Image and Vision Computing 38 (2015) 24–32

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Visual tracking based on online sparse feature learning ☆

Zelun Wang [a], Jinjun Wang [b,*], Shun Zhang [b], Yihong Gong [b]

[a] Texas A&M University, College Station, TX 77843, USA
[b] Xi'an Jiaotong University, 28 Xianning West Road, Xi'an, Shaanxi, 710049, China

ARTICLE INFO

Article history:
Received 17 July 2014
Received in revised form 19 December 2014
Accepted 2 April 2015
Available online 24 April 2015

Keywords:
Visual tracking
Sparse coding
Sparse feature
Bayesian classifier
Haar-like features

ABSTRACT

Various visual tracking approaches have been proposed for robust target tracking, among which using sparse representation of the tracking target yields promising performance. Some earlier works in this line used a fixed subset of features to compress the target's appearance, which has limited modeling capacity between the target and the background, and could not accommodate their appearance change over long period of time. In this paper, we propose a visual tracking method by modeling targets with online-learned sparse features. We first extract high dimensional Haar-like features as an over-completed basis set, and then solve the feature selection problem in an efficient L_1-regularized sparse-coding process. The selected low-dimensional representation best discriminates the target from its neighboring background. Next we use a naive Bayesian classifier to select the most-likely target candidate by a binary classification process. The online feature selection process happens when there are significant appearance changes identified by a thresholding strategy. In this way, our proposed method could work for long tracking tasks. At the same time, our comprehensive experimental evaluation has shown that the proposed methods achieve excellent running speed and higher accuracy over many state-of-the-art approaches.

1. Introduction

Visual tracking is currently one of the most important research topics in the field of computer vision, especially for the application of video surveillance, vehicle navigation, and human computer interaction. In practical problems, analyzing video sequences by human labor force can be impractical due to the explosive growth of video volume. Although many tracking algorithms have been proposed, it remains a challenging problem due to factors such as occlusions, illumination changes, pose changes, view point variations, etc. One of the key issues to separate the foreground targets from the background is to propose suitable appearance models. A model with high dimensional features is effective because it can preserve adequate information of the target, but these features are often redundant and often limit the speed for processing. Several methods have been proposed to find the compressive features out of the high dimensional features as sparse representation. These compressive features are low-dimensional and can preserve most information of the targets. Several tracking methods based on sparse representation have been proposed. Zhang et al. [1] introduced in their compressive tracking method a non-adaptive random matrix to project high dimensional features to a low-dimensional space. The data-independent projection matrix can achieve high processing speed and low computational cost on one hand, but on the other hand, its performance can be unstable due to the random characteristic of the matrix. Mei et al. [2] proposed a method by casting tracking as a sparse approximation problem in a particle filter framework, in which the target is represented in the space spanned by target templates and trivial templates, and the sparsity is achieved by solving an L_1-regularized least squares problem. Jia et al. [3] introduced a structural local sparse appearance model which used sparse codes of local image patches with spatial layout in an object, and employed a template update strategy which combines incremental subspace learning and sparse representation. However, these methods require to discover basis functions from the unlabeled data and can be computationally expensive.

In this paper, we model the targets with sparse Haar-like features. At the beginning, high dimensional Haar-like features are extracted in order to preserve sufficient information of the target. Since these features might be redundant and may hinder the speed for tracking, we next introduce sparse coding into the tracking process for dimensionality reduction. Every dimension of the feature can be viewed as a basis function, thus we would only need to solve an L_1-regularized least squares problem to get the sparse coefficients [4]. The process is a ranking mechanism that evaluates the large set of Haar-like features, and all of the coefficients corresponding to the basis functions should vanish except for a few. With the sparse features, we construct a naive Bayesian classifier to evaluate the target candidates [5] selected from near the current target. Positive and negative features extracted from the neighborhood of the target are used to update the classifier online. This approach can be viewed as a combination of a generative tracker

☆ This paper has been recommended for acceptance by Ming-Hsuan Yang.

* Corresponding author. Tel./fax: +86 29 83395146.
E-mail address: jinjun@mail.xjtu.edu.cn (J. Wang).

http://dx.doi.org/10.1016/j.imavis.2015.04.005
0262-8856/

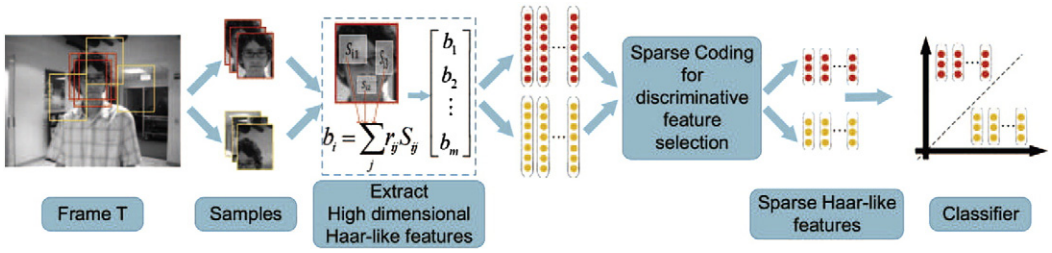


Fig. 1. Sparse feature selection.

and a discriminative tracker. Furthermore, since the appearance of the target changes through the video sequences, we also introduce an adaptive feature update scheme which compares the latest observation with previous target template, i.e., sparse coding is carried out again when target appearance changes significantly. During the tracking process, this method guarantees that the selected features are the most discriminative one. Experiments on several public datasets demonstrate that the proposed tracking method performs favorably against several state-of-the-art methods, and at the same time achieves high tracking speed.

The main contributions of this paper include:

- An online sparse feature selection method for modeling tracking target from its neighboring background,
- An automatically feature updating strategy to accommodate significant appearance changes of the target,
- More stable and accurate tracking results compared to several state-of-the-art methods, as well as real-time processing speed

The rest of the paper is organized as follows. First we review some most relevant works on target tracking in Section 2. Then we introduce the sparse feature selection process in Section 3. We elaborate the construction and updating of the naive Bayesian classifier in Section 4 and next we introduce the tracking process and the online feature selection strategy in Section 5. In Section 6, we list the evaluation results of our algorithm on 7 public dataset, and finally in Section 7, we conclude our work.

2. Related work

According to the type of the adopted appearance model, visual tracking algorithms can be categorized into generative, discriminative, or hybrid approaches. Generative trackers locate the targets using a maximum-likelihood or maximum-a-posterior formulation relying only on the target appearance model. These appearance models represent object appearance without considering its discriminative power with respect to the appearance of the background or other targets. Jepson et al. [6] introduced an appearance model that involves a mixture of stable image structure, learned over long time courses, along with 2-frame motion information and an outlier process. In [7], Matthews et al. introduced a template update method that can reduce the drifting problem by aligning with the first template to reduce drifts. Kwon et al. [8] proposed a method that decomposed the observation model and motion model into multiple basic observation models and basic motion models that are constructed by sparse principle component analysis (SPCA) of a set of templates. In [9], Ross et al. presented a tracking method that incrementally learns a low-dimensional subspace representation and adapt online to the changes in the appearance of the target.

Discriminative trackers aim to distinguish the targets from the background using a classifier that learns a decision boundary between the appearance of the target and that of the background or other targets. Avidan proposed [10] an ensemble tracking method that constantly updates a collection of weak classifiers to separate the foreground object from the background. Tang et al. [11] introduced a semi-supervised learning approach that built an online support vector machine (SVM) for each independent feature and fuses the classifiers by combining the confidence map from each classifier. Babenko et al. [12] introduced a discriminative learning paradigm called multiple instance learning (MIL) that puts all ambiguous positive and negative samples into bags to learn a discriminative model for tracking. Grabner and Bischof proposed [13] an online boosting based feature selection framework.

Hybrid trackers use a combination of the previous two approaches, in which a generative model and a discriminative classifier are combined to capture appearance changes and allow reacquisition of an object after total occlusion. Yu et al. [14] proposed a generative model using a number of low dimension linear subspaces to describe the target appearance, as well as a discriminative classifier using an online support vector machine which is trained to focus on recent appearance variations. In [15], Zhang et al. proposed a hybrid compressive tracking algorithm. The targets are represented by a multiscale convolution with rectangle filters. Then they employed non-adaptive random projections over filtered images using a very sparse measurement matrix, and then used the projected features to formulate the tracking task as a binary classification via a naive Bayesian classifier. They also introduced a coarse-to-fine target search algorithm, which reduces the computational complexity. In [16], Zhong et al. developed a sparsity-based discriminative classifier (SDC) and a sparsity-based generative model (SGM) that exploited both holistic templates and local representations. Notice that Zhong's objective function for SDC is very similar to ours. However, the entire workflows are significantly different. In [16], the SDC learns a sparse classification model while in our work, Eq. (3) is only used for feature selection while a more robust Bayesian classifier

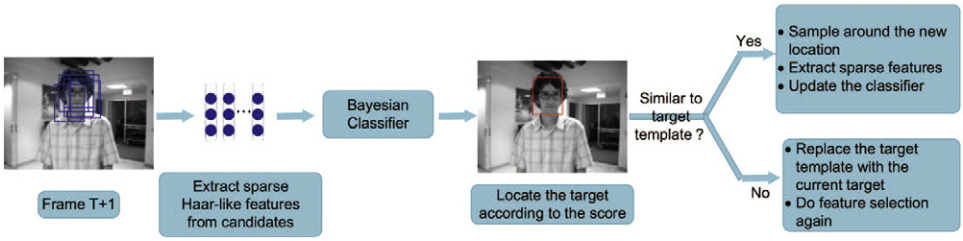


Fig. 2. Target search and feature updating.

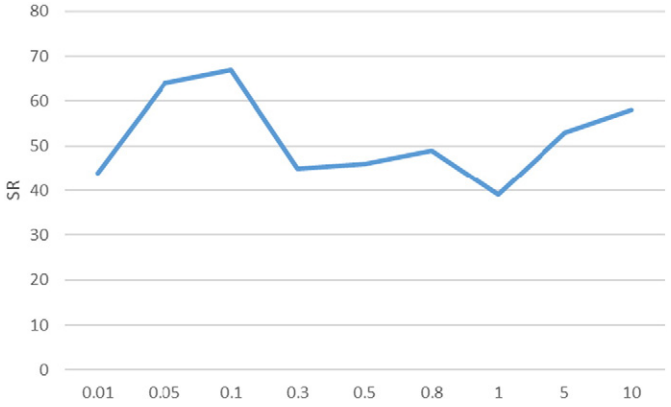


Fig. 3. The success rate (SR) changes according to γ.

is used for recognizing the foreground. In fact, our system is implicitly both generative and discriminative in that, the Bayesian classifier is discriminative while the feature selection process is generative.

Sparse representation of targets has received more and more attention. In [17], Zhang et al. formulated object tracking in a particle filter framework as a multi-task sparse learning problem, and particles are modeled as linear combinations of dictionary templates. Liu et al. [18] proposed a method that is based on L1 trackers. It also uses a sparse approximation over a template set, and adds an l_2 norm regularization on the coefficients associated with the trivial templates. Liu et al. proposed a local sparse appearance model [19], which models the target with a static sparse dictionary and a dynamically online updated basis distribution. A dictionary learning algorithm called K-Selection is also introduced. In [20], Wang et al. introduced a generative tracking algorithm which adopts l_1 regularization into the principal component analysis (PCA) reconstruction, and represents an object by sparse prototypes that explicitly take occlusion and motion blur into account for appearance updates. Furthermore, Wang et al. [21] introduced a generative tracking algorithm based on linear regression, which models the error term with the Gaussian–Laplacian distribution. They also introduced an update scheme to capture the appearance change of targets. Mei et al. [22] proposed a bounded particle resampling-L1 tracker which employs a two-stage sample probability scheme. The more comprehensive surveys and evaluation about recent tracking algorithms can be found in [23–25].

3. Sparse feature representation

The framework of the feature selection process is illustrated in Fig. 1.

First, we initialize the position and scale of the target manually or by a detector at the first frame of a video sequence, and represent the target

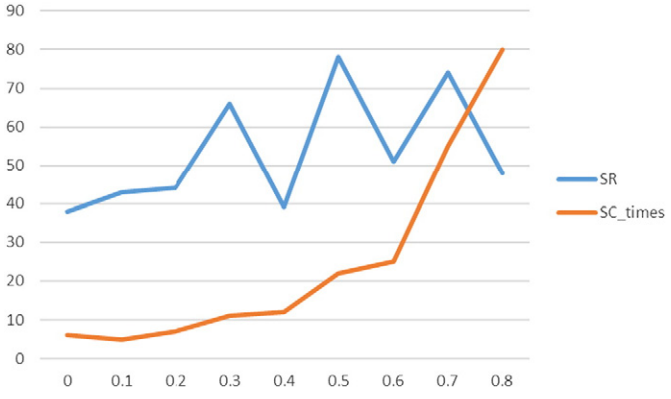


Fig. 4. The success rate (SR) and the times for sparse coding (SC_times) change according to r_0.

Table 1
Success rate (%), the higher the better. Bold font indicates the best performance.

Video clip	OSF	FCT	CT	MIL	OAB	semiB	Frag	l_1-track	TLD	Struck
David indoor	100	98	89	68	31	46	8	41	98	98
Girl	97	31	78	50	71	50	68	90	57	**99**
Twinnings	**98**	98	89	72	98	23	69	83	46	98
Occluded face	**100**	99	89	97	49	41	54	96	87	97
Tiger1	**93**	52	78	39	24	28	19	13	65	73
Tiger2	**95**	72	60	45	37	17	13	12	41	22
Cliffbar	**100**	99	89	65	23	65	22	38	67	70
Sylvester	**100**	77	75	80	70	68	34	46	94	87

with $z_0 \in \mathbb{R}^{w \times h}$, where w and h represent the width and height of the target, and the location of z_0 with $\mathbf{l}(z_0)$. z_0 is saved as the initial target template. We then model the target with high-dimensional features. In order to do this, a bunch of training samples are automatically extracted from the current frame. We first extract a set of samples from a small neighborhood around the current target as a positive bag: $D^{\alpha} = \{z | \|\mathbf{l}(z) - \mathbf{l}_T\| < a\}$ (red bounding boxes in Fig. 1), and then extract a set of samples far away from the target center as the negative bag: $D^{\zeta,\beta} = \{z | \zeta < \|\mathbf{l}(z) - \mathbf{l}_T\| < \beta\}$ with $\alpha < \zeta < \beta$ (yellow bounding boxes in Fig. 1).

Then we extract high dimensional Haar-like features, denoted as $\vec{B}$, from these samples to learn the appearance model, where every dimension of the Haar-like feature $b_i \in \vec{B}$ is selected randomly at the first time. From each of these samples, we extract a high dimensional Haar-like feature vector $\vec{b}_i \in \mathbb{R}^m$, and a corresponding label $y_i \in \{-1, 1\}$ ($+1$ corresponds to a positive sample and -1 corresponds to a negative sample). The extracted features can be denoted as a matrix $\vec{B} = [b_1, \cdots, b_p]^T \in \mathbb{R}^{p \times m}$, in which m is the dimension of the features and p is the number of samples. The corresponding label vector can be denoted as $\vec{Y} \in \mathbb{R}^{p \times 1}$. Each element $b_i \in \vec{B}$ is a weighted linear combination of 2 to 4 spatially distributed rectangle features at different scales:

$$b_i = \sum_j r_{ij} S_{ij} \tag{1}$$

where $j \in \{2, 3, 4\}$, $r_{ij} \in \mathbb{R}$ is a random number between $[-1, 1]$, and S_{ij} is the sum of pixels to a random rectangle. S_{ij} can be calculated efficiently by the integral image trick introduced in [26].

The high dimension feature can preserve adequate appearance information of the target. However, dealing with high dimension features requires high computational cost. In fact the features are always redundant and compressible. Thus, we adopt the sparse coding algorithm to help reducing the dimension and select only the most discriminative features. Assuming the use of L_1 penalty as the sparsity function, this problem can be formulated as an L_1-regularized least squares problem. Specifically, the high dimensional features $\vec{B}$ are used as known bases and $\vec{Y}$ as the input vector. Each element $y_i \in \vec{Y}$ is succinctly represented

Table 2
Center location error (in pixels), the lower the better. Bold font indicates the best performance.

Video Clip	OSF	FCT	CT	MIL	OAB	semiB	Frag	l_1-track	TLD	Struck
David indoor	**7**	11	16	19	57	37	73	42	12	9
Girl	16	40	21	25	23	50	26	13	–	**10**
Twinnings	9	10	9	14	**7**	70	15	10	15	7
Occluded face	**12**	12	19	17	36	39	57	17	24	15
Tiger1	**6**	23	10	27	42	39	39	48	24	12
Tiger2	**6**	10	13	18	22	29	37	57	40	22
Cliffbar	**5**	6	7	14	33	56	34	35	70	20
Sylvester	**6**	9	9	10	12	14	47	42	7	9

Table 3
Speed evaluation. *FPS* refers to frame per second. *SC_Times* refers to the number of times of doing sparse coding in a video sequence.

Video clip	FPS	SC_Times	Frame number
David indoor	20	11	462
Girl	24	8	502
Twinnings	28	4	472
Occluded face	27	14	888
Tiger1	24	24	354
Tiger2	13	26	365
Cliffbar	15	40	472
Sylvester	28	48	1345
Average	22.4	27 frames → 1 sparse coding	

using basis vector $\vec{b}_1, \cdots, \vec{b}_p$, and a sparse vector of weights or "coefficients" $\vec{S} \in \mathbb{R}^m$ such that

$$y_i \approx \sum_{j=1}^{m} b_j^{(i)} s_j, \tag{2}$$

where $s_j \in \vec{S}$ and $b_j^{(i)} \in \vec{b}_i$. With such an assumption, we can model the problem as the following convex optimization problem:

$$minimize_{\vec{s}} \quad \frac{1}{2}\left\|\vec{Y} - \vec{B}\vec{S}\right\|^2 + \gamma\left\|\vec{S}\right\|. \tag{3}$$

Eq. (3) can be solved efficiently by the feature-sign search algorithm proposed in [4].

The solution vector $\vec{S}$ contains sparse coefficients, which enables itself to be used as a classifier. However, it may fail when there exist similar objects or occlusions in the scene, because it is unable to utilize the information from the former frames. An incremental naive Bayesian classifier, fortunately, can properly handle this problem, as elaborated in Section 4. Notice that each column in $\vec{B}$ denotes the same Haar-like features (extracted in the same way but from different samples), and corresponds to one item in $\vec{S}$. The columns that correspond to the non-zero items in $\vec{S}$ are the most discriminative features. We thus delete the columns in $\vec{B}$ where the corresponding item in $\vec{S}$ is zero. We denote the remained features as $\vec{V}\left(\vec{S}\right) \in \mathbb{R}^{p \times n}$, where n is the dimension of the sparse features. Although the dimension is low, these features are rather salient and can almost reconstruct the original features.

4. Bayesian classifier

The sparse feature matrix $\vec{V}\left(\vec{S}\right) = \left[\vec{v}_1, \cdots, \vec{v}_p\right]^T$ is used for classifier construction and updating. We assume that every element in $\vec{v}_i \in \mathbb{R}^n$ is independently distributed and is Gaussian, so we can model them with a naive Bayesian classifier,

$$H\left(\vec{V}\right) = log\left(\frac{\Pi_{i=1}^{n} p(v_i|y=1)p(y=1)}{\Pi_{i=1}^{n} p(v_i|y=-1)p(y=-1)}\right) = \sum_{i=1}^{n} log\left(\frac{p(v_i|y=1)}{p(v_i|y=-1)}\right), \tag{4}$$

where we assume uniform prior, i.e., $p(y=1) = p(y=-1)$, and $y \in \{1, -1\}$ is the sample label. Since we assume that every element is Gaussian, the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=-1)$ can be denoted by four parameters $\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0$,

$$p(v_i|y=1) \sim \mathbf{N}(\mu_i^1, \sigma_i^1), p(v_i|y=-1) \sim \mathbf{N}(\mu_i^0, \sigma_i^0), \tag{5}$$

where μ_i^1 (μ_i^0) and σ_i^1 (σ_i^0) are mean and standard deviation of the positive (negative) bag, respectively. The scalar parameter in Eq. (5) is incrementally updated by

$$\begin{aligned} \mu_i^1 &\leftarrow \lambda\mu_i^1 + (1-\lambda)\mu^1 \\ \sigma_i^1 &\leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2}, \end{aligned} \tag{6}$$

where $\lambda > 0$ is a learning parameter, $\sigma^1 = \sqrt{\frac{1}{p}\sum_{k=0|y=1}^{p-1}(v_i(k) - \mu^1)^2}$ and $\mu^1 = \frac{1}{p}\sum_{k=0|y=1}^{p-1} v_i(k)$. Parameters μ_i^0 and σ_i^0 are updated with similar rules. Since we assume the variables to be independent, the n-dimensional multivariate problem is reduced to the n univariate estimation problem, and thus requires fewer tracking samples to obtain accurate estimation than estimating the covariance matrix in the multivariate estimation. Also, since we use a scheme of positive and negative bags, the distribution parameters can be updated more robustly.

5. Tracking and feature updating

Fig. 2 shows the framework of our tracking process and feature updating strategy. Since the motion of a target is always continuous in a video sequence, the position of the target in frame $T+1$ is always close to the position in frame T. We thus adopt a window search strategy that extracts a set of target candidates Z from $D^\delta = \{z | \|\mathbf{l}(z) - \mathbf{l}_T\| < \delta\}$ in frame $T+1$, where δ is the search radius. We extract directly the sparse features $v_i \in \mathbb{R}^n$ from each of these candidates, and evaluate them with the Bayesian classifier respectively. The tracking is thus treated as a binary classification problem, i.e., the candidate with the highest score

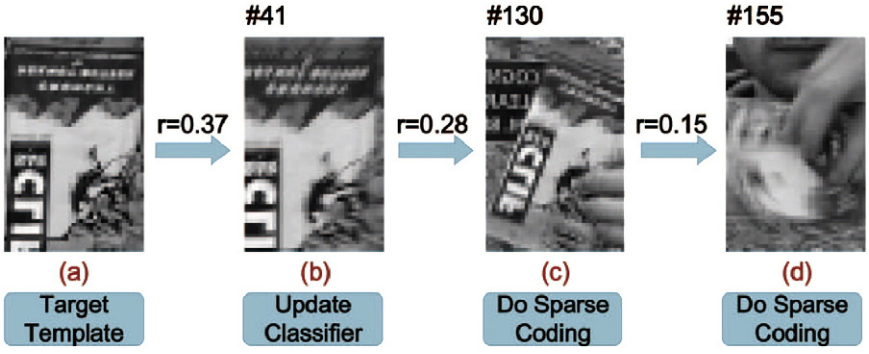


Fig. 5. Feature updating strategy. (a), (b), (c), and (d) are the same target in different frames. The correlations between (a) and (b), (a) and (c), and (c) and (d) are 0.37, 0.28, and 0.15 respectively. Sparse coding is only needed at frame 130 and frame 155, when the correlation is below 0.3.

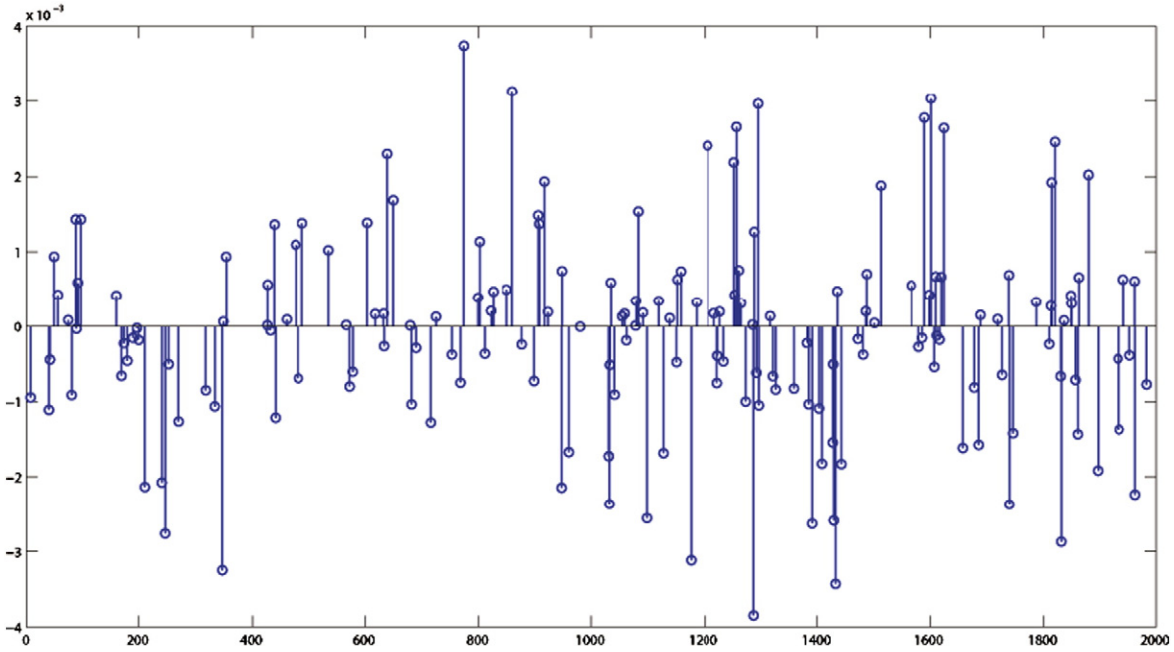


Fig. 6. Feature response to sparse coding. Two thousand features are displayed on the horizontal axis. The vertical axis shows the corresponding response values. A total of 175 out of the 2000 features have non-zero weights.

will be separated from the background as the foreground target in the frame $T+1$, denoted as $z_1 \in Z$.

At this point, we adopt an adaptive updating strategy which determines whether to update the sparse features or not. We would use the correlation between the current target z_1 and the target template z_0 as a measurement of similarity:

$$r = \frac{z_1}{\|z_1\|} * \frac{z_0}{\|z_0\|}. \tag{7}$$

Higher correlation r indicates higher similarity, and vice versa. The correlation value may vary to image densities. To deal with this, we normalize the target and template before computing their correlation. In this way, the correlation value can give us a coherent measurement of similarity. If r is higher than a threshold r_0, i.e., z_1 and z_0 are similar enough, it would not be necessary to update the sparse features. We would only need to extract positive and negative bags around the target location $\mathbf{l}(z_1)$ and extract the sparse features $\vec{V}\left(\vec{S}\right)$ to update the parameters of the classifier. However, if r is lower than the threshold, we need to do the sparse feature selection process again. Specifically, we should extract positive and negative bags around $\mathbf{l}(z_1)$ and extract high dimensional Haar-like features $\vec{B}$ from them. Then we should carry out the sparse coding algorithm again, gain a new sparse

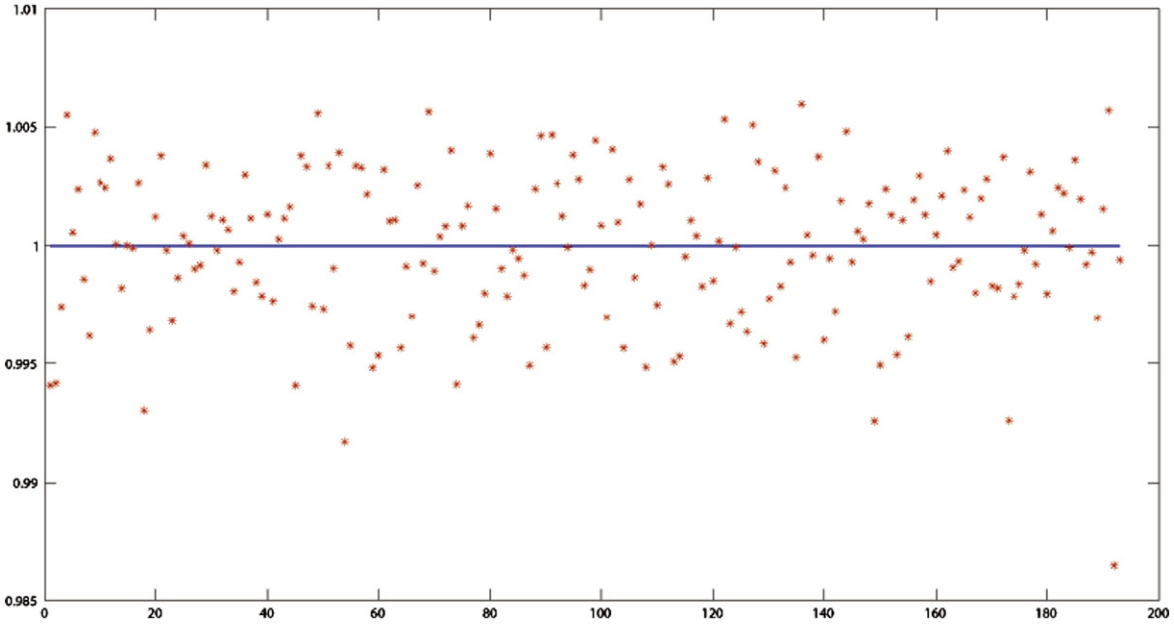


Fig. 7. Reconstructed label values. There are 195 red points in the figure. Each point represents a positive label value reconstructed by the salient features. The values are all very close to "+1". For precise observation, the figure only shows the positive part. The reconstructed negative label values, similarly, are all very close to "−1".

coefficients vector $\vec{S}_{new}$, and extract a new set of sparse features $\vec{V}\left(\vec{S}_{new}\right)$, which is the most salient in the current frame. The old classifier should be discarded and a new classifier should be initialized based on the new sparse features $\vec{V}\left(\vec{S}_{new}\right)$. Also, the target template should be replaced with the current target ($z_1 \rightarrow z_0$).

Notice that since the parameters of the Bayesian classifier are updated continuously at a learning rate of λ, the information from the former frames is properly utilized. However, when the correlation r is low and the sparse features are replaced with new ones, we would need to retrain the parameters for the new classifier. In order to utilize the former information, we keep a feature window which contains some of the positive and negative high-dimensional Haar-like features from several former frames, and use them to retrain the new classifier whenever sparse coding is carried out.

6. Experiment results

In this section, we perform experiments with our proposed method (OSF) on 8 challenging public datasets: David indoor, Girl, Twinnings, Occluded face, Tiger1, Tiger2, Cliffbar, and Sylvester. These sequences cover most challenging situations in object tracking: heavy occlusion, motion blur, in-plane and out-of-plane rotation, large illumination change, scale variation and complex background. We compare our tracking algorithm against 9 state-of-the-art methods: FCT [15], CT [1], MIL [12], OAB [27], semiB [28], Frag [29], l_1-track [30], TLD [31], and Struck [32]. The results from these methods are already reported in [1] and [15]. Each tracking task has been initialized by manually marking the target object in the first frame. Tracking has been applied to sequences consisting of 4717 frames. Some visual results of the 8 datasets are displayed in Fig. 3. All experiments are performed with a MATLAB implementation on a common PC with an Intel Core i7, 3.40 GHz CPU and 16 GB RAM, where we achieve 22.4 fps tracking speed on average.

The following parameters are fixed throughout our experiment and are presented as follows. The dimension of the high dimensional Haar-like features $m = 2000$, the threshold for the correlation $r_0 = 0.3$, and γ in Eq. (3) is set to 0.1. The learning rate λ is a critical parameter and is typically set to 0.85, but is adjusted in the experiment for different datasets. For example, if the appearance of targets changes fast, a smaller λ is needed.

We did some investigation into the effect of the two parameters γ and r_0. These experiments are performed on the David indoor dataset. From Fig. 3, we find that when $\gamma = 0.1$, the success rate is high, and the selected features are sparse at this point. Fig. 4 shows that a higher r_0 leads to a higher success rate. This is because more feature updating is performed, which demonstrate the effectiveness of our feature updating strategy. However, this requires more sparse coding and can significantly lower down the speed. We set r_0 to 0.3, which improves the performance and remains high processing speed at the same time.

6.1. Quantitative comparison

We evaluate our algorithm and 9 other approaches with two evaluation metrics: center location error and success rate [33]. Success rate is defined as, $score = \frac{area(ROI_T \cap ROI_G)}{area(ROI_T \cup ROI_G)}$, where ROI_T is the bounding box of tracking and ROI_G is the bounding box of ground truth. A tracking result is considered success only when $score > 0.5$. Center location error (CLE) is defined as the Euclidean distance between the central locations of the bounding box of tracking and the bounding box of ground truth. Tables 1 and 2 show the comparison results of success rate and center location error respectively.

Table 1 shows that our approach has achieved 100% success rate on David indoor, Occluded face, Cliffbar and Sylvester. None of the other 9 approaches have achieved 100% accuracy on these sequences. Also, the success rate of our approach on Girl, Twinnings, Tiger1 and Tiger2 is all above 90%. Table 2 shows that the CLE of our approach is the best on David indoor, Occluded face, Tiger1, Tiger2, Cliffbar and Sylvester, and is the third best on Girl and Twinnings. It is observed that the performance of the proposed method is overall superior to the other 9 state-of-the-art methods.

6.2. Discussion

6.2.1. Tracking speed

An evaluation of tracking speed of our approach is listed in Table 3. We achieve an average speed of 22.4 fps (frame per second), and sparse coding is carried out every 27 frames on average. The speed varies between the 8 video clips because of different target sizes and different rates of appearance change. If the appearance changes drastically, e.g., the target in Fig. 8(b), more sparse coding process is required during tracking. Compared with the simple classifier updating process, the sparse coding process requires more computational costs. However, these costs are alleviated by our adaptive feature updating strategy, which is demonstrated in Fig. 5. In frame 41, e.g., the target is similar to the target template (the correlation $r = 0.37$), thus we would only need to update the parameters in the classifier. This process lasts until frame 130, when the correlation is below 0.3 ($r = 0.28$). At this time, we do the sparse feature selection process again, train a new classifier, and replace the template with current target. It does not last long before another sparse coding process is required in frame 155 (when $r = 0.15$), because this is a period when the target undergoes a rotation and the appearance changes very fast. This shows that with the help of sparse coding, we can utilize the most salient features and successfully track the target against the drastic appearance change.

6.2.2. Effectiveness of sparse features

Sparse coding provides us with a method to find succinct representations of the original high-dimensional features. We demonstrate the effectiveness of this method by analyzing our implementation on the David indoor dataset. The dimension of the original features is 2000. At the first frame, there are 195 positive samples and 33 negative samples. The label vector $\vec{Y} \in \mathbb{R}^{228\times 1}$ (contains 195 "+1" and 33 "−1") is represented approximately as a weighted linear combination of a small number of "salient features", refer to Eq. (2). The weight vector $\vec{S}$ is solved by the sparse coding process. Most items in $\vec{S}$ are zero, which indicates that the corresponding feature has no response. Otherwise, the corresponding features have its response values and are considered salient. This is shown in Fig. 6. The horizontal axis shows that there are 2000 features, but only 175 of them have response, and the response values vary.

We only retain the features that have response to build the sparse features. Fig. 7 demonstrates that the selected sparse features are salient enough and can almost reconstruct the original features. Ideally, all values of red points in Fig. 7 should be exactly "+1". However, we discard most of the original features, which would certainly lead to reconstruct errors. The average reconstruct error is 0.0023, which is

Fig. 8. Tracking examples on the 8 data sets. Red bounding boxes denote our tracking results.(a) Tracking results on sequence David indoor with illumination change, size change and appearance change.(b) Tracking results on sequence Girl with rotations, pose change and heavy occlusions.(c) Tracking results on sequence Cliffbar with complex background, motion blur and rotations.(d) Tracking results on sequence Tiger1 with heavy occlusions and pose change.(e) Tracking results on sequence Tiger2 with fast motion and pose change.(f) Tracking results on sequence Occluded face with heavy occlusions.(g) Tracking results on sequence Sylvester with drastic illumination change and pose change.(h) Tracking results on sequence Twinnings with rotations and size change.

(a) Tracking results on sequence David indoor with illumination change, size change and appearance change

(b) Tracking results on sequence Girl with rotations, pose change and heavy occlusions

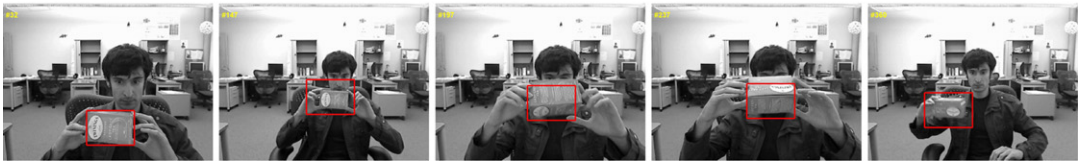
(c) Tracking results on sequence Cliffbar with complex background, motion blur and rotations

(d) Tracking results on sequence Tiger1 with heavy occlusions and pose change

(e) Tracking results on sequence Tiger2 with fast motion and pose change

(f) Tracking results on sequence Occluded face with heavy occlusions

(g) Tracking results on sequence Sylvester with drastic illumination change and pose change



(h) Tracking results on sequence Twinnings with rotations and size change

low enough for us to represent the high-dimensional features with the sparse features.

6.2.3. Robustness

6.2.3.1. Occlusion. Occlusion is one of the primary problems in object tracking. Several of our tested video clips contain heavy occluded situations. For example, in Fig. 8(b), a man's face appears in front of the woman's face for several frames around frame 465. In Fig. 8(f), a woman use a book to block her face frequently. These heavy occlusions often lead to drifting problems. In our approach, however, we can detect the significant appearance change when heavy occlusion occurs, and sparse coding is carried out to update the sparse features and adapt to the occlusion situations.

6.2.3.2. Motion blur. Fast motion of target often leads to blurred target appearance which is difficult to deal with in object tracking. The book on the man's left hand in Fig. 8(c) moves so fast that the characters on the book are blurred and unable to recognize. Many trackers fail in this situation because they are unable to distinguish the target from the background, especially in this sequence with such a complex background. The proposed method can handle this situation well by selecting the most discriminative features so that our classifier can better separate the target from the background.

6.2.3.3. Rotation. Rotation is a very challenging situation because the appearance of the target can totally change during the process. In Fig. 8(b), the target is originally the girl's face. However, as she turns around, we can only see her hair and her face is totally unobservable. In Fig. 8(h), the man rotates the box frequently, while different sides of the box are not similar at all. Our tracker is still able to track the target correctly on such situations due to two facts: 1) our algorithm would replace the old features with new ones when appearance changes drastically, 2) negative samples are used to train the classifier, which helps prevent drifting to the background.

6.2.3.4. Complex background. The sequence *Cliffbar* is challenging not only because of the target rotation, but because of the complexity of the background. From Fig. 8(c) we can see that the background is very similar to the target. Many trackers fail because they may consider background pixels as foreground object through straightforward update schemes. Our tracker tracks the target accurately because the selected sparse features are salient enough to discriminate the foreground target from the background.

6.2.3.5. Other challenging situations. Besides the challenges mentioned above, there also lie some other challenging problems in these datasets, such as pose change, illumination change, and size change. For example, pose change almost occurs in every datasets. Illumination change also challenges the robustness of trackers. In Fig. 8(a), David walks from a dark room to another room with a lamp. In Fig. 8(g), the Sylvester toy moves right under a lamp, which causes severe illumination change. Last but not least, the size of target changes constantly due to the distance change between the target and the camera. Our tracker can successfully track the targets throughout these sequences as it can extract salient sparse features and update the classifier online.

7. Conclusions

In this paper, we propose and demonstrate an efficient and robust tracking method based on online-learned sparse features. High-dimensional Haar-like features are extracted from the target, and are then reduced to low-dimensional discriminative features by sparse coding. An adaptive feature updating strategy is also introduced to control the rate for sparse coding. Finally, the target search is formulated as a binary classification via a naive Bayesian classifier. Experiment results on several challenging video clips demonstrate the effectiveness of our tracker.

Our future work will focus on the color information and scale problem. If necessary, we will also introduce more effective classifiers.

Acknowledgments

This work is supported by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2014AA015205, and the National Science Foundation of China under Grant No. 61332018.

References

[1] K. Zhang, L. Zhang, M.-H. Yang, Real-time compressive tracking, Computer Vision—ECCV 2012, Springer 2012, pp. 864–877.
[2] X. Mei, H. Ling, Robust visual tracking using l1 minimization, Computer Vision, 2009 IEEE 12th International Conference on, IEEE 2009, pp. 1436–1443.
[3] X. Jia, H. Lu, M.-H. Yang, Visual tracking via adaptive structural local sparse appearance model, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012 IEEE.
[4] H. Lee, A. Battle, R. Raina, A.Y. Ng, Efficient sparse coding algorithms, Adv. Neural Inf. Process. Syst. 19 (2007) 801.
[5] A.Y. Ng, M.I. Jordan, On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes, Adv. Neural Inf. Process. Syst. 2 (2002) 841–848.
[6] A.D. Jepson, D.J. Fleet, T.F. El-Maraghi, Robust online appearance models for visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1296–1311.
[7] I. Matthews, T. Ishikawa, S. Baker, et al., The template update problem, IEEE Trans. Pattern Anal. Mach. Intell. 26 (6) (2004) 810–815.
[8] J. Kwon, K.M. Lee, Visual tracking decomposition, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE 2010, pp. 1269–1276.
[9] D.A. Ross, J. Lim, R.-S. Lin, M.-H. Yang, Incremental learning for robust visual tracking, Int. J. Comput. Vis. 77 (1-3) (2008) 125–141.
[10] S. Avidan, Ensemble tracking, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 261–271.
[11] F. Tang, S. Brennan, Q. Zhao, H. Tao, Co-tracking using semi-supervised support vector machines, Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, IEEE 2007, pp. 1–8.
[12] B. Babenko, M.-H. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1619–1632.
[13] H. Grabner, H. Bischof, On-line boosting and vision, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, Vol. 1 2006, pp. 260–267 IEEE,.
[14] Q. Yu, T.B. Dinh, G. Medioni, Online tracking and reacquisition using co-trained generative and discriminative trackers, Computer Vision—ECCV 2008, Springer 2008, pp. 678–691.
[15] K. Zhang, L. Zhang, M.-H. Yang, Fast compressive tracking, IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 1 2014, p. 1, http://dx.doi.org/10.1109/TPAMI.2014.2315808.
[16] W. Zhong, H. Lu, M.-H. Yang, Robust object tracking via sparsity-based collaborative model, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on 2012, pp. 1838–1845 (EEE).
[17] T. Zhang, B. Ghanem, S. Liu, N. Ahuja, Robust visual tracking via multi-task sparse learning, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE 2012, pp. 2042–2049.
[18] C. Bao, Y. Wu, H. Ling, H. Ji, Real time robust l1 tracker using accelerated proximal gradient approach, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE 2012, pp. 1830–1837.
[19] B. Liu, J. Huang, L. Yang, C. Kulikowsk, Robust tracking using local sparse appearance model and k-selection, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE 2011, pp. 1313–1320.
[20] D. Wang, H. Lu, M.-H. Yang, Online object tracking with sparse prototypes, Image Process. IEEE Trans. 22 (1) (2013) 314–325.
[21] D. Wang, H. Lu, M.-H. Yang, Least soft-threshold squares tracking, Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE 2013, pp. 2371–2378.
[22] X. Mei, H. Ling, Y. Wu, E.P. Blasch, L. Bai, Efficient minimum error bounded particle resampling l1 tracker with occlusion detection, Image Process. IEEE Trans. 22 (7) (2013) 2661–2675.
[23] S. Salti, A. Cavallaro, L. Di Stefano, Adaptive appearance modeling for video tracking: survey and evaluation, Image Process. IEEE Trans. 21 (10) (2012) 4334–4348.
[24] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. (CSUR) 38 (4) (2006) 13.
[25] S. Zhang, H. Yao, X. Sun, X. Lu, Sparse coding based visual tracking: review and experimental comparison, Pattern Recogn. 46 (7) (2013) 1772–1788.
[26] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, Computer Vision and Pattern Recognition, 2001. CVPR 2001, Proceedings of the 2001 IEEE Computer Society Conference, Vol. 1, 2001 (IEEE).
[27] H. Grabner, M. Grabner, H. Bischof, Real-time tracking via on-line boosting, BMVC, Vol. 1 2006, p. 6.
[28] H. Grabner, C. Leistner, H. Bischof, Semi-supervised on-line boosting for robust tracking, Computer Vision—ECCV 2008, Springer 2008, pp. 234–247.

[29] A. Adam, E. Rivlin, I. Shimshoni, Robust fragments-based tracking using the integral histogram, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, Vol. 1 2006, pp. 798–805 (IEEE).
[30] X. Mei, H. Ling, Robust visual tracking and vehicle classification via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (11) (2011) 2259–2272.
[31] Z. Kalal, J. Matas, K. Mikolajczyk, Pn learning: bootstrapping binary classifiers by structural constraints, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE 2010, pp. 49–56.
[32] S. Hare, A. Saffari, P.H. Torr, Struck: structured output tracking with kernels, Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE 2011, pp. 263–270.
[33] M. Everingham, L. Gool, C. Williams, A. Zisserman, Pascal visual object classes challenge results, Available from www. pascal-network. org.