Visual Tracking Using Strong Classifier and Structural Local Sparse Descriptors

Bo Ma, Jianbing Shen, *Senior Member, IEEE*, Yangbiao Liu, Hongwei Hu, Ling Shao, *Senior Member, IEEE*, and Xuelong Li, *Fellow, IEEE*

***Abstract*—Sparse coding methods have achieved great success in visual tracking, and we present a strong classifier and structural local sparse descriptors for robust visual tracking. Since the summary features considering the sparse codes are sensitive to occlusion and other interfering factors, we extract local sparse descriptors from a fraction of all patches by performing a pooling operation. The collection of local sparse descriptors is combined into a boosting-based strong classifier for robust visual tracking using a discriminative appearance model. Furthermore, a structural reconstruction error based weight computation method is proposed to adjust the classification score of each candidate for more precise tracking results. To handle appearance changes during tracking, we present an occlusion-aware template update scheme. Comprehensive experimental comparisons with the state-of-the-art algorithms demonstrated the better performance of the proposed method.**

***Index Terms*—Local descriptor, sparse representation, strong classifier, visual tracking.**

I. Introduction

VISUAL tracking is an important problem in multimedia processing and has been widely used in applications such as video surveillance, robotics, human computer interaction, and medical image analysis. Although steady progresses [6], [19], [43], [31], [12], [54]–[56] have been made to the efficiency and robustness of object tracking in recent years, it is still a difficult task due to appearance changes of the target object caused by factors such as illumination variation, occlusion, background clutter, pose variation and shape deformation [5], [7], [9], [15], [21].

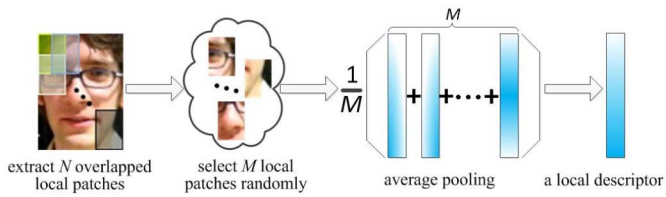

Fig. 1. Process of generating a local descriptor.

Recently, sparse coding based appearance models have been successfully applied to visual tracking [49]. Feature pooling computes the final feature vector based on certain statistics of sparse codes such as average pooling [27] and concatenating pooling [38], which is critical for building a good appearance model. The main problem with current pooling methods for visual tracking lies in the fact that final features considering the sparse codes of all patches are usually sensitive to occlusion and other interfering factors. When the target undergoes a long-time partial occlusion or heavy deformation, performance of these methods would degrade greatly. In addition, these existing pooling methods fail to take full advantage of the spatial configuration of local patches and the discriminative power of sparse codes. To address these issues, we extract structural local sparse descriptors from a fraction of all patches by performing a pooling operation. The proposed method first samples overlapped local image patches inside the target region, and then each image patch is represented with its sparse codes. As shown in Fig. 1, the target is represented by a series of local descriptors generated by average pooling sparse codes of a few selected patches. Intuitively, when the target is partially occluded or deformed, some local descriptors generated by occluded or deformed patches may lose discrimination, but the un-occluded local patches can still generate some discriminative local descriptors. Moreover, these local descriptors can retain enough spatial information. Therefore, the proposed local descriptors are able to deal with partial occlusion and deformation.

The nonlinear theory using local coordinate coding [45] has provided theoretical foundation for us to advocate discriminative tracking using sparse representation. In order to distinguish the target from the background, we formulate the visual tracking task as a classification problem using the proposed structural local sparse descriptors. Descriptors of positive and negative training samples are extracted from the target region and the background regions respectively. Each type of descriptors is

Manuscript received February 25, 2015; revised June 11, 2015; accepted July 27, 2015. Date of publication July 30, 2015; date of current version September 15, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61472036 and Grant 61272359, by the National Basic Research Program of China (973 Program) under Grant 2013CB328805, by the Key Research Program of the Chinese Academy of Sciences under Grant KGZD-EW-T03, and by the Fok Ying-Tong Education Foundation for Young Teachers' Specialized Fund for Joint Building Program of Beijing Municipal Education Commission. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eckehard Steinbach.

B. Ma, J. Shen, Y. Liu, and H. Hu are with Beijing Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: bma000@bit.edu.cn; shenjianbing@bit.edu.cn).

L. Shao is with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: ling.shao@ieee.org).

X. Li is with the Center for Optiical Imagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelong_li@opt.ac.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/TMM.2015.2463221

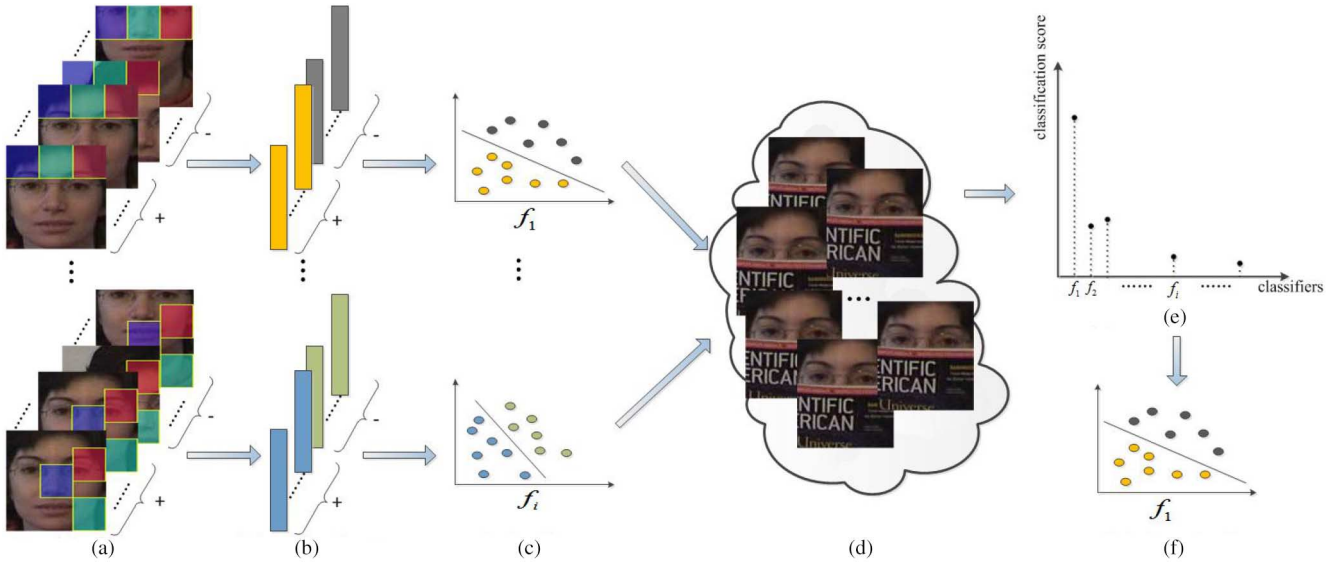

Fig. 2. Process of training the best classifier using local descriptors. (a) Extract a set of overlapped local image patches for each training image and select several local patches randomly (take three patches as an example). (b) Generate local descriptors using selected local patches. (c) Train classifiers using local descriptors. (d) Sample a test image set from the current frame, and generate corresponding local descriptors and test classifiers. (e) Comparing the classification scores; the classifier f_1 has the highest score because local patches used for training f_1 are not occluded in the test image set. (f) Obtain the best classifier. (a) Training image set. (b) Local descriptors. (c) Classifiers. (d) Test image set. (e) Classification score. (f) The best classifier.

used to train a weak classifier, and the final strong classifier will be obtained using a boosting method (Fig. 2). Since each weak classifier only explains partial appearance of the visual target, a boosting method can be utilized to produce a strong and discriminative appearance model for robust visual tracking.

Many sparse coding based tracking methods determine the current target by the reconstruction error. To make full use of the reconstruction information of the target, we adjust the classification score of each candidate by assigning a weight which is calculated by its structural reconstruction error. When the classification confidence might become inaccurate with great appearance variations, the proposed weight computation method using the structural reconstruction error can alleviate and moderate the drifting problem. In addition, a template update strategy using incremental principal component analysis (PCA) and an occlusion handing scheme is introduced to capture appearance changes of the target, and the proposed strong classifier is updated with new samples collected from the estimated target location.

Our tracking method can handle appearance changes of the target caused by different challenges. To handle occlusion during tracking, we propose the structural local sparse descriptor to represent the target region. When the target suffers from partial occlusion, our structural local descriptors with local patches that are not occluded can achieve good discrimination. Moreover, an occlusion-aware template update scheme can handle appearance changes during tracking especially caused by occlusion. For background clutter, the proposed tracker considers the background information to train our classifiers, which ensures the good separating capacity of distinguishing the target from the cluttered background. Our tracker could handle fast motion efficiently under a particle filter framework, where the candidates could cover the target area in the current frame with larger affine parameters with the smooth assumption of adjacent frames. Besides, a weight calculation method using the structural reconstruction error is introduced to adjust the classification score of each candidate, where a penalty term is added to the reconstruction error by considering the spatial layout among local patches. Our source code will be publicly available online.[1]

The main contributions are summarized as follows.

1) We propose a target representation using the collection of local sparse descriptors, where each descriptor represents partial appearance of the target.
2) A robust visual tracking approach using a strong classifier and structural local sparse descriptors is presented under the boosting framework. An update scheme that is robust to occlusion for the strong classifier is presented in our work.
3) A weight calculation method using the structural reconstruction error is introduced to adjust the classification score of each candidate, where a penalty term is added to the reconstruction error by considering the spatial layout among local patches.

II. Related Works

A lot of tracking methods have been proposed to deal with the appearance changes during tracking, and readers can refer to the surveys [44], [46], [26] and a recent benchmark [40]. Most recent tracking algorithms can be roughly categorized as either generative or discriminative [2], [4].

Generative tracking methods search for the most similar region to the target model by some metric. These methods update the target appearance model dynamically to make the model fit for the target appearance changes and reduce the drifting problem. Ross *et al.* [43] learned the dynamic appearance of

[1][Online]. Available: http://github.com/shenjianbing/boostingtrack

the target via incremental low-dimensional subspace representation to adapt online to changes of the target appearance. Kwon *et al.* [24] proposed a visual tracking decomposition for robust object tracking and applied different types of feature templates for object representation. Zhou *et al.* [51] presented a tracking scheme for the abrupt motion problem based on stochastic approximation Monte Carlo. Motivated by sparse representation in face recognition, Mei *et al.* [29] successfully applied it to visual tracking and solved the occlusion problem using trivial templates. In [37], a visual tracking method based on robust non-negative dictionary learning was proposed. Wang *et al.* [35] proposed a least soft-thresold squares tracking algorithm by modeling the error term with the Gaussian-Laplacian distribution. In [52], a tracking algorithm was proposed based on a discriminative sparse similarity map which establishes the relationship between candidates and templates. Wang *et al.* [36] exploited PCA and sparse representation to develop appearance models for object tracking.

Discriminative tracking methods usually treat tracking as a binary classification task which separates the object from its surrounding background. These methods first train a classifier in an online manner employing information from both the target and the background, then the classifier is applied to candidate targets sampled from the next frame. Avidan [3] integrated the Support Vector Machine (SVM) classifier into an optic-flow-based tracker. In [5], an ensemble of weak classifiers was trained online to distinguish the object from the background. Babenko *et al.* [6] used multiple instance learning (MIL) by putting put the positive and negative samples into some positive and negative bags respectively to learn a discriminative model to solve the ambiguity problem. Kalal *et al.* [23] developed a semi-supervised learning approach where tracking results were regarded as unlabeled and samples were selected with structural constraints. Grabner *et al.* [17] used the online AdaBoost feature selection algorithm to select the most discriminating features for tracking. In [18], Grabner *et al.* proposed a semi-supervised online boosting algorithm to handle the drifting problem. Zhang *et al.* [47] utilized a random sparse compressive matrix to reduce dimensionality of Haar-like features, and then trained a naive Bayes classifier with the low-dimensional compressive features. In [16], a tracking method was proposed to overcome spatial distraction using discriminative spatial attention. In [42], a tracker based on a discriminative appearance model and superpixels was proposed. Zhang *et al.* [48] proposed a discriminative feature selection method which coupled the classifier score with the sample importance. Some hybrid methods that combine generative and discriminative approaches to get more robust results were proposed in [50], [14], [28].

Feature pooling has been widely used in sparse coding based appearance models. In [38], Wang *et al.* directly concatenated all sparse codes of a target to represent it, and then a linear classifier was learned. However, their method for representing an object could result in the curse of dimensionality. Wang *et al.* [39] computed the feature vector by the max pooling operator which may miss the spatial information, then they used multiscale max pooling to preserve the spatial information. In [27], Liu *et al.* employed local sparse representation which was computed by average pooling of sparse codes of local patches to model the target appearance. Their method achieved good results especially in dealing with partial occlusion, but the histogram only considered little spatial information among patches. In [38], an alignment-pooling method across local patches was proposed to obtain the subsampling of the sparse representation by patches at the same position of different templates. It can achieve good tracking performance when the object undergoes moderate or small deformation, but may fail when large deformation occurs.

To deal with the drifting problem when updating the templates, numerous approaches have been proposed. In [22], Jia *et al.* used sparse representation and incremental subspace learning to reconstruct a new template and then exploited it to replace an old template. It was efficient but still had a problem that the updated eigenbasis vectors would degenerate gradually if noise or occlusion existed. Wang *et al.* [35] presented a simple update scheme by replacing outliers with corresponding parts of the mean vector. In order to make full use of the spatial structural information of local image patches, we extract local sparse descriptors with sparse codes and each descriptor can represent partial appearance of the target. Then we present a boosting-based strong classifier to implement the tracking algorithm using the collection of local sparse descriptors. A generative weight model for each candidate is constructed via the structural reconstruction error for more accurate tracking results. To handle occlusion during tracking, the proposed strong classifier is updated by reconstructing a new template.

III. Proposed Visual Tracking Algorithm

In this section, the proposed tracking algorithm is described in detail. We first show how the local descriptors of the target are generated by local sparse codes. Next, we describe the classifier training process and the weight computing method. The update strategy of template and classifier is then presented.

A. Local Descriptors by Local Sparse Codes

Given an object image I, we can extract a set of overlapped local image patches $X = \{x_i | i = 1, 2, \cdots, N\} \in R^{d \times N}$ inside the target region with a sliding window, where x_i is the i-th column local vectorized patch extracted from image I, d is the dimension of image vectors and N is the number of local patches. If we have a set of templates $T = [T_1, T_2, \cdots, T_n]$, we extract local patches from T in the same way. Then a dictionary $D = [d_1, d_2, \cdots, d_{N \times n}] \in R^{d \times (N \times n)}$ to encode local patches of candidate targets can be obtained, where n is the number of templates. Each item of the dictionary is a d-dimensional vector corresponding to a local patch extracted from T. The first several frames are tracked using the real-time compressive tracking algorithm [47] and the tracking result (normalized to 32×32) of each frame is treated as a template—then we obtain the set of templates T.

Each local image patch x_i in X can be encoded with the dictionary D by solving

$$\min_{\alpha_i} \|x_i - D\alpha_i\|_2^2 + \lambda \|\alpha_i\|_1 \tag{1}$$

where $\alpha_i \in R^{(N \times n) \times 1}$ is the sparse code, corresponding to local patch x_i. λ is a regularization parameter that controls sparsity and reconstruction error.

In order to fully describe the candidate X, we generate a set of structural local sparse descriptors for X with the sparse coefficients. Given a candidate object region X, we can extract a set of overlapped local patches (denoted as $x_1, x_2, \cdots, x_N$), and calculate a sparse code for each local patch according to (1). We select M patches from all N local patches of the object region randomly (denoted as $x_{i_1}, x_{i_2}, \cdots, x_{i_M}$) and pool the sparse codes of these selected patches using average-pooling

$$f_i = \frac{1}{M} \sum_{j=1}^{M} \alpha_{i_j} \tag{2}$$

where α_{i_j} is the sparse code of the local patch x_{i_j}. Then, we get a local descriptor of the object region: $f_i \in R^{(N \times n) \times 1}$.

As shown in Fig. 1, the locations of local patches contain the spatial structural information of the target region. If different local patches are selected, these selected patches contain the spatial structural configuration of the target region. In our experiments, we use average pooling to compute the final descriptor, which reduces the dimension of the final feature by simplifying a portion of the spatial information. Therefore, the object region can generate $m (m = C_N^M)$ local descriptors totally. The number m of local descriptor groups depends on M. Nevertheless, M is carefully chosen, making it neither too large nor too small. The reason is that when the target suffers from partial occlusion or local deformation, a large M will introduce occluded patches in the final descriptor with a high probability, which makes the formed descriptor inaccurate, while a small M leads to insufficient representation of our descriptor. The proposed descriptor is named as structural local sparse descriptor.

B. Boosting-Based Strong Classifier

To construct a discriminative appearance model using the best of the local descriptors, we present the boosting-based strong classifier. To initialize the classifier, we first get training sample set S which is composed of N_p positive samples and N_q negative samples from the first n frames. We draw $p (p = 9)$ positive samples around the tracking result of each frame via perturbation of a pixel around the target position, and then $N_p (N_p = n \times p)$ positive samples are obtained. We select N_q negative samples near the target in the n-th frame with a Gaussian perturbation. Each training sample can generate m local descriptors. After that, our AdaBoost classifier is learned with the local descriptors in the training sample set.

Most traditional boosting methods [53] train their classifiers with the whole training set S in each iteration. We find that those correctly classified samples make less contribution to train a weak classifier in the next iteration. Therefore, we choose 2/3 samples from the entire training set according to their weights to train the weak classifier in the next iteration. Those samples with high weights of probability are more likely to be selected, and vice versa. Each selected example has m local descriptors, so we can train m weak classifiers using these local descriptors of selected examples. Next, according to the classification scores that are estimated on the entire training set S and the test set (Fig. 2), the best weak classifier with the highest classification score is selected. We denote the best weak classifier as $h_1(x)$. Then, all training samples of S are re-weighted so that the misclassified samples can get higher weights. We then obtain some weak classifiers $h_2(x), \cdots, h_k(x)$ by performing this process repeatedly. The final strong classifier $H(x)$ is defined as follows:

$$H(x) = \sum_{i=1}^{k} \rho_i h_i(x) \tag{3}$$

where ρ_i is the weight of $h_i(x)$.

We then use a linear classifier as the weak classifier by solving the following optimization problem:

$$w^* = \arg\min_w \frac{1}{L} \sum_{i=1}^{L} \log\left(1 + e^{-y_i w^T z_i'}\right) + \frac{\eta}{2} \|w\|_2^2 \tag{4}$$

where w is the classifier parameter, L is the number of selected training examples, $z_i' = [z_i^T, 1]^T$ and $z_i \in R^{(N \times n) \times 1}$ is a local descriptor, y_i represents the property of the local descriptor z_i, i.e., $+1$ for positive training example and -1 for negative training example and η is a regularization term.

C. Weight Calculation by Reconstruction Error

We draw some samples around the target location in the previous frame as the candidate targets of the current frame and each candidate target has a classification score with the strong classifier $H(x)$. A simple approach is that the candidate target with the largest score is treated as the tracking result of the current frame. However, the classification result may not be accurate when the target experiences great appearance variations, because the samples used to train the classifier are sampled from the previous frame.

In order to improve the accuracy, we assign a weight to each candidate target to adjust its classification score. The weight value of a candidate target is calculated by the reconstruction error under the dictionary and reflects the similarity between the candidate target and templates. From Section III-A, we know that the dictionary can be denoted as

$$D = \left[d_1, \cdots, d_N, d_{N+1}, \cdots, d_{2N}, \cdots, d_{(n-1)N+1}, \cdots, d_{nN}\right]. \tag{5}$$

If the candidate X is perfect, the local patch x_i in X should be represented well by sub-dictionary $D_i = [d_i, d_{N+i}, \cdots, d_{(n-1)N+i}] \in R^{d \times n}, 1 \leq i \leq N$. The sparse code under D_i is denoted as $\beta^i = [\alpha_i^i, \alpha_i^{N+i}, \cdots, \alpha_i^{(n-1)N+i}]^T \in R^{n \times 1}$, where α_i^j is the jth item of α_i.

The reconstruction error ε_i of the local patch x_i under D_i is calculated by

$$\varepsilon_i = \left\|x_i - D_i \beta^i\right\|_2^2. \tag{6}$$

In order to calculate the reconstruction error more conveniently, (6) can be rewritten as

$$\varepsilon_i = \|x_i - D(\omega_i \otimes \alpha_i)\|_2^2 \tag{7}$$

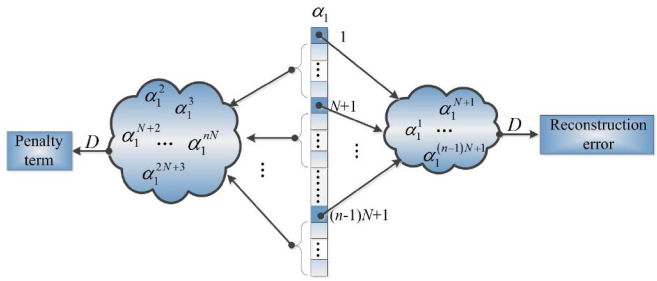

Fig. 3. Calculation process of the reconstruction error and the penalty term. α_1 is the sparse code of local patch x_1 under D, where darker color elements have larger values.

where $\alpha_i \in R^{(N\times n)\times 1}$ is the sparse code of x_i under dictionary D, $\otimes$ is the element-wise multiplication

$$\omega_i = \left[\omega_i^1, \omega_i^2, \cdots, \omega_i^{(N\times n)}\right]^T \in R^{(N\times n)\times 1}, \tag{8}$$

$$\omega_i^j = \begin{cases} 1, & j = i, N+i, \cdots, (n-1)N+i \\ 0, & \text{others.} \end{cases} \tag{9}$$

In order to make full use of the sparse code α_i, we add a penalty term $\|D\cdot((1-\omega_i)\otimes\alpha_i)\|_1$ to (7), so (7) can be rewritten as follows:

$$\varepsilon_i = \|x_i - D(\omega_i\otimes\alpha_i)\|_2^2 + \gamma\|D\cdot((1-\omega_i)\otimes\alpha_i)\|_1 \tag{10}$$

where γ controls the strength of the penalty term. If x_i can be represented well by sub-dictionary D_i, the penalty term will be very small, otherwise very large. The main advantage lies in its taking the spatial layout between local patches into account, and the calculation process is shown in Fig. 3.

After reconstruction errors of all patches in candidate X are obtained, the weight W of X can be calculated by

$$W = \sum_{i=1}^{N} \exp(-\beta\varepsilon_i) \tag{11}$$

where β is a constant and N is the number of local patches in X.

When partial occlusion happens to the target, the occluded patches may have large reconstruction errors, but the other patches still have small reconstruction errors, so the weight of the target still retain a relative large value. If a candidate cannot be reconstructed by target templates well, its weight is smaller than that of the target because each patch of this candidate has a large reconstruction error.

D. Strong Classifier Update

Occlusion-Aware Template Update: Templates should be updated dynamically to adapt to appearance changes [29]. In our work, each template T_i has a weight a_i and its initial value is 1. After getting the target of each frame, we update the weight of each template via $a_i = a_i\cdot e^{-\theta}$, where θ is the angle between T_i and the target. Template update is carried out every t frames and we choose the template with the least weight to be replaced by a new template. Many authors [22], [35] have used sparse representation and incremental subspace learning to reconstruct a new template and then exploit it to replace an old template. It is efficient but it does not consider the occlusion occurred during tracking, which means that the template set may be updated by a polluted new template and cause tracking failure. Before a new template is reconstructed, the tracking results are employed to incrementally update the eigenbasis vectors. If noise or occlusion exists in tracking results, the updated eigenbasis vectors will degenerate gradually. Therefore, the occlusion in tracking results should be first handled.

After getting the best candidate target of each frame, we reconstruct the target by

$$[\hat{z}, \hat{s}] = \arg\min_{z,s} \frac{1}{2}\|\bar{y} - Uz - s\|_2^2 + \lambda_1\|s\|_1 \tag{12}$$

where $\bar{y} = y - \mu$, y represents the observation vector, U is composed of PCA basis vectors, μ is the mean vector, z denotes the coefficients of $\bar{y}$ under U and s is the noise term. Then y is reconstructed by

$$y_r^i = \begin{cases} y^i, & s^i = 0 \\ \mu^i, & s^i \neq 0 \end{cases} \tag{13}$$

where y_r^i denotes the i-th item of y_r which is the reconstructed observation vector. y_r^i is collected and then we incrementally update U and μ.

When updating templates every t-frames, we first compute the coefficient $\hat{z}$ of the current observation vector by (12), then reconstruct a new template by

$$T^* = U\hat{z} + \mu \tag{14}$$

where T^* is the new template for updating the template with the least weight.

Classifier Update: We draw $p(p = 9)$ positive samples around the tracking result of each frame via perturbation of a pixel, then replace some old positive samples, and update the negative samples every t frames via drawing some samples near the target. In order not to introduce bad samples to positive samples, we should check whether each new sample is reasonable before updating positive samples. The reconstruction errors between new samples and templates are calculated, and the sample with a large reconstruction error is regarded as occlusion or unreasonable. The reconstruction error of a new sample is then obtained by

$$\delta = \sum_{i=1}^{N} \|x_i - D\alpha_i\|_2^2 \tag{15}$$

where x_i is the ith local patch extracted from the new sample, and N is the number of local patches. If $\delta > \delta_0$, the new sample is regarded as occlusion, where δ_0 is a predefined threshold which determines whether the new sample is occluded or not.

The AdaBoost classifier is then retrained by the updated training sample set. When the target is occluded, the new samples collected around the tracking result should not be added to the positive samples, which help our AdaBoost classifier avoid the adverse impact of occlusion. For each candidate target, the classification score using the AdaBoost classifier can be obtained, meanwhile, the weight value is calculated using the structural reconstruction error. The likelihood function of candidate targets is constructed by

$$p = W\cdot H(x) \tag{16}$$

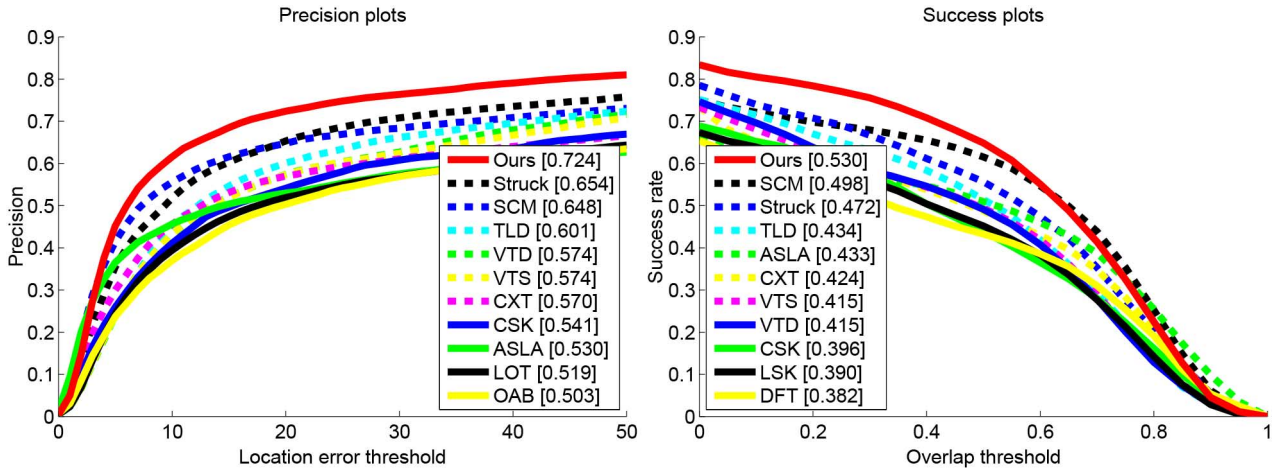

Fig. 4. Precision plots (left) and success plots (right) over all 51 video sequences. The legends show the precision scores and AUC scores for each tracker, respectively.

and the candidate with the highest likelihood value is regarded as the tracking result. Algorithm 1 summarizes our method.

Algorithm 1 The Proposed Tracking Algorithm

Input: sequence $o_1, \cdots, o_M$; object state s_1 at the first frame; the number of templates n; update frequency u;
Output: tracking results $s_t, t = 2, \cdots, M$.
1: **Initialization:**
template set T and positive training set N_p from the first n frames; negative training set N_q from the nth frame; reconstructed target set $\Phi = \emptyset$.
2: extract local descriptors for $[N_p, N_q]$.
3: construct the appearance model using local descriptors by the strong classifier $H(x)$ with (3).
4: **while** $t = n + 1, \cdots, M$ **do**
5: generate candidate targets X.
6: extract local descriptors for X.
7: calculate classification score and weight for X.
8: select tracking result s_t using (16).
9: draw p positive samples $\Rightarrow N_p$.
10: reconstruct s_t using (12) and (13), and then $\Phi = [\Phi, s_t]$.
11: **if** size$(\Phi) == u$ **then**
12: update U and μ using Φ, then $\Phi = \emptyset$.
13: reconstruct T^* using (14) and update T.
14: draw q negative samples $\Rightarrow N_q$.
15: extract local descriptors for $[N_p, N_q]$.
16: retrain $H(x)$.
17: **end if**
18: **end while**

IV. Experimental Results

Our tracking algorithm is evaluated with 29 trackers in a recent benchmark [40], where each method is tested on 51 challenging videos. We use the precision plot and success plot [40] to measure the overall performance. The precision plot indicates the percentage of frames whose center location error (the distance between center location of tracking result and that of ground truth) is less than a given threshold. The precision score of each tracker is represented with the score under the threshold $t = 20$ pixels. The success plot demonstrates the ratios of successful frames whose bounding box overlap is larger than a given threshold. The area under curve (AUC) of each success plot is used to measure the trackers.

The overall performances of the competing trackers on the 51 videos are displayed in Fig. 4, where only parts of the competing trackers are presented for clarity. The trackers shown in Fig. 4 are: Struck [19], SCM [50], TLD [23], ASLA [22], CXT [14], VTD [24], VTS [25], CSK [20], LSK [27], DFT [33], LOT [30], OAB [17]. Other trackers (such as L1APG [8], ORIA [41], SMS [10], VR-V [11], Frag [1], SemiT [18], BSBT [34]) are not displayed. For convenience, we directly use the results of these trackers provided with [40] to conduct comparative experiments with our results.

V. Experimental Setup

The proposed algorithm runs at 1.1 frames per second on an Intel Core i7 3.4 GHz with 4G memory using our unoptimized Matlab implementation. The number of templates is 8, training samples and candidate targets are all normalized to 32×32 pixels, and then 9 overlapped local patches with the size of 16×16 are extracted within the region with 8 pixels as step length. When constructing descriptors with local sparse codes, we select 3 from 9 local sparse codes to perform average pooling, which results in 84 local descriptors. For learning the classifier, we collect $N_p = 72$ (9 positive samples per frame and 8 consecutive frames) positive samples and $N_q = 150$ negative samples. We choose 2/3 samples randomly from all training samples to get 84 weak classifiers, and select the best weak classifier.

In our experiments, we obtain $k = 100$ weak classifiers after 100 iterations to establish our boosting-based strong classifier. However, for computational simplicity, we first sort these weak classifiers in ascending order according to their classification errors, and then the first 45 weak classifiers are used to form the final strong classifier $H(x)$. By this way, we can obtain comparable tracking accuracy with less computational time. The templates and the AdaBoost classifier are updated every five frames. Other parameters are set as follows. The variable λ in (1), γ in (10), β in (11), λ_1 in (12) and δ_0 are set to 0.01, 0.01, 5, 0.1 and

3.5 respectively. The affine transformation with six parameters is fixed to [8, 8, 0.01, 0, 0, 0]. The number of particles is set to 600. All the parameters mentioned in this section are fixed for all sequences.

A. Overall Performance

Fig. 4 shows the precision plots and success plots which illustrate the overall performance of our tracker and the competing trackers on 51 videos. For precision plots, we rank the trackers according to the result at error threshold of 20 pixels. For success plots, the trackers are ranked by the AUC scores. The precision scores and AUC scores for each tracker are shown in the legend of Fig. 4. Only the top 10 of the competing trackers and our tracker are displayed for clarity.

Our tracker achieves the best performance among all compared trackers (Fig. 4). In the precision plot, our algorithm performs 7.0% better than Struck, 7.6% better than SCM and Struck is slightly better than SCM. When the error threshold is reduced to 10 pixels, the SCM performs better than Struck, and our method still obtains the highest score. In the success plot, our tracker outperforms SCM by 3.2% and Struck by 5.8%. When given a specific overlap threshold (e.g., 0.5), our method still achieves the best performance. We can also observe that SCM achieves higher precision when the error threshold is relatively small and a higher success rate when the overlap threshold is relatively large. This is because SCM integrates holistic templates and local representations based on sparse code to handle appearance variations. Struck achieves higher precision scores than SCM, but lower AUC scores. The main reason for this is that Struck only predicts the location of the target and ignores scale variations.

Overall, our tracker performs better than the aforementioned well-known trackers. The main reasons are explained as follows. First, the proposed method generates the structural local descriptors for the target with good discrimination. Even if the target is partially occluded or contaminated, some local descriptors generated by the parts which are not contaminated still possess good discrimination. Our method can select the discriminative local descriptors to train the classifier, which ensures the accuracy of the strong classifier. Second, the weight value is calculated by the structural reconstruction error of the candidate target to adjust the classification score, and a small weight value is assigned to the bad candidate. Therefore, the weight model improves the robustness of our tracker. Third, our strong classifier update scheme is robust to heavy occlusion and alleviates the drifting problem during the tracking process.

B. Different Pooling Methods and Weak Classifiers

In order to represent a type of spatial configuration of the target, we select M patches from all N local patches of the object region randomly. Each patch has its own sparse codes. To obtain a more compact feature for these patches, we pool the sparse codes of these selected patches. Average-pooling is used in our setting; nevertheless, other pooling methods are suitable for our method as well. We compare the overall tracking performance of our method with average-pooling, concatenate pooling and max-pooling with all the other parameters fixed. As shown in Fig. 5, the performance scores using these three pooling methods are close to each other, and average-pooling gets a slightly higher performance score than the other two pooling methods. It proves that the pooling method is a small part of the reason for the success of our approach.

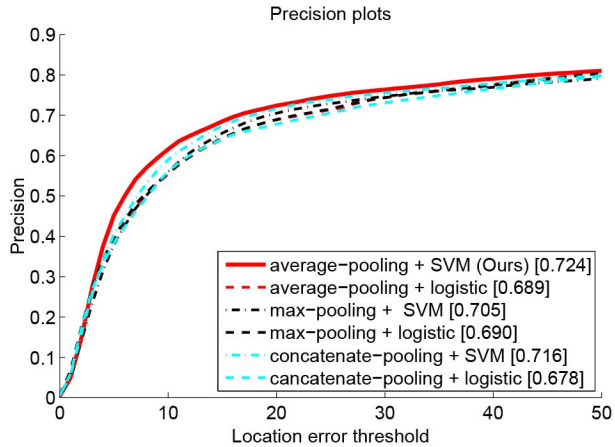

Fig. 5. Comparison of the overall tracking performance for our tracking framework with different pooling methods and different weak classifiers.

In fact, other tracking methods using pooling methods are included in the overall estimation in the benchmark, for example, SCM with average-pooling and ASLA with aliment-pooling. But our tracking method gets higher performance scores than others. To make our final strong classifier, the linear weak classifier is used as in (4). To verify the tracking accuracy of the proposed approach under different weak classifiers, we compare the tracking performance of our method with the weak linear classifier and the weak logistic regression classifier in our experiments. As shown in Fig. 5, the tracking performance scores with these two weak classifiers are close to each other, and the performance of the tracking method with the weak linear classifier is slightly higher than that with the weak logistic weak classifier. We find that different weak classifiers have a small effect on the overall performance in our tracking framework.

C. Performance Analysis per Attribute

The performance of a tracker is affected by many factors which can be divided into 11 attributes [40]. The 51 videos are annotated with the 11 different attributes and one sequence can be annotated with several attributes. Based on the annotation, 11 subsets with different attributes are constructed. Each subset can be utilized to evaluate the robustness of trackers against specific challenging factors.

We compare our tracker with other methods on the 51 video sequences with respect to the 11 attributes mentioned above. Our tracker performs well in 7 of the 11 video subsets: background clutters, in-plane rotation, deformation, occlusion, motion blur, out-of-plain rotation and illumination variation and also performs well in the other 4 subsets. Fig. 6 shows the precision plots of our tracker and the competing trackers in these 11 attributes. These results show that our tracker is robust to appearance changes of the target object caused by some factors. On the occlusion subset, our method and the SCM perform favorably compared to other trackers, which suggests that local sparse representation is more

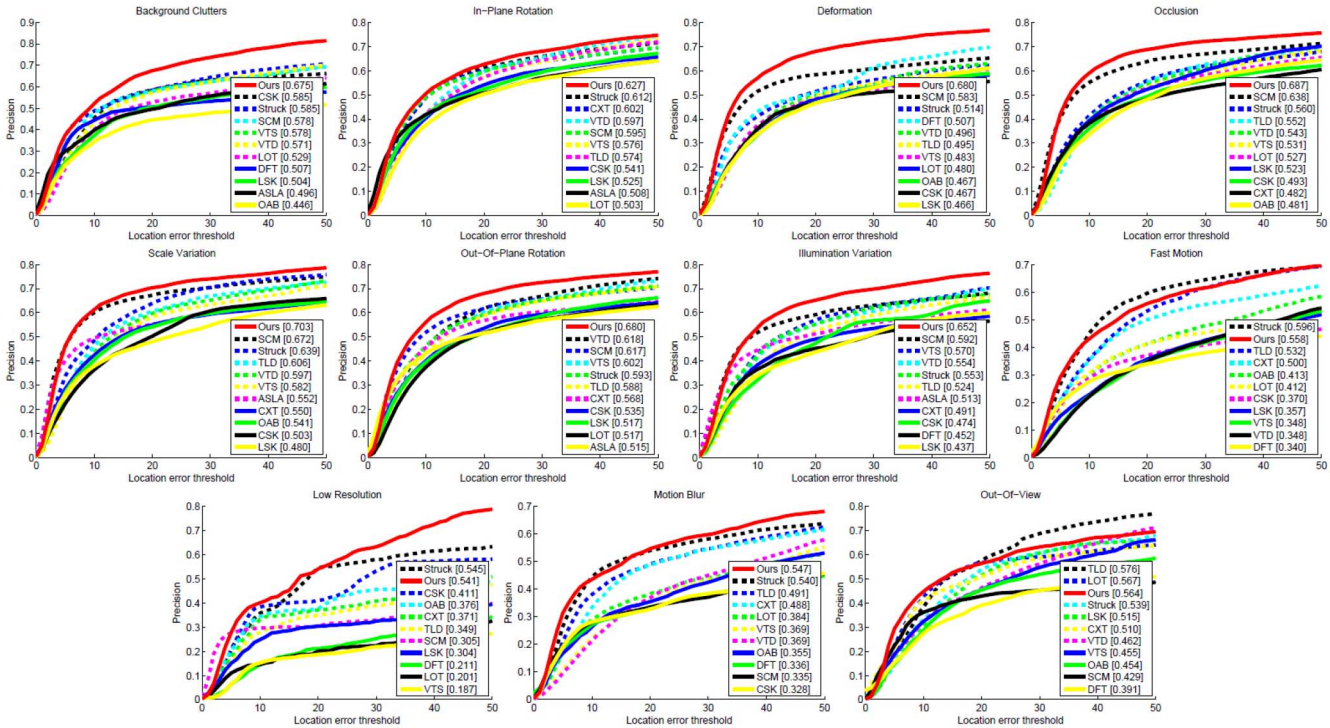

Fig. 6. Performance analysis using precision plots. The attributes are: background clutters, in-plane rotation, deformation, occlusion, scale variation, out-of-plain rotation, illumination variation, fast motion, low resolution, motion blur, out-of-view.

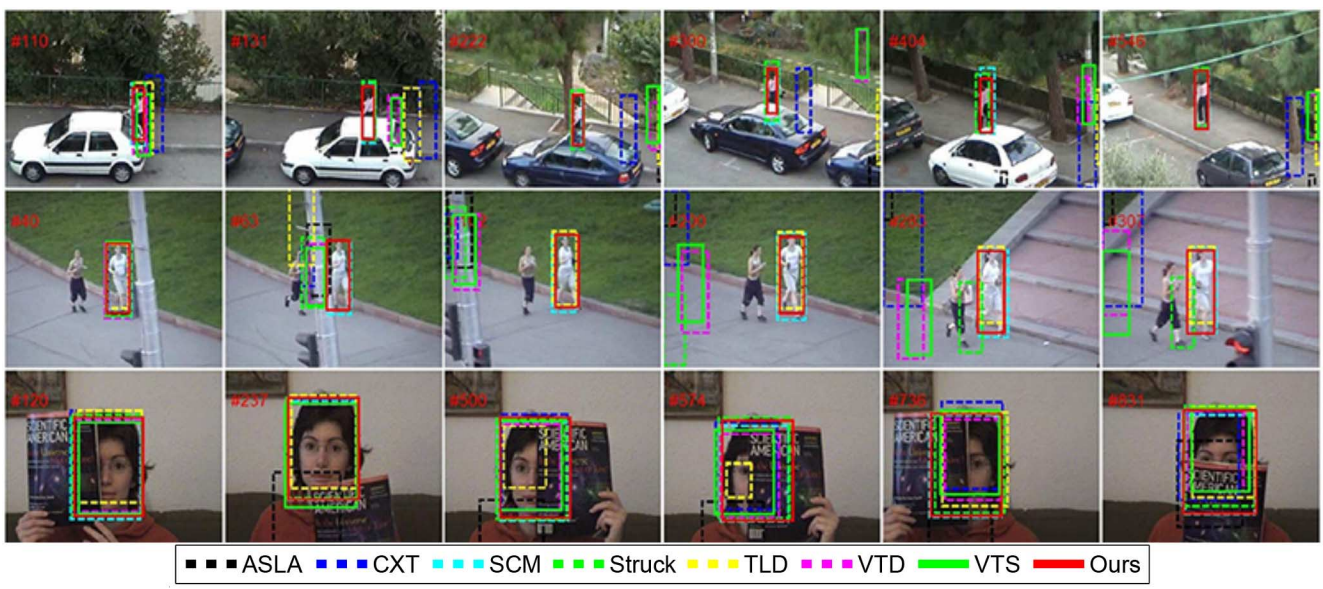

Fig. 7. Comparison results of the competing trackers on sequences *woman*, *jogging2*, and *faceocc1*. Objects are heavily occluded.

effective in dealing with occlusion compared with holistic representation. On the background clutters subset, the SCM, Struck and our method provide much better results. It can be attributed to the fact that background information is critical for robust visual tracking. On the deformation subset, our method provides superior results than others. It can be attributed to the proposed local descriptors that are robust to the target deformation.

D. Qualitative Evaluation

We select ten challenging sequences that include occlusion, background clutter, illumination variation, fast motion, deformation, out-of-plane rotation and so on. Some tracking results about these ten videos are shown in Fig. 7 to Fig. 9. We mainly discuss occlusion, background clutter, illumination variation and fast motion as follows.

Occlusion: As shown in Fig. 7, the tracked objects in the *woman*, *jogging2* and *faceocc1* sequences undergo heavy occlusion or long-time partial occlusion. In the *woman* sequence, the target undergoes long-time partial occlusion by cars together with out-of-plane rotation. The ASLA, CXT, TLD, VTD and VTS methods fail to track the object when the woman is partially occluded (e.g. #131). Only the Struck, SCM and our method are able to track the target throughout the entire sequence (e.g., #546). In the *jogging2* sequence, most trackers fail to track

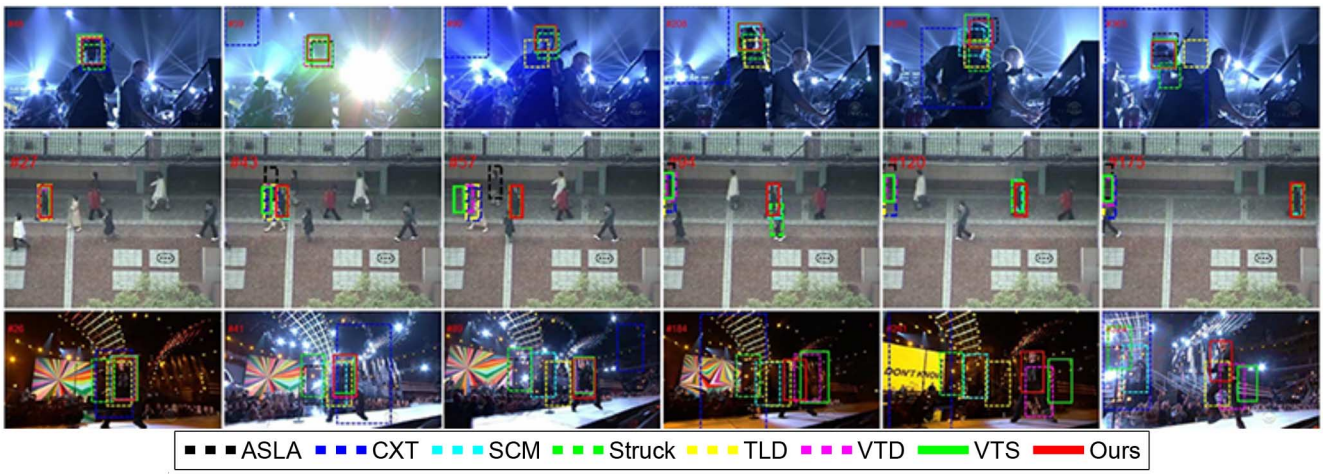

Fig. 8. Comparison results on sequences *shaking*, *subway*, and *singer2*. Objects appear in background clutters.

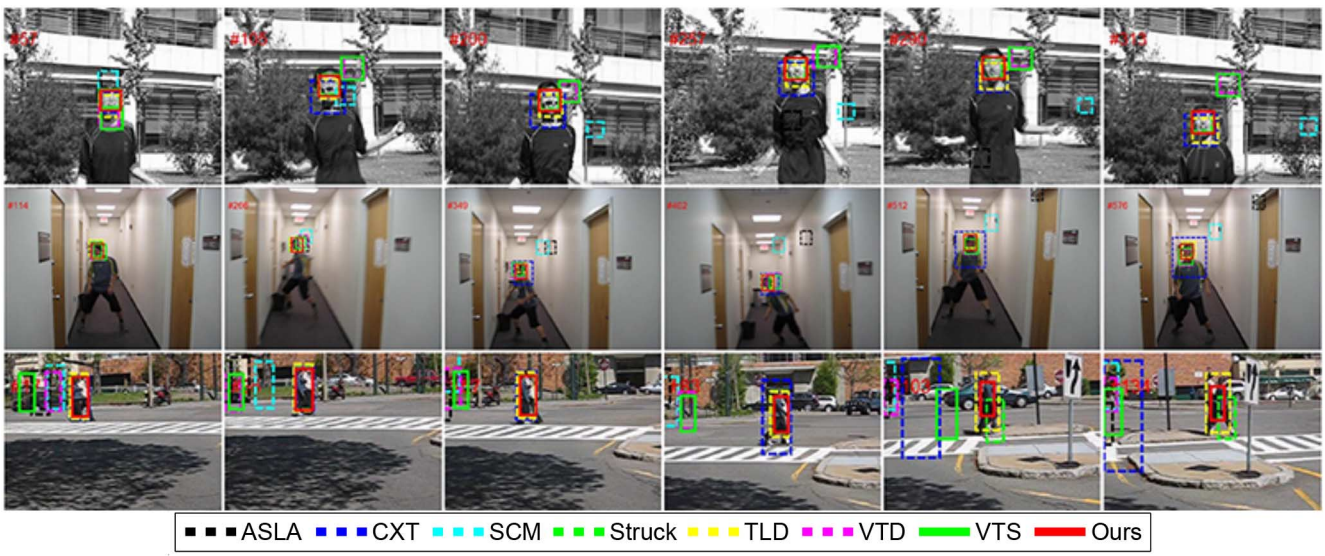

Fig. 9. Comparison results of the competing trackers on sequences *jumping*, *boy*, and *couple*. Objects undergo fast motion.

the object when the person is completely occluded, while our method can obtain the correct object again when the person appears again (e.g., #63). After about thirty frames, the TLD also obtains the object (#102), because it re-initializes the tracker using a detector. Our tracker does not drift away owing to the strong classifier and the robust template update scheme which can handle the occlusion in tracking results. In the *faceocc1* sequence, the target undergoes long-time heavy occlusion. Most trackers are able to track the target to some extent but the SCM, Struck and our method perform favorably compared to other trackers (e.g., #736, #831). Our structural local descriptors and strong classifier focus more on the uncontaminated local patches, which makes our tracker more accurate and robust.

Background clutter: Fig. 8 shows how the proposed method performs in the *shaking*, *subway* and *singer2* sequences when the targets appear in background clutters. In the *shaking* sequence, the target not only has the very similar color as the background but also undergoes large illumination variations. The CXT method locks on a wrong target due to the interference of illumination variations (e.g., #59). The ASLA, Struck and TLD methods drift away when the target shakes drastically, whereas the VTD, VTS, SCM and our method perform well in this sequence (e.g., #365). For *subway* sequence, the ASLA, CXT, VTD, VTS and TLD trackers fail to track the target when the tracking person is occluded by another walking person (e.g., #43). The SCM, Struck and the proposed method are able to keep track of the target to the end. The *singer2* sequence is challenging as the background is cluttered and the target experiences illumination variations, deformation and out-of-plane rotation. When the target rotates (e.g., #89), the ASLA, CXT, SCM, Struck and TLD methods fail to track the object. The VTD and VTS trackers drift away along with the move and rotation of the target (e.g., #184, #281). In contrast, our method performs best in this sequence.

Fast motion: Fig. 9 demonstrates the tracking results in three sequences (i.e., *jumping*, *boy* and *couple*) with fast motion. In the *jumping* sequence, the motion of the target is so drastic that the ASLA, SCM, VTD and VTS methods fail before frame #57. The TLD, Struck, CXT and our method can keep track of the target to the end, but our method achieves more accurate tracking results. The main reason is that our tracker can select the discriminative local descriptors to train the classifier. Meanwhile, the weight model (11) assigns smaller weights to the background. In the *boy* sequence, the target experiences fast motion together with out-of-plane rotation. While most trackers

are able to track the target to the end, our method obtains the lowest tracking error and the best tracking results. The motion of the target in the *couple* sequence is full of uncertainty that the ASLA, VTD and VTS fail at the beginning. Our tracker can successfully lock on the target in the tracking frames.

VI. Conclusion

In this paper, we employed sparse codes of local patches to generate local sparse descriptors of the object, and then these descriptors are combined into a boosting-based strong classifier with the discriminative appearance model. This strong classifier is applied to candidate targets to separate the object from its surrounding background. In order to adapt the proposed strong classifier to appearance changes of the target, we assign a weight to each candidate target to adjust its classification score. The weight is computed using the reconstruction error under a generative model and it reflects the similarity between the candidate target and templates. In addition, a robust occlusion-aware strong classifier update scheme is proposed to improve the tracking performance. Comparisons with the state-of-the-art trackers on a comprehensive benchmark show better performance of the proposed method.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, vol. 1, pp. 798–805.

[2] X. Zhou, L. Xie, Q. Huang, S. J. Cox, and Y. Zhang, "Tennis ball tracking using a two-layered data association approach," *IEEE Trans. Multimedia*, vol. 17, no. 2, pp. 145–156, Feb. 2015.

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

[4] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang, "Object tracking with multi-view support vector machines," *IEEE Trans. Multimedia*, vol. 17, no. 3, pp. 265–278, Mar. 2015.

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

[6] 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, Aug. 2011.

[7] Y. Bai and M. Tang, "Robust tracking via weakly supervised ranking SVM," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1854–1861.

[8] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1830–1837.

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

[10] R. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, Jun. 2003, vol. 2, pp. 228–234.

[11] 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, Oct. 2005.

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

[13] T. B. Dinh and G. G. Medioni, "Co-training framework of generative and discriminative trackers with partial occlusion handling," in *Proc. WACV*, 2011, pp. 642–649.

[14] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 1177–1184.

[15] J. Fan, X. Shen, and Y. Wu, "Scribble tracker: A matting-based approach for robust tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1633–1644, Aug. 2012.

[16] J. Fan, Y. Wu, and S. Dai, "Discriminative spatial attention for robust tracking," in *Proc. ECCV*, 2010, pp. 480–493.

[17] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. BMVC*, 2006, pp. 1–10.

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

[19] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 263–270.

[20] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. ECCV*, 2012, pp. 702–715.

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

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

[23] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 49–56.

[24] J. Kwon and K. Lee, "Visual tracking decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 1269–1276.

[25] J. Kwon and K. Lee, "Tracking by sampling trackers," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1195–1202.

[26] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, pp. 1–48, 2013.

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

[28] R. Liu, J. Cheng, and H. Lu, "A robust boosting tracker with minimum error bound in a co-training framework," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep.–Oct. 2009, pp. 1459–1466.

[29] X. Mei and H. Ling, "Robust visual tracking using L1 minimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep.–Oct. 2009, pp. 1–8.

[30] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1940–1947.

[31] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. ECCV*, 2002, pp. 661–675.

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

[33] L. Sevilla-Lara and E. G. Learned-Miller, "Distribution fields for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1910–1917.

[34] S. Stalder, H. Grabner, and L. van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, Sep.–Oct. 2009, pp. 1409–1416.

[35] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2371–2378.

[36] D. Wang, H. Lu, and M.-H. Yang, "Online object tracking with sparse prototypes," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 314–325, Jan. 2013.

[37] N. Wang, J. Wang, and D. Yeung, "Online robust non-negative dictionary learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 657–664.

[38] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Online discriminative object tracking with local sparse representation," in *Proc. WACV*, 2012, pp. 425–432.

[39] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang, "Transferring visual prior for online object tracking," *IEEE Trans. Image Process.*, vol. 21, no. 7, pp. 3296–3305, Jul. 2012.

[40] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2411–2418.

[41] Y. Wu, B. Shen, and H. Ling, "Online robust image alignment via iterative convex optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1808–1814.

[42] F. Yang, H. Lu, and M.-H. Yang, "Robust superpixel tracking," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1639–1651, Apr. 2014.

[43] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. Hengel, "Part-based visual tracking with online latent structural learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2363–2370.

[44] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surveys*, vol. 38, no. 4, p. 13, 2006.

[45] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," in *Proc. NIPS*, 2009, pp. 2223–2231.
[46] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomput.*, vol. 74, no. 18, pp. 3823–3831, 2011.
[47] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. ECCV*, 2012, pp. 864–877.
[48] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time object tracking via online discriminative feature selection," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4664–4677, Dec. 2013.
[49] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse coding based visual tracking: Review and experimental comparison," *Pattern Recog.*, vol. 46, no. 7, pp. 1772–1788, 2013.
[50] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1838–1845.
[51] X. Zhou and Y. Lu, "Abrupt motion tracking via adaptive stochastic approximation Monte Carlo sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 1847–1854.
[52] B. Zhuang, H. Lu, Z. Xiao, and D. Wang, "Visual tracking via discriminative sparse similarity map," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1872–1881, Apr. 2014.
[53] S. Avidan, "Ensemble Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.
[54] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 1090–1097.
[55] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.
[56] B. Christian, P. Alain, and S. Didier, "A superior tracking approach: Building a strong tracker through fusion," in *Proc. ECCV*, 2014, pp. 170–185.

Bo Ma is an Associate Professor with the School of Computer Science, Beijing Institute of Technology, Beijing, China. His research interests include computer vision and pattern recognition.

Jianbing Shen (M'11–SM'12) is a Full Professor with the School of Computer Science, Beijing Institute of Technology, Beijing, China. He has authored or coauthored about 50 journal and conference papers such as the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Cybernetics, the IEEE Transactions on Multimedia, the IEEE Conference on Computer Vision and Pattern Recognition, and the IEEE International Conference on Computer Vision. His research interests include computer vision and multimedia processing.

Prof. Shen is on the editorial board of *Neurocomputing*. He has also obtained many flagship honors including the Fok Ying Tung Education Foundation from Ministry of Education, the Program for Beijing Excellent Youth Talents from Beijing Municipal Education Commission, and the Program for New Century Excellent Talents in University from Ministry of Education.

Yangbiao Liu is currently working toward the M. S. degree in computer science at the Beijing Institute of Technology, Beijing, China.

Her current research interests include visual tracking algorithms.

Hongwei Hu is currently working toward the M. S. degree in computer science at the Beijing Institute of Technology, Beijing, China.

Her current research interests include visual tracking algorithms.

Ling Shao (M'09–SM'10) is a Full Professor and Head of the Computer Vision and Artificial Intelligence Group with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, U.K., and an Advanced Visiting Fellow with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K. His research interests include computer vision, image processing, pattern recognition, and machine learning.

Prof. Shao is a Fellow of the British Computer Society, a Fellow of the IET, and a Life Member of the ACM. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and other journals.

Xuelong Li (M'02–SM'07–F'12) is a Full Professor with the Center for Optical Imagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.