Pattern Recognition 46 (2013) 1772–1788

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Sparse coding based visual tracking: Review and experimental comparison

Shengping Zhang, Hongxun Yao *, Xin Sun, Xiusheng Lu

School of Computer Science and Technology, Harbin Institute of Technology, China

ARTICLE INFO

Available online 29 October 2012

Keywords:
Visual tracking
Appearance modeling
Sparse coding
Sparse representation
Dictionary learning

ABSTRACT

Recently, sparse coding has been successfully applied in visual tracking. The goal of this paper is to review the state-of-the-art tracking methods based on sparse coding. We first analyze the benefits of using sparse coding in visual tracking and then categorize these methods into appearance modeling based on sparse coding (AMSC) and target searching based on sparse representation (TSSR) as well as their combination. For each categorization, we introduce the basic framework and subsequent improvements with emphasis on their advantages and disadvantages. Finally, we conduct extensive experiments to compare the representative methods on a total of 20 test sequences. The experimental results indicate that: (1) AMSC methods significantly outperform TSSR methods. (2) For AMSC methods, both discriminative dictionary and spatial order reserved pooling operators are important for achieving high tracking accuracy. (3) For TSSR methods, the widely used identity pixel basis will degrade the performance when the target or candidate images are not aligned well or severe occlusion occurs. (4) For TSSR methods, ℓ_1 norm minimization is not necessary. In contrast, ℓ_2 norm minimization can obtain comparable performance but with lower computational cost. The open questions and future research topics are also discussed.

1. Introduction

Visual tracking is the process that continuously infers the state of a target from an image sequence. Usually, it is formulated as a search problem that aims at finding the candidate most matching to the target template as the tracking result. A typical tracking process contains several stages as shown in Fig. 1. A target template is maintained over time and may be updated online once the tracking result is available. Before starting tracking at the current time, a set of candidates are sampled around the state of the target at the last time. Both the target template and candidates are represented using an appearance model. Then, a target searching strategy is used to find the candidate most matching to the template appearance as the tracking result.

Although visual tracking has been studied for more than 30 years, it is still a very overwhelming research topic due to some unsolved challenging issues arising from both the appearance modeling and target searching. From the point of view of appearance modeling, discriminating the target from the background is a very basic ability and plays a key role in complex scenes where the contrast between the target and background is low. To achieve reliable tracking performance, it is also very important to handle target appearance variations during tracking, which contain both the intrinsic variations such as pose changes and shape deformation and extrinsic variations such as illumination and occlusion. To handle these variations, a good appearance model is desired to meet two requirements: adaptivity that adapts to the intrinsic appearance variations and robustness that is invariant to the extrinsic appearance variations. From the point of view of target searching, computation complexity is an very important issue since the real-time tracking speed is a practical requirement of most subsequent high-level applications such as action recognition and retrieval. In addition, it is also possible to handle appearance variations in target searching stage, which is ignored by most existing methods.

In the literature, a number of tracking algorithms have been proposed (for example [1–6]; see [7,8] for detailed reviews). The methods most related to those discussed in this paper are learning based methods. Jepson et al. [9] proposed a framework to learn an adaptive appearance model, which adapts to the changing appearance over time. In Collins et al. [10], an online feature selection method was proposed to select features that are able to discriminate the target from the background. Since the feature selection is online when new observations are available, the selected features adapt to environment changes very well. In Ross et al. [4], a tracking method was proposed to incrementally learn a low-dimensional subspace representation, which efficiently adapts to target appearance variations. Discriminative methods that formulate tracking as a classification problem have also been attracting much attention. For example, Grabner and

* Corresponding author.
E-mail address: h.yao@hit.edu.cn (H. Yao).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.10.006

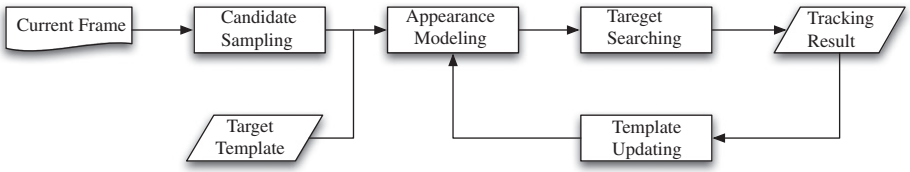

Fig. 1. A typical tracking flowchart. This paper mainly focuses on the use of sparse coding in appearance modeling and target searching.

Bischof [11] proposed an online boosting method to update discriminative features to distinguish the target from the background. However, the update process may introduce errors due to inaccurate tracking results, which can finally lead to tracking failure (drifting). To overcome the drifting problem, Grabner and Leistner [12] further proposed a semi-supervised boosting method to combine decisions of a given prior and an on-line classifier. In Babenko et al. [13], multiple instance learning (MIL) was used to treat ambiguous positive and negative samples into bags to learn a discriminative classifier which can further overcome the drifting problem. Kuo et al. [14] proposed an AdaBoost based algorithm to learn a discriminative appearance model for multi-target tracking, which allows the model to adapt to target appearance variations over time. The common property of these complex appearance models is that they try to achieve both the discriminative ability and robustness in a single appearance representation. However, it is difficult to find a good tradeoff between the discriminative ability and robustness. Usually, the good robustness is achieved while the discriminative ability is lost in some extent.

Recently, motivated by the popularity of compressive sensing in signal processing [15,16], an elegant and working model, named *sparse coding* [17], has been attracting much attention in computer vision. Very recently, inspired by the success of sparse representation in face recognition [18], some researchers also tried to use sparse representation in visual tracking and reported state-of-the-art performance [19]. On the other hand, motivated by the biologically inspired object representation model [20,21] proposed to use the responses of sparse coding to model target appearance for visual tracking. For the past several years, increasing attention has been paid along these directions and some improvements have also been proposed to further enhance the tracking performance.

Although a variety of tracking methods based on sparse representation or sparse coding have been proposed, there is no work reviewing these methods and answering several important questions: (1) What is the connection and difference between these methods? In this work, we classify these methods according to which stage (appearance modeling or target searching) sparse coding is used in. Particularly, we emphasize the difference between sparse representation and sparse coding. Sparse representation, in fact, a sub-process of sparse coding, can be used to perform target searching (TSSR), which is the motivation of the pioneering work [19]. On the other hand, sparse coding learns local representations of image patches, which can be used to model target appearance (AMSC). Classifying different tracking methods into TSSR and AMSC as well as their combination facilitates the understanding of the connection and difference among these methods. (2) Why sparse coding would be useful for visual tracking? Although a huge number of tracking methods based on sparse coding has been proposed, there is no work trying to analyze the rationales behind these methods. In this work, we try to answer this question by analyzing the roles of sparse representation from the point of view of signal processing and the roles of sparse coding from the point of view of biologically inspired representation mechanism of simple cells in visual cortex. (3) Does sparse coding really benefit visual tracking? Although previous publications reported state-of-the-art tracking performance, the limited number of test sequences and comparison methods as well as different implementation frameworks restrict the fair comparison to reveal the benefits of using sparse coding in visual tracking. In this work, we collected a total of 11 trackers based on sparse coding and four widely used baseline trackers to perform a comprehensive experimental comparison on a total of 20 test sequences. The comparison results indicate: (1) AMSC methods significantly outperform TSSR methods. (2) For AMSC methods, both discriminative dictionary and spatial order reserved pooling operators are important for achieving high tracking accuracy. (3) For TSSR methods, the widely used identity pixel basis will degrade the performance when the target or candidate images are not aligned well or severe occlusion occurs. (4) For TSSR methods, ℓ_1 norm minimization is not necessary. In contrast, ℓ_2 norm minimization can obtain comparable performance but with lower computational cost.

The rest of the paper is organized as follows. In Section 2, we give brief introduction to sparse coding and its roles in visual tracking. In Section 3, we review the tracking methods based on sparse coding in the literature. We conduct experiment comparison and analysis in Section 4. Finally, conclusion and future work are summarized in Section 5.

2. Sparse coding and visual tracking

2.1. Overview of sparse coding

Let $\mathbf{x}\in\mathbb{R}^D$ be an vector obtained by stacking all pixel intensities of an image into a column vector. Sparse coding represents $\mathbf{x}$ as a linear combination of a set of basis functions $\mathbf{V}=[\mathbf{v}_1,\ldots,\mathbf{v}_K]\in\mathbb{R}^{D\times K}$

$$\mathbf{x}=\sum_{k=1}^{K}u_k\mathbf{v}_k+\mathbf{n} \tag{1}$$

where u_k is the coefficient of the kth basis function and $\mathbf{n}\in\mathbb{R}^D$ is the noise. Basis function set $\mathbf{V}$ is also called as dictionary and each basis function is called as an atom. Let $\mathbf{u}=[u_1,\ldots,u_K]^T\in\mathbb{R}^K$ be the coefficient vector. In general, there are many solutions of $\mathbf{u}$ that satisfy Eq. (1) when the dictionary $\mathbf{V}$ is overcomplete where $D<K$. In order to compute a reasonable solution, a sparse constrain is usually placed on $\mathbf{u}$ as suggested in Field [22]. In this case, $\mathbf{u}$ can be solved by the following ℓ_0-norm minimization

$$\mathbf{u}=\arg\min_{\mathbf{u}}\|\mathbf{u}\|_0 \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{V}\mathbf{u}\|_2^2\le\varepsilon, \tag{2}$$

where $\|\cdot\|_0$ denotes the ℓ_0-norm that counts the number of nonzero entries in a vector, $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector, and ε is the noise level. This optimization problem is NP-hard and there is no known procedure that can find the sparsest solution more efficiently than exhausting all possible $\mathbf{u}$. As the closest convex function to the ℓ_0-norm minimization, the ℓ_1-norm minimization is widely used to replace the ℓ_0-norm minimization

$$\mathbf{u}=\arg\min_{\mathbf{u}}\|\mathbf{u}\|_1 \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{V}\mathbf{u}\|_2^2\le\varepsilon \tag{3}$$

It was shown that the ℓ_0-norm and ℓ_1-norm minimizations are equivalent if the coefficients are sufficiently sparse [23].

Using the Lagrangian method, the ℓ_1-norm minimization problem can be rewritten as

$$\mathbf{u} = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{x}-\mathbf{V}\mathbf{u}\|_2^2 + \lambda\|\mathbf{u}\|_1 \quad (4)$$

where λ is the Lagrangian multiplier, which controls the relative importance of the sparseness to the reconstruction error.

The linear system defined by Eq. (1) and its solution Eq. (4) are called *sparse representation*, which was widely used in computer vision, especially in face recognition. In the literature, there are two slightly different methods to cast Eq. (4) as a quadratic programming problem, namely, gradient projection sparse representation (GPSR) [24] and truncated Newton interior-point method (TNIPM) [25]. The interested readers can refer to related papers for the algorithm details.

In addition to compute coefficients of sparse representation, the other important issue in sparse coding is *dictionary learning* which learns a dictionary $\mathbf{V}$ from a collection of natural images $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$ such that any image $\mathbf{x}_i$ can be sparsely represented by the learned dictionary. Let $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N] \in \mathbb{R}^{K\times N}$ be the matrix consisting of coefficient vectors of all training images. The dictionary $\mathbf{V}$ can be learned by solving the following optimization problem:

$$\min_{\mathbf{U},\mathbf{V}} \sum_{i=1}^{N} \|\mathbf{x}_i-\mathbf{V}\mathbf{u}_i\|_2^2 + \lambda\|\mathbf{u}_i\|_1 \quad (5)$$

After being assigned initial values, the dictionary $\mathbf{V}$ can be learned by alternating two phases: (1) given the dictionary $\mathbf{V}$, the coefficients $\mathbf{U}$ are computed using Eq. (4) for all training images and (2) given the coefficients $\mathbf{U}$, the dictionary $\mathbf{V}$ is updated using gradient descent. There are many efficient dictionary learning methods such as MOD [26] and K-SVD [27].

From the introduction of sparse coding above, we can see that sparse coding, in fact, contains two processes: sparse representation and dictionary learning as shown in the left of Fig. 2. To clearly understand the difference between sparse coding and sparse representation, in this work, we emphasize that sparse representation focuses on representing an input signal (maybe not natural image) using a given dictionary with sparsity constrain on the representation coefficients. However, sparse coding focuses on learning a dictionary from natural images to represent the underling structure primitives in images. After representing an image using the learned dictionary, the representation coefficients can be used as features to describe the appearance of the image, which is inspired by the properties of the receptive fields of simple cells in visual cortex.

2.2. Motivations of using sparse coding in visual tracking

In this section, we introduce the motivations of using sparse coding in visual tracking. The relationship between sparse coding and visual tracking is shown in Fig. 2.

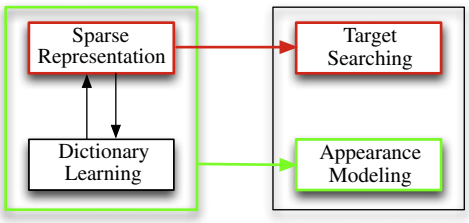

Fig. 2. Relationship between sparse coding and visual tracking. Sparse representation can be used to perform target searching. Sparse coding including both sparse representation and dictionary learning can be used to model the target appearance for visual tracking.

2.2.1. Local appearance representation

The sparse coding model was initially proposed to model natural image statistics. Given the dictionary learned from a collection of natural image patches, different image patches can be represented by the dictionary with different coefficients. Therefore, the representation coefficients can be used as features to describe the appearance of image patches. We call this ability of sparse coding as local appearance representation (LAR). In this section, we explain why LAR is useful to model target appearance from the point of view of biologically inspired object representation.

Motivated by object recognition models in cortex [28,29], Riesenhuber and Poggio proposed an biologically inspired representation model called HMAX [20] which starts with a grayscale image layer and alternates between "S" and "C" layers. The "S" layer uses Gaussians derivative filters to compute higher-order features by combining different types of units in the previous layers, which simulates the receptive fields of simple cells in visual cortex. The "C" layer achieves invariance by pooling units of the same type in the previous layer over local ranges. The original HMAX model attracted much attention in object recognition. Several modified models [30,31] have also been proposed to improve the recognition performance. The difference between the original HMAX model and its modified versions mainly focuses on the filters used in the "S" layer. Although the responses of the Gaussians derivative filters used in the original HMAX model have similar properties with the simple cells in striate cortex, it is still non-biological because it neglects the response saturation of V1 cells. The modified HMAX models use Gabor filters [32] to replace the Gaussians derivative filters because Gabor filters have been extensively used to model the receptive fields of simple cells. On the other hand, Gabor filters have more parameters and allow more accurate tuning than the Gaussians derivative filters. However, Gabor filters still have their disadvantages. First, it is difficult and time-consuming to set a large number of parameters to effectively simulate the response properties of receptive fields of simple cells in visual cortex. Second, Gabor filters are hand-designed with fixed formulation for any image datasets. However, the structure primitives underling in different image datasets are different. Therefore, it is not reasonable to use the hand-designed Gabor filters to compute the responses of images from different datasets.

In contrast to Gabor filters with fixed formulation, sparse coding basis functions are directly learned from natural images and are capable of adapting to different image datasets. In addition, there is only a few parameters needed to be tuned and the learned basis functions captured the structure primitives underling in the image datasets. Most importantly, the responses of sparse coding are localized, oriented and bandpass, which are more similar with the receptive fields of simple cells in visual cortex than Gabor filters [33]. Therefore, sparse coding is more suitable for biologically inspired appearance modeling for visual tasks. Although sparse coding was used in object recognition [34], it was only used to replace the *K*-means procedure in the bag of words models [35], which maps the low dimensional SIFT descriptors [36] into high dimensional space to increase the discriminative ability. There is a little work that used the responses of sparse coding on natural images as features to model image appearance. Motivated by the successes of hand-designed Gabor filters in biologically inspired object representation and sparse coding's advantages compared with Gabor filters, it is possible to believe that the responses of sparse coding are more suitable to model target appearance for visual tracking.

2.2.2. Minimal subspace searching

Assume that there are a total of M subspaces and the mth subspace is spanned by N_m D-dimensional vectors $\mathbf{X}_m = [\mathbf{x}_{m,1}, \ldots, \mathbf{x}_{m,N_m}] \in \mathbb{R}^{D\times N_m}$. If a signal $\mathbf{y} \in \mathbb{R}^D$ belongs to the

mth subspace, it should be approximately represented by $\mathbf{X}_m$ as

$$\mathbf{y} \approx \mathbf{X}_m \mathbf{u}_m \tag{6}$$

where $\mathbf{u}_m = [u_{m,1}, \ldots, u_{m,N_m}]^T$ is the coefficient vector.

If we do not know which subspace the signal $\mathbf{y}$ belongs to, we can linearly represent $\mathbf{y}$ using all subspaces

$$\mathbf{y} \approx \mathbf{X}_1\mathbf{u}_1 + \mathbf{X}_2\mathbf{u}_2 + \cdots \mathbf{X}_M\mathbf{u}_M = \mathbf{X}\mathbf{u} \tag{7}$$

where $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_M]$ and $\mathbf{u}^T = [\mathbf{u}_1^T, \ldots, \mathbf{u}_M^T]$. The coefficient vector $\mathbf{u}$ is sparse as long as $\mathbf{y}$ belongs to one of the M subspaces. Therefore, we can compute $\mathbf{u}$ using the ℓ_1-norm minimization (Eq. (4)). The index of the subspace that $\mathbf{y}$ belongs to can be determined by searching which subspace can reconstruct $\mathbf{y}$ with minimal error

$$\arg\min_m \|\mathbf{y} - \mathbf{X}_m\mathbf{u}_m\|_2^2 \tag{8}$$

Therefore, sparse representation can be used to find the minimal subspace the input signal belongs to from all subspaces. We call this ability as *minimal subspace searching* (MSS). In the context of face recognition, when the mth subspace is spanned by the training face images from the mth class and the input signal is a test face image, MSS can be used to recognize which class the test face image belongs to Wright et al. [18].

The MSS ability of sparse representation can be easily simplified to perform target searching for visual tracking. For example, considering two subspaces, one consists of target template images. The other consists of background images. Given a set of target candidates, the aim of visual tracking is to find which candidate is the target. This can be solved by recognizing each candidate as the target or background. If one candidate is recognized as the target, it can be treated as the tracking result. This idea motivates the pioneering work [19]. In addition to casting visual tracking as such a supervised classification problem, the other manner is to search the target template from all target candidates. It is easy to modify the MSS to perform such a target search task. For example, the target template can be linearly represented by all target candidates. A reasonable assumption is that at least one candidate is similar enough with the target. Therefore, the representation coefficients are sparse and can be computed using Eq. (4). The coefficients can be directly used to measure the similarities between the target template and candidates. The candidate with the largest coefficient can be chosen as the tracking result.

One of the most challenging issues in visual tracking is how to handle target appearance variations during tracking. From the analysis above, the MSS ability of sparse representation can be used to solve this problem. For example, appearance variations can be treated as the representation error $\mathbf{n}$ in Eq. (1), which can also be represented by a dictionary $\mathbf{O}$ and the corresponding coefficients $\mathbf{z}$

$$\mathbf{y} = \mathbf{X}\mathbf{u} + \mathbf{n} = \mathbf{X}\mathbf{u} + \mathbf{O}\mathbf{z} = [\mathbf{X}, \mathbf{O}]\begin{bmatrix}\mathbf{u}\\ \mathbf{z}\end{bmatrix} = \tilde{\mathbf{X}}\tilde{\mathbf{u}} \tag{9}$$

When the suitable dictionary $\mathbf{O}$ is chosen, the representation coefficient vector $\tilde{\mathbf{u}}$ is also sparse and can be computed using Eq. (4). Then the target searching can be finished as like in the case without appearance variations discussed previously.

3. Visual tracking based on sparse coding

According to the motivations of using sparse coding in visual tracking and the stages of a general tracking system as shown in Fig. 1, visual tracking methods based on sparse coding can be roughly classified into three classes: (1) appearance modeling based on sparse coding (AMSC), (2) target searching based on sparse representation (TSSR) and (3) combination of both AMSC and TSSR. In this section, we first introduce the basic tracking framework in each class and then review some improvement work to overcome their disadvantages.

3.1. Appearance modeling bases on sparse coding

3.1.1. Milestone work

Zhang et al. [21] proposed an appearance model based on the response distribution of basis functions learned using independent component analysis (ICA) [37]. It should be noted that ICA is a special case of sparse coding and also has similar properties with receptive fields of simple cells in visual cortex. In particular, after collecting a set of 8×8 color image patches randomly sampled from a training image set, a set of 192 basis functions and the corresponding filters were learned using ICA. The learned basis functions are called "general basis functions" because the training images are not from the tracking sequence but from any natural images. To make the general basis functions adapt to the tracking task, a basis selection strategy based on entropy gain was further used to select those basis functions which code the dominant features in the target. The rational behind this strategy is that the selected dominant features are more robust to local appearance variations. Finally, the appearance model of the tracked target was represented by the response distribution of the selected basis functions. The response ratio of the kth basis function can be computed as

$$z_k = \frac{1}{C}\sum_{i=1}^{N} u_{ik}^2 \tag{10}$$

where u_{ik} is the response of the ith patch to the kth basis function and $C = \sum_{k=1}^{K} u_{ik}^2$ is the normalization constant to make response energies of one patch to all basis functions sum to one.

The appearance model of the target was represented by the distribution $\mathbf{z} = [z_1, \ldots, z_K]$, which can be analogous to the well-known color histogram [38]. The difference is that color histogram is obtained by computing the activation frequencies of the hand-designed one-dimensional basis functions when coding each single pixel. However, the response distribution computes the activation frequencies of the learned basis functions, which is used to code each image patch. The reported experiment results in Zhang et al. [21] show that the proposed appearance model is robust to appearance variations especially to partial occlusion due to the feature selection strategy. However, this model is not discriminative enough to distinguish the target from the background because the learned basis functions are too general to code the difference between the target and background patches.

Motivated by this work, we can abstract a basic framework of the appearance model based on sparse coding as shown in Fig. 3, which contains three main layers: image layer, coding layer and pooling layer. The image layer provides input to the coding layer, which can be a gray image or color image or a set of feature descriptors such as SIFT [36] extracted from the color or gray image. The coding layer uses sparse coding (or its special cases, such as, ICA or K-means) to compute the coefficients of representing each image patch using the learned dictionary. The pooling layer computes the statistics of all coefficients to obtain a final feature representation of the input image. In the literature, several related work was proposed, which are consistent with this framework. In the following subsection, we will introduce some recent developments based on this framework from two aspects: dictionary learning methods in the coding layer and pooling operators in the pooling layer.

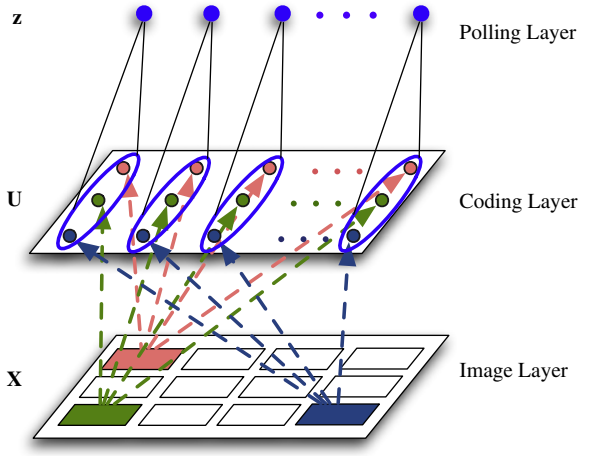

Fig. 3. A basic framework of appearance modeling based on sparse coding. The image layer **x** is sampled to get a collection of patches. Each patch is sparsely coded to produce a set of codes, which are further pooled in the pooling layer to form the final feature vector **z** to describe the appearance of the input image.

3.1.2. Recent developments

Dictionary learning: the goal of dictionary learning is to learn a dictionary with which the coding coefficients of the target or background patches can be used as features to distinguish the target from the background well. The difference between different dictionary learning methods in visual tracking is mainly the collection procedure of the training data. Wang et al. [39] learned the dictionary from a set of SIFT descriptors [36] extracted from the VOC2010[1] and Caltech101[2] datasets. Because these two datasets contain similar structures with the tracked target, for example, human shape and car contour, the learned dictionary can be used to sparsely represent the target patches.

To further increase the discriminative ability, a more discriminative appearance model based on sparse coding was proposed in Liu et al. [40] where the dictionary was directly learned using the patches sampled from the target image. Compared with the general basis functions used in [21,39], the learned basis functions are more specific to represent the patches sampled from the target. On the other hand, the learned dictionary will result in a more non-sparse representation for the background patches. Therefore, the representation coefficients are easier to distinguish the target from the background compared with the learned dictionary in Zhang et al. [21], Wang et al. [39]. Wang et al. [41] also used the patches sampled from the target image to learning the dictionary.

Feature pooling: in the pooling layer, the role of pooling operator is to compute the final feature vector **z** based on some statistics of the local codes $u_{i,k}$ obtained in the coding layer to model the appearance of the target. In Wang et al. [41] the authors obtained the final feature vector by concatenating all local codes, $\mathbf{z}^T=[\mathbf{u}_1^T,\ldots,\mathbf{u}_N^T]$. We called this pooling operator as *concatenating pooling*. The advantage of the concatenating pooling is that the spatial order of local codes is preserved, which significantly increases the discriminative ability of the resulting feature vector. However, there are two drawbacks: one is that it is very sensitive to image noise because each local code is an element of the concatenated feature vector. Once noise appears in the image, the resulting feature vector will be directly affected by the noise. The other one is that the dimensionality of the final feature vector is extremely high. For example, if the size of the dictionary is 64 and the number of sampled patches from the target image is 100, the dimensionality of the obtained feature vector is 6400. This will cause hight computation complexity, especially when the feature vectors are further used to learn a classifier.

In [39], a max pooling operator is used to compute the final feature vector, $z_k=\max\{|u_{1,k}|,\ldots,|u_{N,k}|\}$. This max pooling operator is motivated by the biophysical evidence in visual cortex [30]. Although it is successfully used in object recognition [34] due to its invariant properties, such as position and scale invariances, it is not suitable for visual tracking due to the lose of discriminative ability because each dimension of the resulting feature vector is the maximum coefficients of its corresponding basis function over all patches, which ignores other non-maximum coefficients.

The histogram-like feature vector computed using Eq. (10) in Zhang et al. [21] exploits an average pooling on local codes after first normalizing them using a square function. Similar to Zhang et al. [21], Liu et al. [40] also use the average pooling to compute the final feature vector except that they normalize the local codes using an absolute function. Although the average pooling operator uses all local codes to compute the final feature vector, which increases the discriminative ability compared with the max pooling, it loses the spatial order of local codes, which is similar to the disadvantage of the widely used color histogram representation.

In Jia et al. [42], a structural local sparse coding model was proposed, which uses a set of target templates. Each template image was divided into a set of overlapped local image patches. The patches sampled from all target images were used as the dictionary to sparsely represent the patches sampled from the candidate using the same way. Because each patch represents one part of the target, therefore, all local patches from the target with a fixed spatial relationship can reflect the target structure. To compute a final feature vector which captures the possible structure from the candidate image, all local codes computed from the candidate image were then pooled by an alignment pooling operator, which computes the feature element for each patch as the sum of the codes corresponding to the basis functions at the same position with the patch. With this strategy, the feature element reflects the similarity between the candidate and the target template. Although the reported experimental results are appealing, there is still large space to improve this structural sparse coding model. For example, the dictionary should also include patches from the background image to describe the structure of the background, which should be useful to discriminate the target from the background.

3.2. Target searching bases on sparse representation

3.2.1. Milestone work

In Wright et al. [18], a sparse representation based face recognition method was proposed which exploits the minimal subspace searching (MSS) ability introduced in Section 2.2.2. Motivated by this work, Mei and Ling [19] considered visual tracking as a two-subspace searching problem where the first subspace was spanned by a set of target templates and the second subspace was spanned by a set of trivial templates. The trivial templates are column vectors of an identity matrix, also called identity pixel basis. In particle filter framework, the purpose of visual tracking is to search which subspace each target candidate belongs to using Eq. (8). If the search result is the first subspace, the candidate is considered as the tracking result.

Different from searching each candidate, Zhang et al. [43] proposed to search the target template from two subspaces. The first subspace was spanned by a set of candidates similar to

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/index.html
[2] http://www.vision.caltech.edu/Image Datasets/Caltech101/

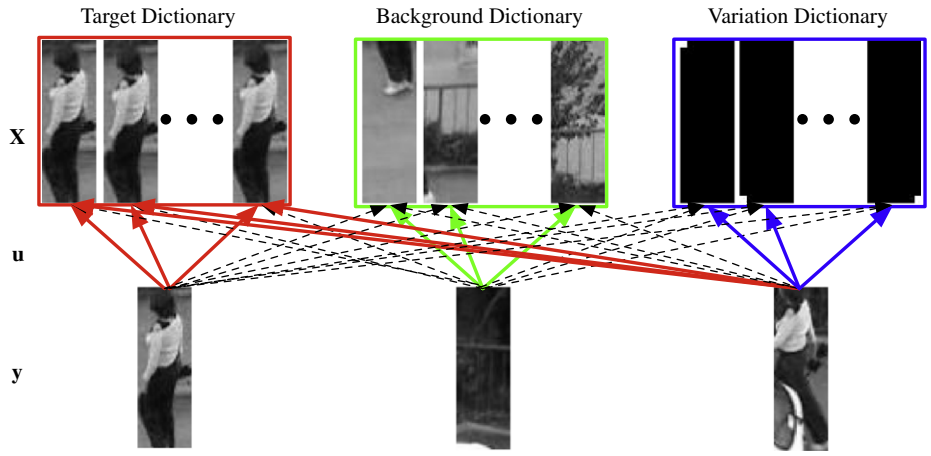

Fig. 4. Illustration of target searching based on sparse representation. When represent a candidate **y** using dictionary **X**, the red, green and blue solid lines indicate non-zeros coefficients. The black dashed lines indicate zero coefficients. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

the target template. The second subspace was spanned by a set of candidates corresponding to the background. When representing the template using all candidates from two subspaces, the candidate corresponding to the largest coefficient is considered as the tracking result. In Mei and Ling [19], the number of ℓ_1-norm minimization equals to the number of candidates, which is very large in most visual tracking applications, e.g., 600 in their experiments. Therefore, the computational cost is very expensive. In contrast, Zhang et al. [43] use all candidates to represent the target template. The number of ℓ_1-norm minimization is just one, which significantly reduces the computation complexity.

Upon these work, we can abstract a general target searching framework based on sparse representation as shown in Fig. 4. The dictionary **X** consists of three parts: target dictionary **T**, background dictionary **B** and variation dictionary **V**. Given any candidate **y**, dictionary **X** associated with coefficients **u** will produce a sparse representation of the candidate. Compared with the work in Mei and Ling [19], this framework is more general. First, the dictionary contains background images. When the candidate is from the background, the background dictionary will be used to represent it, which will result in the non-zero coefficients corresponding to the background dictionary. Therefore, the coefficients are more discriminative. In addition, the variation dictionary is more general than the identity pixel basis used in Mei and Ling [19]. In the next subsection, we will review the related work focusing on how to learn the dictionary as well as how to reduce the computation complexity.

3.2.2. Further improvement

Dictionary learning: in the tracking framework mentioned above, the dictionary is the collection of a number of vectorized images. Most work uses the downsampled gray images to obtain a low dimensional representation. However, gray images are sensitive to noises especially in outdoor scenes. On the other hand, a target is more easy to be discriminated from other targets when some sophisticated features are used. In Tzimiropoulos et al. [44], gradients along the horizontal and vertical directions were used to replace the downsampled intensity features. However, it is difficult to represent a sample as a linear combination of a subspace in the angle domain. To use angular data, each angle vector was mapped onto a new vector by concatenating its cosine and sine values.

Multiple feature descriptors are complementary for searching the target. Instead of using one kind of feature descriptor to build the dictionary, Wu et al. [45] proposed to use multiple feature descriptors in the dictionary. Each dictionary atom is obtained by concatenating multiple feature descriptors extracted from the target image. When computing the representation coefficients using Eq. (4), the sparseness constrain plays an important role to make the different descriptors compete with each other to represent the target candidate, which achieves the purpose of fusing multiple kinds of feature descriptors.

In the methods introduced above, the variation dictionary is the identity matrix, which served as an identity pixel basis to model which spatial pixels of the target are occluded. These methods [19,46] have a assumption that the occluded pixels occupy only a small part of the entire target region in order to use the sparse prior to calculate the representation coefficients. However, such methods have two main drawbacks. First, the size of the identity pixel basis is the same with the dimension of the target image, which is usually very large. Therefore, the computational cost of inferring the representation coefficients with the identity pixel basis is expensive. On the other hand, the assumption that only a small number of pixels are occluded is always not true. For example, the target could be occluded severely in many practical tracking scenes. In addition, when partial occlusion occurs, the identity pixel basis will be useful to hand occlusion. However, when there is no occlusion, the identity pixel basis would be activated to represent any image patches. In Bao et al. [47], the authors proposed to constrain the identity pixel basis to be activated when there is no occlusion. These drawbacks also exist in face recognition [18]. To overcome them, a learned occlusion dictionary based on Gabor features was proposed for face recognition [48]. Motivated by this work, Zhang et al. [49] proposed to learn the variation dictionary online for visual tracking. At each time, after the tracking result is available, the reconstruction errors between the tracking result and all candidates are collected as samples to learn the variation dictionary using an online dictionary learning method [50]. The learned dictionary is more compact and representative than the identity pixel basis.

A fixed dictionary cannot capture the appearance variation, therefore, the subspace have to been updated over time when the tracking result is available. In Mei and Ling [19], a template update scheme was used to online update the target dictionary. First, each target template is assigned a weight to specify its importance in the linear representation. The weights are assigned with equal values at the first time and then updated online based on their coefficients of representing the current tracking result.

If a template update is necessary, the template with the least weight will be replaced by the current tracking result. Updating of a template is based on the distance between the current tracking result and the template with the largest coefficient, which can be a cosine angle between two vectors. In Ling et al. [51], the authors found such an update scheme is vulnerable to drift problems when track vehicle on infrared videos where both the image quality and contrast between target and background are low. To overcome this problem, they proposed a probabilistic template update scheme where the weight update occurs only when the observation likelihood of the current tracking result is greater than a threshold.

Reducing computation complexity: to reduce the computation complexity, two kinds of methods can be adopted. One of them is to reduce the computational cost of each ℓ_1-norm minimization. The other one is to reduce the number of ℓ_1-norm minimizations. Instead of reducing the dimension of ℓ_1-norm minimization by downsampling the cropped images for both the target and candidates, Liu et al. [52] proposed a feature selection method to choose low dimensional but more discriminative features. In Wu et al. [53], to reduce the dimension of the linear representation, the authors proposed to use covariance matrix to represent the target or candidate. The covariance matrix has several significant advantages. For example, it enables efficient fusion of different types of features, where the spatial and statistical properties as well as their correlation are characterized, and its dimension is small. In Li et al. [54], a real-time tracking algorithm was proposed, which exploits the restricted isometry property (RIP) [16] in compressive sensing to reduce the dimension of the dictionary by multiplying a hashing matrix that guarantees the RIP in the two sides of Eq. (1). It should be noted that the latest CT tracker [55] also exploits RIP to reduce the dimension of the target representation and achieves real-time tracking. However, in this paper, we did not classify CT tracker into one of the three categorizations reviewed in this paper. The main reason is that all the trackers reviewed in this paper are based on a generative model, e.g., the image patch is modeled as a linear superposition of basis functions in AMSC. However, CT tracker is based on a linear transformation model, which compresses the high dimensional feature vector into a low dimensional space while preserve some kind of distance metric. From the point of view of coding, CT tracker is significantly different with the reviewed methods.

In addition to reduce the dimension of dictionary, other methods that can be used to reduce the computation complexity is to reduce the size of the dictionary, especially the size of the occlusion dictionary. Recently, a robust sparse representation model was proposed in Yang et al. [56], which removes the occlusion dictionary from the sparse representation and uses a weighted LASSO algorithm to handle occlusion. In Yan and Tong [57], the authors proposed to use this robust sparse representation to perform visual tracking. Compared with Mei and Ling [19], their method does not need to use the identity pixel basis to represent the errors caused by occlusion, which significantly reduces the size of the dictionary and therefore reduces the computation time.

In all the methods mentioned above, each target candidate is represented by the dictionary. Therefore, the number of ℓ_1 minimization equals to the number of candidates. To reduce the number of ℓ_1-norm minimization, in Mei et al. [46], the number of ℓ_1-norm minimization is reduced by exploiting the fast computational lower bound of the reconstruction error to exclude the unimportant particles. In Liu et al. [58], the authors integrated a motion model into the sparse representation. Starting from an initial state of the target, a gradient based optimization procedure is iteratively used to find the sparse representation coefficients and the corresponding gradient vector. During each iteration, a new candidate is obtained based on the calculated gradient vector and the candidate at the last time. After a small number of iterations, the local minimal value of the ℓ_1-norm minimization is obtained. Clearly, the number of ℓ_1-norm minimizations is significantly reduced.

3.3. Combining both TSSR and AMSC methods

In Sections 3.1 and 3.2, we review existing tracking methods based on AMSC and TSSR, respectively. In addition to these methods, some researchers tried to combine them together to build a sparsity based collaborative model for visual tracking [59]. The TSSR uses the target dictionary and background dictionary to represent each candidate. Let $\mathbf{u}=[\mathbf{u}_T^{\mathbf{u}_T}]$ be the coefficients of representing candidate $\mathbf{y}$ using dictionary $\mathbf{X}=[\mathbf{T},\mathbf{B}]$, the weight of the candidate is computed as $w_{TSSR}=\exp(-(\varepsilon_T-\varepsilon_B)/\delta)$ where $\varepsilon_T=\|\mathbf{y}-\mathbf{T}\mathbf{u}_T\|_2^2$ is the reconstruction error when represent the candidate using the target dictionary $\mathbf{T}$ with the associated coefficient $\mathbf{u}_T$, $\varepsilon_B=\|\mathbf{y}-\mathbf{B}\mathbf{u}_B\|_2^2$ is the reconstruction error when represent the candidate using the background dictionary $\mathbf{B}$ with the associated coefficients $\mathbf{u}_B$ and δ is a parameter. The AMSC method uses the concatenating pooling to obtain a histogram-like feature vector $\tilde{\mathbf{z}}$ for the target template and $\mathbf{z}$ for the candidate. The candidate's weight based on AMSC is computed by histogram interaction as $w_{AMSC}=\sum_{j=1}^{N\times K}\min(\tilde{\mathbf{z}}_j,\mathbf{z}_j)$. The final weight of the candidate is computed by a simple fusion $w=w_{TSSR}\times w_{AMSC}$.

Their work combines the advantages of both TSSR and AMSC and reported better tracking performance compared with other state-of-the-art methods. However, as we analyzed in the last section, the feature vector obtained by the concatenating pooling is sensitive to noise. In addition, they just adopt a simple method to fusion the weights of TSSR and AMSC. A more reasonable fusion method should be used, for example, assign higher confidence on TSSR or AMSC according to their tracking performance in previous times.

4. Experimental comparison

In this section, we conducted both quantitative and qualitative experiments on a total of twenty test sequences to evaluate the benefits of using sparse coding in visual tracking. In the following subsections, we first introduce the experimental setup including comparison methods, parameters, test sequences and evaluation criteria, and then present the performance comparison in details. All the MATLAB source codes and datasets are available at http://www.shengping.us/CSTracking.html.

4.1. Experimental setup

4.1.1. Comparison methods

To demonstrate the effectiveness of using spare coding in visual tracking, we select several representative trackers in AMSC or TSSR as well as their combination. In particularly, for TSSR, we select L1 [19], BL1 [19], T2CL1 [43] and OT2CL1 [49]. For AMSC, we select BRD [21], OLSR [41], and SLSA [42]. For the combination of both TSSR and AMSC, we select SCM [59]. To get a more fair comparison between TSSR and AMSC, we divide the SCM tracker into SCM_C and SCM_G, which respectively uses TSSR and AMSC but keep other parts the same. This allows us to focus on the benefits of using sparse coding in target searching or appearance modeling. In addition, we also select five baseline trackers, Frag [60], IVT [4], MIL [13], OAB [11] and CT [55] for comparison.

4.1.2. Parameters

We use the source codes provided by the authors to run experimental results. For particle filter based trackers L1, BL1, T2CL1, OT2CL1, OLSR, SLSA, SCM_C, SCM_G, SCM, to fairly

compare their performance, we further modified the source codes to use the same state parameters and motion model as in Mei and Ling [19]. In particularly, the state of the tracked target (particle) is represented by four deformation parameters and two translation parameters. By applying an affine transformation, each particle can be used to crop an image region and then normalized to be a fixed size (32×32 for sequences where the target is approximate square, otherwise, 16×64 or 64×16 according to the aspect ratio of the target.) The six parameters of the affine transformation are assumed to be independent. The particle motion is modeled by a Gaussian distribution with the particle in the last time as the mean and an initial covariance matrix. The covariance matrix used in the motion model is set according to the motion speed of the tracked target. We initialize the covariance matrix for each sequence and keep it unchanged for all particle filter based trackers. The number of particles is set to be 600 as like in Mei and Ling [19]. For TSSR trackers L1, BL1, T2CL1 SCM_C, the sizes of the target and background dictionaries are 50 and 200, respectively. For AMSC trackers BRD, OLSR, SLSA and SCM_G, the patches with size 6×6 are sampled by sliding across the images with step 2. λ in Eq. (4) is set to 0.01. We use the SPAM package[3] to compute the coefficients in Eq. (4) and learn the dictionary in Eq. (5). For five baseline trackers, we use the same parameters as the authors used in their papers for all the experiments. It should be noted that although we use the source codes provided by the authors, we cannot exactly reproduce the results reported in their papers because it is difficult to tune the parameters for each sequence as like those used in their experiments.

4.1.3. Test sequences

The experiments were conducted on twenty publicly available video sequences, which include *car4*, *david_outdoor*, *david_indoor*, *sylv*, *box*, *board*, *liquor*, *lemming*, *CAVIAR*, *faceocc2*, *PETS*, *seq_mb*, *face*, *woman*, *singer*, *basketball*, *bird_1*, *bird_2*, *running*, *girl*.[4] These video sequences were captured from different scenes, which contain a variety of object motion events (e.g., human walking and car running) and appearance variations (e.g., pose change, partial occlusion and illumination). The resolution, length, target size and challenges in each sequence were summarized in Table 1.

4.1.4. Evaluation criteria

For quantitative performance comparison, two popular evaluation criteria were used, namely, *center location error* (CLE) and *tracking success rate* (TSR). The CLE was computed as the distance between the predicted center position and the ground truth center position. The TSR was computed as the ratio of the number of frames the target was successfully tracked to the number of frames in the sequence. To define whether the target is successfully tracked at a frame, we use the score in the PASCAL VOC challenge [61], which can be computed as $score = area(\mathcal{R}^* \cap \mathcal{R}_{gt})/area(\mathcal{R}^* \cup \mathcal{R}_{gt})$, where $\mathcal{R}^*$ is the bounding box obtained by a tracker, $\mathcal{R}_{gt}$ is the corresponding ground truth bounding box and $area(\mathcal{R})$ denotes the area of the region $\mathcal{R}$. If the PASCAL score is larger than 0.5, the target is considered as being successfully tracked.

4.2. Comparison results and analysis

4.2.1. Overall performance

We first show the overall performance of all trackers on 20 test sequences. The TSR and average CLE of each tracker on all sequences are shown in Tables 2 and 3, respectively. Fig. 5 shows the CLE of each tracker over time on each sequence. From Table 2, we can see that there is at least one tracker achieving more than 95% TSR on the first 12 sequences. In contrast, on the remaining eight sequences, all trackers achieve bad performance, especially on *bird_1*, *board* and *box* sequences. We categorize the first twelve sequences as simple sequences and the remaining eight sequences as complex sequences. In Table 2, the highest TSR values for each sequence are highlighted in red bold font. We count the numbers of the highest TSR values for baseline trackers, TSSR trackers, AMSC trackers as well as their combination on simple sequences and complex sequences, respectively. Similarly, Table 3 shows the average CLE values of all trackers on all test sequences. We also highlighted the smallest average CLE in red bold font and then count the number of the smallest average CLE for different classes of trackers on simple sequences and complex sequences, respectively.

As we can see from these two tables, AMSC trackers significantly outperform TSSR trackers on twelve simple sequences. Therefore, we can conclude that sparse coding is more useful to model target appearance than perform target searching for visual tracking. This conclusion can be more convincingly supported by analyzing SMC_C and SMC_G trackers on these sequences. SMC_C and SMC_G are implemented in the same framework but just with different target searching and appearance modeling methods. Therefore, they are much easier and fairer to compare the benefits of using sparse coding in target searching and appearance modeling. As seen from Table 2, SMC_C tracker achieves the best performance on three simple sequences and SCM_G trackers achieves the best performance on eight simple sequences.

For the eight complex sequences, although all trackers do not achieve perfect tracking performances, AMSC trackers are still superior to TSSR trackers in terms of the numbers of the largest TSR values shown in Table 2 and the number of the smallest average CLE shown in Table 3. Fig. 6 shows some tracking results, from which we can see that most tracking failures occur when the target is occluded or there are similar objects around the target. For example, the girl is occluded by the man in the *girl* sequence. There are other runners similar to the tracked runner in the *running* sequence. On the other hand, all trackers based on sparse coding are slightly worse than the baseline trackers on these complex sequences, which indicates the trackers based on sparse coding still have large space for further improvements.

4.2.2. Performance of AMSC methods

From Section 4.2.1, we can see that the overall performance of AMSC methods are better than TSSR methods. One of the most important reasons is that the dictionary learned by sparse coding from image patches are capable of describing the underlying structure primitives (e.g., edge) in the image patches. When an image patch is represented by the learned dictionary, the representation coefficients can be used as local features to represent the image patch. When a pooling operator is used on the local codes, the resulting feature vector is effective to model the global appearance of the image. Compared with global TSSR methods, local AMSC methods are more stable, which are not sensitive to misalignment and noise. In this section, we analyze how

[3] http://spams-devel.gforge.inria.fr/

[4] These sequences can be downloaded from http://www.cs.toronto.edu/dross/ivt/, http://gpu4vision.icg.tugraz.at/subsites/prost, http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/, http://vision.ucsd.edu/bbabenko/project_miltrack.shtml, http://www.cvg.rdg.ac.uk/PETS2001/, http://www.ces.clemson.edu/~stb/research/headtracker/, http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm, http://cv.snu.ac.kr/research/~vtd/index.html, http://ice.dlut.edu.cn/lu/Project/iccv_spt_webpage/iccv_spt.htm, http://cv.snu.ac.kr/research/~bhmctracker/index.html, http://www.ece.northwestern.edu/~mya671/VehicleVideo/cvpr07_testvideo.rar.

Table 1
The used test sequences and their challenges as well as the size of the tracked target, frame size and frame length.

Sequences	Challenges	Target size	Frame size	Frame length
faceocc2	Pose change, partial occlusion	98 × 82	320 × 240	594
woman	Severe occlusion	93 × 31	352 × 288	333
singer	Severe illumination change, pose change	157 × 59	320 × 240	351
face	Severe occlusion	102 × 67	352 × 288	178
car4	Illumination change	88 × 104	360 × 240	630
david_outdoor	Illumination change, pose change	65 × 45	320 × 240	250
david_indoor	Illumination change, pose change, scale change, low contrast	79 × 61	320 × 240	460
CAVIAR	Severe occlusion, distraction from similar objects	115 × 31	384 × 288	500
PETS	Severe occlusion, pose change, distraction from similar objects	79 × 24	768 × 576	412
seq_mb	Severe pose change, partial occlusion, distraction from similar objects	45 × 31	128 × 96	500
sylv	Illumination change, pose change, scale change, cluttered background	61 × 51	320 × 240	800
bird_2	Severe pose change, low contrast	73 × 69	720 × 400	99
bird_1	Severe pose change, low contrast	37 × 31	720 × 400	408
girl	Severe pose change, severe occlusion	151 × 39	640 × 480	400
running	Severe pose change, partial occlusion	61 × 26	640 × 360	150
basketball	Partial occlusion, pose change, distraction from similar objects	81 × 34	576 × 432	725
liquor	Severe occlusion, distraction from similar objects	210 × 73	640 × 480	1741
lemming	Illumination change, partial occlusion	103 × 61	640 × 480	1336
board	Partial occlusion, distraction from similar objects	153 × 195	640 × 480	698
box	Partial occlusion, distraction from similar objects	112 × 86	640 × 480	500

Table 2
The TSR on 20 test sequences. The best result on each sequence is shown in bold font. The number of best results in each class are also shown in the middle and bottom rows. Their maximal values are shown in bold fonts.

	Baseline tracker					Target search based sparse representation					Appearance modeling based sparse coding				Combination
Sequence	Frag (%)	IVT (%)	MIL (%)	OAB (%)	CT (%)	L1 (%)	BL1 (%)	T2CL1 (%)	OT2CL1 (%)	SCM_C (%)	BRD (%)	OLSR (%)	SLSA (%)	SCM_G (%)	SCM (%)
faceocc2	68.18	96.63	54.04	83.00	87.37	62.63	90.24	59.26	95.79	79.46	94.95	85.86	64.81	64.81	**97.14**
woman	38.44	14.11	15.02	15.02	14.11	17.12	13.21	44.14	95.50	78.08	15.02	14.71	**100.00**	**100.00**	99.40
singer1	24.79	27.35	24.79	24.79	24.79	27.35	22.22	92.59	30.48	**100.00**	25.93	**100.00**	65.53	**100.00**	**100.00**
face	84.83	78.65	82.02	75.28	58.43	81.46	57.87	72.47	93.82	99.44	**100.00**	98.31	25.28	**100.00**	**100.00**
car4	27.30	97.78	27.62	27.46	27.46	24.60	30.95	68.41	63.17	**100.00**	32.38	**100.00**	**100.00**	55.24	**100.00**
david_outdoor	24.80	32.00	20.40	13.60	22.80	65.60	62.40	66.80	92.40	20.80	19.60	58.80	79.20	**95.20**	94.80
david_indoor	20.87	91.09	18.91	11.30	89.13	48.26	24.13	46.09	61.30	27.83	29.35	22.61	71.09	**99.57**	95.65
CAVIAR	35.40	**100.00**	38.80	38.60	35.80	38.00	89.20	87.00	97.60	**100.00**	79.00	39.80	40.40	**100.00**	**100.00**
PETS	50.49	**99.76**	53.16	58.01	46.36	0.24	0.24	59.47	55.34	98.54	10.68	62.62	0.24	83.74	94.90
seq_mb	77.60	32.00	37.00	70.40	16.80	68.20	65.60	73.40	**95.20**	86.00	30.80	9.60	80.80	64.20	54.60
sylv	91.12	76.12	91.38	79.50	94.38	46.75	48.38	70.50	**100.00**	55.88	53.50	90.12	97.75	**100.00**	**100.00**
bird_2	34.34	14.14	50.51	89.90	59.60	44.44	41.41	45.45	43.43	48.48	62.63	63.64	57.58	**95.96**	78.79
			# 2						# 5				**# 13**		# 6
bird_1	22.79	2.21	**46.57**	27.21	25.49	0.25	0.49	0.98	1.23	14.71	3.19	7.35	1.72	21.81	20.34
girl	24.75	26.00	14.75	27.75	22.25	21.00	15.00	44.25	31.25	41.50	**65.50**	21.50	27.75	45.25	29.50
running	8.67	2.00	4.00	8.67	0.00	2.00	0.67	7.33	15.33	6.00	**82.00**	5.33	2.00	2.67	58.67
basketball	45.10	44.55	28.14	9.93	**82.34**	25.66	14.34	23.59	17.93	36.00	35.17	11.72	6.48	8.28	8.41
liquor	27.51	**68.52**	20.96	20.96	21.19	54.74	45.43	63.01	64.45	22.92	20.28	20.96	23.03	39.17	48.31
lemming	50.07	40.19	38.40	38.40	54.34	38.40	41.84	56.29	**89.07**	25.00	48.13	3.37	18.11	62.57	48.28
board	11.03	0.86	5.30	4.30	3.72	1.72	0.14	2.15	2.44	1.72	17.77	14.61	**19.34**	16.05	14.76
box	9.40	11.40	3.40	6.00	**18.40**	4.60	5.00	9.20	10.60	6.40	10.60	8.80	12.20	15.20	12.60
			# 4						# 1				# 3		# 0

dictionary learning and pooling operator affect the performance of AMSC methods. We select three dictionary learning methods: (1) learning dictionary using image patches sampled from target image, **T**, (2) learning dictionary using patches sampled from both target and background images, (**T**+**B**) and (3) first learning dictionary using patches sampled from target image and background image respectively and then combining them to form the final dictionary, **[T, B]**. We chose five pooling operators: (1) concatenation, **Con**; (2) max pooling, **Max**; (3) average pooling, **Ave**; (4) multi-scale max pooling, **Mul-Max**; (5) multi-scale average pooling, **Mul-Ave**. The multi-scale versions of max pooling and average pooling are based on spatial pyramid representation with scales [0,1,2] as in Yang et al. [34].

We conducted experiments on eight complex test sequences. The TSR measures on these sequences are shown in Table 4, from which we can see that when use the same pooling operators, the dictionary method **[T, B]** achieves the best performance than **T** and (**T**+**B**). On the other hand, dictionary learning methods **T** is better than (**T**+**B**). The performance difference between these three dictionary learning methods is due to the discriminative abilities of the learned dictionary. The **[T, b]** method is the most discriminative. The dictionary learning method **T** just uses the samples from the target image to learn the dictionary, which can sparsely represent the samples from the target image with minimal reconstruction error but can not sparsely represent the samples from the background image. Therefore, the learned dictionary still has discriminative ability in some extent. In contrast, the dictionary learning method (**T**+**B**) uses samples from both the target and background images to learn the dictionary, which can sparsely represent the samples from both the target and background images. Therefore, the codes have the least

Table 3
Average CLE on 20 test sequences. The best result on each sequence is shown in bold font. The numbers of best results in each class are also shown in the middle and bottom rows. Their maximal values are shown in bold fonts.

	Baseline tracker					Target search based sparse representation					Appearance modeling based sparse coding				Combination
Sequence	Frag	IVT	MIL	OAB	CT	L1	BL1	T2CL1	OT2CL1	SCM_C	BRD	OLSR	SLSA	SCM_G	SCM
faceocc2	25.15	6.46	19.18	14.72	16.25	35.72	11.85	30.07	**5.89**	10.63	8.77	9.15	24.95	24.70	8.13
woman	92.26	146.67	117.70	105.42	106.80	94.37	88.42	44.82	2.93	5.40	80.77	140.83	2.49	1.76	**1.74**
singer1	14.15	7.42	18.59	18.37	15.44	61.26	78.08	4.94	59.66	2.47	18.58	2.59	4.44	2.24	**2.10**
face	9.26	12.30	14.20	14.82	22.73	23.37	45.80	28.18	7.31	5.68	6.15	4.91	53.61	**3.35**	3.50
car4	133.35	5.78	90.92	49.93	81.56	92.78	160.22	27.18	26.61	**2.67**	142.58	3.17	3.18	10.68	3.36
david_outdoor	79.17	89.20	55.35	67.21	66.20	24.49	39.72	25.17	**5.78**	78.39	74.34	28.99	14.14	6.31	6.28
david_indoor	42.18	**3.61**	25.85	21.34	6.91	25.23	68.09	44.66	35.58	66.35	82.19	116.28	15.91	4.56	5.48
CAVIAR	57.96	3.06	57.47	23.01	58.99	58.09	7.17	8.52	2.48	2.32	8.85	61.78	45.75	**2.02**	2.08
PETS	9.30	**2.13**	4.49	4.26	5.68	332.38	379.08	78.75	39.50	2.19	272.01	32.05	14.07	3.13	2.25
seq_mb	6.30	17.72	13.77	10.14	18.91	11.54	13.02	17.62	3.68	**3.15**	27.49	20.72	6.91	8.46	11.17
sylv	5.89	28.12	7.74	13.38	7.09	46.81	42.89	29.29	**1.36**	41.54	41.36	7.67	6.73	4.29	4.09
bird_2	50.24	99.37	21.54	12.13	19.97	51.11	36.72	52.84	66.70	53.36	13.85	15.27	20.42	**7.73**	11.98
			# 2					**# 5**					# 3		# 2
bird_1	266.54	162.90	**19.96**	132.27	94.60	182.26	152.18	104.87	169.70	55.23	339.46	67.33	168.61	77.52	72.24
girl	111.87	104.28	154.38	97.54	113.00	198.06	180.80	101.91	181.26	119.54	**13.80**	135.38	156.12	20.38	91.59
running	90.50	279.80	271.82	254.78	233.26	104.05	101.44	255.48	68.66	273.66	**8.43**	281.10	260.17	290.91	13.90
basketball	70.61	**7.29**	66.20	120.09	12.68	98.06	136.30	72.61	100.31	45.83	108.34	132.65	145.34	185.03	198.03
liquor	96.59	61.69	149.86	146.41	137.02	90.39	115.92	72.83	80.59	113.05	191.62	190.20	103.08	70.45	**56.10**
lemming	93.26	117.98	109.70	84.30	55.59	108.38	106.47	125.48	**16.14**	121.95	97.50	213.01	162.30	93.23	132.17
board	350.20	330.74	315.44	333.15	289.08	286.65	550.41	333.26	342.34	**279.01**	356.01	367.88	346.78	350.68	391.86
box	328.17	334.64	459.00	326.64	328.09	296.37	448.04	324.82	369.72	424.50	370.54	**265.77**	354.89	353.79	368.93
			# 2					# 2					**# 3**		# 1

discriminative ability. Here, we conducted an experiment to validate the discriminative abilities of three different dictionary learning methods. We sample 10,000 8×8 positive patches from one target image and 10,000 8×8 negative patches from 50 background images to learn the dictionaries using three dictionary learning methods. The target image and 50 background images are represented by histograms obtained by average pooling on local codes. The similarity between the target image and one background image is computed as the histogram intersection between their histograms. We then compute the average of similarities between the target image and 50 background images. The smaller the average of similarities is, the high discriminative ability is. We show the average of similarities over different sizes of dictionary in Fig. 8. From this figure, we can see that (1) for all three dictionary learning methods, the discriminative abilities increase when the size of the dictionary increase except slight decrease for **(T+B)** and **[T, B]** with size 96. (2) for any fixed size of the dictionary, the dictionary learning method **[T, B]** achieves the highest discriminative ability.

From Table 4, we can see that the **Con** pooling operator is better than the **Sum** and **Ave** pooling operators, which validates that the **Con** pooling is more discriminative than the **Sum** and **Ave** pooling operators. The best tracking performance on all test sequences are achieved when the **Mul-Ave** pooling is used. The reason is that **Mul-Ave** pooling is more discriminative than other pooling operators. The high discriminative ability comes two aspects: (1) the multiple scale spatial pyramid representation reserves the spatial order of local codes. (2) the average pooling used in each bock at each scale is also more discriminative than the sum pooling.

4.2.3. Performance of TSSR methods

As observed in Section 4.2.1, TSSR methods get worse performance than AMSC methods. In this section, we chose BL1 tracker to further analyze why it does not perform well on most sequences. The first reason is that the template update is not effective. For example, on the *car4* sequence which contains severe illumination, especially when the car passes beneath a bridge and under trees, the target and background occurs severe illumination changes. The target templates can not be updated immediately to reflect the appearance changes. Therefore the tracker failed to accurately track the target. The second reason is that the ℓ_1-norm minimization is very sensitive to background cluster. For example, in the *box* sequence, similar background regions will affect the tracking performance. When the similar background pixels appear in the candidate, the identity pixel basis will be activated to capture these background pixels. We show the first failure of the *box* sequence in Fig. 9. In this figure, we can clearly see that the candidate marked by the red rectangle is significantly different with all templates. However, the BL1 failed to treat this candidate as the tracking result. In contrast, the ground truth candidate marked by blue rectangle is very similar to the template set. However, the tracker failed to track it. We show the coefficients when represent two candidates with the ten target templates and 180 trivial templates. We found that there is only one nonzero coefficient corresponding to the target templates. For the ground truth candidate, the only one nonzero coefficient corresponding to the target templates is 1.2713. For the wrong candidate, the coefficient is 2.0422. The reconstructed candidates using the 10 target templates and their associated coefficients are shown above the plots and we can see that they are very similar just with different intensity levels determined by the nonzero coefficients. The reconstruction errors of these two candidates are 0.9864 and 0.8109, respectively. The BL1 tracker computed the similarity between the candidate and the target templates based on the reconstruction error; therefore, it failed to track the right target. We further analyze why the reconstruction error of the wrong candidate is smaller than the ground truth candidate. We found the main reason is due to the downsampling and zero-mean-unit-norm normalization used in Mei et al. [46]. Both the downsampling and normalization reduce the difference between the reconstructed image and the candidate itself.

In addition, although TSSR methods are motivated by face recognition work, there are some researchers claimed that face recognition may be not a sparse representation problem [62,63] and their experimental results on face recognition indicated that non-sparse representation based on ℓ_2 norm minimization (e.g., $\mathbf{u}=\arg\min_{\mathbf{u}}\|\mathbf{x}-\mathbf{V}\mathbf{u}\|_2^2$) is more effective than ℓ_1 norm minimization (Eq. (4)). Is target searching in visual tracking a sparse representation problem? There was initial work [64] that points

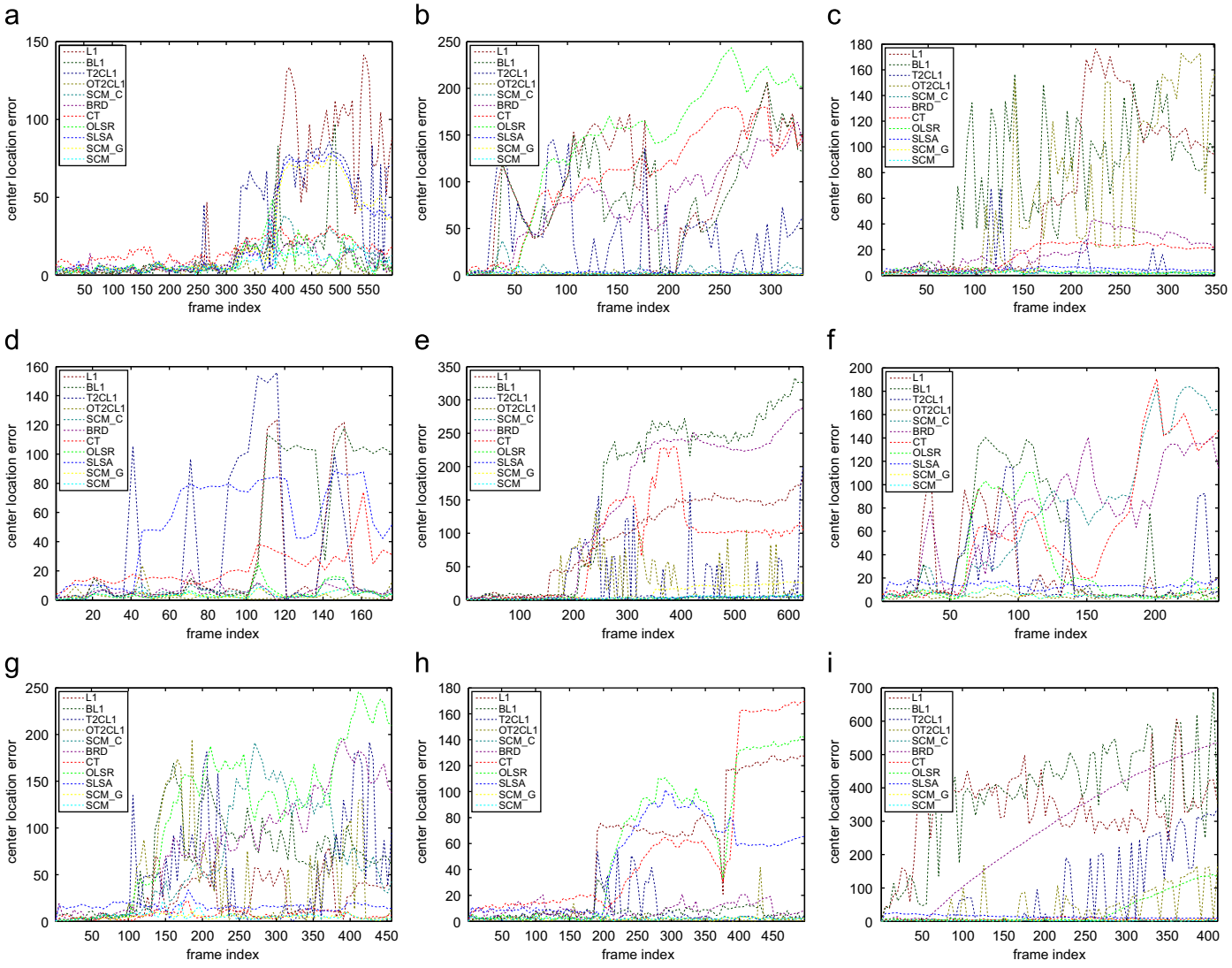

Fig. 5. Quantitative comparison of different trackers in CLE on all 20 sequences. (a) *faceocc2*. (b) *woman*. (c) *singer*. (d) *face*. (e) *car4*. (f) *david_outdoor*. (g) *david_indoor*. (h) *CAVIAR*. (i) *PETS*. (j) *seq_mb*. (k) *sylv*. (l) *bird_1*. (m) *bird_1*. (n) *girl*. (o) *running*. (p) *basketball*. (q) *liquor*. (r) *lemming*. (s) *board*. (t) *box*.

out this problem but without performing detailed evaluation. Here, we also conduct some experiments to compare the effectiveness of sparse representation based on ℓ_1 norm minimization and non-sparse representation based on ℓ_2 norm minimization. We modified the SCM_L1 tracker to obtain a new tracker named SCM_L2 tracker which computes the representation coefficients by minimizing the ℓ_2 norm based reconstruction error instead of minimizing the sum of ℓ_2 norm reconstruction error and ℓ_1 norm sparsity measure in SCM_L1 tracker. In addition, we also wish to validate whether the identity pixel basis is necessary when both target and background images are contained in the dictionary. Therefore, we modified the SCM_L1 tracker to let the dictionary contains identity pixel basis. The modified tracker is called SCM_L1I. For SCM_L1, SCM_L2 and SCM_L1I. The most important parameters that affect the tracking performance are the sizes of target and background dictionaries. We set 10 pairs of these two parameters (10, 40), (20, 80), (30, 120), (40, 160), (50, 200), (60, 240), (70, 280), (80, 320), (90, 360), (100, 400) and compared their performance on 12 simple sequences. The TSR values on each sequence and different pairs of dictionary sizes are shown in Fig. 7. We can see that SCM_L1I is very close to SCM_L1, which indicates that the identity pixel basis is not necessary for sparse representation based target searching. On sequences *CAVIAR*, *PETS* and *seq_mg*, SCM_L2 is slightly worse than SCM_L1. However, on remaining sequences, they get very close performance, which indicates that ℓ_1 norm minimization used in existing TSSR methods can be replaced by ℓ_2 norm minimization when both target and background dictionaries are used to represent target candidates. Because the ℓ_2 norm minimization can be more efficient to be solved than ℓ_1 norm minimization, therefore, ℓ_2 norm minimization is more suitable for real-time visual tracking than the widely used ℓ_1 norm minimization.

4.2.4. *Performance of combination of AMSC and TSSR methods*

So far, there is only one work [59] that combines both AMSC and TSSR methods. As shown in Table 2, the SCM_G tracker that only uses the AMSC method achieves the best overcome performance in all trackers. However, the SCM_C trackers achieves worse performance than the SCM_G tracker. The SCM tracker that combines both AMSC and TSSR achieve the performance better than SCM_G tracker but worse than SCM_G tracker. From this point, we can see that the combination of both AMSC and TSSR seemly does not improve the tracking performance compared with using AMSC alone. The one possible reason is that [59] just use a simple fusion manner to combine AMSC and TSSR, which results in the final performance of

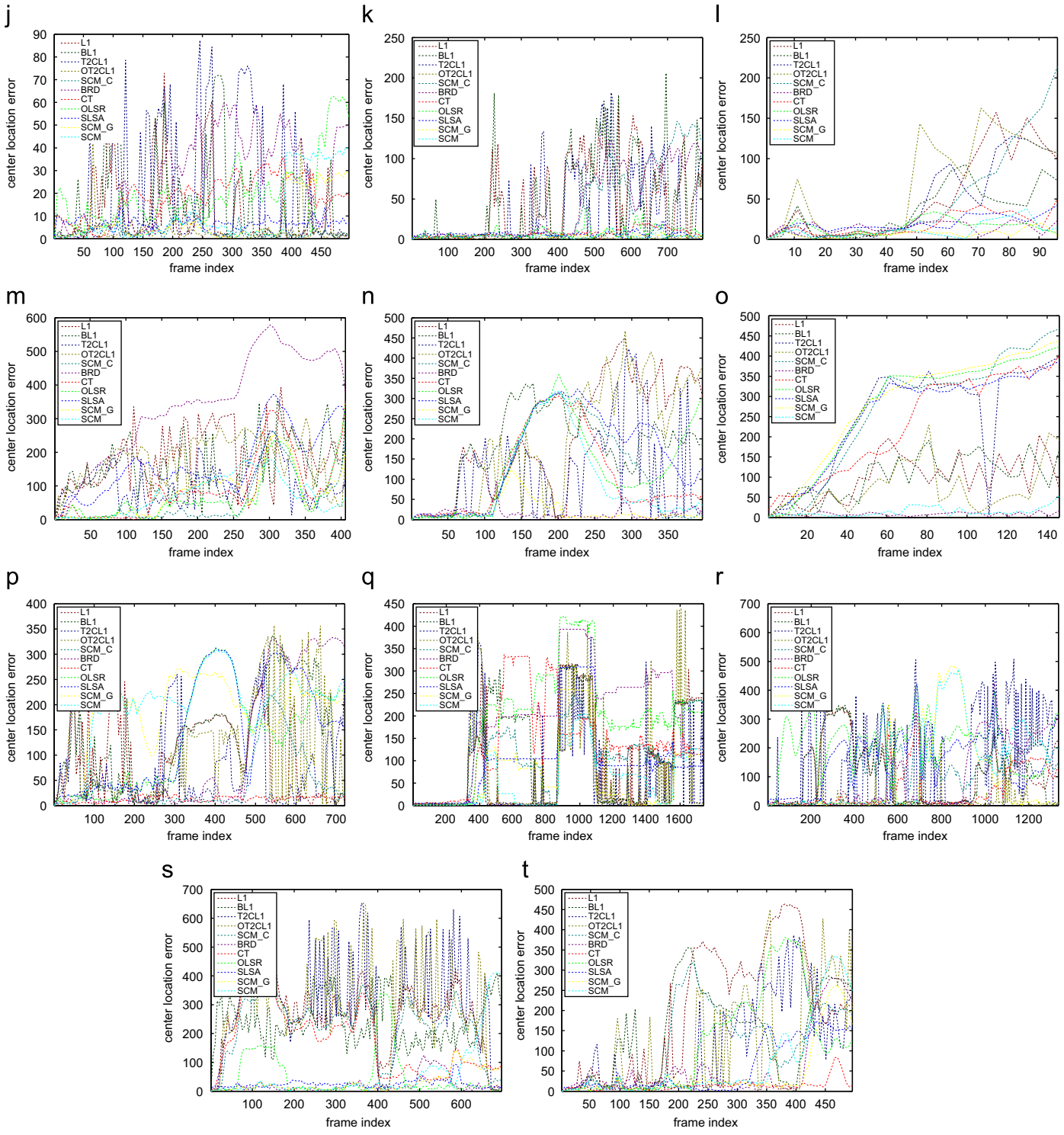

Fig. 5. Continued.

the SCM tracker being between the SCM_C and SCM_G trackers. On the other hand, this comparison experiment also validates the effectiveness of the AMSC trackers compared with TSSR trackers.

4.2.5. *Complexity comparison*

Computation complexity is a very important issue for real-time visual tracking. As well know, sparse coding is an optimization problem, which makes both sparse representation and dictionary learning computational expensive. As introduced in the previous sections, for AMSC methods, the dictionary is only learned at the first time. Therefore, for both AMSC and TSSR methods, the computational cost is mainly from the solving of ℓ_1-norm minimization. Both the number of solving ℓ_1-norm minimization and the cost of solving each ℓ_1-norm minimization significantly affect the running speed of both AMSC and TSSR trackers. The computation complexity of solving each ℓ_1-norm minimization is $\mathcal{O}(D^2K^{3/2})$ where D is the dimension of each basis

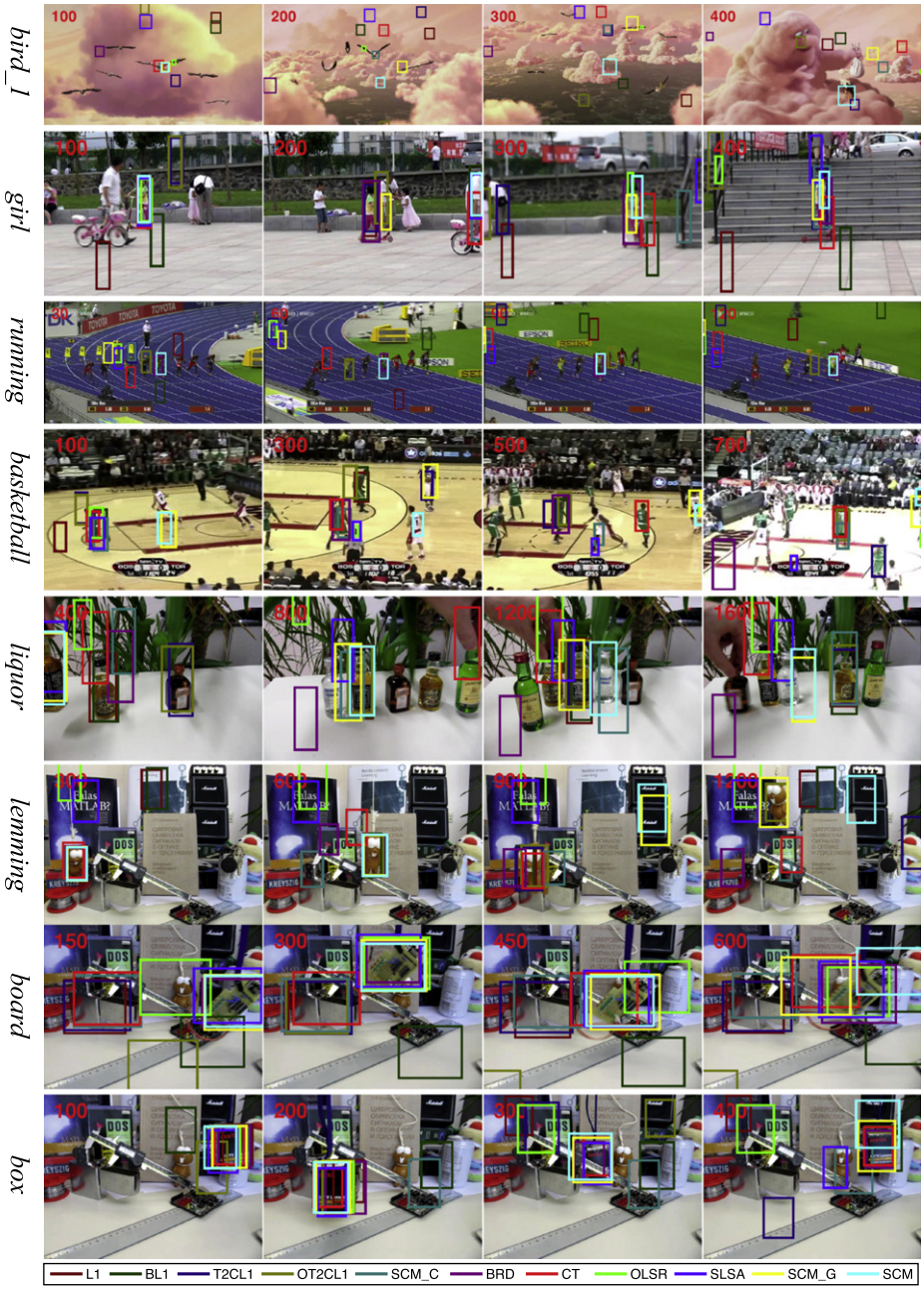

Fig. 6. Some tracking results on eight complex sequences.

function and K is the number of basis functions. Let N denote the number of solving ℓ_1-norm minimization. For L1 tracker, $N=600$ is the number of target candidates. For T2CL1, $N=1$ is the number of target tempts. For SCM_G tracker, $N=729$ is the number of sampled patches from the target template or candidates. The computation complexity of a tracker is $\mathcal{O}(ND^2K^{3/2})$. In our experiments, for AMSC trackers, $D=36$, $K=36$. For TSSR trackers, $D=1024$, $K=250$.

We compare the running time (seconds/frame) of the selected trackers on *faceocc2* sequence using Matlab 2011b on a laptop with 2.4 GHz Intel Core i5 processor and 4 GB memory. The comparison results are shown in Table 5, from which we can see that: (1) AMSC trackers are significantly faster than TSSR trackers. (2) For TSSR trackers, T2CL1 and OT2CL1 are faster than L1 and BL1. (3) SCM_L2 is significantly faster than SCM_L1. SCM_L1I is more slow than SCM_L1. These quantitative results are consistent with the analysis in previous sections.

5. Conclusion and future work

In this work, we reviewed the recently proposed tracking methods based on sparse coding and conducted extensive experiments to analyze the benefits of using sparse coding for visual tracking. The main contributions of this work are three-folds:

- The first contribution of this work is to explain the motivations of using sparse coding in visual tracking. In particularly, we emphasized the difference between sparse representation and sparse coding and analyzed the benefits of using them in different stages of a typical tracking process.
- We proposed to categorize the state-of-the-art tracking methods based on sparse coding into AMSC and TSSR as well as their combination, which provides readers a reference about these methods and also helps them to understand the connections and differences between these methods.

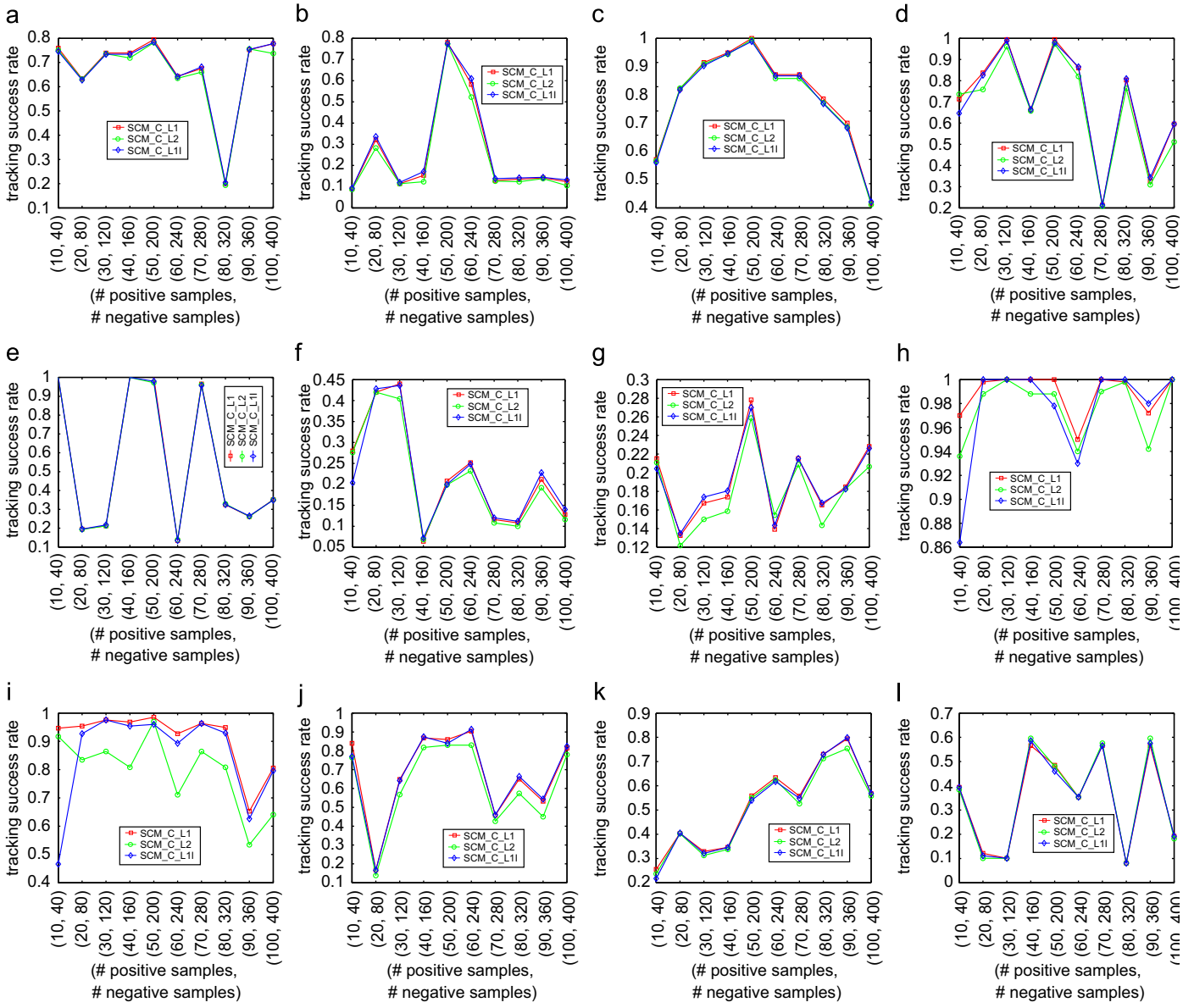

Fig. 7. The TSR of SCM_L1, SCM_L1l and SCM_L2 trackers on 12 simple sequences. (a) *faceocc2*. (b) *woman*. (c) *singer*. (d) *face*. (e) *car4*. (f) *david_outdoor*. (g) *david_indoor*. (h) *CAVIAR*. (i) *PETS*. (j) *seq_mb*. (k) *sylv*. (l) *bird_1*.

Table 4
The TSR on eight complex sequences using different dictionary learning methods and pooling operators. The best results are shown in bold fonts.

Sequence	T/ Con (%)	T/ Sum (%)	T/ Ave (%)	T/Mul-Sum (%)	T/Mul-Ave (%)	(T+B)/ Con (%)	(T+B)/ Sum (%)	(T+B)/ Ave (%)	(T+B)/Mul-Sum (%)	(T+B)/ Mul-Ave (%)	[T, B]/ Con (%)	[T, B]/ Sum (%)	[T, B]/ Ave (%)	[T, B]/Mul-Sum (%)	[T, B]/ Mul-Ave (%)
bird_1	23.57	7.68	19.13	10.25	28.31	15.98	5.65	10.20	7.18	12.45	30.45	10.11	20.09	15.89	**45.62**
girl	50.08	32.45	40.07	43.56	59.86	35.28	17.32	26.43	20.87	40.66	65.12	42.78	45.89	50.12	**65.46**
running	12.33	5.62	8.78	6.84	15.45	7.12	4.87	7.25	6.49	10.28	15.44	12.83	17.65	10.58	**19.47**
basketball	10.72	4.88	6.32	5.46	11.27	4.02	3.65	5.93	4.77	7.13	14.85	10.30	12.36	11.36	**15.65**
liquor	40.17	23.53	34.25	27.34	36.89	20.13	11.78	19.79	21.68	27.34	45.98	30.66	40.12	34.58	**52.15**
leming	65.34	43.25	45.89	45.23	53.08	40.28	29.30	36.58	29.97	36.34	70.34	55.70	60.27	58.90	**72.35**
board	20.13	9.68	20.45	15.08	30.12	15.75	12.31	15.78	10.29	19.38	35.21	20.08	40.58	30.75	**35.82**
box	18.96	8.24	15.37	14.72	25.29	11.87	5.96	13.24	11.98	20.63	26.78	12.17	20.98	25.86	**34.28**

- We conducted extensive experimental comparisons between eleven trackers based on sparse coding and four baseline tracker on a total of 20 challenging sequences. In addition to compare the overall performance of these trackers, we also validated the benefits of using different components including ℓ_1 norm minimization, dictionary learning, feature pooling. Computational complexities of several representative trackers are also compared.

Based on the categorization of these methods and the extensive experimental results, we concluded the application of sparse coding in visual tracking, including

- AMSC methods significantly outperform TSSR methods, which indicates that sparse coding is more suitable for modeling target appearance than performing target searching. In addition, for AMSC methods, both dictionary learning and pooling operator affect the final tracking performance. When discriminative dictionary is used, the resulting local codes are more easy to distinguish the target patches from the background patches. When multiple scale pooling operator based on pyramid structure is used, the resulting global appearance model reserves rich spatial order information of the local codes.
- For TSSR methods, when the dictionary contains identity pixel basis, the tracking performance is very sensitive to similar background or occlusion. In addition, when the dictionary consists of both target and background images, the identity pixel basis is not necessary and can be removed from the dictionary but without significant performance degradation. On the other hand, visual tracking may be not a sparse representation problem. When the sparsity constrain on the representation coefficients is removed, the traditional ℓ_1 norm minimization tracking framework is reduced to a ℓ_2 norm non-sparse representation tracking framework, which achieves similar tracking performance but with lower computational complexity.

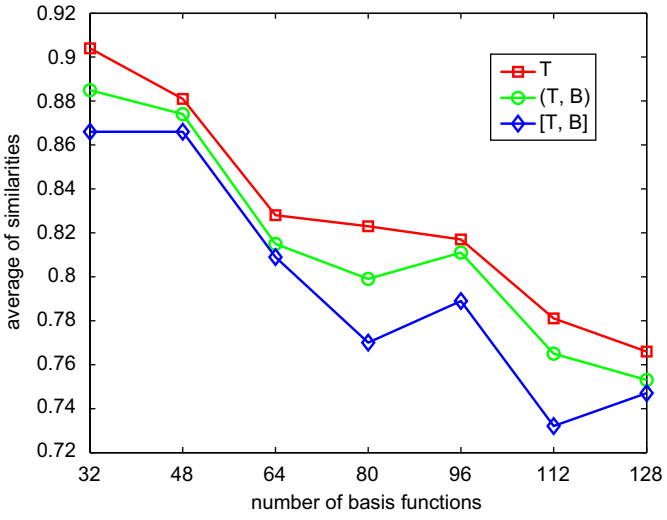

Fig. 8. Illustration of discriminative abilities of three dictionary learning methods over different dictionary sizes.

Although sparse coding has been used in visual tracking for several years, the tracking performance is still not satisfactory particularly on eight complex sequences shown in Fig. 6. To further improve tracking performance, the following topics are worthing investigating:

- In the past, most attentions had been paid on how to use sparse representation to perform target searching, which is motivated by the success of using sparse representation in face recognition. However, the latest work in face recognition indicated that face recognition may be not a sparse representation problem. Our experimental results also validated that visual tracking may be not a sparse representation problem. Therefore, in the future work, we will investigate how to use non-sparse representation to perform target searching. The most desired advantage of non-sparse representation is its low computational complexity, which makes it especially suitable for developing real-time visual tracking methods.
- Appearance modeling based on sparse coding has been attracting much more attention in recent. However, the dictionary learning and pooling operators used in existing methods can be further improved to enhance the discriminative ability of the appearance model. For dictionary learning, there are some effective discriminative dictionary methods [65,66] proposed in object recognition, which could be modified to be used in visual tracking. For featuring pooling, the existing pooling operators in visual tracking either lose spatial information of local codes such as max pooling and average pooling or reserve simple spatial information such as multiple scale pooling. A more reasonable pooling operator should exploit the global structure information [67] to model target appearance. In

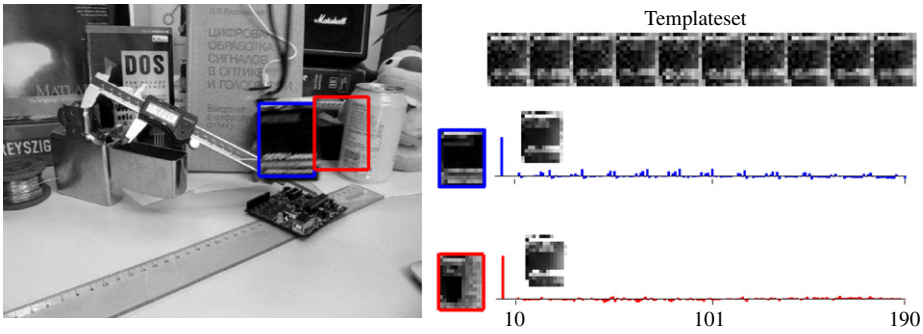

Fig. 9. An illustration of the first failure on the *box* sequence by BL1 tracker. The ground truth and the tracking result in this frame are marked by the blue and red rectangles, respectively. These two candidates are also shown in the middle column. The 10 templates are shown on the top right. The coefficients of representing two candidates using these templates are shown by the blue and red bar plots, respectively. The reconstructed image using the templates and the associated coefficients are shown above the plots. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Table 5
Computation speeds (seconds/frame) of the selected trackers on *faceocc2* sequence. It should be noted that our Matlab codes were not optimized.

Tracker	L1	BL1	T2CL1	OT2CL1	BRD	CT	SLSA	SCM_G	SCM_L1	SCM_L2	SCM_LiI
Speed (seconds/frame)	5.56	1.97	0.94	1.13	0.15	0.04	0.42	0.24	6.22	0.24	7.85

addition, motivated by the success of deep learning in object recognition, hierarchical sparse coding [68] should also be investigated in visual tracking.

- The latest work that combines both appearance modeling based on sparse coding and target searching based on sparse representation reported some initial results. But the results are not satisfactory due to the simple combination strategy. Future work can be focused on how to develop effective combination methods that exploit their advantages together. A possible solution is to use multi-task sparse coding [69] to treat them as different tasks and then integrate them together.

Acknowledgments

The authors would like to thank the editor and reviewers for their valuable comments. The work was supported by the National Natural Science Foundation of China (No. 61071180) and Key Program (No. 61133003). Shengping Zhang was also supported by the Short-Term Overseas Visiting Scholar Program of Harbin Institute of Technology funding when he was an visiting student researcher at Redwood Center for Theoretical Neuroscience, University of California, Berkeley, United States.

References

[1] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: Proceedings of the European Conference on Computer Vision, 2002, pp. 661–675.
[2] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5) (2003) 564–577.
[3] H. Zhou, T. Liu, J. Zheng, F. Lin, Y. Pang, J. Wu, Tracking non-rigid objects in video sequences, International Journal of Information Acquisition 3 (2) (2006) 131–137.
[4] D. Ross, J. Lim, R. Lin, M. Yang, Incremental learning for robust visual tracking, International Journal of Computer Vision 77 (8) (2007) 125–141.
[5] H. Zhou, Y. Yuan, C. Shi, Object tracking using sift features and mean shift, Computer Vision and Image Understanding 113 (3) (2009) 345–352.
[6] X. Sun, H. Yao, S. Zhang, A novel supervised level set method for non-rigid object tracking, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2011, pp. 3393–3400.
[7] H. Zhou, H. Hu, Human motion tracking for rehabilitation—a survey, Biomedical Signal Processing and Control 3 (1) (2008) 1–18.
[8] H. Yang, L. Shao, F. Zheng, L. Wang, Z. Song, Recent advances and trends in visual tracking: a review, Neurocomputing 74 (18) (2011) 3823–3831.
[9] A. Jepson, D. Fleet, T. EI-Maraghi, Robust online appearance models for visual tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (10) (2003) 1296–1311.
[10] R. Collins, Y. Liu, M. Leordeanu, On-line selection of discriminative tracking features, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (10) (2005) 1631–1643.
[11] H. Grabner, H. Bischof, On-line boosting and vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2006, pp. 260–267.
[12] H. Grabner, C. Leistner, Semi-supervised on-line boosting for robust tracking, in: Proceedings of the 10th European Conference on Computer Vision, 2008, pp. 234–247.
[13] B. Babenko, M. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1619–1632.
[14] C. Kuo, C. Huang, R. Nevatia, Multi-target tracking by on-line learned discriminative appearance models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1–8.
[15] D. Donoho, Compressed sensing, IEEE Transactions on Information Theory 52 (4) (2006) 1289–1306.
[16] E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Transactions on Information Theory 52 (2) (2006) 489–509.
[17] B. Olshausen, D. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (1996) 607–609.
[18] J. Wright, A. Yang, A. Ganesh, S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 210–227.
[19] X. Mei, H. Ling, Robust visual tracking using L1 minimization, Proceedings of the 12th International Conference on Computer Vision, 2009, pp. 1436–1443.
[20] M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex, Nature Neuroscience 2 (11) (1999) 1019–1025.
[21] S. Zhang, H. Yao, S. Liu, Robust visual tracking using feature-based visual attention, in: Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing, 2010, pp. 1150–1153.
[22] D. Field, What is the goal of sensory coding ? Neural Computation 6 (4) (1994) 559–601.
[23] D. Donoho, For most large underdetermined systems of linear equations the minimal L1-norm solution is also the sparsest solution, Communications on Pure and Applied Mathematics 59 (6) (2006) 797–829.
[24] M. Figueiredo, R. Nowak, S. Wright, Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, IEEE Journal of Selected Topics in Signal Processing 1 (4) (2007) 586–597.
[25] S. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale ℓ_1-regularized least squares, IEEE Journal of Selected Topics in Signal Processing 1 (4) (2007) 606–617.
[26] K. Engan, S. Aase, J. Husoy, Frame based signal compression using method of optimal directions, Proceedings of the IEEE International Symposium on Circuits and Systems 4 (1999) 1–4.
[27] M. Aharon, M. Elad, A. Bruckstein, Svd: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on Signal Processing 54 (11) (2006) 4311–4322.
[28] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological Cybernetics 36 (4) (1980) 193–202.
[29] D. Perrett, M. Oram, Neurophsyiology of shape processing, Image and Vision Computing 11 (6) (1993) 317–333.
[30] T. Serre, L. Wolf, T. Poggio, Object recognition with features inspired by visual cortex, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 994–1000.
[31] J. Mutch, D. Lowe, Object class recognition and localization using sparse features with limited receptive fields, International Journal of Computer Vision 80 (1) (2008) 45–57.
[32] D. Gabor, Theory of communication, Journal of the Institution of Electrical Engineers 94 (73) (1947) 429–459.
[33] B. Olshausen, D. Field, Sparse coding with an overcomplete basis set: a strategy employed by V1? Vision Research 37 (1997) 3311–3325.
[34] J. Yang, K. Yu, Y. Gong, T. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1794–1801.
[35] L. Fei-Fei, P. Perona, A Bayesian hierarchical model for learning natural scene categories, in: Proceedings of International Conference on Computer Vision and Pattern Recognition, 2005, pp. 524–531.
[36] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[37] A. Bell, T. Sejnowski, The independent components of natural scenes are edge filters, Vision Research 37 (23) (1997) 3327–3338.
[38] M. Swain, D. Ballard, Color indexing, International Journal of Computer Vision 7 (1) (1991) 11–32.
[39] Q. Wang, F. Chen, J. Yang, W. Xu, M. Yang, Transferring visual prior for online object tracking, IEEE Transactions on Image Processing 21 (7) (2012) 3296–3305.
[40] B. Liu, J. Huang, L. Yang, C. Kulikowsk, Robust tracking using local sparse appearance model and K-selection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1313–1320.
[41] Q. Wang, F. Chen, W. Xu, M. Yang, Online discriminative object tracking with local sparse representation, in: Proceedings of the IEEE Workshop on the Applications of Computer Vision, 2012, pp. 425–432.
[42] X. Jia, H. Lu, M. Yang, Visual tracking via adaptive structural local sparse appearance model, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1822–1829.
[43] S. Zhang, H. Yao, X. Sun, S. Liu, Robust object tracking based on sparse representation, in: Proceedings of the International Conference on Visual Communications and Image Processing, pp. 77441N-1-8, 2010.
[44] G. Tzimiropoulos, S. Zafeiriou, M. Pantic, Sparse representations of image gradient orientations for visual recognition and tracking, in: Proceedings of the CVPR Workshop for Human Behaviour Analysis, 2011, pp. 26–33.
[45] Y. Wu, E. Blasch, G. Chen, L. Bai, H. Ling, Multiple source data fusion via sparse representation for robust visual tracking, in: Proceedings of the 13th Conference on Information Fusion, 2011, pp. 1–8.
[46] X. Mei, H. Ling, Y. Wu, E. Blasch, L. Bai, Minimum error bounded efficient l1 tracker with occlusion detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1257–1264.
[47] C. Bao, Y. Wu, H. Ling, H. Ji, Real time robust l1 tracker using accelerated proximal gradient approach, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1830–1837.
[48] M. Yang, L. Zhang, Gabor feature based sparse representation for face recognition with Gabor occlusion dictionary, in: Proceedings of the 11th European Conference on Computer Vision, 2010, pp. 448–461.
[49] S. Zhang, H. Yao, H. Zhou, X. Sun, S. Liu, Robust visual tracking based on online learning sparse representation, Neurocomputing 100 (2013) 31–40.
[50] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online dictionary learning for sparse coding, in: Proceedings of the 26th Annual International Conference on Machine Learning, vol. 382, 2009, pp. 689–696.
[51] H. Ling, L. Bai, E. Blasch, X. Mei, Robust infrared vehicle tracking across target pose change using L1 regularization, in: Proceedings of the 13th Conference on Information Fusion, 2010, pp. 1–8.

[52] B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, C. Kulikowski, Robust and fast collaborative tracking with two stage sparse optimization, in: Proceedings of the 11th European Conference on Computer Vision, 2010, pp. 624–637.
[53] Y. Wu, H. Ling, E. Blasch, L. Bai, G. Chen, Visual tracking based on log-Euclidean Riemannian sparse representation, in: Proceedings of the Seventh International Symposium on Advances in Visual Computing, 2011, pp. 738–747.
[54] H. Li, C. Shen, Q. Shi, Real-time visual tracking using compressive sensing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1305–1312.
[55] K. Zhang, L. Zhang, M. Yang, Real-time compressive tracking, European Conference on Computer Vision, in: Proceedings of the 12th European Conference on Computer Vision, 2012, pp. 864–877.
[56] M. Yang, L. Zhang, J. Yang, D. Zhang, Robust sparse coding for face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 625–632.
[57] J. Yan, M. Tong, Weighted sparse coding residual minimization for visual tracking, in: Proceedings of the Second International Conference on Internet Multimedia Computing and Service, 2011, pp. 1–4.
[58] H. Liu, F. Sun, M. Gao, Visual tracking using iterative sparse approximation, in: Proceedings of the Eighth International Symposium on Neural Networks, 2011, pp. 207–214.
[59] W. Zhong, H. Lu, M. Yang, Robust object tracking via sparsity-based collaborative model, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1838–1845.
[60] A. Adam, E. Rivlin, I. Shimshoni, Robust fragments-based tracking using the integral histogram, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 798–805.
[61] M. Everingham, L. Gool, C. Williams, J. Winn, A. Zisserman, The Pascal visual object classes challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.
[62] R. Rigamonti, M. Brown, V. Lepetit, Are sparse representations really relevant for image classification, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1545–1552.
[63] Q. Shi, A. Eriksson, A. Hengel, C. Shen, Is face recognition really a compressive sensing problem, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2011, pp. 553–560.
[64] X. Li, C. Shen, Q. Shi, A. Dick, A. Hengel, Non-sparse linear representations for visual tracking with online reservoir metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1760–1767.
[65] Q. Zhang, B. Li, Discriminative K-SVD for dictionary learning in face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1–8.
[66] Z. Jiang, Z. Lin, L. Davis, Learning a discriminative dictionary for sparse coding via label consistent k-svd, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1697–1704.
[67] J. Huang, T. Zhang, D. Metaxas, Learning with structured sparsity, Journal of Machine Learning Research 12 (2011) 3371–3412.
[68] K. Yu, Y. Lin, J. Lafferty, Learning image representations from the pixel level via hierarchical sparse coding, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1713–1720.
[69] X. Yuan, S. Yan, Visual classification with multi-task joint sparse representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3493–3500.

Shengping Zhang received his MS and PhD degrees in computer science from Harbin Institute of Technology, Harbin, China. Currently, he is a lecturer at Harbin Institute of Technology, Weihai, China. He was also an visiting student researcher at Redwood Center for Theoretical Neuroscience at University of California, Berkeley. His research interests focus on computer vision and pattern recognition, especially on moving objects detection, tracking and action recognition.

Hongxun Yao received the BS and MS degrees in computer science from the Harbin Shipbuilding Engineering Institute, Harbin, China, in 1987 and in 1990, respectively, and the PhD degree in computer science from Harbin Institute of Technology in 2003. Currently, she is a professor with the School of Computer Science and Technology, Harbin Institute of Technology. Her research interests include pattern recognition, multimedia technology, and human–computer interaction technology. She has published three books and over 100 scientific papers.

Xin Sun received the BS and MS degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2008 and 2010, respectively. Currently, she is pursuing the PhD degree. Her research interests focus on computer vision and pattern recognition, especially on moving objects detection and tracking.

Xiusheng Lu received the BS in computer science from the Harbin Institute of Technology, Harbin, China, in 2010. He is currently a master student in the same school.