RESEARCH

Efficient Visual Tracking via Low-Complexity Sparse Representation

Weizhi Lu[1*], Jinglin Zhang[2], Kidiyo Kpalma[1] and Joseph Ronsin[1]

[*]Correspondence:
weizhi.lu@insa-rennes.fr
[1] UMR 6164, IETR, INSA, Université Européenne de Bretagne (UEB), 35708, Rennes, France
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

Thanks to its good performance on object recognition, sparse representation has recently been widely studied in the area of visual object tracking. Up to now, little attention has been paid to the complexity of sparse representation, while most works are focused on the performance improvement. By reducing the computation load related to sparse representation hundreds of times, this paper proposes by far the most computationally efficient tracking approach based on sparse representation. The proposal simply consists of two stages of sparse representation, one is for object detection and the other for object validation. Experimentally, it achieves better performance than some state-of-the-art methods in both accuracy and speed.

Keywords: object tracking; sparse representation; low complexity

1 Introduction

Object tracking is a challenging task in computer vision due to the constant changes of object appearance and location. Sparse representation has recently been introduced in this area for its robustness in recognizing objects with high corruption [1]. Although related tracking works have been proposed with competitive performance, the application efficiency of sparse representation has not received enough attention. This paper is thus proposed to address this problem.

Sparse representation is mainly developed within the framework of particle filter, where it is used to measure the similarity between the particle and the dictionary with the representation error. Currently, this method still faces some challenges in terms of complexity and performance. To be specific, it should be noted that sparse representation has to be calculated for each particle, while the number of particles is often of the level of hundreds (e.g. 600 particles in [2–4]). Obviously it is a considerable computation cost, especially in the setting where each sparse solution is also computationally expensive. Precisely, to represent the particle with relatively little error, sparse representation usually requires a relatively large dictionary (with a trivial template) and relatively dense coefficients, which both will increase the solution complexity. Regarding the tracking performance, it is necessary to point out that sparse representation cannot resolve the problem of identity drift, if it is simply used to weight the particle. There are two major reasons. First, sparse representation cannot provide a reliable similarity measure due to the potential overfitting solution, which tends to introduce excessive nonzero coefficients to reduce the representation error. In practice, it seems difficult to completely avoid the overfitting problem, because the sparsity of sparse solution is usually unknown. Second, the similarity

threshold between object and background is hard to be determined with sparse representation, since the similarity level usually varies with the change of object appearance.

To address the aforementioned problems, this paper develops a simple but effective tracking-by-detection scheme by exploring the distribution of sparse coefficients instead of the sparse representation error for similarity measure. The proposed scheme consists of two-stage sparse representation. In the first step, the object of interest is detected by the largest sparse coefficient; in the second step, the detected object is validated with a binary classifier based on sparse representation, which outputs the decision in terms of the distribution of sparse coefficients.

Compared with the traditional framework of particle filter, the proposed scheme will not only significantly reduce the computation cost related to sparse representation, but also effectively avoid identity drift. The computation advantage mainly results from the following two facts. First, as stated before, the proposed scheme only involves two-step sparse representation; in contrast, the particle filter usually has to repeat sparse representation hundreds of times. Second, note that in the proposed scheme sparse representation is explored with the distribution of sparse coefficients rather than the representation error. Precisely, sparse representation is concerned only with the largest few coefficients which cover a great proportion of the energy of sparse solution. From the example in Figure 1, it can be observed that the distribution of the few largest coefficients is not sensitive to the representation error. This allows sparse representation to be carried out with a relatively small dictionary and a relatively high representation error, which also implies a relatively low solution cost. Regarding the robustness to identity drift, it mainly benefits from the binary classifier based on sparse representation, which can effectively identify the background sample even when the background model is not well trained.

The rest of this paper is organized as follows. In the next Section, the tracking works related to sparse representation are briefly reviewed. In Section III, a brief summary about sparse representation is presented. In Section IV, the proposed tracker with two-step sparse representation is described and analyzed. In Section V, extensive experiments are conducted with comparison to the state-of-the-art. Finally, a conclusion is given in Section VI.

2 Related Work

Extensive literature has been proposed on object tracking. Due to the limited writing space, we mainly review the tracking works related to sparse representation in terms of performance and complexity.

Sparse representation is introduced into the tracking mainly for improving the performance of recognition or feature selection. Mei and Ling [5] first explored sparse representation into an on-line tracking system, where a trivial template with high dimension is introduced to approximate noise and occlusion. Later, to improve the high-dimensional feature selection, Liu et al. [6] attempt to learn discriminative high-dimensional features using dynamic sparsity group. To reduce the sensitivity to background noise in the selected object area, Wang et al. [2] and Jia et al. [7] applied the sparse coding histogram based on local patches to describe objects. Zhong et al. [8] proposed a collaborative model that weights particles by combining

the confidences of local descriptor and holistic representation. Note that, the tracking methods described above are mainly focused on the performance improvement, while ignoring the complexity of implementation. In the traditional tracking framework of particle filter, the computation cost introduced by sparse representation usually cannot be ignored, because it has to be calculated for each particle. In this sense, it is of practical interests to reduce the complexity related to sparse representation. Mei et al. [3] proposed to discard insignificant samples by limiting their linear least square errors before resampling particles with more computationally expensive ℓ_1-regularization. In terms of compressed sensing theory, random projection is introduced to reduce the feature dimension in [9, 10]. Strictly speaking, the random projection based feature selection arises from random projection theory rather than compressed sensing theory [11]. In this paper, we also apply the similar method for feature selection. According to random projection theory, we will implement random projection with random matrices, rather than fixed matrices as applied in [9, 10]. By this means, the feature selection performance of random projection should be improved [12]. In [13], Bao et al. developed a fast sparse solution solver with the accelerated proximal gradient approach. However, their solver is sensitive to a parameter termed as Lipschitz constant, which is computational load during the template updating. Liu and Sun [14] attempted to weight each particle only with corresponding sparse coefficient such that sparse representation needs to be conducted only once. This method seems very attractive in complexity; however, it should be noted that, the magnitude of each coefficient in fact cannot be ensured 'proportional' to the similarity/correlation between the corresponding particle and the query object. Mathematically, with the principle of least square, we can derive that the exact 'proportion' exists only when the sub-dictionary corresponding to sparse coefficients is orthogonal. Obviously, this condition is hard to be satisfied by the realistic dictionaries. With the erroneous similarity measure, however, the method in [14] still presents relatively good performance. This is because empirically the particles corresponding to large coefficients tend to be similar to the query object. In this case, the selected particles with high weights are not inclined to change the attribute of particle filter, then the tracking performance will not be influenced. Besides the theoretical limitation, this method in [14] also holds a critical performance limitation: it is sensitive to identity drift, because the object out of the scene can hardly be detected only with the distribution of sparse coefficients. Zhang et al. [15] proposed to jointly represent particles by using multi-task learning to explore the interdependencies between particles. In addition, to detect an occlusion, the nonzero coefficients in the trivial template were used to locate occluded pixels in [3]. However, this method seems unreliable due to the potential overfitting solution. In particular, when the overfitting solution occurs as in Figure 1(c), all pixels are likely to be classified as occlusion, though in fact there is no occlusion. In this paper, to reduce the complexity of sparse representation, we also exploit the sparse coefficients instead of representation error for similarity measure. However, we successfully avoid the limitations mentioned above by developing a novel tracking scheme.

To account for the change of object appearance, almost all the trackers mentioned above explore an adaptive appearance model for the on-line sample updating [5]

or learning [16, 17]. It is known that the adaptive model is likely to lead to the identify drift, if the background sample cannot be effectively excluded. However, to effectively discriminate the background sample from the object is still a challenging task due to the change of object appearance. For instance, in practice it is hard to set a decision threshold between the object and background with representation error [4, 16, 18, 19]. To address this problem, it is better to introduce a binary classifier which involves a definitive decision threshold [20–23]. Thus in the proposed approach, we specially develop a binary classifier based on sparse representation. Compared with traditional binary classifiers, as will be shown later, the proposed classifier is competitive in both performance and complexity.

3 Sparse representation-based classification

This section briefly introduces sparse representation-based classification with the typical face recognition as an example. Let vector $y \in \mathbb{R}^{m\times 1}$ denote a test face, and matrix $\mathbf{D} = [D_{G_1}, D_{G_2}, \ldots, D_{G_N}] \in \mathbb{R}^{m\times n}$ be a dictionary consisting of N classes of labeled face vectors, where the i-th sub-matrix $D_{G_i} = [D_{i_1}, D_{i_2}, \ldots, D_{i_{n_i}}]$ includes n_i samples and $\sum_{i=1}^{N} n_i = n$. Then we ideally suppose that the test face can be approximated by a linear combination of few labeled face vectors, namely

$$y = \mathbf{D}\beta + \epsilon \tag{1}$$

where β is required to hold at most $k \ll n$ nonzero *positive* entries; and ϵ is the tolerated error. Subsequently, the feature vector $\mathbf{y}$ is viewed as close to the subspace of labeled samples corresponding to the nonzero entries of β. In other words, it can be identified as the class

$$\hat{i} = \underset{i}{\operatorname{argmax}}\{\delta_i(\beta)|1 \le i \le N\}, \tag{2}$$

where $\delta_i(\beta)$ is a function that sums the elements of β corresponding to D_{G_i}. The solution to k-sparse vector β can be simply derived with greedy algorithms of complexity $\mathcal{O}(mnk)$, such as OMP [24] or LARS [25]. Note that, to reduce the representation error, the dictionary $\mathbf{D}$ is often further concatenated with a trivial template consisting of two identity matrices $[I - I] \in R^{m\times 2m}$, thereby dramatically increasing the solution complexity [2, 3, 5–7, 9, 15].

It is worth mentioning a special case where the test face is novel and out of the database. In this case, the nonzero entries of β empirically incline to scatter among some different classes instead of focusing on a specific class. So the novel face can be detected with a threshold as

$$\max\{\delta_i(\beta)|1 \le i \le N\} < \gamma \sum_{j=1}^{n} \beta_j \tag{3}$$

where $0 < \gamma < 1$ is an empirical parameter. As will be detailed in section 4.3, the feature of detecting novel objects can be used to detect the outlier during the object tracking.

4 Proposed tracking scheme

The proposed tracking scheme mainly consists of two steps: object detection and object validation, as sketched in Figure 2. In this section, we first introduce the feature selection method, then detail the tracking scheme in subsections 4.2 and 4.3. The computation advantage of the proposed tracking scheme is analyzed in subsection 4.4.

4.1 Random-projection based feature selection

In tracking scenarios, it is usually hard to obtain ideal object features. The raw image and color histogram have been two kinds of popular features. In this paper, we use the random projection of raw image as a feature. Specifically, before performing sparse representation, we first project the vector of raw image to a relative low dimension with a random matrix [12]. This simple feature selection method has achieved competitive performance for face recognition [1]. In this case, the sparse representation of formula (1) can be reformulated as

$$\mathbf{R}y = \mathbf{RD}\beta + \epsilon \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{d\times m}$ is a random projection matrix, $d < m$. The random matrix $\mathbf{R}$ is commonly constructed with elements i.i.d drawn from the Gaussian distribution. Here, for simpler computation, we exploit a more sparse random matrix which holds exactly one nonzero entry being ± 1 equiprobably in each column. This kind of matrix has shown better feature selection performance than Gaussian matrices [11], and it also performs well in the following tracking work. Note that to obtain relatively reliable feature selection, for each given y, random projection usually requires to be carried out several times and then to consider the average result [12]. In this process, the matrix $\mathbf{R}$ is random generated. More precisely, in our approach the random projection together with sparse representation will be repeated five times for each given y. Then the average value of five sparse solutions β is used to make a decision for y.

Despite its low implementation complexity, random projection clearly is not the best feature selection tool in terms of performance. However, considering the variation of object appearance, it is reasonable to argue that, the feature comparison based on the sum of few randomly selected pixels is probably more robust than the conventional pixel-wise comparison. This also explains why random projection can present satisfactory recognition performance in the proposed tracker.

4.2 Object detection

In this section, the object detection is simply implemented by approximating the known object with local patches detected from current frame. To be specific, as illustrated in Figure 3, let y be a known/query object extracted from the former adjacent frame, and $\mathbf{D}$ consisting of overlapping local patches collected from current frame. Here the local patches are extracted from a rectangle region of interest with a sliding window of fixed size. Suppose y can be sparsely approximated with the dictionary $\mathbf{D}$. Then the candidate object can be located with the element of $\mathbf{D}$ which corresponds to the largest component of sparse solution β, as illustrated in Figure 3.

Algorithm 1 Object detection

Definitions: Let $\mathbf{Y} = [Y_s\ Y_d]$ be a set of known object samples from former frames, where Y_s represents the subset of static samples from the initial frames and Y_d denote the subset of dynamic samples from the recent frames, and $\mathbf{D}$ be the dictionary consisting of overlapping local patches extracted from some regions of current frame, with a fixed space layout. And then suppose $\{y^j : 1 \leq j \leq N_c\}$ is a set of N_c query samples randomly selected from Y_s or Y_d.

1. The average value of sparse coefficient $\bar{\beta} = \frac{1}{N_c}\sum_{j=1}^{N_c}\beta^j$ is derived with respect to $\mathbf{R}y^j = \mathbf{RD}\beta^j + \epsilon$, where $\mathbf{R}$ is the random projection matrix for feature selection.
2. The candidate object is located by the local patch with index $\hat{i} = \text{argmax}_i\{\bar{\beta}_i\}$, where $\bar{\beta}_i$ indicates the i-th component of vector $\bar{\beta}$.

It is clear that the detection performance depends heavily on the reliability of query object y extracted from the former frame. Here, we define a special template $\mathbf{Y} = [Y_s\ Y_d]$ to model the query object y. As it appears in state-of-the-art [4, 8], Y_s denotes a static appearance model consisting of ground-truth manually or automatically detected from the first frame as well as its perturbations with small Gaussian noises, and Y_d represents a dynamic appearance model collecting some object samples extracted from recent frames. To account for the object appearance change, in this paper a set of query samples, rather than one, are randomly selected from the two models above. The average sparse solution of the query objects selected above is used to determine the detection result. Note that, to avoid false detection, it is suggested to collect more query samples from the static model Y_s than from the dynamic model Y_d. The local patches of the dictionary $\mathbf{D}$ are collected with a sliding window of the size of the initialized object. Considering the continuity of the object movement, the searching region of the sliding window allows to be restricted to a relatively small area, e.g. twice or three times the object size. If the object is lost, the searching region can be temporarily expanded. For better understanding, the object detection flow is sketched in Algorithm 1.

It is necessary to point out that the detection corresponding to the largest coefficient is not always reliable or correct. An obvious evidence is that, there would remain an 'object area' defined by the largest coefficient, even though the object has been occluded or out of the image. To avoid such kind of false detection, we have to introduce a binary classifier to further validate the detection, as detailed in the sequel.

4.3 Object validation and template updating

In this section, a binary classifier based on sparse representation is developed for discriminating the object from the background. Here sparse representation is adopted for the following four reasons:

- It is computationally competitive, since it only involves simple operations of matrix-vector product.
- The decision can be easily derived in terms of the distribution of sparse coefficients.
- Compared with traditional binary classifiers, it has an exclusive advantage: it can detect the outliers which is not included in current background model,

because in this case the sparse coefficients tend to scatter rather than focus [1]. This property has been verified in our recent work on multi-object tracking, in which the novel object is detected as an outlier according to the scattering sparse coefficients [26]. This implies that we can detect the background sample, even when the background model is not robust or large. Then the computation and storage loads on background modelling can be significantly reduced.

- In practice, the discrimination between the object and the background seems to be a multi-class classification problem rather than a binary classification problem, since the complex and dynamic background usually involves kinds of feature subspaces, some of which might be similar to the object feature. In this case, the two opposite half-spaces trained by traditional binary classifiers, like SVM [26], probably overlap with each other, thereby deteriorating the classification performance. In contrast, sparse representation is robust to this problem, because it partially explores the similarity between individual samples rather than directly dividing the sample space into two parts [1].

The proposed binary classifier can be briefly described as follows. Suppose y_c is the candidate object detected from the former step, which can be sparsely approximated by a dictionary $\mathbf{Z}$, namely

$$\mathbf{R}y_c = \mathbf{RZ}\beta + \epsilon. \tag{5}$$

Here $\mathbf{Z} = [Z_{G_p}\ Z_{G_n}]$ consists of two parts, which represent the subset of positive samples and the subset of negative samples, respectively; and the two subscripts G_p and G_n are the subsets of column indexes. Then we can determine the attribute of the candidate object with the distribution of sparse coefficients. Precisely, as illustrated in Figure 4(a)(b), the candidate object will be regarded as positive if the sparse coefficients mainly focus on the part of positive samples; otherwise, the decision is negative.

The parameters of the proposed classifier are further detailed as follows. The positive samples Z_{G_p} come from the aforementioned static and dynamic appearance models $\mathbf{Y}$, and the negative samples Z_{G_n} are collected by a sliding overlapping window from the neighborhood of tracked object, where partial object region is included as opposed to relatively complete object region in positive samples. Correspondingly, the sparse solution β is also divided into two parts: $\beta = [\beta_{G_p}\ \beta_{G_n}]$. Note that, the classifier is not sensitive to the representation error, and so β can be very sparse, e.g., it holds at most 10 nonzero entries in our experiments. In terms of the distribution of sparse coefficients, we propose *two* rules to define the *positive* output. *One* is that the largest coefficient of sparse solution β corresponds to a positive sample of Z_{G_p}; namely, $\text{argmax}_i\{\beta_i\} \in G_p$. And *the other* is that the sparse coefficients corresponding to positive samples, β_{G_p}, take higher energy than the sparse coefficients corresponding to negative samples, β_{G_n}; that is, $||\beta_{G_p}||_1/||\beta||_1 > 0.5$. Empirically, the latter criterion is more strict than the former, since it measures the similarity between the candidate object and the whole positive subspace, instead of the individual positive samples. In this paper, to present a relatively fluent tracking trajectory, the detection is positively labeled, when either of the two criterions above is satisfied. But for the template updating, we only apply the second criterion with a stricter threshold, i.e. $||\beta_{G_p}||_1/||\beta||_1 > 0.8$, which provides more reliable

Algorithm 2 Object validation and template updating

Definitions: Let y_c denote the candidate object derived in Algorithm 1, and $\mathbf{Z} = [Z_{G_p}\ Z_{G_n}]$ be the dictionary consisting of object samples and background samples, where the G_p and G_n denote the index sets corresponding to above two classes of samples. Note that $Z_{G_p} \subseteq \mathbf{Y}$ of Algorithm 1.

1. The average of sparse solutions $\bar{\beta} = [\bar{\beta}_{G_p}\ \bar{\beta}_{G_n}] = \frac{1}{N_r}\sum_{j=1}^{N_r}\beta^j$ is derived with $\mathbf{R}_j y_c = \mathbf{R}_j\mathbf{Z}\beta^j + \epsilon$, where $\mathbf{R}_j$ denotes the j-th random projection with $j \in \{1, 2, ..., N_r\}$, and correspondingly β^j is the sparse solution. Here N_r is the frequency of repeating random projection.
2. The candidate object is labeled, if $\text{argmax}_i\{\bar{\beta}_i\} \in G_p$ and $0.5 < ||\bar{\beta}_{G_p}||_1/||\bar{\beta}||_1 < 0.8$, where $\bar{\beta}_i$ denotes the i-th element of vector $\bar{\beta}$.
3. The candidate object is labeled and updated for dynamic object model Y_d, and its neighboring background patches are updated for Z_{G_n}, if $\text{argmax}_i\{\bar{\beta}_i\} \in G_p$ and $||\bar{\beta}_{G_p}||_1/||\bar{\beta}||_1 > 0.8$.
4. If both steps 2 and 3 cannot be performed, the object is assumed to keep still or move with a constant velocity.
5. If step 4 is active in a few consecutive frames, object detection in Algorithm 1 is performed again in a larger region.

features and then prevents the template from identity drift. In practice, the threshold value needs to be tuned empirically. Recall that random projection needs to be carried out several times to achieve better feature selection performance for the unique candidate object [12]. In our experiments, the random projection together with sparse representation is repeated five times, and then the average value of five sparse solutions β is used for the final decision.

It is necessary to emphasize that the proposed classifier holds an exclusive advantage: it can detect the outlier. Typical outliers include the dynamic background samples and the sudden and great changes of object appearance, which usually cannot be well described with current background model. In this case, as shown in Figure 4(c), the sparse coefficients incline to scatter among the positive and negative subspaces rather than focusing on one of them, namely $||\beta_{G_p}||_1/||\beta||_1 \approx 0.5$. Then the outliers can be easily excluded from the template updating, if a relatively strict threshold is adopted, e.g., $||\beta_{G_p}||_1/||\beta||_1 > 0.8$. This advantage allows us to build a background model of relatively few samples, since the classifier is not very sensitive to the robustness of the background model. Finally, it is necessary to discuss the case where the object is lost. In this case, the object will be searched again within a larger region. If the search fails, the object will be assumed to move with a constant velocity or keep still in the following few frames. The whole flow of this part is summarized in Algorithm 2.

4.4 Computation cost related to sparse representation

There are two major factors affecting the computation cost related to sparse representation: 1) the repetition number of sparse representation and 2) the complexity of sparse solution. Compared with the traditional tracking framework of particle filter, the proposed approach presents obvious advantages on these two factors, as detailed below.

- The proposed scheme only involves two-step sparse representation, in which sparse representation allows to be performed only twice, if the feature is robust. To obtain relatively reliable features, in our experiments sparse representation along with random projection is repeated dozens of times, namely $(N_c+1)N_r$ times. In contrast, the traditional framework of particle filter has to be carried out for each particle, such that the repetition times of sparse representation is usually of the level of hundreds, e.g. about 600 times in [2–4].
- Recall that the solution complexity is roughly proportional to the dictionary size $m \times n$ and the sparsity k. In the traditional methods, the dictionary size is usually large, since to reduce representation error, it has to involve a high-dimensional trivial template $[I-I]$ with the size of object feature. The sparsity k cannot be restricted, unless the representation error is small enough. In contrast, the proposed approach is not sensitive to the representation error. Thus, in the paper we significantly reduce the dictionary column size n by excluding the trivial template, while restricting the sparsity k to a relatively low value, e.g. $k = 10$ in our experiments. Furthermore, the dictionary row size m is also drastically reduced with random projection.

Recently, some trackers based on sparse representation have been proposed with 'real-time' performance by reducing the complexity of sparse solution [9] [13]. However, these trackers cannot reduce the repetition number of sparse representation due to their framework of particle filter. Thus, their computational gain is still limited compared with our approach. For a better understanding, here we analyze two typical real-time trackers: RTCST [9] and APGL1 [9]. For the tracker RTCST, compressed sensing theory is exploited to reduce the feature dimension with linear projection, thereby reducing the complexity of sparse solution. The strategy is also adopted in our approach with random projection theory [11]. So compared with our approach, the tracker RTCST has no computational advantage. Furthermore, it is worth noting that the linear projection based feature selection is based on random projection theory rather than compressed sensing theory [11]. The tracker APGL1 is developed by exploring the accelerated proximal gradient (APG) approach for sparse solution. The APG approach seems to be computationally attractive, since it does not requires the operation of matrix inversion, which is of complexity $\mathcal{O}(k^3)$ and often involved in current solution algorithms. However, it should be noted that the convergence performance of the APG approach is sensitive to a parameter termed the Lipschitz constant, which needs to be evaluated with the largest singular value of the dictionary $\mathbf{D}$. This implies that the singular value of the dictionary has to be calculated for each dictionary updating, while the solution of singular value holds a relatively high complexity $\mathcal{O}(n^3)$. Then we can say that the updating of Lipschitz constant will drastically degrade the computational advantage of the APG approach, especially in the complex scene where the dictionary requires to be frequently updated.

5 Experiments

We evaluate the proposed tracker on ten challenging videos, among which eight are publicly available[1] and two are produced by ourselves. Their attributes are

[1] `http://www.cvg.rdg.ac.uk/PETS2009/`; `http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html`; `http://faculty.ucmerced.edu/mhyang/pubs.html`

summarized in Table 1. They are exploited here especially to validate the robustness of the proposed approach on identity preservation. To present more significant challenges of occlusion than the public available videos, we specially produce two videos, termed 'paper' and 'face_hand', in which the occlusions share the similar feature with the targets, and last a long-time. For the video results, please see the link below[2].

For comparison, we perform four known trackers: IVT tracker [16], L1 tracker [5], PLS tracker [4] and SCM tracker [8]. These four trackers all explore an adaptive appearance model. The first two trackers are mainly focused on the updating of dynamic appearances, while the latter two trackers further introduce the static model to prevent the identity drift. Note that the trackers L1 and SCM are both developed based on sparse representation. L1 is the first tracker that explores sparse representation, and to the best of our knowledge, SCM is currently known the best tracker based on sparse representation [27]. For fair comparison, the four trackers are all implemented with their original codes, and the tracked object is initialized with same position. Regarding the parameters tuning, we use the default parameters of L1 and PLS. Since the trackers IVT and SCM have provided some options of parameters for some popular videos, we adopt their default parameters for the videos they have evaluated, and select proper parameter options for other videos, e.g., using their parameters for 'head' and 'pedestrian' to track 'head' and 'pedestrian' in other videos. Recall that the proposed tracker cannot cope with scales. In contrast, the other four trackers all exploit the technique of affine transform. Empirically, some trackers are probably sensitive to the initialization. For fair and comprehensive performance evaluation, we exploit two initialization methods: one-pass evaluation (OPE) and spatial robustness evaluation (SRE) [27]. The OPE method simply initializes the tracker with the ground-truth. The SRE method sets the initialization region by scaling the ground truth with five ratios 0.8, 0.9, 1, 1.1 and 1.2, and then the average tracking performance is considered.

The parameters related to the two-step sparse representation are detailed as follows. In the step of object detection, 10 query samples are collected from the static model Y_s and 5 query samples collected from the dynamic model Y_d. The step length of the sliding window is around 4 pixels. Both the object retrieval region and background sampling region are not more than three times the object size. In the step of object validation, the classifier consists of 50 positive samples and 100 negative samples. The positive detection cannot be used for model updating unless $||\beta_{G_p}||_1/||\beta||_1 > 0.8$. In the two steps above, the sparse representation based on random projection is repeated 5 times for each instance. The random projection matrix is of size $(d = 200, m = 32 \times 32)$. It implies that the object is extracted and represented with a vector of size 32×32, which is further reduced to the dimension of 200 by random projection before performing sparse representation. The upper bound of sparsity k is set to 10 during the sparse solution.

5.1 Computational efficiency

The proposed tracker is mainly implemented with MATLAB code, except for the sparse solution which is derived with the LARS algorithm of SPAMS package [28].

[2] https://sites.google.com/site/wzlusd/home/tracking-project

Here we compare its speed with other three popular trackers based on sparse representation: L1 [5], SCM [8] and APGL1 [13]. Also, these trackers are mainly implemented with MATLAB code, except for some computationally expensive parts. Precisely, similarly to the proposed tracker, the trackers L1 and SCM also explore the SPAMS package for sparse solution. So the comparison with them is fair. The real-time APGL1 tracker implements the technique of affine transform with C code. For a fair comparison, all the trackers are tested with the same video 'girl' on a PC with intel i7 CPU (2.67GHZ). The result is shown in Table 2, where the speed is evaluated with respect to varying feature size. Recall that the feature size means the length of the feature vector which is extracted from the image to represent the object. It corresponds to the column size m of the random projection matrix $\mathbf{R}$ with size $d \times m$. The increasing feature size will increase the computation load of sparse solution, thereby reducing the speed of the tracker. Note that, to demonstrate the advantage of random projection, our approach is evaluated in terms of two cases: 'Ours_1' and 'Ours_2'. Precisely, for the case of Ours_1, the random matrix of size $d \times m$ is applied without introducing dimension reduction by setting $d = m$; for the case of Ours_2, with random projection, the feature vector of length m is projected to the low dimension $d = 50 < m$. Theoretically, the case of Ours_2 should run faster than the case of Ours_1 due to its dimension reduction, which is validated with the results of Table 2. This implies that the random projection can indeed significantly improve the tracker's speed, as the projection dimension d decreases. Note that the smaller d tends to lead to the worse feature selection. Thus the value of d cannot be too small in practice. Compared with other three trackers, as it is expected, both Ours_1 and Ours_2 present much higher speed. For instance, the frame rate is improved dozens of times in Ours_2. From Table 2, it can be observed that their speed advantages become more obvious as the feature size increases. This is because the increasing computation of sparse solution gradually turns into the tracker's time bottleneck, and then our low-complexity on sparse representation is fully displayed. In summary, the low complexity of the proposed tracker is fully verified by its obvious speed advantage over state-of-the-art.

5.2 Quantitative evaluation

We measure the tracking accuracy of aforementioned trackers based on the center location error and the overlap rate [29]. The center location error is the Euclidean distance between the central points of tracked object R_T and the corresponding manually labeled ground-truth R_G. The OPE center location error plots on all test sequences are shown in Figure 5. The average location errors are given in Table 3. The overlap rate, defined as $area(R_T \cap R_G)/area(R_T \cup R_G)$, evaluates the success rate, which is shown in Table 4. From Figure 5, it can be observed that the proposed approach achieves relatively persistent and stable tracking, and obtains better overall performance than other four trackers on the ten videos. However, its performance advantage is not sufficiently mirrored in Table 4, since we simply exploit a fixed-size rectangle to represent an object with varying size. Precisely, even if we have successfully captured the object, like in the sequence *david*, the overlap rate is still small, because our tracking window is often larger or smaller than the ground-truth.

By comparing the results of OPE and SRE in each video of Table 3, we can see that the performance of the proposed tracker is relatively stable in the two cases. This implies that the proposed tracker is robust to the scale of initialization region. In fact, it presents poor performance only in the video 'PETS09_s2l1', where it fails in the scaling cases of SRE. In contrast, the other four trackers seem sensitive to the initialization: this might be explained by the following fact. These trackers all exploit the technique of affine transform, which inclines to gradually converge to the local part of the target when the feature is not robust. From Table 4, it can be observed that the the SRE result is a little worse than the OPE result in each video, although as shown in Table 3, the proposed tracker in fact presents comparable performance in these two cases. As explained before, this is because the SRE method introduces a relatively large difference between the initialized tracking window and the ground-truth in the first frame.

5.3 Qualitative evaluation

In addition to quantitative evaluation, the qualitative evaluation, as illustrated in Figure 6, is presented in terms of the following several typical challenges.

Occlusion: The object occlusion has been the major challenge of on-line visual object tracking, which probably leads to the false object detection and gradually drift the identity of object model. However, in the proposed approach the binary classifier based on sparse representation can effectively detect and prevent the occlusion from the updating of object model. In practice, the proposed approach is robust to object occlusion. To highlight the advantage of the proposal, we specially produce two challenging videos against long-time complete occlusions: sequences *face_hand* and *paper*. In the sequence *face_hand*, the target face is completely occluded with hands for a long period. In the sequence *paper*, the target paper is completely occluded twice by another similar paper. In addition, the partial or complete occlusion cases can also be observed in the sequences *PETS09_s2l1*, *Tudcrossing*, *face_man*, *face_woman* and *girl*.

The proposed approach performs well on the videos mentioned above. In contrast, the other four trackers fail, when the long-time complete occlusion occurs or the occlusion shares similar feature with the target. For instance, in the sequence *face_hand*, IVT, L1 and PLS early drift to the background when a short-time occlusion occurs, and SCM finally drifts to one hand which covers the face for a long period. In the sequence *paper*, the four trackers all drift to the occlusion or background. It is interesting to note that, SCM is robust to short-time occlusion due to the application of static object model. Nevertheless, it remains sensitive to the long-time occlusion, as demonstrated in the sequence *face_hand*. This implies that SCM cannot effectively detect the collusion, which finally modifies the attribute of the dynamic object model by the accumulation of false samples.

Motion & Blur: The fast or abrupt motion has been a great challenge for the traditional framework of particle filter, whose motion estimation parameters are usually continuous. However, this problem can be easily addressed within the proposed tracking-by-detection scheme by expanding the object retrieval region. It is known that the blur caused by fast motion is unfavorable for object recognition. However, the fluent tracking results in sequences *animal* and *jumping* validate that

the proposed approach works well in this case. It indicates that the random projection of raw image is robust to the blur. By exploring the sparse coding histogram as feature, SCM also performs well in this case. In contrast, the remaining three methods perform relatively worse. They all drift from the target in the sequence *jumping*.

Scale & Rotation: There are drastic scale changes, and in-plane or out-of-plane rotations in the two sequences *david* and *girl*. They pose great challenges to the proposed approach which only holds a fixed-sized tracking window. In this case, the object detection of the proposed approach is usually false. However, the false detection can be effectively identified with object validation. This will help the proposed tracker effectively avoid the identity drift caused by scale or rotation. So in the sequences *david* and *girl*, the proposed approach successfully recaptures the object after severe scalings or rotations. In contrast, the other four methods incline to lose the target for ever in the presence of severe scale changes or rotations, e.g. the out-of-plane rotation in the sequence *girl*.

Illumination: In theory, the proposed approach should not be sensitive to the illumination change, since the feature vector collected by random projection allows to be linearly scaled during the sparse solution. In practice, the proposed approach performs well together with other four methods. For instance, in the sequence *david*, the five methods all successfully track the object walking from the dark to the light in the early few frames.

Overall performance: The proposed approach shows better overall performance than others due to the robustness of sparse representation on both object detection and validation. The two trackers SCM and PLS both explore a static object model to identify the object, while they cannot prevent the false detection from updating the dynamic object model. So they perform worse than the proposed tracker in our experiments. Note that SCM obviously outperforms PLS. This can be explained by the fact that SCM explores both static and dynamic features to weight particles, while PLS only adopts the dynamic feature. The remaining two trackers IVT and L1 cannot cope with severe appearance changes, since the ground-truth is not preserved in their template updating.

6 Conclusion

This paper has proposed an efficient tracking-by-detection scheme based on two-stage sparse representation. In order to evaluate the proposed approach, extensive experiments are conducted on ten benchmark videos comprising various challenges like light change, fast motion, scale and rotation, partial occlusion, and complete occlusion. Compared with traditional trackers based on sparse representation, the proposed tracker presents obvious advantages on both accuracy and complexity. Specifically, it significantly reduces the computation cost related to sparse representation, thereby presenting much higher speed than state-of-the-art. Thanks to its robustness to identity drift, it also achieves better tracking performance than state-of-the-art especially in the presence of severe occlusions.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
Text for this section ...

Acknowledgements
Text for this section ...

Author details
[1] UMR 6164, IETR, INSA, Université Européenne de Bretagne (UEB), 35708, Rennes, France. [2] School of Atmospheric Science, Nanjing University of Information Science & Technology, 210044, Nanjing, China.

References

1. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(2), 210–227 (2009)
2. Wang, Q., Chen, F., Xu, W., Yang, M.-H.: Online discriminative object tracking with local sparse representation. In: IEEE Workshop on Application of Computer Vision (WACV) (2012)
3. Mei, X., Ling, H., Wu, Y., Blasch, E., Bai, L.: Minimum error bounded efficient l_1 tracker with occlusion detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
4. Wang, Q., Chen, F., Xu, W., Yang, M.-H.: Object tracking via partial least squares analysis. IEEE Transactions on Image Processing **21**(10), 4454–4465 (2012)
5. Mei, X., Ling, H.: Robust visual tracking using l_1 minimization. In: IEEE International Conference on Computer Vision (2009)
6. Liu, B., Yang, L., Huang, J., Meer, P., Gong, L., Kulikowski, C.: Robust and fast collaborative tracking with two stage sparse optimization. In: European Conference on Computer Vision (ECCV) (2010)
7. Jia, X., Lu, H., Yang, M.-H.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
8. Zhong, W., Lu, H., Yang, M.-H.: Robust object tracking via sparsity-based collaborative model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
9. Li, H., Shen, C., Shi, Q.: Real-time visual tracking using compressive sensing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
10. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: Proceedings of the 12th European Conference on Computer Vision. ECCV'12, pp. 864–877 (2012)
11. Lu, W., Li, W., Kpalma, K., Ronsin, J.: Sparse Matrix based Random Projection for classification. arXiv:1312.3522 (2014)
12. Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: A cluster ensemble approach. in Proceedings of the 20th International Conference on Machine Learning (2003)
13. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust L_1 tracker using accelerated proximal gradient approach. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1830–1837 (2012)
14. Liu, H., Sun, F.: Visual tracking using sparsity induced similarity. In: IEEE International Conference on Pattern Recognition (ICPR), pp. 1702–1705 (2010)
15. Zhang, T., Ghanem, B., Liu, S., Ahuja, N.: Robust visual tracking via multi-task sparse learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
16. Ross, D., Lim, J., Yang, R.-S.L.M.-H.: Incremental learning for robust visual tracking. International Jounal of Computer Vision **77**(1), 125–141 (2008)
17. Wang, D., Lu, H., Yang, M.-H.: Online object tracking with sparse prototypes. IEEE Transactions on Image Processing **22**(1), 314–325 (2013)
18. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
19. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
20. Babenko, B., Yang, M.-H., Belongie, S.: Visual tracking with online multiple instance learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
21. Avidan, S.: Ensemble tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
22. Grabner, H., Bischof, H.: On-line boosting and vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
23. Kalal, Z., Matas, J., Mikolajczyk, K.: P-N learning: Bootstrapping binary classifiers by structural constraints. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
24. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, pp. 40–441 (1993)
25. Efron, B., Hastie, T., Tibshirani, R.: Least angle regression. Annals of Statistics **32**, 407–499 (2004)
26. Lu, W., Bai, C., Kpalma, K., Ronsin, J.: Multi-object tracking using sparse representation. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2312–2316 (2013)
27. Wu, Y., Lim, J., Yang, M.-H.: Online object tracking: A benchmark. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2411–2418 (2013)
28. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research **11**, 19–60 (2010)
29. Everingham, M., Gool, L.V., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Jounal of Computer Vision **88**(2), 303–338 (2010)

Figures
Tables

Figure 1: An object vector of size 300×1 (left in (a)) is sparsely approximated by a dictionary of ten objects (right in (a)) with sparse solution in (b), or by a dictionary combining the ten objects and a trivial template $[I - I]$ with sparse solution in (c). Here I is an identity matrix of size 300×300. This means that the sparse solution is of length equal to 10 in (b) and 310 in (c). The sparse solution in (b) has 4 nonzero entries and the representation error is about 10^{-1}. In contrast, the solution in (c) holds 300 nonzero coefficients, thereby lowering the representation error to 10^{-3}. For clarity, the first ten coefficients in (c) corresponding to the ten objects above are detailed in (d). The similarity between (b) and (d) implies that the distribution of the largest few coefficients is not sensitive to the representation error.

Figure 2: The block diagram of the proposed tracking scheme.

Figure 3: The labeled object from the former frame can be sparsely approximated by a dictionary consisting of overlapped local patches in some area of current frame. The local patch corresponding to the largest coefficient is prone to indicate the position of the candidate object.

Figure 4: The sparse solutions of binary classifier with input being an object sample in (a), a background sample in (b) and an outlier in (c).

Figure 5: The OPE center location errors of five trackers on all test sequences. The vertical axis indicates the location error and the horizontal axis is the frame index.

Figure 6: Tracking examples of five methods on ten challenging videos

Table 1: The attributes of the ten tested videos on light change (LC), fast motion (FM), scale and rotation (SR), partial occlusion (PO), and complete occlusion (CO).

Video clip	LC	FM	SR	PO	CO
Face_hand				✓	✓
Paper				✓	✓
PETS09_s2l1					✓
Tudcrossing					✓
Face_man			✓	✓	
Face_woman				✓	
Animal		✓			
Jumping		✓			
David	✓		✓		
Girl	✓		✓		✓

Table 2: The implementation speed (frames per second) of the four trackers based on sparse representation with varying feature size. The best and second best results are labeled in red and blue, respectively. The $\star$ indicates that the software cannot run with the corresponding feature size.

Feature size	10×10	20×20	30×30	40×40	50×50
L1 [5]	2.0172	0.3263	0.0166	0.0039	0.0019
SCM [8]	$\star$	1.9025	0.9760	0.5648	0.3512
APGL1 [13]	18.0518	6.7513	0.7084	0.3160	0.1329
Ours_1	18.1232	8.5850	2.5802	0.9241	0.4040
Ours_2	18.3729	15.7857	13.0183	11.0276	7.8666

Table 3: Average center location errors (in pixel). The best and second best results are shown in red and blue fonts, respectively.

	Face_hand		Paper		PETS09_s2l1		Tudcrossing		Face_man		Face_woman		Animal		Jumping		David		Girl	
	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE
IVT	57.80	51.23	38.77	38.64	51.26	42.17	29.11	22.30	7.41	18.01	39.99	30.41	10.02	21.27	15.25	15.10	5.49	6.74	36.50	30.22
L1	43.98	45.12	70.42	55.28	45.04	45.22	11.49	11.17	28.96	23.47	24.06	20.30	81.02	33.08	55.68	52.03	58.32	57.00	24.53	45.13
PLS	62.08	59.30	34.77	39.11	42.98	44.71	43.11	37.60	9.09	17.10	22.57	20.00	91.94	32.90	66.93	72.44	77.73	78.17	39.51	54.00
SCM	23.82	18.17	40.89	37.01	32.12	23.00	4.36	6.35	3.06	6.81	4.68	4.58	25.75	14.22	3.73	3.66	47.20	40.90	110.49	51.18
Ours	4.08	6.22	4.45	4.12	8.47	40.06	4.23	6.29	9.74	13.40	10.45	9.79	6.39	9.73	3.77	4.15	8.65	10.21	22.68	26.43

Table 4: Average overlap rates between the tracked region and the ground-truth. The best and second best results are shown in red and blue fonts, respectively.

	Face_hand		Paper		PETS09_s2l1		Tudcrossing		Face_man		Face_woman		Animal		Jumping		David		Girl	
	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE	OPE	SRE
IVT	0.40	0.37	0.48	0.40	0.20	0.21	0.15	0.15	0.56	0.32	0.46	0.45	0.74	0.58	0.52	0.45	0.64	0.63	0.37	0.38
L1	0.35	0.30	0.12	0.16	0.39	0.35	0.54	0.51	0.36	0.32	0.54	0.54	0.23	0.37	0.11	0.14	0.26	0.24	0.48	0.25
PLS	0.35	0.29	0.44	0.39	0.38	0.35	0.09	0.11	0.52	0.34	0.43	0.40	0.21	0.37	0.07	0.07	0.25	0.25	0.39	0.27
SCM	0.70	0.61	0.54	0.48	0.38	0.47	0.69	0.70	0.67	0.61	0.82	0.78	0.57	0.62	0.80	0.80	0.37	0.36	0.13	0.28
Ours	0.93	0.77	0.82	0.73	0.64	0.30	0.74	0.69	0.56	0.47	0.67	0.59	0.84	0.69	0.82	0.73	0.39	0.35	0.51	0.48