Generalized Pooling for Robust Object Tracking

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

***Abstract*— Feature pooling in a majority of sparse coding-based tracking algorithms computes final feature vectors only by low-order statistics or extreme responses of sparse codes. The high-order statistics and the correlations between responses to different dictionary items are neglected. We present a more generalized feature pooling method for visual tracking by utilizing the probabilistic function to model the statistical distribution of sparse codes. Since immediate matching between two distributions usually requires high computational costs, we introduce the Fisher vector to derive a more compact and discriminative representation for sparse codes of the visual target. We encode target patches by local coordinate coding, utilize Gaussian mixture model to compute Fisher vectors, and finally train semi-supervised linear kernel classifiers for visual tracking. In order to handle the drifting problem during the tracking process, these classifiers are updated online with current tracking results. The experimental results on two challenging tracking benchmarks demonstrate that the proposed approach achieves a better performance than the state-of-the-art tracking algorithms.**

***Index Terms*— Object tracking, feature pooling, Fisher kernel, local coordinate coding, sparse coding.**

I. Introduction

VISUAL tracking has many important practical applications in different areas, e.g. traffic transportation [1], video compression [2], and human computer interaction [3]. Although it has been studied for decades [4]–[6], it is still one of the most challenging tasks in computer vision. The main difficulty lies in establishing an effective appearance model to account for challenging appearance changes caused by factors, such as illumination variation, shape deformation, background clutter, and motion blur. Different visual descriptors are used to establish these appearance models for robust tracking. Recently, sparse representation based tracking algorithms [7] have attracted much attention in face recognition [8], image deblurring [9] and classification [10].

Manuscript received March 13, 2015; revised December 4, 2015, February 23, 2016 and May 16, 2016; accepted June 27, 2016. Date of publication July 7, 2016; date of current version July 21, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61472036, Grant 61272359, and Grant 61528106, in part by the National Basic Research Program of China (973 Program) under Grant 2013CB328805, and in part by the 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 Prof. Wen Gao. *(Corresponding author: Jianbing Shen.)*

B. Ma, H. Hu, J. Shen, and Y. Liu are with the 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; huhongwei@bit.edu.cn; shenjianbing@bit.edu.cn; liuyangbiao@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).

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/TIP.2016.2588329

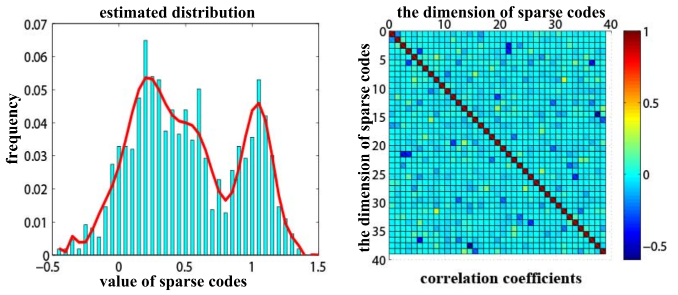


Fig. 1. Illustration of the estimated distribution and the correlation coefficients map. Left: the estimated distribution (the red solid line) of sparse codes in one dimension does not obey a single Gaussian distribution but more likely a mixture Gaussian distribution. Right: correlation coefficients by sparse codes corresponding to different dictionary items further prove that different dimensions are correlated.

It is well-known that a probabilistic distribution function can better characterize sparse codes of target patches, leading to a more generalized pooling framework for visual tracking. Since direct matching between distributions is time prohibitive in tracking, we propose to extract the Fisher vector from sparse codes and use the probabilistic model as the target representation. To instantiate our proposed pooling framework, we assume that sparse codes obey Gaussian mixture distributions, and develop a visual tracking algorithm using semi-supervised Fisher kernel classifiers. To make our approach computationally feasible, we further assume that each component of GMM takes on a diagonal covariance matrix. This assumption is obviously distinct from traditional sparse pooling methods. Taking average pooling [11] as an example, it not only assumes that sparse codes corresponding to different dictionary items are statistically independent, but also considers that each sparse code obeys a single Gaussian distribution. Thus, average pooling approach gives rise to one mean value as representative of all the sparse coefficients corresponding to the same dictionary item. Unfortunately, these simplifications do not hold in many cases and distributions of sparse codes can be arbitrarily complex as illustrated in the left part of Fig. 1. At least in the case of an overcomplete visual dictionary, sparse coefficients are more likely to be correlative rather than independent as illustrated in the right part of Fig. 1.

Under a particle filtering framework, our algorithm includes three main modules. The first module computes sparse codes for target patches using local coordinate coding (LCC) [12], [13]. By encoding patches with local

1057-7149 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

anchors in the manifold, LCC maintains locality of coordinate coding and produces similar sparse coordinate codes for similar patches. The second module performs a generalized pooling operation on sparse codes by extracting the Fisher vector as the target representation for random sparse code vectors [14], [15]. Fisher vectors have shown excellent results on image retrieval and classification tasks [16]–[18]. The main idea is to characterize a signal by its normalized gradient vector from a generative probability model. Compared with the Bag-of-Words model (BoW), Fisher vector can not only characterize the probabilities of occurrences of different image features, but also encode the high-order statistics of the distribution for image features. These works often assume that the image features are generated independently by GMM. While our method is under the framework of sparse coding and assume that sparse codes obey a probabilistic model. As shown in Fig. 1, we estimate the distribution of sparse codes in one dimension and find the distribution more likely obeys a mixture Gaussian distribution. After we get the probabilistic distribution function using these sparse codes, this function could be considered as the target representation. Fisher vector is used to avoid direct matching between distributions. As the third module, semi-supervised Fisher kernel classifiers are developed to compute the likelihood score for a candidate target.

The proposed method is significantly different to the existing Fisher vector based image retrieval and classification methods [17], [18]. Firstly, they aim to find a discriminative representation for a sample to be classified using GMM based Fisher vector. In contrast, we establish a generalized pooling framework by considering both high-order statistics of sparse codes and correlations between coding coefficients. In our method, Fisher vector is used to avoid the direct matching between distributions for computation efficiency. Secondly, they assume that the image features are generated independently by GMM, while our method assumes that sparse codes obey a probabilistic model. Thirdly, the linear SVMs are learned to classify samples in their work, but we learn a semi-supervised classifier by considering the similarity of samples for both labeled and unlabeled samples.

Tracking methods using sparse representation usually determine the current target by minimizing the holistic appearance reconstruction error or using a sparse pooling method. In sparse coding based tracking algorithms [7], sparse pooling methods usually encode local patches sampled from target regions, and then extract summary features from sparse codes of patches as the target representation. Previous pooling methods such as concatenating pooling, average pooling, and max pooling only use low-order statistics or extreme responses of sparse codes to derive the target visual representation. In contrast, we consider high-order statistics of sparse codes and correlations between coding coefficients corresponding to different dictionary items. Thus, we propose a probabilistic distribution function to better characterize sparse codes of target patches by a more efficient matching representation between distributions. Finally, we introduce GMM as the probabilistic distribution function to extract Fisher vectors for computation efficiency. Feature pooling using GMM based Fisher vector can be viewed as a special case of the proposed generalized pooling framework. Our source code will be publicly available online.[1]

The main contributions are summarized as follows:

- A generalized pooling method based on a probabilistic distribution function is proposed to extract summary features for sparse codes of target patches for object tracking. We propose to extract Fisher vectors from sparse codes to derive compact and discriminative visual descriptors.
- We instantiate our generalized pooling method by developing a Fisher tracker, where GMM models sparse code vector distributions, and a semi-supervised Fisher kernel classifier is utilized for classification. The extensive experiments on two challenging tracking datasets demonstrate better performance of our method compared with the state-of-the-art tracking algorithms.

II. Related Works

Appearance modeling is one of the most significant issues in visual object tracking. Typically, appearance modeling consists of statistical modeling and visual representation [20]. The main task of statistical modeling is to establish mathematical models by statistical learning theories, while visual representation constructs robust object descriptors with different features.

Various tracking methods based on different statistical modeling techniques were designed to represent different statistical properties of the object. For example, Jiang *et al.* [21], [22] incorporated an adaptive metric learning method into the tracking framework to obtain the optimal distance metric of different training samples. In [23], an ensemble of weak classifiers was trained online to distinguish the object from the background. But it is susceptible to drifting because the tracker updates its weak classifiers rapidly every frame. In order to overcome drifting, Babenko *et al.* [24] used multiple instance learning to learn a discriminative model to handle ambiguous positive and negative samples. However, this method would inevitably learn a poor classifier by introducing incorrect examples in the current frame, leading to tracking failure. Kalal *et al.* [25] developed a tracker by bootstrapping binary classifiers with structural constraints. In [26], Grabner *et al.* also proposed a semi-online boosting algorithm by combining a given prior and the classifier to alleviate the drifting problem. Different methods have been proposed to handle other challenges. For example, Oron *et al.* [27] presented locally orderless tracking to estimate the amount of local disorder in the object, which specializes in nonrigid and deformable objects. Zhang *et al.* [28] modeled the drift problem by a multi-expert restoration scheme for robust tracking. To address the significant appearance changes in long-term tracking, Ma *et al.* [29] solved translation and scale estimation of objects using correlation with temporal context. Hong *et al.* [30] built a representation that adapt to variations of object appearance during tracking by a cognitive psychology principle.

In visual tracking, researchers have proposed many global and local visual descriptors to model the statistical characteristics of the target appearance. Ross *et al.* [31] and Ho *et al.* [32] flattened the intensity of the target region into a

[1] *http://github.com/shenjianbing/poolingtrack*

high-dimensional vector directly. Zhao *et al.* [33] introduced the color distribution to describe the object appearance, and Danelljan *et al.* [34] extended the CSK tracker [35], [36] with color attributes on both photometric invariance and discriminative power. Ma *et al.* [37] proposed structural local sparse descriptors to learn a strong classifier for tracking. Zhou *et al.* [38] used local regions of the object and the mean shift for tracking. Li *et al.* [39] combined spatio-temporal visual information by incorporating Dempster-Shafer information to estimate the state of an object. Kwon and Lee [40] proposed a visual tracking decomposition and applied different types of feature templates for robust tracking.

In particular, sparse representation based tracking methods [7] have achieved great success by considering both local and global target appearances. Mei and Ling [41] introduced sparse representation into visual tracking as a pioneering work by assuming that the visual target is a sparse linear combination of holistic and trivial templates. Wang *et al.* [42], [43] assumed that the noise term obeys the Gaussian-Laplacian distribution and proposed a least soft-thresold squares algorithm. Hu *et al.* [44] solved original $l0$ norm minimization problem for tracking. Generally, confidences of target candidates of these methods are calculated based on reconstruction errors, leading to low discriminative power. To overcome this, sparse pooling methods by summarizing sparse codes of target local patches have been well studied. Wang *et al.* [45] learned sparse codes from raw patches to obtain the final feature representation by concatenating all codes, which is termed concatenating pooling. Zhong *et al.* [46] constructed a histogram with local sparse codes for each candidate using concatenating pooling. Liu *et al.* [11] modeled the target with local sparse codes by k-selection using average pooling. Wang *et al.* [47] learned the visual prior from generic real-world images and computed the feature vectors by max pooling. Jia *et al.* [48] exploited sparse coding and alignment-pooling to extract the target feature representation containing object structural information. Ma *et al.* [49] learned sparse codes and classifiers jointly under the linearization to nonlinear learning theory. Zhuang *et al.* [50] evaluated the scores of target candidates based on a discriminative sparse similarity map by multi-task reverse sparse representation, where an additive pooling method was proposed to extract discriminative information from this map. Ma Huang and Shao [51] implemented a tracking method on tensor sparse coding where target is represented as a tensor instead of vector. These pooling methods make use of low order statistics or extreme responses of sparse codes to model the target appearance, ignoring high order statistics and correlations between responses to different dictionary elements. To establish a more generalized feature pooling framework for visual tracking, we propose to utilize a probabilistic function to model the statistical distribution of sparse codes.

III. Preliminaries

In this section, we first introduce the Fisher vector [17], [18] which is extracted for image classification by sparse codes, and a sparse coding algorithm named local coordinate coding.

A. Fisher Vector

Assume that a set of d-dimensional local sparse descriptors $\mathbf{X} = \{\mathbf{x}_i, i = 1, \cdots, p\}$ for an image region are generated by a parametric distribution $\mathcal{P}_\lambda(\mathbf{x})$ independently. In our work, GMM is chosen as the distribution to model the generation process of local features in an image. Then, $\mathcal{P}_\lambda(\mathbf{x})$ can be written as

$$\mathcal{P}_\lambda(\mathbf{x}) = \sum_{k=1}^{K} \omega_k p_k(\mathbf{x}), \tag{1}$$

where p_k is the k-th component of GMM with

$$p_k(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)}, \tag{2}$$

where $\lambda = \{\omega_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1,\cdots,K}$ denotes the parameters of GMM. To ensure a valid distribution for $\mathcal{P}_\lambda(\mathbf{x})$, each ω_k is equal to or greater than zero and the sum of all ω_ks is required to be one. Thus, the explicit constraints could be avoided by re-parameterizing ω_k as

$$\omega_k = \frac{e^{\alpha_k}}{\sum_{i=1}^{K} e^{\alpha_i}}. \tag{3}$$

Furthermore, we assume diagonal covariance matrices for each Gaussian component as δ_k. The Fisher vector is a sum of gradient statistics with respect to each parameter of GMM calculated from local descriptors, and describes the contribution of different parameters to the generative process. After derivation [18], the following gradients are obtained

$$\mathcal{G}_{\alpha_k} = \frac{1}{\sqrt{\omega_k}} \sum_{i=1}^{n} (\beta_i(k) - \omega_k), \tag{4}$$

$$\mathcal{G}_{\boldsymbol{\mu}_k} = \frac{1}{\sqrt{\omega_k}} \sum_{i=1}^{n} \beta_i(k) \left(\frac{\mathbf{x}_i - \boldsymbol{\mu}_k}{\delta_k}\right), \tag{5}$$

$$\mathcal{G}_{\delta_k} = \frac{1}{\sqrt{\omega_k}} \sum_{i=1}^{n} \beta_i(k) \frac{1}{\sqrt{2}} \left(\frac{(\mathbf{x}_i - \boldsymbol{\mu}_k)^2}{\delta_k^2} - 1\right), \tag{6}$$

where $\mathcal{G}_{\alpha_k}$, $\mathcal{G}_{\boldsymbol{\mu}_k}$ and $\mathcal{G}_{\delta_k}$ are normalized gradients of local descriptors with respect to α_k, $\boldsymbol{\mu}_k$ and δ_k respectively, and the division and exponentiation of vectors are all element-wise operations. $\beta_i(k)$ is the soft assignment of $\mathbf{x}_i$ for the k-th component of GMM with

$$\beta_i(k) = \frac{\omega_k p_k(\mathbf{x}_i)}{\sum_{j=1}^{K} \omega_j p_j(\mathbf{x}_i)}. \tag{7}$$

The Fisher vector is obtained by concatenating the gradients $\mathcal{G}_{\alpha_k}, \mathcal{G}_{\boldsymbol{\mu}_k}$, and $\mathcal{G}_{\delta_k}$. Note that the dimension of the obtained Fisher vector is $(2d + 1)K$ which does not depended on the sample size p. $\mathcal{G}_{\alpha_k}, \mathcal{G}_{\boldsymbol{\mu}_k}$, and $\mathcal{G}_{\delta_k}$ could be seen as the zero-order, first-order and second-order statistics with respect to $\mathbf{x}_i$. Because of the fact that the gradient with respect to the weight parameters brings little additional information, the Fisher vector is obtained by only concatenating the gradients $\mathcal{G}_{\boldsymbol{\mu}_k}$ and $\mathcal{G}_{\delta_k}$. Therefore, the dimension of the Fisher vector is $2dK$ in our method. In addition, the normalization is necessary for Fisher vectors and we execute the power normalization first and then the L2 normalization [17].

B. Local Coordinate Coding

Although the data of many real problems are represented in high dimensional space, they usually lie on a manifold with much smaller intrinsic dimensionality [12]. Furthermore, the characteristics of manifold make it possible to keep locality of coordinate coding, which means that a sample can be encoded by its local anchors in the manifold.

Assume that the nonlinear function $f(\mathbf{x})$ defined in high dimensional space is (β, δ, p)-Lipschitz smooth. Given an arbitrary coordinate coding (γ, D), where $D \subset \mathcal{R}^d$ is a set of anchor points and γ maps point $\mathbf{x}$ to $\boldsymbol{\alpha} \in \mathcal{R}^{|C|}$ with $\sum_{i=1}^{|D|} \alpha_i = 1$ where α_i is the i-th element of $\boldsymbol{\alpha}$. This leads to an approximation of point $\mathbf{x}$, namely, $\gamma(\mathbf{x}) = \sum_{i=1}^{|D|} \alpha_i \mathbf{d}_i$ where $\mathbf{d}_i \in D$. It can be proved that,

$$\left| f(\mathbf{x}) - \sum_{i=1}^{|D|} \alpha_i f(\mathbf{d}_i) \right| \leq \beta \left\| \mathbf{x} - \gamma(\mathbf{x}) \right\| + \delta \sum_{i=1}^{|D|} |\alpha_i| \left\| \mathbf{d}_i - \gamma(\mathbf{x}) \right\|^{1+p}.$$

It can be seen that a nonlinear classification function can be approximated by a linear function with respect to coordinate coefficients of samples. And the upper bound of the approximation error is the reconstruction error of samples and the affinity between samples and anchor points. Therefore, the function $f(\mathbf{x})$ can be approximated by minimizing the right side of the above equation.

Given samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{R}^d$ is a data point and y_i is its corresponding label, to learn the approximation nonlinear function $f(\mathbf{x})$, we first obtain a dictionary (anchor points) $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_k] \in \mathcal{R}^{d \times k}$ which will be used to encode every sample. A code $\boldsymbol{\alpha}_i$ should be able to minimize the reconstruction error and preserve locality when encoding a sample $\mathbf{x}_i$. For each sample $\mathbf{x}_i \in \mathcal{R}^d$, the LCC is defined as

$$\min_{\mathbf{D}, \boldsymbol{\alpha}_i} \frac{1}{2} ||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||^2 + \mu \sum_{j=1}^{k} |\alpha_i^j| ||\mathbf{d}_j - \mathbf{x}_i||^2, \tag{8}$$

where α_i^j is the j-th element of $\boldsymbol{\alpha}_i$ and μ a constant that keeps balance between the reconstruction error and locality.

IV. PROPOSED TRACKING APPROACH

We now introduce our tracking approach. Firstly, we show the target appearance model in Section IV-A. Then, we present the classifier learning method in Section IV-B and the tracking framework of our algorithm in Section IV-C. Finally, the updating scheme is introduced to handle appearance changes during tracking in Section IV-D.

A. The Generalized Pooling Method

The initial target is located manually or by an object detection algorithm without loss of generality. We crop n templates $T = \{T_1, T_2, \cdots, T_n\}$ with a normalized size by moving the target center within a small scope. Templates are then divided into overlapped local patches with size $s \times s$. Most sparse representation based tracking methods extract intensity features for image patches only, which ignore the related spatial location information of each. To tackle this issue, we vectorize these patches by flattening their intensities and concatenating with their relative coordinates as the original local descriptors, i.e., for each patch, it is represented by

$$\mathbf{y} = [\mathbf{I}, a, b], \tag{9}$$

where $\mathbf{I}$ is the vectorial intensity feature, and a, b are the related coordinate positions of x direction and y direction with regard to the target location respectively. In this way, local descriptors can model both image information and spatial information. For computational efficiency, we calculate the dictionary first and then code each patch respectively. K-means clustering is performed to achieve the dictionary $\mathbf{D}$ for local coordinate coding with these original local descriptors. To improve tracking performance, we establish several GMMs to estimate the Fisher vector by separating the whole object into several overlapping parts. As shown in Fig. 2, the appearance model consists of global representation and partial representation.

1) Global Pooling Representation: Denote the original local feature set for all templates as $Y = \{\mathbf{y}_i\}_{i=1}^p$, where p is the number of overlapped patches. We encode each sample $\mathbf{y}_i$ with dictionary $\mathbf{D}$ by LCC, and obtain p local codes $\{\mathbf{x}_i\}_{i=1}^p$ as the new representations for local patches. With this operation, these elements larger than zero in the local coordinate code correspond to dictionary items that lie in the neighborhood of this sample. We find that similar samples acquire similar sparse coordinate codes We assume that sparse codes corresponding to different dictionary items are statistically independent, and each obeys the Gaussian mixture distributions instead of a single Gaussian distribution. We compute the final target representation not by traditional sparse pooling methods, but by a GMM leading to a generalized sparse pooling framework. Therefore, these local codes are used to train a global GMM. Actually, we can apply direct matching between two GMMs by Kullback-Leibler divergence [52], but it is time consuming which is not suitable for visual tracking. A Fisher kernel method is introduced to extract the final the feature vector for target template with GMM. For each template, we compute its Fisher vector by substituting its local codes into Eqs. (4), (5) and (6). The calculated Fisher vector is regarded as the global representation of a template, which is shown in the upper portion of Fig. 2.

2) Partial Pooling Representation: Considering only global target representation is susceptible to local appearance variations caused by partial occlusion, out-of-plane rotation, etc. In order to establish a more discriminative appearance model, we extract b partial Fisher vectors for the target appearance to handle local changes. Firstly, we split up each template $T_i \in T$ into b overlapped blocks. We get blocks using slide widow method with a certain step size. Let T_i^j denote the j-th block of the i-th template. We collect all the local patches belonging to the j-th block for all templates in T, and fit a local GMM for the j-th block using its corresponding local codes of the collected local patches. Thus, b GMMs are obtained in total for partial appearance representation. We calculate a Fisher vector for each block in templates using

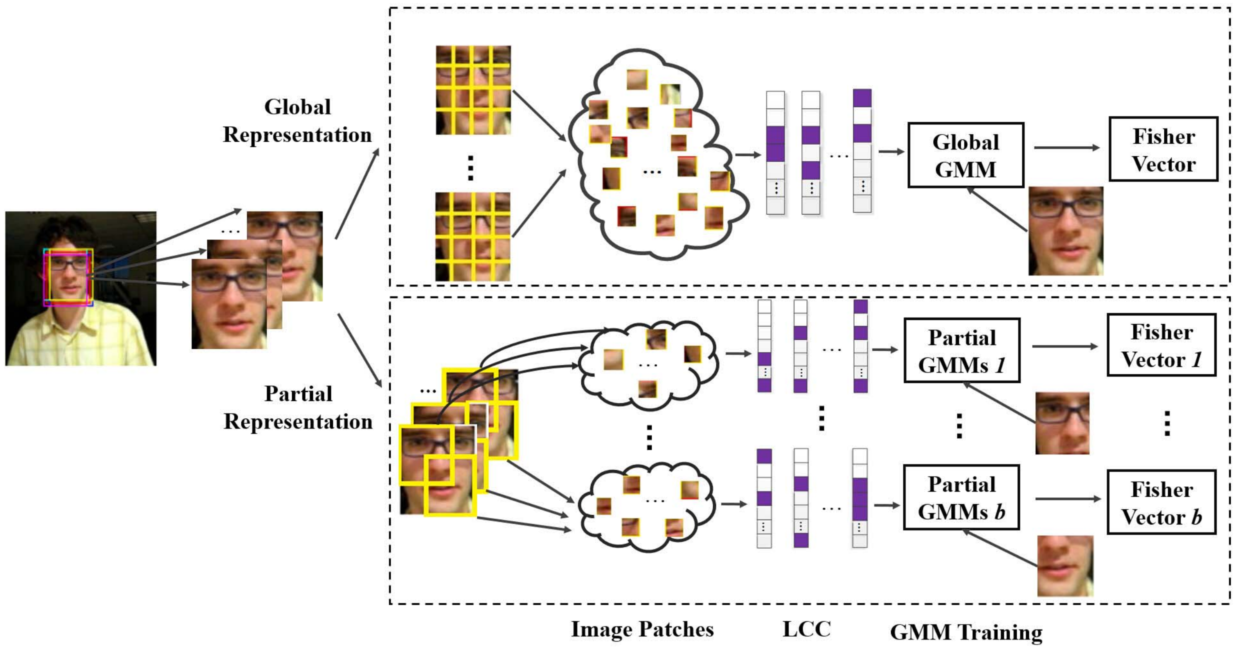


Fig. 2. Illustration of the proposed generalized pooling method. $(b+1)$ Fisher vectors which are the final vectors are obtained for each target candidate.

the above global representation. For each template, b local representations of the Fisher vector are obtained. The work flow of our appearance modeling is shown in the right-lower part of Fig. 2.

Considering both global and partial representations, we calculate $(b+1)$ Fisher vectors for each target template. Note that Fisher vectors can also characterize the high-order statistical properties of coding coefficients apart from low-order statistics compared with other sparse pooling methods. More importantly, it can also model the correlations between sparse coefficients corresponding to different dictionary items.

B. Classifier Learning

Visual tracking is treated as a binary classification problem, which separates the target region from the background. Fisher vectors can be employed as the training samples of a semi-supervised classifier to train samples for superior classification results. We train several classifiers using global and local Fisher vector representations.

Given a set of M labeled training data $\{(\mathbf{z}_i, l_i) \mid \mathbf{z}_i \in \mathcal{X}, l_i \in \{1,0\}\}_{i=1}^{M}$ and a set of N unlabeled training data $\{\mathbf{z}_i\}_{i=M+1}^{M+N}$, where the candidates in the current frame are treated as unlabeled training data. We learn the semi-supervised classifier $\mathcal{C} : \mathcal{X} \to \mathcal{R}$ by minimizing the following optimization problem,

$$\frac{1}{2}\sum_{i=1}^{M}(l_i - \mathcal{C}(\mathbf{z}_i))^2 + \lambda_1 \|\mathcal{C}\|_K^2 + \frac{\lambda_2}{2}\sum_{i,j=1}^{M+N}\left(\mathcal{C}(\mathbf{z}_i) - \mathcal{C}(\mathbf{z}_j)\right)^2 W_{ij}, \tag{10}$$

where $\|\cdot\|_K$ is the norm defined in $\mathcal{H}_K$, and $\mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) that is associated with a positive definite Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$. W is a $(M+N) \times (M+N)$ similarity matrix with entries W_{ij} indicating the adjacency weights between data points $\mathbf{z}_i$ and $\mathbf{z}_j$,

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{\sigma_i \sigma_j}\right), & i \in Q_k^j \ or \ j \in Q_k^i \\ 0, & otherwise \end{cases} \tag{11}$$

where Q_k^i is the index set of the k nearest neighbors of $\mathbf{z}_i$ in $\{\mathbf{z}_i\}_{i=1}^{M+N}$, $\sigma_i = \left\|\mathbf{z}_i - \mathbf{z}_i^{(k)}\right\|_2$, and $\mathbf{z}_i^{(k)}$ is the k-th nearest neighbor of $\mathbf{z}_i$.

The solution $\mathcal{C}(\mathbf{z})$ of Eq. (10) is an expansion of kernel functions over both labeled and unlabeled data,

$$\mathcal{C}(\mathbf{z}) = \sum_{i=1}^{M+N} \omega_i K(\mathbf{z}_i, \mathbf{z}), \tag{12}$$

and we adopt a linear kernel in the classifier. The corresponding classification score of the one classifier is calculated as $e^{-\|1-\mathcal{C}(\mathbf{z})\|^2}$.

In order to collect training data for classifiers, we cropped n' positive samples around the target center within a small scope and m' negative samples out of the target region within a larger scope randomly. Global and local Fisher vectors for each training sample and its corresponding blocks are extracted from the trained global and local GMMs in Section IV-A. In our experiments, we train a global classifier $\mathcal{C}_g$ using global Fisher vector representations and b local classifiers $\{\mathcal{C}_l^i\}_{i=1}^{b}$ with local Fisher vectors of all b blocks. Given a target candidate, we calculate its global Fisher vector f_g and b local Fisher vectors $\{f_l^1, f_l^2 \cdots, f_l^b\}$ using the trained GMMs. Then, as illustrated in Fig. 3, the final classification score of

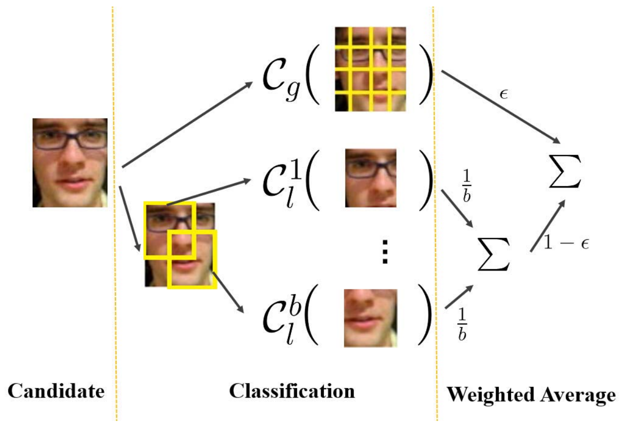


Fig. 3. A candidate is classified by a global classifier and b partial classifiers, and the final confidence score is a weighted sum of these classifiers.

a candidate is defined as

$$H(f_g, f_l) = \epsilon \mathcal{C}_g(f_g) + (1-\epsilon)\frac{1}{b}\sum_{i=1}^{b} \mathcal{C}_l^i(f_l^i), \tag{13}$$

where ϵ controls the balance between contributions of global features and local features. A sample with the highest classification score indicates that it is most likely to be the current target, which is considered as the tracking result for the current frame.

C. Our Tracking Method

Given the observation $\mathbf{o}_{1:t} = \{\mathbf{o}_i\}_{i=1}^{t}$, the motion parameter of object $\mathbf{s}_t$ is calculated by maximum a posteriori estimation (MAP)

$$arg\,\underset{\mathbf{s}_t}{max}\; p(\mathbf{s}_t \mid \mathbf{o}_{1:t}). \tag{14}$$

It can be inferred under the Bayesian theorem

$$p(\mathbf{s}_t|\mathbf{o}_{1:t}) \propto p(\mathbf{o}_t|\mathbf{s}_t)\sum_{i=1}^{t} p(\mathbf{s}_i|\mathbf{s}_{i-1})p(\mathbf{s}_{i-1}|\mathbf{o}_{1:i-1}), \tag{15}$$

where $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is the dynamic model and $p(\mathbf{o}_t|\mathbf{s}_t)$ is the likelihood model. For simplicity, the candidates are considered to be sampled from a proposal distribution $q(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{o}_{1:t}) = p(\mathbf{s}_t|\mathbf{s}_{t-1})$. In this work, the posterior $p(\mathbf{s}_t|\mathbf{o}_{1:t})$ is approximated by a set of samples $\{\mathbf{s}_t^1, \mathbf{s}_t^2, \cdots, \mathbf{s}_t^N\}$ with their corresponding weights $\{w_t^1, w_t^2, \cdots, w_t^N\}$.

An affine image warp is used to model the target motion between two consecutive frames. The state is defined as $\mathbf{s}_t = [\xi_x, \xi_y, \theta, s, \eta, \phi]$, where (ξ_x, ξ_y) is the target center coordinate in the image, and θ, s, η, ϕ are the parameters of rotation angle, scale, aspect ratio and skew, respectively. We apply a Gaussian distribution to model the dynamic model which is denoted as $p(\mathbf{s}_t|\mathbf{s}_{t-1}) = \mathcal{N}(\mathbf{s}_t; \mathbf{s}_{t-1}, \Sigma)$, where Σ is a diagonal covariance matrix. The likelihood model is constructed as

$$p(\mathbf{o}_t|\mathbf{s}_t) \propto H(f_g, f_l), \tag{16}$$

where $H(f_g, f_l)$ is the classification score function defined in Eq. (13). The pseudocode of the proposed tracking approach is summarized in Algorithm 1.

Algorithm 1 The Proposed Tracking Algorithm

```
Input: Video frames $V_1, V_2, \cdots, V_{num}$, initial target state $\mathbf{s}_1$
 1: $t = 1$
 2: while $t <= num$ do
 3:     if $t == 1$ then
 4:         Collect positive and negative templates $\{T_i\}_{i=1}^{n}$ from $V_t$.
 5:         Compute global and partial representations based on the proposed generalized pooling method introduced in Section IV-A , and obtain labeled training samples.
 6:     else
 7:         Sample candidates as unlabeled training samples.
 8:         Train a global classifier $\mathcal{C}_g$ and $b$ partial classifiers $\{\mathcal{C}_l^i\}_{i=1}^{b}$ using the corresponding labeled and unlabeled data by Eq. (12).
 9:         Compute final classification score of each candidate by Eq. (13).
10:         Determine current target state $\mathbf{s}_t$.
11:     end if
12:     Collect positive and negative templates based on current target state $\mathbf{s}_t$.
13:     Update template set $T_{i\,i=1}^{\,n}$ by Eq. (19).
14:     Recalculate dictionary $\mathbf{D}$ and GMMs.
15:     Update labeled samples.
16: end while
Output: target state $\mathbf{s}_2, \mathbf{s}_3, \cdots, \mathbf{s}_{num}$
```

D. Update Scheme

Tracking with fixed dictionary and classifiers is prone to drifting in real world scenes, because the target appearance is changing overtime due to illumination variations, global and partial occlusions, shape deformation, etc. In our work, we maintain two sample sets: one is used to train the GMMs with only target templates, and the other is collected to train the semi-supervised classifiers. We now introduce the update scheme of these two sample sets.

1) Target Templates Update: Numerous template updating algorithms have been proposed to handle appearance changes during tracking. Jia *et al.* [48] proposed to update the template set by a reconstructed image with sparse representation and incremental subspace learning. This method updates eigenbasis vectors only with the estimated tracking results. But the estimated targets may be polluted by noise or occlusions. In this paper, we filter out occlusions and outliers by reconstructing a new template. We remove outliers in tracking results $\mathbf{y}$ using a Laplacian noise component $\mathbf{s}$ calculated by minimizing

$$[\mathbf{q}, \mathbf{s}] = \arg\min_{\mathbf{q},\mathbf{s}} \frac{1}{2}\|\bar{\mathbf{y}} - \mathbf{U}\mathbf{q} - \mathbf{s}\|_2^2 + \gamma\,\|\mathbf{s}\|_1, \tag{17}$$

where $\mathbf{U}$ consists of PCA basis vectors of the target templates, and $\bar{\mathbf{y}} = \mathbf{y} - \mathbf{u}$ where $\mathbf{u}$ is the mean vector of all the target templates, and $\mathbf{q}$ is the coefficients of $\bar{\mathbf{y}}$ with respect to $\mathbf{U}$. We compute the i-th element of the reconstructed target $\tilde{y}$ as

$$\tilde{y}_i = \begin{cases} y_i, & s_i = 0 \\ u_i, & s_i \neq 0, \end{cases} \tag{18}$$

where u_i and s_i denote the i-th elements of mean vector $\mathbf{u}$ and noise term $\mathbf{s}$ respectively. The basis vector matrix $\mathbf{U}$ and mean vector $\mathbf{u}$ are updated with the processed target incrementally.

The target template set is updated with a reconstructed template

$$\bar{T} = \mathbf{U}\mathbf{q} + \mathbf{u}. \tag{19}$$

We then recalculate dictionary $\mathbf{D}$ for LCC and the GMMs to compute Fisher vectors with the new template set.

2) Training Samples Update: In order to train the semi-supervised linear kernel classifiers, we maintain a set of labeled training samples collected from pervious frames. To update the positive samples in them, we crop several new positive samples around the current tracking result and update the old positive samples with the new ones. However, the target during tracking often suffers from appearance changes due to different internal and external reasons. Therefore, we reconstruct each new sample under target templates to train GMMs, where these new templates with high reconstruction errors are treated to be occluded or polluted. That is to say, for each new target template T_t, we calculate its reconstruction error γ by

$$\gamma = \sum_{i=1}^{b} \|\mathbf{y}_t^i - \mathbf{D}_p \mathbf{x}_t^i\|_2^2, \tag{20}$$

where $\mathbf{y}_t^i$ is the feature vector of i-th block of T_t, and $\mathbf{x}_t^i$ is its corresponding sparse code and $\mathbf{D}_p$ is the dictionary learned with partial representations of target templates. If γ is larger than a constant γ_0, this new sample template is regarded as inferior ones, and we drop it. If not, it will be applied to replace the current positive template with the lowest classification confidence score by current classifiers.

During tracking, the background of the target is also changing with the target moving, and negative templates are also important to the semi-supervised classifiers. Therefore, we crop negative templates far away from the current target center randomly, and replace all these old negative templates with the new sampled ones. These semi-supervised classifiers are retrained with the training samples extracted in the current sample set as in Section IV-B.

V. Experimental Results and Discussion

Our tracker was tested on both the tracking benchmark [19] (TB) which consists of 50 video sequences and the VOT2014 dataset [53] with substantial variation. Our tracking approach runs 0.29 fps using our un-optimized Matlab implementation on an Intel Core2 3.0GHz CPU with 2GB RAM. The number n of target templates cropped around the target region within 2 pixels was set to 25. All the templates were normalized to 32×32, and the local patch size $s = 6$ with step size 3. The block size is set to 16×16, and we get blocks using the sliding window method with a certain step size 8. Thus, each template is divided into $b = 9$ blocks. We obtained $d = 30$ dictionary items by K-means for LCC. The numbers of components for global and local GMMs were set to 4 and 2 manually. We used the implementation of an open library [54] to train GMMs and compute Fisher vectors. For training samples to learn classifiers, $n' = 9$ positive samples and $m' = 200$ negative samples were collected. Under the particle filter framework, the number of particles was set to 600. We fixed all these parameters through this experiment and demonstrated the good performance of the proposed tracker.

A. Visual Comparisons

We show some typical examples of our tracking results compared with several popular tracking algorithms, and explain the advantages according to different tracking challenges.

In Fig. 4, we show the tracking result on sequences where targets are confronted with occlusions and illumination variation, fast motion and motion blur. In sequence 'david3', the target suffers from global occlusion when it walks behind a tree. TLD and DSSM fail to track the target even when it only suffers from partial occlusion because of a lamp pole. Although CSK can track the target before it walks back, it is lost at the end of the sequence. Only TGPR and our tracker are able to track the target until the end of the whole sequence successfully. For sequence 'shaking', the target experiences large illumination changes such as in frame #56. Struct drifts when the appearance varies, but VTD, SCM, TGPR and our tracker perform well. Note that VTD and TGPR drift in several frames such as in frame #366, while our tracker can keep tracking with high accuracy and less drifting. In the 'deer' sequence, the tracked deer moves abruptly and the target motion between adjacent frames is smooth. VTD, TGPR and DSSM fail to track the fast moving target, but TGPR, KCF and our method could track the target that moves rapidly successfully. CSK and KCF could sample candidates densely in a large scope using the circulant matrices theory, and this method could capture the target, although it undergoes fast motion. In the 'jump' sequence, the target undergoes motion blur in several frames along with background cluttering. The texture feature of this target is blurred and the extracted visual cues in blurred frames are very different from that in the initial frame. TGPR, Struct, TLD, KCF and our tracker have the ability to track the interesting target precisely. The proposed method also performs well benefiting from the use of a semi-supervised classifier. The semi-supervised classifier considers both labeled and unlabeled samples, and these blurred candidates can participate during training as unlabeled samples.

B. Evaluation on Tracking Benchmark (TB) [19]

The test sequences in the tracking benchmark [19] cover almost all difficulties encountered during tracking, and they are annotated with different attributes such as illumination variation, scale variation, occlusion, and deformation. We have tested our tracking algorithm on these video sequences and carried out both overall estimation and attribute-based comparisons with 29 popular trackers and four recently proposed

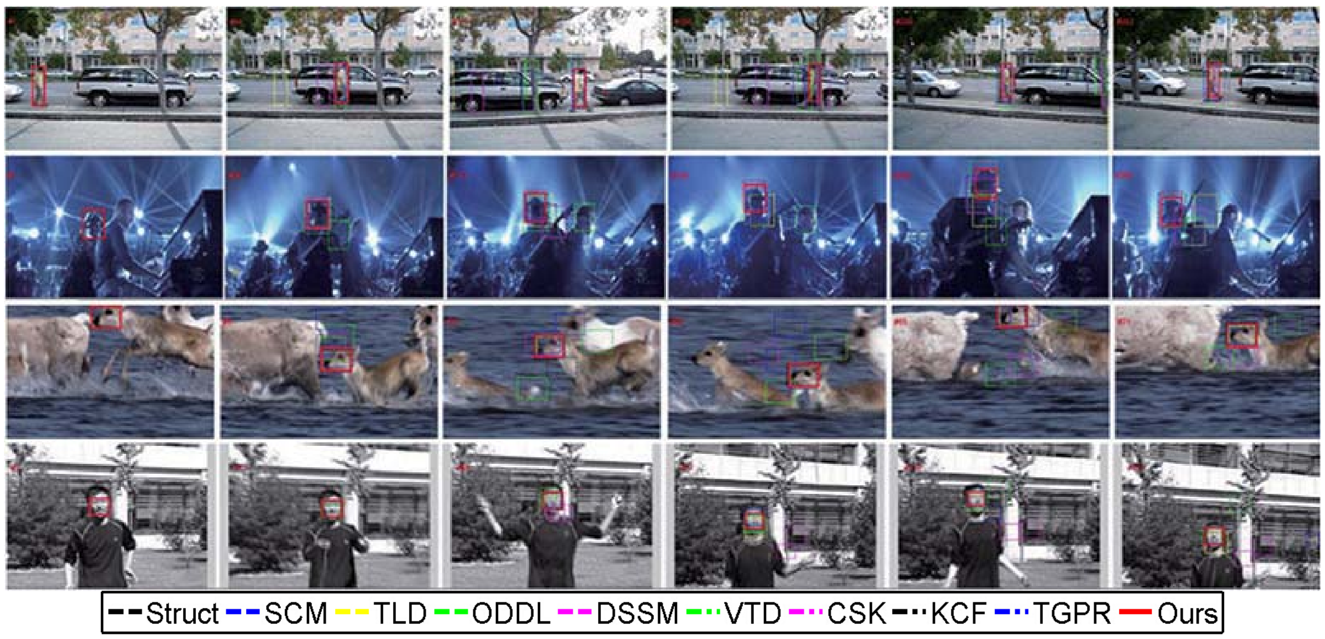


Fig. 4. Handling occlusion illumination variation, fast motion and motion blur. Tracking results over video sequences 'david3', 'shaking', 'deer' and 'jump' from top to bottom, where targets are confronted with serious occlusion, illumination variations, fast motion and motion blur, respectively.

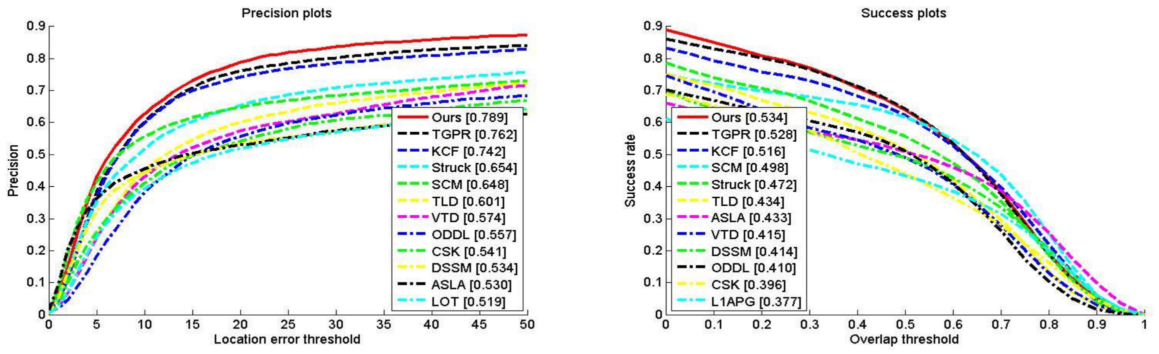


Fig. 5. Overall performance comparison of precision plot (left) and success rate (right) for these trackers.

TABLE I
THE PERFORMANCE SCORES OF TRACKERS(%)

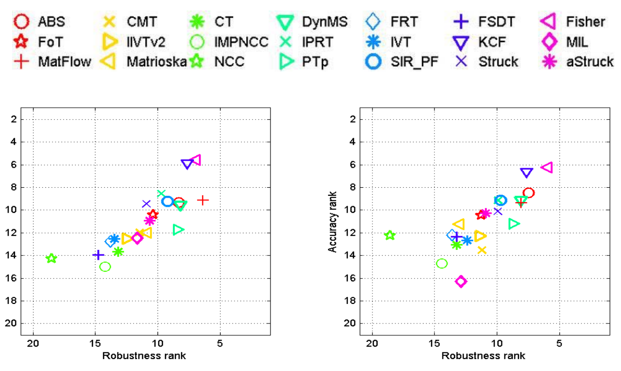
	Struck	SCM	TLD	VTD	ODDL	DSSM	CSK	ASLA	LOT	KCF	LSK	TGPR	Ours
DP	65.4	64.8	60.1	57.4	55.7	53.4	54.1	53.0	51.9	74.2	49.9	76.2	78.9
OP	47.2	49.8	43.4	41.5	41.0	41.4	39.6	43.3	36.6	52.9	39.0	52.8	53.4

trackers including KCF [36], ODDL [55], DSSM [50] and TGPR [56].

1) Evaluation Metrics: To evaluate the performance of our tracker quantitatively, the tracking results were estimated by distance precision (DP) and overlap precision (OP) [19]. DP is a measurement that presents the relative number of frames when the center location error is smaller than a threshold in a video, and OP is used to measure the percentage of frames that the overlap between ground truth and the tracking bounding box is larger than a threshold.

We show the precision plots and success rate plots in Fig. 5. We rank these trackers according to their corresponding scores. Due to the limitation of space, we only list curves of the first 10 trackers that gain the best performance. The curves of our tracker are displayed with red solid lines.

2) Overall Estimation: The overall center location error performance scores at 20 pixels are presented in the legend of the precision plots, and the overlap performance scores calculated with the areas under curves are shown in the legend of the success plots in Fig. 5. The rankings of trackers in precision plots and success plots are slightly different because of different metrics. For a clearer comparison, we list all these scores in Table I. TGPR [56], KCF [36], Struck [57], SCM [46] and TLD [25] obtain good performance, but our method obtains better performance on both precision plots (78.9%) and success plots (53.4%). TGPR gets the second best performance with DP value 76.2% and OP value 52.8%, while these values of our method are higher than it by about 2 and 1 percent on the two values respectively. It is an



(a) (b)

Fig. 6. The robustness-accuracy ranking plots of our tracker ('Fisher') and the state-of-the-art tracking methods in VOT2014. The better trackers are located at the upper right corner, and our tracker gets the best performance. (a) baseline. (b) region_noise.

impressive performance gain compared with these popular existing tracking algorithms.

C. Evaluation on VOT2014 [53]

We further test our tracker on nineteen video sequences in a more challenging benchmark VOT2014 including the two experiments mentioned in [53]. In the baseline setting, each tracker is initialized with the ground truth bounding box, and in the region_noise setting, it is initialized by a perturbed bounding box centered around the ground truth bounding box randomly. Each experimental setting is executed for 15 times. The estimation toolkit[2] reports two evaluation results including accuracy and robustness. The accuracy measures the bounding box overlap ratio with ground truth, and the robustness assesses the number of failures which indicate when the overlap measure equals zero. In order to reduce the robustness measure bias, the target is re-initialized 5 frames after tracking failure. To further reduce the bias, 10 frames after re-initialization are ignored.

Our tracker is compared with 20 trackers in VOT2014 (please refer to [53] for more details about these trackers). As shown in Fig. 6, we show the results of all trackers under the baseline experiment and the region noise experiment. The better trackers are located at the upper right corner on these two figures. In both the baseline experiment and region noise experimental settings, our tracker (denoted as 'Fisher') obtains the average best performance, whose ranking score is 5.93 and 6.44 respectively.

VI. Conclusion and Future Work

A generalized pooling framework for sparse code vectors has been proposed for visual tracking in this paper. We propose to use a probabilistic function to model sparse codes of the visual target. For the consideration of computational efficiency, we extract Fisher vectors from the data and the distribution model to get a compact and discriminative representation. We instantiate the proposed framework by designing a visual tracker that makes use of semi-supervised Fisher kernel classifiers, which shows better tracking performance than state-of-the-art algorithms. There are still some limitations in our current tracking method. First, we use GMMs with a fixed number of components to characterize sparse codes of target patches, which may not model the real probabilistic distribution of them well. Secondly, we apply Fisher vector as the final representation of a template for computational simplicity, but it may not be enough to reflect the distribution of sparse codes corresponding to different dictionary items.

[2]https://github.com/vicoslab/vot-toolkit

References

[1] J.-C. Tai, S.-T. Tseng, C.-P. Lin, and K.-T. Song, "Real-time image tracking for automatic traffic monitoring and enforcement applications," *Image Vis. Comput.*, vol. 22, no. 6, pp. 485–501, 2004.

[2] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 19–31, Feb. 1997.

[3] S. Paschalakis and M. Bober, "Real-time face detection and tracking for mobile videoconferencing," *Real-Time Imag.*, vol. 10, no. 2, pp. 81–94, Apr. 2004.

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

[5] S. Salti, A. Cavallaro, and L. Di Stefano, "Adaptive appearance modeling for video tracking: Survey and evaluation," *IEEE Trans. Image Process.*, vol. 21, no. 10, pp. 4334–4348, Oct. 2012.

[6] 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, Nov. 2011.

[7] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse coding based visual tracking: Review and experimental comparison," *Pattern Recognit.*, vol. 46, no. 7, pp. 1772–1788, Jul. 2013.

[8] 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.

[9] L. Xu, S. Zheng, and J. Jia, "Unnatural L_0 sparse representation for natural image deblurring," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1107–1114.

[10] S. Gao, I. W.-H. Tsang, L.-T. Chia, and P. Zhao, "Local features are not lonely–Laplacian sparse coding for image classification," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3555–3561.

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

[12] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 22. 2009, pp. 2223–2231.

[13] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1215–1222.

[14] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 487–493.

[15] S. I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, 1998.

[16] M. Douze, A. Ramisa, and C. Schmid, "Combining attributes and fisher vectors for efficient image retrieval," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 745–752.

[17] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 143–156.

[18] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vis.*, vol. 105, no. 3, pp. 222–245, 2013.

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

[20] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Van Den Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, p. 58, Sep. 2013.

[21] N. Jiang, W. Liu, and Y. Wu, "Learning adaptive metric for robust visual tracking," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2288–2300, Aug. 2011.

[22] N. Jiang and W. Liu, "Data-driven spatially-adaptive metric adjustment for visual tracking," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1556–1568, Apr. 2014.

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

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

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

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

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

[28] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Computer Vision–ECCV*. 2014, pp. 188–203.

[29] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5388–5396.

[30] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "Multi-store tracker (MUSTer): A cognitive psychology inspired approach to object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 749–758.

[31] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, 2008.

[32] J. Ho, K.-C. Lee, M.-H. Yang, and D. Kriegman, "Visual tracking using learned linear subspaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun./Jul. 2004, pp. 782–789.

[33] Q. Zhao, Z. Yang, and H. Tao, "Differential earth mover's distance with its applications to visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 274–287, Feb. 2010.

[34] 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 Recognit.*, Jun. 2014, pp. 1090–1097.

[35] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. IEEE Eur. Conf. Comput. Vis.*, Jun. 2012, pp. 702–715.

[36] 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.

[37] B. Ma, J. Shen, Y. Liu, H. Hu, L. Shao, and X. Li, "Visual tracking using strong classifier and structural local sparse descriptors," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1818–1828, Oct. 2015.

[38] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using SIFT features and mean shift," *Comput. Vis. Image Understand.*, vol. 113, no. 3, pp. 345–352, Mar. 2009.

[39] X. Li, A. Dick, C. Shen, Z. Zhang, A. van den Hengel, and H. Wang, "Visual tracking with spatio-temporal Dempster–Shafer information fusion," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3028–3040, Aug. 2013.

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

[41] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 1436–1443.

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

[43] D. Wang, H. Lu, and M.-H. Yang, "Robust visual tracking via least soft-threshold squares," *IEEE Trans. Circuits Syst. Video Technol.*, to be published, doi: 10.1109/TCSVT.2015.2462012, 2016.

[44] H. Hu, B. Ma, and Y. Jia, "Multi-task ℓ_0 gradient minimization for visual tracking," *Neurocomputing*, vol. 154, pp. 41–49, Apr. 2015.

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

[46] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1838–1845.

[47] 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.

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

[49] B. Ma, H. Hu, J. Shen, Y. Zhang, and F. Porikli, "Linearization to nonlinear learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Jun. 2015, pp. 4400–4407.

[50] 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.

[51] B. L. J. Ma Huang and L. Shao, "Discriminative visual tracking using tensor pooling," to be published, *IEEE Trans. Cybern.*, doi: 10.1109/TCYB. 2015.2477879, 2016.

[52] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2007, pp. 317–320.

[53] M. Kristan *et al.*, "The visual object tracking vot2014 challenge results," in *Computer Vision–ECCV Workshops*. 2014, pp. 191–217.

[54] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proc. 18th Int. Conf. Multimedia*, 2010, pp. 1469–1472.

[55] F. Yang, Z. Jiang, and L. S. Davis, "Online discriminative dictionary learning for visual tracking," in *Proc. Conf. Appl. Comput. Vis.*, 2014, pp. 854–861.

[56] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," *Computer Vision–ECCV*. 2014, pp. 188–203.

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