordingly. Commonly, the keypoint-based trackers [29], [33], [34] will build the appearance model of the target by the foreground keypoints. However, a fraction of the background keypoints, which appear inside the target bounding box, will introduce noise into appearance model and degrade the performance of the tracker. In order to overcome this problem, [33] proposes to use nearest neighbor classifier to determine a keypoint belonging to the foreground set or the background set. It employs the KD-trees to efficiently perform the nearest neighbor searching based on the Euclidean distance between descriptor vectors of the SIFT keypoints. Similar to [33], we also use the nearest neighbor classifier but with sparse residual to measure how a keypoint close to the foreground/background keypoints sets. The performance comparison between [33] and the proposed method is presented in the experimental section, which shows the advantage of our approach.

III. Learning Discriminative Object Model with Sparse Representation

Sparse representation has been widely used in visual tracking such as [5], [12], and [13]. In [12], the author employs an over-complete dictionary to encode the local patches inside the object, then generates its corresponding sparse code. Inspired by [12], but unlike to the dictionary only encoding the information of the target, we construct a dictionary $\mathbf{D} = [\mathbf{D}_p, \mathbf{D}_n]$ that contains both the foreground dictionary and the background dictionary that help to distinguish the object target from the background. After calculating the sparse codes of the object and background patches, a linear SVM is used to train a discriminative model to separate the target object from the background. We will give the details of this method in this section.

A. Local Sparse Coding

To construct the foreground dictionary $\mathbf{D}_p$, we extracts the overlapped image patches using the sliding window method from the target object region in the first frame, then aggregates the vectorized patches to form the target basis set $T_f = [\mathbf{t}_1, \ldots, \mathbf{t}_n] \in \mathbb{R}^{d\times n}$, where $\mathbf{t}_i$ is the ith vectorized patch and d denotes its dimensionality, n is the number patches extracted from object region. The over-complete foreground dictionary is defined as $\mathbf{D}_p = T_f \in \mathbb{R}^{d\times n}$. Similarly, we use the sliding window to sample the patches around the target and vectorize them to form the background basis set $T_b = [\mathbf{t}_1, \ldots, \mathbf{t}_m] \in \mathbb{R}^{d\times m}$. Then, the over-complete background dictionary is represented by $\mathbf{D}_n = T_b \in \mathbb{R}^{d\times m}$.

With the overall dictionary $\mathbf{D} = [\mathbf{D}_p, \mathbf{D}_n] \in \mathbb{R}^{d\times(n+m)}$, the sparse codes of any image region can be computed by the following steps. Firstly, extract N image patches from image region with each patch vectorized to $\mathbf{x} \in \mathbb{R}^{d\times 1}$, then the image region can be denoted by $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d\times N}$. Secondly, the sparse code $\alpha_i \in \mathbb{R}^{n+m}$ corresponding to $\mathbf{x}_i$ is calculated by

$$\min_{\alpha_i} \frac{1}{2}\|\mathbf{x}_i - D\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1 \tag{1}$$

where the l_1-norm leads to sparsity in coefficient α_i. Equation (1) is actually the Lasso [14] regression that can be solved efficiently by LARS [15]. Finally, when $\{\alpha_i\}_{i=1}^N$ are computed for all patches $\{\mathbf{x}_i\}_{i=1}^N$, the sparse code $\mathbf{z}$ of the image region will be achieved by directly concatenating all the $\{\alpha_i\}_{i=1}^N$: $\mathbf{z} = [\alpha_1^T, \ldots, \alpha_N^T]^T$.

B. Learning Discriminative Appearance Model with Sparse Representation

Suppose we are given the location of the target $\mathbf{L}_t = (x_t, y_t)$ in frame t, the positive (foreground) samples L_{pos} are drawn according to the Gaussian perturbation that satisfies $\|L_{pos} - L_t\| < \gamma$, the negative (background) samples L_{neg} are drawn from the annular region specified by $\gamma < \|L_{neg} - L_t\| < \eta$, where γ and η are the sampling radius. Using the proposed discriminative dictionary $\mathbf{D}$, the distribution of the sparse codes will be totally different between the positive samples and the negative samples. Therefore, it is easy to train a classifier on sparse codes to separate the foreground samples from the background samples.

Algorithm 1: Online Dictionary Learning [16]

```
  Input: $x \in \mathbb{R}^m \sim p(x)$, $\lambda \in \mathbb{R}$, $T$
  Output: learned dictionary $\mathbf{D}_T$ and $\mathbf{A}_T, \mathbf{B}_T$
1 if the first frame then
2     $\mathbf{A}_0 \leftarrow 0$, $\mathbf{B}_0 \leftarrow 0$, random initial dictionary $\mathbf{D}_0$;
3 else
4     $\mathbf{A}_0 \leftarrow \mathbf{A}'_T$, $\mathbf{B}_0 \leftarrow \mathbf{B}'_T$, $\mathbf{D}_0 \leftarrow \mathbf{D}'_T$;
5 end
6 for $t = 1$ to $T$ do
7     D
8 end
9 raw $x_t$ from $p(x)$;
10 Sparse coding: $\alpha_t = \mathrm{argmin}_{\alpha\in\mathbb{R}^k} \frac{1}{2}\|x_t - \mathbf{D}_{t-1}\alpha\|_2^2 + \lambda\|\alpha\|_1$,
11 $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \alpha_t\alpha_t^T$,
12 $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + x_t\alpha_t^T$,
13 Compute $\mathbf{D}_t$ using Algorithm 2, with $\mathbf{D}_{t-1}$ so that:
14 $\mathbf{D}_t = \mathrm{argmin}_{D\in C} \frac{1}{t}\sum_{i=1}^t \frac{1}{2}\|x_i - \mathbf{D}\alpha_i\|_2^2 + \lambda\|\alpha\|_1$
15 $= \mathrm{argmin}_{D\in C} \frac{1}{t}\left(\frac{1}{2}Tr(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) + Tr(\mathbf{D}^T\mathbf{B}_t)\right)$ // op1
16 Return $\mathbf{D}_T, \mathbf{A}_T, \mathbf{B}_T$;
```

If we set all training patches denoted by L_{pos} and L_{neg} to the certain scale, which is the same to target object, the sparse codes $\{\mathbf{z}_i\}_{i=1}^M$ of all the training patches will be computed to construct the training data $\{\mathbf{z}_i, y_i\}_{i=1}^M$, where the vector $\mathbf{z} \in \mathbb{R}^{N(n+m)}$, $y_i \in \{+1, -1\}$, and M is the number of training samples.

The optimization of the linear classifier with training data $\{\mathbf{z}_i, y_i\}_{i=1}^M$ can be defined as follows:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda}{M}\sum_{i=1}^M L(y_i, \mathbf{w}, \mathbf{z}_i) \tag{2}$$

where $\mathbf{w}$ is the classifier parameter, $\lambda > 0$ is a constant that controls the tradeoff between training error minimization and margin maximization. The $L(\cdot)$ is a loss function which is defined by

$$L(y, \mathbf{w}, \mathbf{z}) = \log\left(1 + e^{-y\mathbf{w}^T\mathbf{z}}\right). \tag{3}$$

In the testing phase, the classification score of any candidate $\mathbf{z}$ can be computed by

$$f(\mathbf{z}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{z}}}. \tag{4}$$

Once the classifier is initialized, the classification score can be used as the similarity measure for tracking. A sample with large score indicates that it is more likely to belong to the foreground class. The sample with highest classification score is considered as the tracking result for the current frame.

IV. Robust Tracking with Online Dictionary Learning

The dictionary $\mathbf{D}$ mentioned above is the predefined dictionary that only encodes the information of the first few frames.

Algorithm 2: Dictionary Update [16]

```
Input: $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m\times k}$,
       $\mathbf{A} = [\alpha_1, \ldots, \alpha_k] \in \mathbb{R}^{k\times k}$,
       $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{R}^{m\times k}$
Output: $\mathbf{D}$
1 repeat  for $j = 1$ to $k$ do
2     U
3 end
4 pdate the $j$th column to optimize for op1 in Algorithm 1:
5 $\mathbf{u}_j \leftarrow \frac{1}{\mathbf{A}_{jj}}(\mathbf{b}_j - \mathbf{D}\alpha_j) + \mathbf{d}_j$
6 $\mathbf{d}_j \leftarrow \frac{1}{\max(\|\mathbf{u}_j\|_2, 1)}\mathbf{u}_j$.
7 until convergence Return D;
```

However, using such dictionary with off-the-shelf bases may not adapt to the tracking scenario due to the appearance variation of both the object target and the background. Therefore, we need a dictionary updated in the online manner.

Fortunately, J. Mairal *et al.* [16] proposed a new online optimization algorithm based on stochastic approximations for dictionary learning, which can handle large datasets with millions of training samples. Unlike dictionary learning with the predefined bases, this algorithm learned an updated dictionary to adapt to the dynamic training data changing over time, which has been shown to dramatically improve signal reconstruction in practice. The procedure of online dictionary learning is shown in Algorithm 1, and one should refer to [16] for more details.

In the first frame, we achieve the $\mathbf{D}_0$ by random initializing. The matrices $\mathbf{A}_0$ and $\mathbf{B}_0$ that help to learn the dictionary are set to $\mathbf{0}$. After T iteration, we obtain the dictionary $\mathbf{D}_T$ for the first frame with two additional matrices $\mathbf{A}_T$ and $\mathbf{B}_T$ for further usage. In the consequent frames, the online dictionary learning initializes the $\mathbf{D}_0$, $\mathbf{A}_0$, and $\mathbf{B}_0$ in the current training phase by the $\mathbf{D}_T$, $\mathbf{A}_T$, and $\mathbf{B}_T$ obtained from the last frame. In our experiments, we update the foreground dictionary and background dictionary respectively and concatenate them to construct the whole discriminative dictionary. Moreover, we should appoint the number of the bases in the dictionary. The number of bases in the dictionary $\mathbf{D}_p$ is set to 200, the number of bases in $\mathbf{D}_n$ is 500. It is noteworthy that the dictionary $\mathbf{D}$ will degenerate after a few updating iterations. In other words, each base $\mathbf{d}_j$ will tend to zero vector. In order to overcome such degeneration, we drop the past $\mathbf{D}_T$, $\mathbf{A}_T$, and $\mathbf{B}_T$ and resample the dictionary every five frames.

V. Robust Refined Keypoint Matching with Online Dictionary Learning

The SIFT descriptor [32] is of particular interest because it performs well compared with other types of image descriptor in the same class [35]. The sift points matching plays an important role in the feature-based tracking algorithms. This section will depict how to apply the sparse representation and dictionary learning to refine the keypoints matching and eliminate the background keypoints.

Algorithm 3: Matching refinement

```
Input: Candidate pairs $\{s^j_{t-1}, s^j_t\}^K_{j=1}$, frame $I_{t-1}$ and $I_t$
Output: The refined matching set
        $M = \{s^j_{t-1}, s^j_t, \{f^i_{j,t}\}^N_{i=1}\}^{\tilde{K}}_{j=1}, \tilde{K} \in \mathbb{R}$
1 for $j = 1$ to $K$ do
2     Get $\mathbf{S}_{t-1} = \{f^i_{j,t-1}\}^N_{i=1}$ and $\mathbf{S}_t = \{f^i_{j,t}\}^N_{i=1}$ using random
      affine sampling;
3     Calculate $\{\alpha_i\}^N_{i=1}$ and $\{\gamma_i\}^N_{i=1}$ with dictionary $\mathbf{S}_{t-1}$ by 5;
4     if $\min(\{\gamma_i\}^N_{i=1}) < \theta$ then
5         $\{s^j_{t-1}, s^j_t, \{f^i_{j,t}\}^N_{i=1}\}$ insert into $M$;
6     end
7 end
8 Return $M$;
```

A. Keypoint Matching Refinement

In the feature points-based tracker, the first problem to address is the keypoint matching. The common way is to use the Euclidean distance to measure the similarity between the descriptor vectors of the candidate keypoints pair. Then, a useful data-structure called KD-tree can be employed in order to fast matching between frames. However, such method may lead to incorrect matches due to ambiguous features or confused background information. Therefore, a lot of works resort to RANSAC [28] to improve the matching. It is a nondeterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. RANSAC-based matching do not consider the context information of the candidate keypoints pair, then often performs poorly in practice.

We propose a novel keypoint pair refinement method based on the local sparse coding. Suppose $\mathbf{s}^j_{t-1}$ and $\mathbf{s}^j_t$ denote the matched SIFT point pair achieved by RANSAC in the frame $t-1$ and t, respectively. Using random affine sampling around the two keypoints (all the sample patches are scaled to the same size, and then vectorized), we can get two sample sets $\mathbf{S}_{t-1} = \{f^i_{j,t-1}\}^N_{i=1}$ and $\mathbf{S}_t = \{f^i_{j,t}\}^N_{i=1}$ for the jth matched pair $(\mathbf{s}^j_{t-1}, \mathbf{s}^j_t)$, where $f^i_{j,t-1}, f^i_{j,t} \in \mathbb{R}^{d\times 1}$, d is the size of sample patch and N denotes the number of samples (below we will call these sampling patches the keypoint samples). If we consider the $\mathbf{S}_{t-1}$ as the dictionary, the sparse code α_i of the sample $f^i_{j,t}$ in $\mathbf{S}_t$ can be computed by

$$\alpha_i = \arg\min \frac{1}{2}\|f^i_{j,t} - \mathbf{S}_{t-1} * \alpha_i\|^2_2 + \lambda\|\alpha\|_1. \quad (5)$$

For all the samples in $\mathbf{S}_t$, we can achieve their sparse codes $\{\alpha_i\}^N_{i=1}$. Then, their corresponding sparse residual $\{\gamma_i\}^N_{i=1}$ can be calculated as follows:

$$\gamma_i = \|f^i_{j,t} - \mathbf{S}_{t-1} * \alpha_i\|_2. \quad (6)$$

If $\min(\{\gamma_i\}^N_{i=1}) < \theta$, it indicates that the keypoints $\mathbf{s}^j_{t-1}$ and $\mathbf{s}^j_t$ have the similar context information. Therefore, they can be seen as the correct matched points pair, otherwise they are the incorrect matching and will be discarded. The algorithm of keypoint matching refinement is shown in Algorithm 3

```
Algorithm 4: Eliminate background keypoint
  Input: $\mathbf{D}_{p,t-1}$ and $\mathbf{D}_{n,t-1}$, $M_p$, frame $I_{t-1}$ and $I_t$
  Output: The refined foreground set
          $N = \{s^j_{t-1}, s^j_t, \{f^i_{j,t}\}^N_{i=1}\}^{\tilde{K}_f}_{j=1}$, $\tilde{K}_f \in \mathbb{R}$, the $\mathbf{D}_{p,t}$ and
          $\mathbf{D}_{n,t}$
1 for $j = 1$ to $\tilde{K}_p$ do
      // $\tilde{K}_p$ is the number of keypoint in
         $M_p$
2     Calculate $\mathcal{R}_p$ and $\mathcal{R}_n$ with dictionary $\mathbf{D}_{p,t-1}$ and
      $\mathbf{D}_{n,t-1}$ using Eq.();
3     if $\min(\mathcal{R}_p) < \min(\mathcal{R}_n)$ then
4         $\{s^j_{t-1}, s^j_t, \{f^i_{j,t}\}^N_{i=1}\}$ insert into $N$;
5     end
6 end
7 Update $\mathbf{D}_{p,t}$ using $\mathbf{D}_{p,t-1}$ by Algorithm 1,
8 Update $\mathbf{D}_{n,t}$ using $\mathbf{D}_{n,t-1}$ by Algorithm 1,
9 Return $M$,$\mathbf{D}_{p,t}$ and $\mathbf{D}_{n,t}$;
```

B. Online Eliminate Background SIFT Keypoint

Intuitively, one should detect and match the keypoints only inside the object target bounding box. However, in our experiments, incorporating the background keypoints around the object target will help to get the better performance. Because utilizing the information of the keypoints around the object target could impose a constraint on the target location. Moreover, bounding box region will inevitably contain the background keypoints that are the primary cause of leading the tracker to drift. Therefore, recognizing and eliminating the background keypoints inside the bounding box will help to improve the performance.

The properties of the two kinds of keypoint are that the foreground keypoint will always inside the object region, while the background keypoint in the bounding box in a certain frame may go outside in other frames. Therefore, in order to distinct between the foreground keypoint and background keypoint, two dictionaries mentioned above, one for the object target (the foreground dictionary $\mathbf{D}_p$) and the other for the background (the background dictionary $\mathbf{D}_n$), will be useful for distinguishing the kind of a certain keypoint. Then, the keypoint samples can be sparsely represented by $\mathbf{D}_p$ and $\mathbf{D}_n$

$$\mathcal{R}(\mathbf{D}, \{f^i\}^N_{i=1}) = \left\{\|\mathbf{D} * \alpha_i - f^i\|_2\right\}^N_{i=1} = \left\{\mathcal{R}(\mathbf{D}, f^i)\right\}^N_{i=1} \quad \mathbf{D} = \mathbf{D}_p \text{or} \mathbf{D}_n \tag{7}$$

$$\alpha_i = \operatorname{argmin} \frac{1}{2}\|\mathbf{D} * \alpha - f^i\|^2_2 + \lambda\|\alpha\|_1 \tag{8}$$

where the $\mathcal{R}(\mathbf{D}, \{f^i\}^N_{i=1})$ denotes the reconstruction residual of the keypoint samples $\{f^i\}^N_{i=1}$ with dictionary $\mathbf{D}$. The samples of the keypoint contain the context information around the keypoint position. If the keypoint samples can be sparsely represented by the foreground dictionary $\mathbf{D}_p$ with lower residual than by the background dictionary $\mathbf{D}_n$, then we can conclude that this keypoint belongs to the foreground keypoint, otherwise belongs to the background keypoint and will be discarded from the foreground keypoints set. The decision function can be formally expressed by

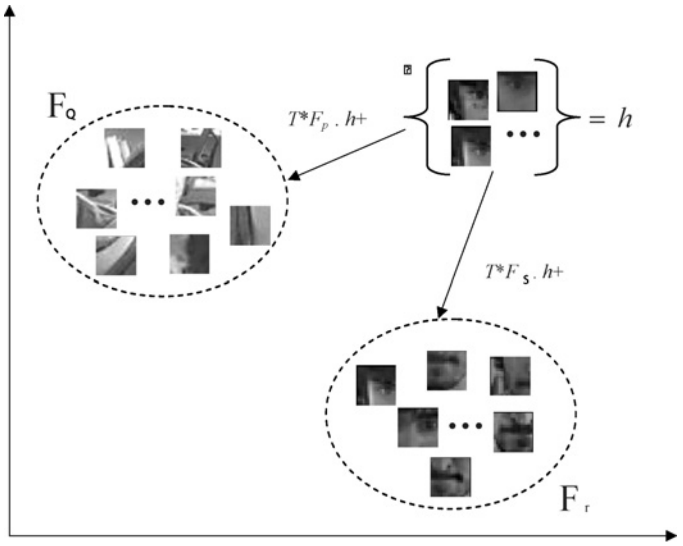

Fig. 1. Illustration of nearest neighbor classifier with sparse residual as similarity measure. f denotes the random samples around the certain keypoint, $\mathbf{D}_p$ is the foreground dictionary and $\mathbf{D}_n$ represents the background dictionary. $R(\mathbf{D}_p, f)$ and $R(\mathbf{D}_n, f)$ denote the minimum sparse residuals from the patches in f to the foreground and background dictionary, respectively.

$$\mathcal{C}(\{f^i\}^N_{i=1}) = sign\left(\min(\mathcal{R}(\mathbf{D}_p, \{f^i\}^N_{i=1})) - \min(\mathcal{R}(\mathbf{D}_n, \{f^i\}^N_{i=1}))\right) \tag{9}$$

where *sign* is the sign function.

Observing (9) carefully, it can be viewed as the decision function of a nearest neighbor classifier, if we consider the sparse residual with respect to a certain dictionary $\mathcal{R}(\mathbf{D}, \{f^i\}^N_{i=1})$ as the distance measure (Fig. 1). Such distance denotes that keypoint sample f^i in $\{f^i\}^N_{i=1}$ can be well sparsely reconstructed by a certain dictionary. At [33], S. Gu *et al.* given a theoretical analysis that shows why the neighbor works better than more sophisticated classifiers in the context of tracking. Suppose that the local information of a sift keypoint can be represented in a certain feature space, according to the ball cover theorem [33], such feature space can be well bounded by the selection criterion defined in (9), while the space in other classifiers, such as linear or nonlinear SVM, is unbounded. In experimental section, the keypoint refinement method with nearest neighbor classifier shows the prospective performance.

Until now we have described elimination of the background keypoints inside the bounding box between two consecutive frames. However, both the foreground and background will be changed during the tracking. Consequently, we should update the foreground dictionary and background dictionary on-the-fly. Fortunately, the online updated discriminative dictionary $\mathbf{D} = [\mathbf{D}_p, \mathbf{D}_n]$ mentioned in Section IV can be straightly used. The pesudo-code of the online eliminating background keypoint is depicted in Algorithm 4.

VI. Proposed Tracking Algorithm

A. Two Complementary Part

The proposed tracker contains two complementary parts. The one is the SOD part mentioned in Section III; the other is the KP part described in Section V. The combination of the two parts can be viewed as integrating the discriminative model and the generative model into an unified framework as mentioned in Section I. Moreover, this combination can also be considered as unifying the global method with local method. On the one hand, the SOD is a kind of model-based object tracking approach. It employs the sparse coding and linear SVM to capture the global feature of the target. The local sparse representation in SOD is only to encode the information of the local patch into its corresponding sparse code. However, all these sparse codes are concatenated together to represent the object target, therefore it models the overall information of the target patch actually. On the other hand, our KP part introduces the local information around the keypoint position into tracker system, which shares the same idea as the part-based tracking method that can handle the partial occlusion. Below, we will discuss how to combine them under the Bayesian framework.

B. Object Tracking by Bayesian Inference

We embed the two complementary parts into the Bayesian inference framework to construct a robust tracking algorithm. Given the observations of the target $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \dots, \mathbf{z}_t\}$ up to time t, the target state $\mathbf{x}_t$ can be computed by the maximum *a posteriori* estimation

$$\widehat{\mathbf{x}}_t = \operatorname*{argmax}_{x_t} p(\mathbf{x}_t|\mathbf{z}_{1:t}). \tag{10}$$

The posterior probability $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be inferred by the Bayesian theorem recursively

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \tag{11}$$

where $p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}$. Within the above formulation, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the motion model and $p(\mathbf{z}_t|\mathbf{x}_t)$ is the observation model. In our algorithm, we model the motion of the target between two consecutive frames by affine transformation. Similar to [17], the state vector $\mathbf{x}_t = (x_t, y_t, \eta_t, s_t, \beta_t, \phi_t)$ at time t contains six parameters of affine transformation, where x_t, y_t denote the 2-D position and $\eta_t, s_t, \beta_t, \phi_t$ represent the rotation angle, scale, aspect ratio, and skew direction, respectively. In the following, we will use the $\mathbf{x}_t^r, r = [x_t, y_t, \eta_t, s_t, \beta_t, \phi_t]$ to denote the part of the state vector at the time t.

In our tracker, we apply the voting result of the keypoint to improve the estimation of target states in consecutive frames. Suppose that at the time $t-1$ the tracker memorizes the relative position of each sift keypoint and object center. At the time t, after refining the matching and eliminating the background keypoint, we can achieve the rough position of the target by the voted center of the foreground keypoints whose relative positions are memorized at frame $t-1$. The voting details are presented in Algorithm 5. Note that in algorithm 5, if the voting results are not centralized, it only returns the state vector of the last frame. Then the transition model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ can be defined in (12),

$$p(\mathbf{x}_t^r|\mathbf{x}_{t-1}^r) = \begin{cases} \mathcal{N}(\mathbf{x}_t^r; w_1 f_{vote}(\{S\}_{i=1}^N, \mathbf{x}_{t-1}^r) + w_2\mathbf{x}_{i-1}^r, \Sigma^r) & r \in [x_t, y_t], \\ \mathcal{N}(\mathbf{x}_t^r; \mathbf{x}_{i-1}^r, \Sigma^r) & r \in [\eta_t, s_t, \beta_t, \phi_t]. \end{cases} \tag{12}$$

Algorithm 5: Voting Rough Position

```
Input: Keypoints set $\{S\}_{i=1}^N$, $\mathbf{x}_{t-1}$, cluster cutoff threshold $T$
Output: Estimated object center $C$
1  for $r = 1$ to $N$ do
2      Every point votes to the object's center $(x_i, y_i)$ using the relative position from the last frame.
3  end
4  Cluster $\{x_i, y_i\}_{i=1}^N$ using cluster cutoff threshold $T$ and get $K$ cluster centers $\{C_i\}_{i=1}^K$;
5  Find the cluster center with most vote $C_j$ (L denotes $C_j$'s vote number);
6  if $L > \frac{N}{\alpha}$ then
7      Return $C_j$ (replace the translation part of state vector);
8  else
9      Return center position of $\mathbf{x}_{t-1}$;
10 end
```

where $\mathcal{N}(\cdot)$ denotes the Gaussian distribution and Σ is a diagonal covariance matrix whose elements are the variances of the affine parameters. The f_{vote} represents the voting algorithm which is the function of the parameters $\{S\}_{i=1}^N$ and the 2-D position part of $\mathbf{x}_{i-1}$ actually. In (12), all the parameters in state vector $\mathbf{x}_t$ except the (x_t, y_t) are governed by a Gaussian distribution around their previous state $\mathbf{x}_{t-1}$. The 2-D position parameters (x_t, y_t) are determined by the weighted sum of the voted position of the foreground keypoints and the previous position of the object target.

The observation model of our tracker is defined as

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto \log\left(1 + e^{-y\mathbf{w}^T\mathbf{z}_t}\right) + \alpha K(\mathbf{z}_t) - \beta L(\mathbf{z}_t) \tag{13}$$

where $K(\mathbf{z}_t)$ and $L(\mathbf{z}_t)$ denote the numbers of the foreground and background sift keypoints inside the patch corresponding to the particle $\mathbf{x}_t$. The first part of the right side of 13 is equal to 3. It is obvious that the (13) includes the information from both the SOD part and the KP part. If there are not matched keypoints between the two consecutive frames, our tracker will reduce to the pure SOD part. The workflow of the proposed tracking algorithm is illustrated in Fig. 2, and the pseudo code is summarized in Algorithm 6.

VII. Experiments

In our experiments, the object target is initialized according to the first frame in ground truth. The motion is characterized by the state vector $\mathbf{x}_t = (x_t, y_t, \eta_t, s_t, \beta_t, \phi_t)$, where x_t, y_t denote the x, y translation and $\eta_t, s_t, \beta_t, \phi_t$ represent the rotation angle, scale, aspect ratio, and skew direction at time t, respectively. All those parameters are similar to the parameters in [17]. We perform the experiments on ten video sequences, all the testing videos are the benchmark challenging sequences that can be downloaded from the **URL**.[1] [2] The

[1]http://www.cs.toronto.edu/ dross/ivt/

[2]http://vision.ucse.edu/ bbabenko/project_miltrack.shtml

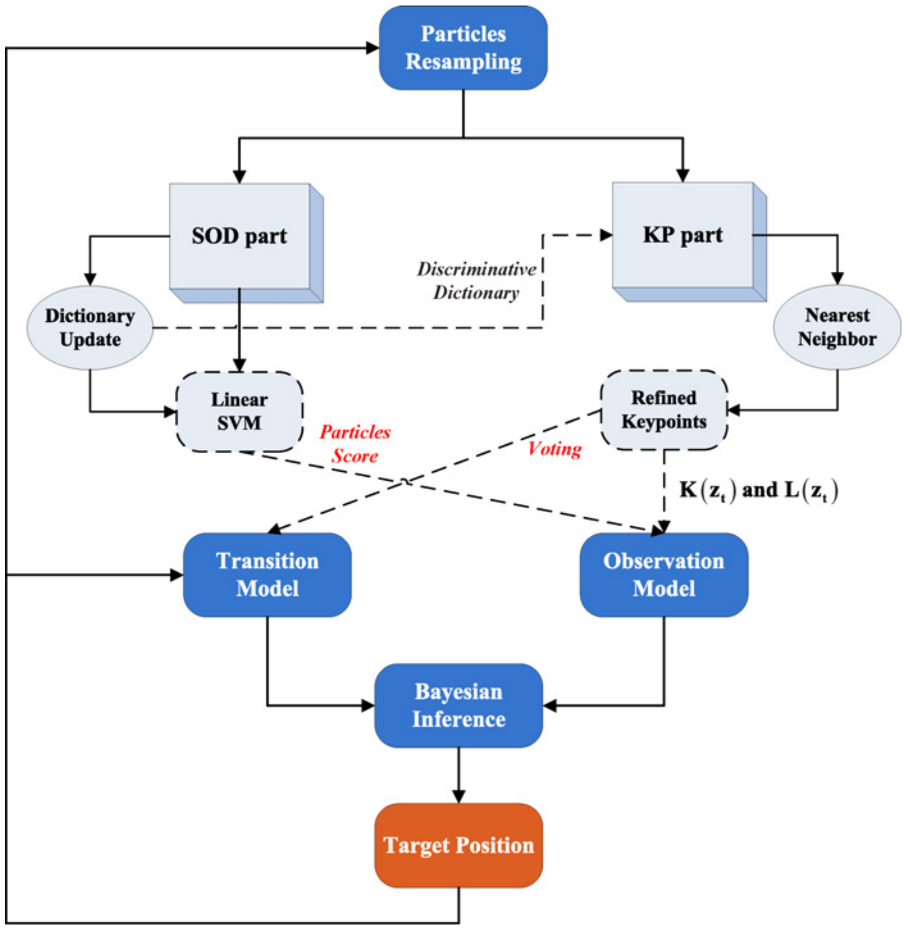

Fig. 2. Overview of the proposed tracker. Suppose the target position x_{t-1} at the time step $t-1$, the tracker draws the particles for both SOD part and KP part. In the SOD part, the discriminative dictionary $\mathbf{D}$ is updated using the Algorithm 1. Then, the scores of the particles are evaluated by the retrained linear SVM. In the KP part, the nearest neighbor classifier is built using the dictionary $\mathbf{D}$ provided by the SOD part in order to refine the keypoints matching and eliminate the background keypoints (Algorithm 3 and 4). The observation model (13) combines the particle score from the SOD part with the identified numbers of each kind of keypoints ($K(\mathbf{z}_t)$ and $L(\mathbf{z}_t)$). The transition model (12) is constructed using the voted position of the refined foreground keypoints and the target position at the previous time step. Finally, the tracker utilizes the Bayesian inference to select the best particle x_t that represents the current state of the object target.

challenges of those sequences are listed in Table I, including pose variation (in the plane or out of plane), illumination changes, occlusions, scaling, abrupt motion, and cluttered background. Implemented in MATLAB with MEX, our tracker runs at 0.1 frames per second with 800 particles on a standard Core 2 2.8 GHz computer.

A. Analysis the Performance of the SOD

In this subsection, we aim to validate that the proposed discriminative dictionary that includes both the foreground and the background information would provide more power than the generative dictionary used in ODLSR tracker [12]. For this purpose, we compare the performance of the SOD part (the proposed tracker without using KP part) with the ODLSR tracker. Two competitors are tested on four sequences (from the whole dataset used in this paper) and their error plots are shown in Fig. 3. Comparing the error curves, the ODLSR tracker achieves comparable results with the SOD only in Tiger1 sequence. However, in Tiger2 sequence, which includes the similar background and object target but with all the challenges enhancement, the performance of the ODLSR drops dramatically (97 average pixels error) while the SOD still keeps relative low error (11 average pixels error). It can be attributed to the discriminative power provided by the discriminative dictionary that consists both the foreground and the background information. It is worth to note that both the ODLSR and the SOD perform poorly in David clip, e.g., gradually drifting after frame #150, as it is shown in Fig. 3(d). The reason is that the dictionary-based tracker may easily introduce the wrong bases (sample patches) into dictionary, when the tracker gets an inaccurate target location. Then the noise appearing in dictionary will affect the capability of the object appearance model. With the additional constraints imposed by the KP part, the tracker will get more accurate target location. In the subsection VII-C, the advantage of combining the SOD with the KP will be illustrated.

B. Analysis the Effect of the Keypoints Refinement

In order to illustrate the effect of the proposed keypoint refinement method, we qualitatively analyze the results of comparison between the RANSAC-based refinement and our approach. Figs. 4 and 5 plot some screenshots of the matched sift point pairs between the consecutive frames in the Sylvester

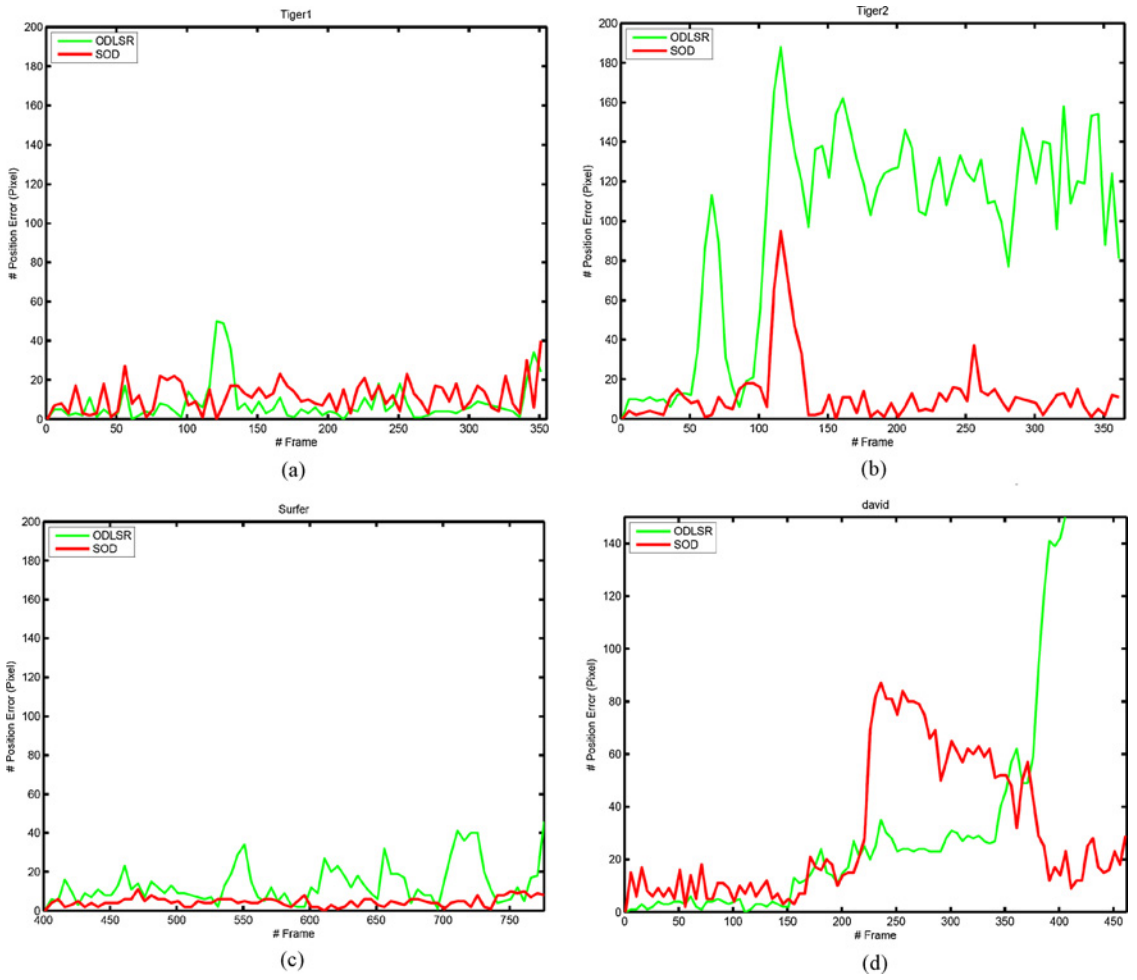

Fig. 3. Performance comparing between ODLSR and SOD (red curve denotes the SOD). (a) Tiger1. (b) Tiger2. (c) Surfer. (d) David.

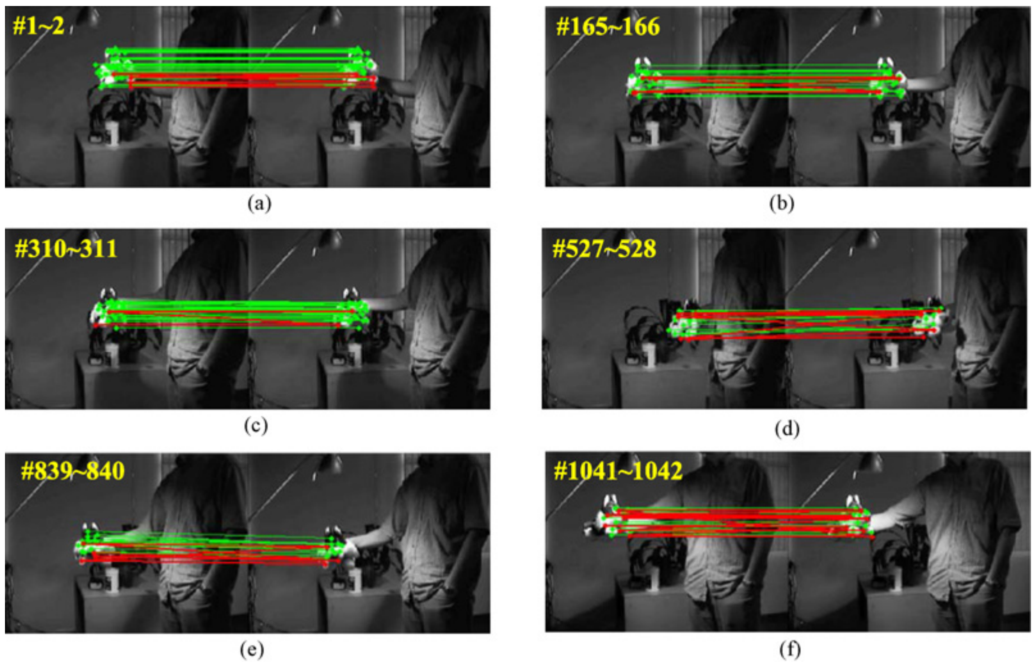

Fig. 4. Keypoints matched between the consecutive frames on Sylvester sequence (green lines denote the correct matches. Red lines denote the outliers).

Algorithm 6: Main Tracking Algorithm

```
**Input**: Frame set $\{F_i\}_{i=1}^{N}$, initial target state $\mathbf{x}_1$,
**Output**: Target states $\{\mathbf{x}_i\}_{i=1}^{N}$
1  Get initial SIFT points $S_1$ from $F_1$ with $\mathbf{x}_1$;
2  Sampling from $F_1$ to create over-complete foreground
   and background dictionary $\mathbf{D}_p$ and $\mathbf{D}_n$ to form the whole
   dictionary $\mathbf{D}_1$ as in Section III-A;
3  Get $M$ positive and negative patches $\mathbf{I}_{pos}$ and $\mathbf{I}_{neg}$, and
   compute sparse codes of all patches $\{\mathbf{z}_i\}_{i=1}^{M}$ with
   dictionary $\mathbf{D}_1$ as in Section III-B;
4  Training linear classifier to get $\mathbf{w}_1$ using 2 with training
   samples $\{\mathbf{z}_i, y_i\}_{i=1}^{m}$;
5  for $i = 2$ to $N$ do
6      Get matched points $S_i$ in $F_i$ corresponding to $S_{i-1}$;
7      Refine the matched points using Algorithm 3 and
       eliminate background keypoints by Algorithm 4 to
       get the refined SIFT points set $\widehat{S_i}$;
8      Calculate rough object position $r$ by Algorithm
           5// Transition Model
9      Get the best particle $\mathbf{x}_t$ using the Observation Model
       defined by 13;
10     Sampling from $F_i$ to update the $\mathbf{D}_p$ and $\mathbf{D}_n$
       respectively to construct the whole dictionary
       $\mathbf{D} = [\mathbf{D}_p, \mathbf{D}_n]$ with Online Dictionary Learning using
       the Algorithm 3;
11     Training the linear classifier to get $\mathbf{w}_i$ using 2;
12 end
13 Return The target states $\{\mathbf{x}\}_{i=1}^{N}$;
```

TABLE I

CHALLENGES OF THE TEST SEQUENCES

Video Clip	Pose	Illumination	Occlusion	Scaling	Motion	Cluttering
Tiger1	√‡[a]	√	√‡	√	√†[b]	√‡
Tiger2	√†	√‡	√‡	√	√‡	√‡
Cliffbar	√‡	×	×	√‡	√†	√‡
Surfer	√	×	×	√	√†	×
David	√‡	√†	×	√‡	√	×
Slvyester	√‡	√‡	×	√	√‡	√†
Faceocc1	×	×	√‡	×	×	×
Faceocc2	√†	√	√‡	×	√	×
Girl	√‡	√	√‡	√†	√‡	√†
Coke Can	√	√‡	√‡	×	√‡	√†

[a] ‡ denotes severe variation, illumination/scale changes, occlusion, fast random motion or cluttered background.

[b] † denotes modest variation, illumination/scale changes, partial occlusion, random motion, or cluttered background.

and Face Occlusion2 sequences, respectively. In these figures, all the keypoints in the left frames are considered as the foreground keypoints. The green lines denote the matched keypoint pairs between the consecutive frames that pass through the RANSAC algorithm, while the red lines present the pairs accepted by the RANSCA but further rejected by the keypoints matching refinement and background keypoints elimination. The matching results of all the frames of these two sequences can be found in the supplemental material.

As is shown clearly in Figs. 4(d) and (e) and 5(a)–(d), some red lines drawn on a slant with high degree are obviously the incorrect matches. With the help of the local sparse coding, our method can recognize the information around a candidate keypoint pair, then measure their similarity by means of the sparse residual. If the residual exceeds a threshold, it indicates that the two candidate keypoints correspond to different locations of the object target. In other words, they are not the correct matched pair. Beside rejecting the incorrect matching, the background keypoints also should be eliminated so as to prevent the appearance model suffering from the noise. In Figs. 4 and 5, some correct matches are also labeled as rejected pairs, e.g., the red lines between keypoint pairs whose location at the leaves which are behind the toy in Fig. 4(d) and (f) and at the computer screen that is behind the head of the person in Fig. 5(d), respectively. Since the foreground and the background information are encoded in an online learned discriminative dictionary, our proposed method can filter the background keypoints by using their sparse residual as the similarity measure. Therefore, even though those background matched pairs pass through the matching refinement, they will be discarded to the maximum extent by the background keypoints elimination. As mentioned in Section VI, the transition model will be affected by the position of the foreground keypoints (12) and the observation model will be partially influenced by the number of both the foreground and background keypoints (13). Thus, the proposed tracker will indeed benefit from such keypoints refinement procedure.

C. Comparative Tracking Results

In this section, we evaluate our proposed tracker on ten challenging sequences, all of which are publicly available. In addition, we test other six state-of-the-art trackers on the same video sequences for comparison. They are Online-AdaBoost (OAB for short) [24], fragment tracking (FragTrack) [20], multiple instance tracker (MIL) [23], nearest neighbor tracker (NNTracker) [33], $L1$ tracker [5] and ODLSR tracker [12]. The code of all those trackers are publicly available, and we keep the parameter settings provided by authors for all the test sequences. All the ten testing videos are the benchmark challenging sequence that can be downloaded from the URL[3].

For quantitative analysis, we use average center location errors as evaluation criteria to compare performance, the pixel error in every frame is defined as follows:

$$\text{error} = \sqrt{(x' - x)^2 + (y' - y)^2} \tag{14}$$

where (x', y') represents the object position given by tracker, (x, y) is the ground truth. The quantitative results are summarized in Table II. In Table II, each row represents average center location errors of seven comparison algorithms testing on a certain video sequence. The number marked with red indicates the best tracker in a certain testing video, blue indicates the second one. As shown in Table II, our proposed method acquires eight bests and two second best. For thorough investigation, we draw the excursion curve according to (14) for each video sequence [Figs. 6 and 11(d)]. In addition, Figs. 7, 8, 9, and 10 show the screen captures for some of the video clips. More details of the experiments will be discussed below.

[3]http://vision.ucsd.edu/bbabenko/project_miltrack.shtml

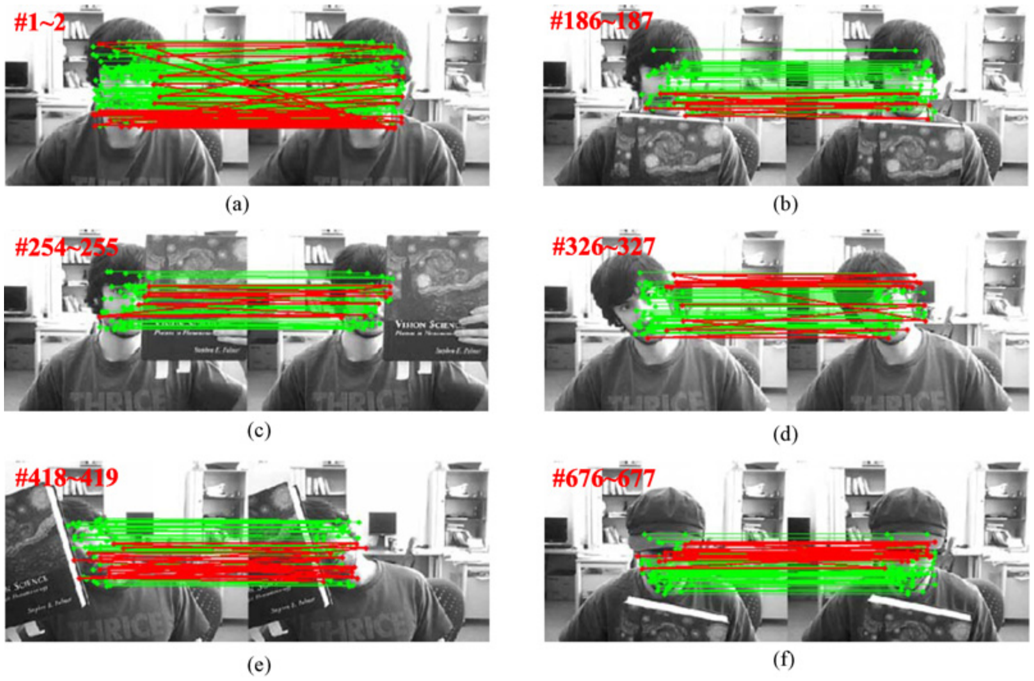

Fig. 5. Keypoints matched between the consecutive frames on Face Occlusion2 sequence (green lines denote the correct matches. Red lines denote the outliers).

a) ***Videos: Tiger1 and Tiger2 :*** Among all the testing videos in our experiments, Tiger1 and Tiger2 include most challenges, such as frequent occlusions, motion blur, background cluttering, drastic appearance changes (tiger opens mouth abruptly, rotates in the plane or out of plane) and illumination changes. The performances of seven methods are presented in Fig. 6(a) and (b), our proposed tracker achieves the best in these two sequence with very low pixel errors. In the subsection VII-A, without the help of the KP part, we have observed that the SOD is inferior to the ODLSR tracker slightly in Tiger1 sequence [Fig. 3(a)]. However, comparing the best two methods on Tiger1 sequence, the performance of the proposed tracker exceeds that of the ODLSR tracker due to enhancement obtained by the KP part [the red and orange curves in Fig. 6(a)]. When testing on Tiger2 clip whose challenges are more intensive than those of Tiger1, the average error of ODLSR increases to 97, while the proposed method only gets the eight average pixel error (see the red bounding box in Fig. 7). It is obvious that the performance of ODLSR degenerates dramatically [see the orange curve in Fig. 6(b)]. Therefore, the proposed tracker is more stable than ODLSR. It is can be explained by the following three factors.

1) The dictionary $\mathbf{D}$ in the SOD part, which including both foreground dictionary ($\mathbf{D}_p$) and background dictionary ($\mathbf{D}_n$), may provide more discriminative power than the dictionary used in ODLSR that encodes the information of the foreground object alone. Hence, our proposed tracker could suffer less drifting than ODLSR.
2) The SOD updates the dictionary in an online manner, while ODLSR takes a pseudo-online schema that drops the old dictionary totally and constructs a new one in each frame. Because the dictionary in SOD contains the past and current information of both the foreground and the background, the SOD tracker can handle the drastic appearance changes of target during tracking.
3) The KP part will integrate the local invariant information of the object target between the consecutive frames, which will enhance the behavior of the proposed tracker.

TABLE II
AVERAGE CENTER LOCATION ERRORS (PIXELS)

Video Clip	OAB	Fragment	NNtracker	MIL	L1	ODLSR	SODKP
Tiger1	42	47	32	20	19	7	5
Tiger2	Lost	38	81	9	35	97	8
Cliffbar	12	33	26	10	69	51	4
Surfer	14	137	38	11	12	13	4
David	65	68	20	24	56	42	5
Sylvester	14	29	20	10	30	14	8
Faceocc1	14	4	10	40	4	4	8
Faceocc2	25	48	18	7	40	8	8
Girl	29	11	12	31	31	11	6
Coke Can	5	35	24	17	53	21	9

OAB tracker loses the target after frame #255.

b) ***Videos: Surfer*** & ***Cliffbar :*** Our proposed tracker beats all the competitors on those test sequences. Note that the average center location errors of our method are only four pixels for the two sequences, both of which are below ten pixels (see the third and fourth rows in Table II). The Cliffbar includes more challenges than the Surfer sequence, so we only take the Cliffbar sequence for example. The goal of video clip Cliffbar is to track an object that changes in scale (e.g, the frame #76 in Fig. 8) and moves against a background that is very similar in texture. The motion of the target in this clip is fast and random, then a lot of frames show the motion blur.

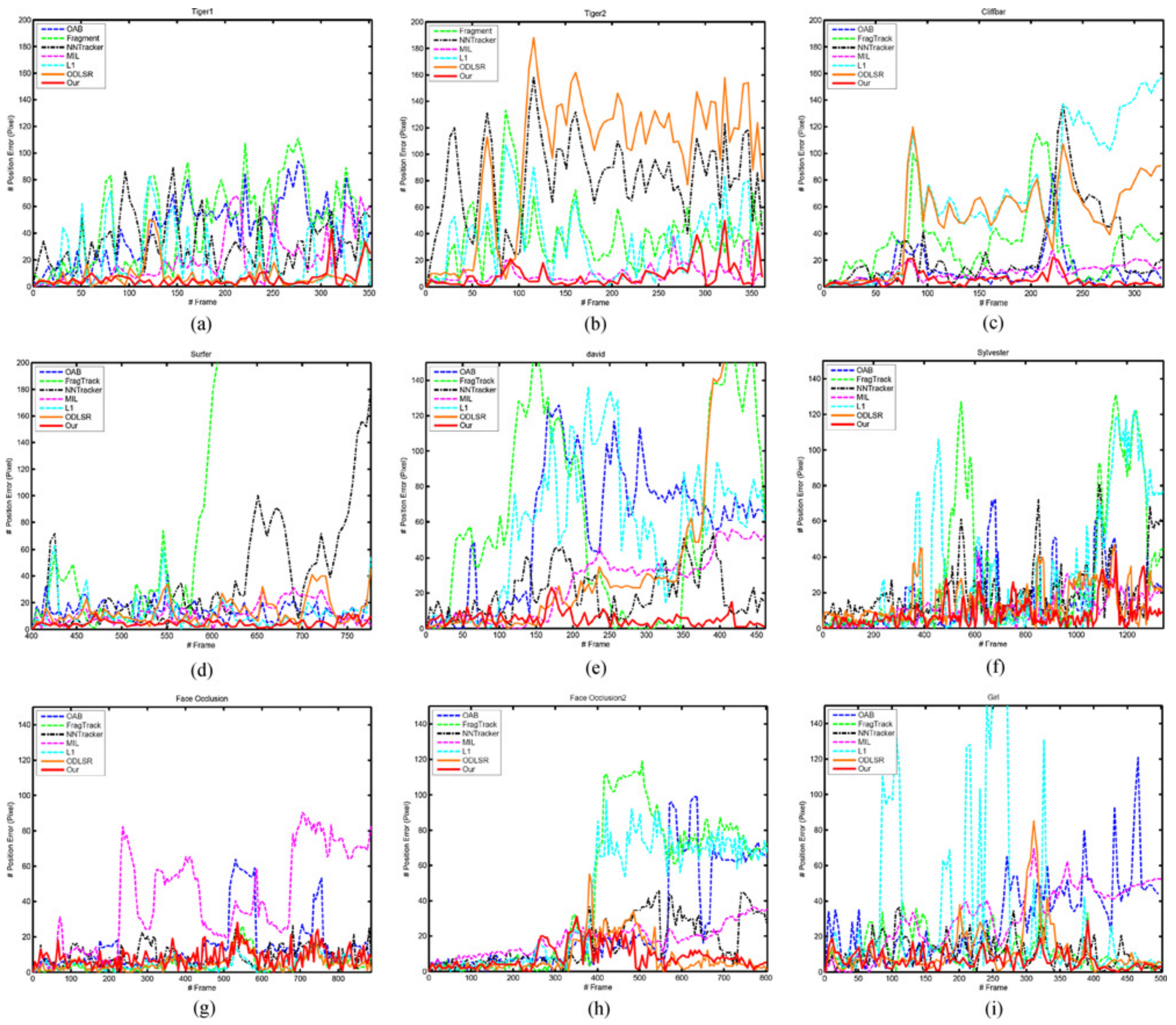

Fig. 6. Error plots for all test sequences (red curve denotes the proposed method). (a) Tiger1. (b) Tiger2. (c) Cliffbar. (d) Surfer. (e) David. (f) Sylvester. (g) Face Occlusion1. (h) Face Occlusion2. (i) Girl.

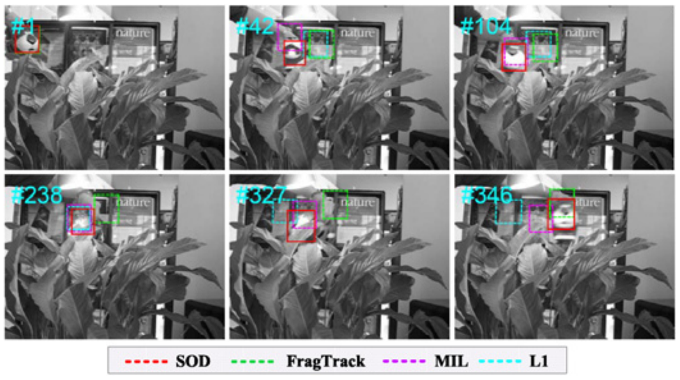

Fig. 7. Screenshots of tracking results of the best four trackers on Tiger2 sequence (red box denotes the proposed method).

Only three trackers get the errors below 20 pixels on this clip, they are OAB, MIL and the proposed tracker (see the 3rd row in Table II). The MIL tracker upgrades the OAB by applying the multiple instance learning, such learning schema can handle the sampling ambiguity in the training phase. Therefore, the drifting of the MIL is less than OAB. The error curve of the MIL is near to the proposed tracker. But if we observe the bounding boxes of the MIL tracker carefully in Fig. 8, they all show the different extent of drifting in the target location. The SOD part of the proposed tracker combines the local sparse coding with the online discriminative dictionary to encode the appearance information of the object target. Due to such combination, the appearance model is more powerful than that of the MIL tracker. That is why our proposed tracker is more accurate.

c) ***Videos: David*** & ***Sylvester*** *:* These video clips consist the similar difficulties including intense illumination change, scale, and pose variety. The most difficult in Sylvester is the frequent out-of-plane pose variety. Due to the the global and local information obtained from the SOD part and KP part respectively, the two parts can complement each other, leading to the position correction of the tracker after the significant

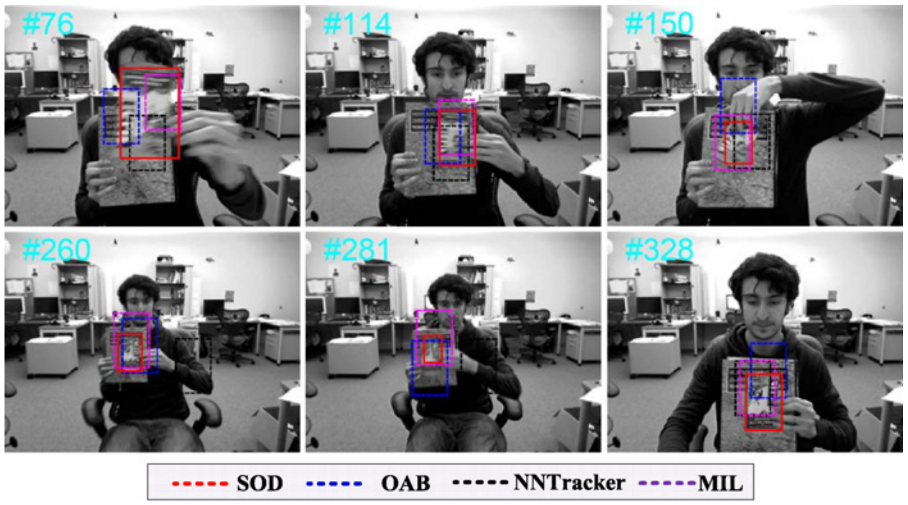

Fig. 8. Screenshots of tracking results of the best four trackers on Cliffbar sequence (red box denotes the proposed method).

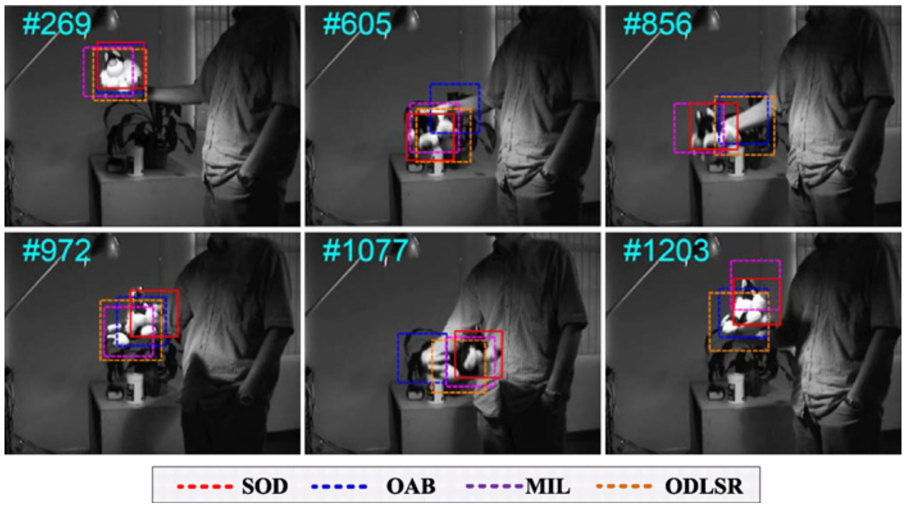

Fig. 9. Screenshots of tracking results of the best four trackers on Sylvester sequence (red box denotes the proposed method).

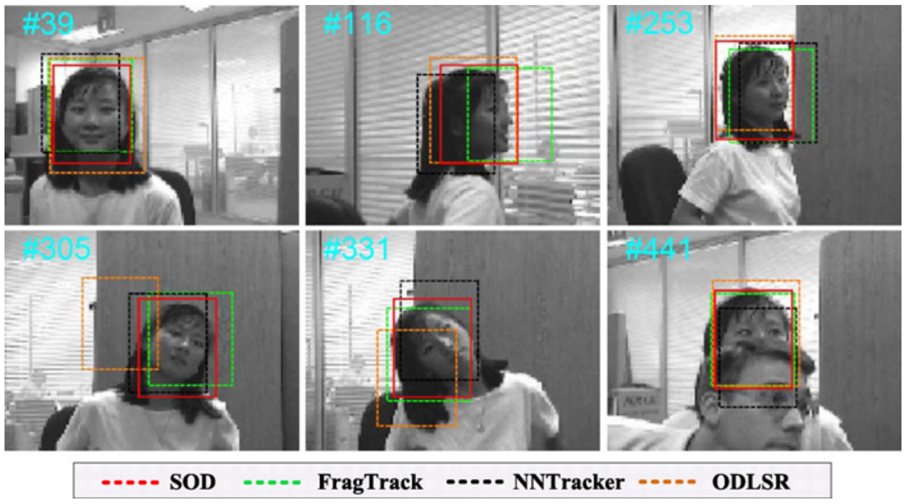

Fig. 10. Screenshots of tracking results of the best four trackers on Girl sequence (red box denotes the proposed method).

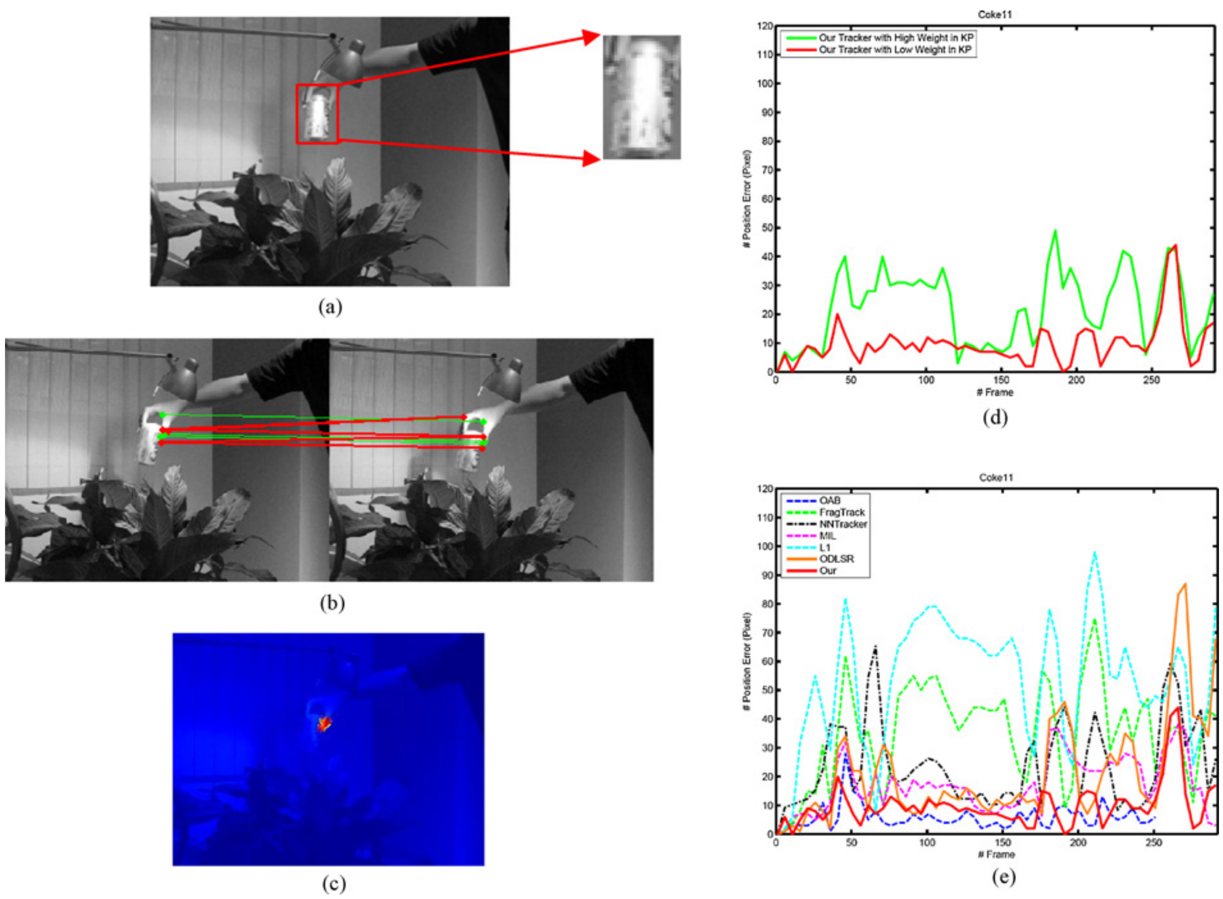

Fig. 11. (a) Object target: a specular Coke can, with its zoomed patch. (b) Sift keypoints appear at the corners or edges of the target; some of them cannot be refined by our KP part (green lines indicate the matched background keypoints pair). (c) Confidence of particles misled by the background keypoints (using the Jet colormap). (d) Performance comparison of different weights in transition model (w_1 = 0.8 and w_1 = 0.2). (e) Performance compared to the another six trackers.

pose change happening [e.g., the red curve in Fig. 6(f)], while the magenta and blue curves show that the second best MIL and OAB tracker present a little drift. Moreover, some screenshot results of all the trackers in the Sylvester sequence are shown in Fig. 9. It demonstrates that the pure discriminative tracker is not enough to handle the large pose variation without utilizing additional information of the object target.

In David sequence, the major challenges include the scaling and pose variety (smiling and take on/off the glasses). Only our tracker can handle those cases very well [see the red curve in Fig. 6(e)]. Note that we have compared the performance of the SOD part with the ODLSR tracker on the David sequence [Fig. 3(d)]. Without the KP part, the SOD cannot capture the target after #150. Moreover, the NNTracker, which is based on the traditional keypoint matching, also tends to drift after the frame #100 where the person turns his face and changes the distance to the camera. Therefore, those results can demonstrate the following two conclusions: 1) the proposed keypoint refinement method can improve the performance of the SOD tracker by integrating the local invariant information of the target; 2) the nearest neighbor classifier employing the sparse residual as similarity measure is more robust than the traditional Euclidean distance based similarity measure. With the help of the KP part, our tracker achieves the performance much better than the NNTracker [compare the red and the black curves in Fig. 6(e)].

d) ***Videos: Faceocc1, Faceocc2, and Girl*** : Two Face Occlusion videos are designed to test whether a tracking algorithm can handle the partial occlusion and pose changes. In Faceocc1 sequence, The FragTrack, $L1$, and ODLSR tracker perform the best with average four pixels errors. The person's face endures severe occlusion but its pose stays unchanged, therefore, FragTrack employs that a static part-based model can handle the case easily. Meanwhile, the dictionary of the ODLSR is sufficient to capture the appearance of the target. Similarly, the target template set of $L1$ tracker is unlikely to update during tracking. Actually, in our experiment, only two patches in the template set are updated in Faceocc1 sequence. However, similarly, but more challenging clip (e.g., the person puts on/off a hat or turns his face) Faceocc2, both FragTrack and $L1$ Tracker perform poorly (see the green and blue curves in Fig. 6(h) after #400 frame). For FragTrack, it could not update in the online fashion, which leads to failure when object appearance changes drastically. For $L1$ Tracker, it is easy to introduce the inaccurate patches into template set so as to cause the drifting problem. The OAB, MIL, NNTracker, and our proposed own the capability of online updating, leading to a good performance in this sequence.

The Girl sequence contains all the challenges listed in Table II. The pure discriminative tracker MIL and OAB failed to handle the combination of all the challenges, then tend to drifting after #200 frame gradually [see the blue and magenta curve in Fig. 6(i)]. The FragTrack, ODLSR, and NNTracker achieve the good performance with average 11, 11, and 12 pixels error, respectively. Here, we will compare the performance of these tracker with our proposal in several frame intervals. Since the static appearance model and the pure keypoints-based method cannot handle drastic appearance change, the FragTrack and NNTracker behave poorly between frames #50 and #150 [the green and black curve in Fig. 6(i)], where the girl turns her head around (the green bounding box in Fig. 10 frame #116). The ODLSR tracker tends to drift during the frames #300 $\sim$ #350, where the girl rotates her face toward left and right (the orange bounding boxes in Fig. 10 frames #305 and #331). Thus, such rotation can not be rapidly captured by the dictionary used in ODLSR as well as our proposed discriminative dictionary. However, the KP part employed in our tracker provides the ability of invariant to rotation, leading our tracker robust to such appearance change. When the girls are gradually occluded by other person between the frames #430 and #470 (the frame #441 in Fig. 10), the NNTracker cannot locate the target accurately while the proposed tracker always keep relative low errors. This demonstrates our tracker could handle the occlusion robustly.

e) ***Videos: Coke :*** In the Coke sequence, the object target is a specular coke that has little texture as shown in Fig. 11(a). Therefore, the amount of the SIFT keypoints within the region of the target will be small and even zero. If our uses the high weight w_1 in the transition model 12, the tracker will gradually drift. Although our robust keypoints matching method can identify the background keypoints within the target bounding box, the KP part might be ineffective when most of the keypoints appear at the corners or edges of the object target [see the sift points in Fig. 11(b)]. Because such kind of keypoints contain the context information from both foreground and background, which makes the nearest neighbor classifier 9 very confused, then the mismatched keypoints will vote to inaccurate position [as shown in Fig. 11(c), the particles sampled by the transition model cannot represent the position of the object target]. To illustrate the influence of the KP part in the case of specular target, we compare our tracker's performance under different weights, whose values are 0.8 (high) and 0.2 (low). As it is shown in Fig. 11(d), the tracker which is less influenced by the KP part strongly outperforms the one with putting more focus on the KP part (average error nine pixels comparing to 22 pixels). In other words, our tracker should only employ the SOD part when handling the low texture object target. We also compare the result of our tracker (with $w_1 = 0.2$ for KP part) to six other algorithms, as the excursion curves shown in Fig. 11(d). OAB wins the competition in coke can sequence. It achieves average location errors five pixels less than our tracker. That is partly because coke can is a specular object whose gradient characteristic is inconspicuous, while the intensity based haar-like feature used in OAB can capture the appearance characters of the object target. However, OAB loses the target completely after frame #255, which shows that our method is stable than OAB.

VIII. Conclusion

In this paper, we proposed a discriminative visual tracker via the sparse representation, online dictionary learning, and refined keypoint matching schema. The proposed tracker combines the local sparse representation with the online discriminative dictionary to encode the appearance information of both the object target and the background. Then the sparse codes of the samples was used to train a linear discriminative appearance model to best separate the target from the background. Additionally, a new keypoint matching schema that was robust to incorrect matches and an outlier keypoints was adopted to improve the performance of the proposed tracker. By applying several difficult benchmark videos, the experimental results demonstrated the effectiveness and stability of our approach compared to some state-of-the-art methods .

There were still some issues in our tracker that can be further improved. The experiments showed that the discriminative dictionary might includes the inaccurate patches (bases) whose appearances did not belong to target actually. This would degrade the performance of the tracker. Therefore, future directions of this paper will introduce a postprocessing method to refine the incorrect bases of the dictionary in the online manner, such as the pruning events and the structural constraints for object manifold in TLD tracker [36].

Acknowledgments

The authors would like to thank H. Wang from the Center for Pattern Analysis and Machine Intelligence, Xiamen University for his valuable suggestion. The authors would also like to thank the anonymous editor and reviewers who gave valuable suggestions that have helped to improve the quality of the manuscript.

References

[1] X. Li, W. Hu, and Z. Zhang, "Robust visual tracking based on incremental tensor subspace learning," in *Proc. Int. Conf. Comput. Vision*, 2007, pp. 1–8.

[2] Y. Wu, J. Cheng, J. Wang, and H. Lu, "Real-time visual tracking via incremental covariance tensor leanring," in *Proc. Int. Conf. Comput. Vision*, 2009, pp. 1631–1638.

[3] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomputing*, vol. 74, no. 18, pp. 3823–3831, 2011.

[4] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[5] M. Xue and L. Haibin, "Robust visual tracking using l_1 minimization," in *Proc. Int. Conf. Comput. Vision*, 2009, pp. 1436–1443.

[6] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, "Robust tracking using local sparse appearance model and k-selection," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 2011, pp. 1313–1320.

[7] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l_1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 2012, pp. 1830–1837.

[8] X. Jia, H. Lu, and M. H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 2012, pp. 1822–1829.

[9] W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Mar. 2012, pp. 23–25.

[10] T. X. Bai, Y. and F. Li, "Robust visual tracking with structured sparse representation appearance model," *Pattern Recogn.*, vol. 45, no. 6, pp. 4399–4404, May 2012.

[11] Z. Han, J. Jiao, B. Zhang, Q. Ye, and J. Liu, "Visual object tracking via sampled-based adaptive sparse representation," *Pattern Recogn.*, vol. 44, no. 9, pp. 2170–2183, Mar. 2011.

[12] Q. Wang, F. Chen, W. Xu, and M. H. Yang, "Online discriminative object tracking with local sparse representation," in *Proc. IEEE Workshop Appl. Comput. Vision*, Jan. 2012, pp. 345–352.

[13] B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. Kulikowski, "Robust and fast collaborative tracking with two stage sparse optimization," in *Proc. Eur. Conf. Comput. Vision*, 2010, pp. 624–637.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc.*, vol. 58, no. 1, pp. 267–288, 1996.

[15] B. Efron, T. Hastie, and I. Johnstone, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–499, 2004.

[16] J. Mairal and F. Bach, "Onlne dictionary learning for sparse coding," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 689–696.

[17] D. Ross, J. Lim, R. S. Lin, and M. H. Yang, "Incremental learning for robust visual tracking," in *Proc. Int. J. Comput. Vision*, vol. 77, no. 1–3, 2007, pp. 125–141.

[18] J. Lim, D. Ross, R. S. Lin, and M. H. Yang, "Incremental learning for visual tracking," in *Proc. Adv. Neural Inform. Process. Syst.*, 2004, pp. 793–800.

[19] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-grid objects using mean shift," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jan. 2000, pp. 142–149.

[20] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jan. 2006, pp. 798–805.

[21] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Dec. 2001.

[22] M. Isard and J. Maccormick, "Bramble: A Bayesian multiple-blob tracker," in *Proc. Int. Conf. Comput. Vision*, vol. 2. 2001, pp. 34–41.

[23] B. Babenko, M. H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, 2011.

[24] H. Grabner and H. Bischof, "Online boosting and vision," in *Proc. Comput. Vision Pattern Recogn.*, 2006, pp. 260–267.

[25] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *Proc. Int. Conf. Comput. Vision Workshop*, 2009, pp. 1393–1400.

[26] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1631–1643, Aug. 2005.

[27] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 11, pp. 412–418, Nov. 1981.

[29] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 448–461, Mar. 2010.

[30] D. N. Ta, W. Chen, N. Gelfand, and K. Pulli, "SURFTrac: Efficient tracking and continuous object recognition using local feature descriptors," in *Proc. IEEE Comput. Vision Pattern Recogn.*, 2009, pp. 2937–2944.

[31] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vision*, 2006, pp. 404–417.

[32] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[33] S. Gu, Y. Zheng, and C. Tomasi, "Efficient visual object tracking with online nearest neighbor classifier," in *Proc. Asian Conf. Comput. Vision*, 2010, pp. 271–282.

[34] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato, "SIFT feature tracking for video stabilization," in *Proc. Int. Conf. Image Anal. Process.*, 2007, pp. 825–830.

[35] K. Mikolajczyk and C. Schmid, "Performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2004.

[36] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, Mar. 2011.

Yuan Xie received the Master's degree from the School of Information Science and Technology from Xiamen University, Xiamen, China, in 2010. He is currently pursuing the Ph.D. degree in the State Key Laboratory of Intelligent Control and Management of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His current research interests include image processing, computer vision, machine learning, and pattern recognition.

Wensheng Zhang received the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, in 2000.

He joined the Institute of Software, CAS, in 2001, where he is currently a Professor of machine learning and data mining and the Director of Research and Development Department, Institute of Automation, CAS. He has published over 32 papers in the areas of modeling complex systems, statistical machine learning and data mining. His current research interests include computer vision, pattern recognition, artificial intelligence, and computer human interaction.

Cuihua Li received the Ph.D. degree in automatic control from Xi'an Jiaotong University, Suzhou, China.

He is currently a Professor of image/signal processing and the Director of Computer Science Department, Xiamen University, Xiamen, China. He has published many papers in the areas of image processing, wavelet and pattern recognition. His current research interests include signal processing, pattern recognition, and wavelet analysis.

Shuyang Lin received the Master's degree from the Department of Computer Science, Xiamen University, Xiamen, China, in 2011.

His current research interests include image processing, computer vision, machine learning, and pattern recognition.

Yanyun Qu (M'05) received the B.Sc. and M.Sc. degrees in computational mathematics from Xiamen University, Xiamen, China, and Fudan University, Shanghai, China, in 1995 and 1998, respectively, and the Ph.D. degree in automatic control from Xi'an Jiaotong University, Suzhou, China, in 2006.

She has been with the Department of Computer Science, Xiamen University since 1998. She was appointed a Lecturer in 2000 and was appointed as an Associate Professor in 2007. Her current research interests include pattern recognition, computer vision, image/video processing, machine learning, and so on.

Dr. Qu is a member of ACM.

Yinghua Zhang received the Bachelor's degree in computer science and technology from Shandong University, Shandong, China, in 2007. She is currently pursuing the Ph.D. degree at the State Key Laboratory of Intelligent Control and Management of Complex Systems, Institute of Automation Chinese Academy of Sciences, Beijing, China.

Her current research interests include artificial intelligence, machine learning, and probabilistic graphic models.