Low-Rank Sparse Learning for Robust Visual Tracking

Tianzhu Zhang[1], Bernard Ghanem[2,1], Si Liu[3], and Narendra Ahuja[1,4]

[1] Advanced Digital Sciences Center of UIUC, Singapore
[2] King Abdullah University of Science and Technology, Saudi Arabia
[3] ECE Department, National University of Singapore
[4] University of Illinois at Urbana-Champaign, Urbana, IL USA
tzzhang10@gmail.com, bernard.ghanem@kaust.edu.sa, dcslius@nus.edu.sg, n-ahuja@illinois.edu

Abstract. In this paper, we propose a new particle-filter based tracking algorithm that exploits the relationship between particles (candidate targets). By representing particles as sparse linear combinations of dictionary templates, this algorithm capitalizes on the inherent low-rank structure of particle representations that are learned jointly. As such, it casts the tracking problem as a low-rank matrix learning problem. This low-rank sparse tracker (LRST) has a number of attractive properties. **(1)** Since LRST adaptively updates dictionary templates, it can handle significant changes in appearance due to variations in illumination, pose, scale, etc. **(2)** The linear representation in LRST explicitly incorporates background templates in the dictionary and a sparse error term, which enables LRST to address the tracking drift problem and to be robust against occlusion respectively. **(3)** LRST is computationally attractive, since the low-rank learning problem can be efficiently solved as a sequence of closed form update operations, which yield a time complexity that is linear in the number of particles and the template size. We evaluate the performance of LRST by applying it to a set of challenging video sequences and comparing it to 6 popular tracking methods. Our experiments show that by representing particles jointly, LRST not only outperforms the state-of-the-art in tracking accuracy but also significantly improves the time complexity of methods that use a similar sparse linear representation model for particles [1].

1 Introduction

Visual tracking is a well-known problem in computer vision with many applications such as surveillance, robotics, human computer interaction, etc. It is challenging to design a robust tracking algorithm due to the presence of occlusion, background clutter, varying viewpoints, and illumination and scale changes. Over the years, many tracking algorithms have been proposed to deal with these difficulties. To survey many of these algorithms, we refer the reader to [2].

Recently, several approaches have successfully applied ℓ_1 minimization for robust visual tracking [1,3]. In these methods, visual tracking exploits the sparse

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 470–484, 2012.

representation of the target candidate using a dictionary of templates that can be updated progressively. An important advantage of using sparse representation is its robustness to a wide range of image corruptions, especially moderate occlusions. These methods demonstrates promising robustness compared with many existing trackers on well-known video sequences, but at a computational cost dominated by ℓ_1 minimization. Furthermore, since the target states are estimated in a particle filter framework, the computational cost grows linearly with the number of particle samples used. This large computational bottleneck precludes the use of these robust trackers in real-time scenarios. Moreover, this family of methods learns the sparse representations of particles separately and thus ignores the relationships among them, which ultimately constrain their representation. For example, most particles are densely sampled at a small distance around the target, so their appearances, and thus their sparse representations, are expected to be similar.

Inspired by the above work, we propose a computationally efficient tracking method that capitalizes on sparse and low-rank representation in a particle filter framework. This method is denoted as the Low-Rank Sparse Tracker (LRST). In visual object tracking, the next state of the tracked object is decided based on its current state and observation. To obtain particle samples, we adopt the independent and identically distributed (i.i.d.) sampling strategy in [4] based on a zero-mean Gaussian model centered around the current object state. The next object state is chosen to be the particle that has the highest similarity with a dictionary of target object templates. As such, an accurate and joint representation of particle samples w.r.t. this dictionary is crucial. To devise this representation, we make the following observations. **(a)** The best particle sample should have the most similar representation with the target object templates in the dictionary. **(b)** Since particles are densely sampled around the current target state, the appearances of many of these particles and, in turn, their representations w.r.t to the dictionary are expected to be similar. So, we observe a correlation among particle representations and *not* their i.i.d. sampled states. This correlation should be exploited instead of being ignored as in other sparse coding trackers [1,3,5]. **(c)** Occlusion and noise can significantly impact tracking performance. To alleviate their effect, representation error should be explicitly incorporated in the tracking process. **(d)** During tracking, a particle sample should be represented using a dictionary of templates composed of both object and background templates, which are updated progressively. This emphasizes the importance of representing what a target is *and* is not. Discriminating the target from the background adds another layer of robustness against possible tracker drift. Generally, a "good" target candidate is effectively represented by the object and not the background templates, thus, leading to a sparse representation. The converse is true for a "bad" target candidate.

Motivated by these considerations, we propose a novel formulation to address the robust object tracking problem. Here, particle samples in the current frame are represented as linear combinations $\mathbf{Z}$ of object and background templates that define a dictionary $\mathbf{D}$. We require that $\mathbf{Z}$ be sparse (only a few templates are

required to represent it well) and low-rank (most of particles are expected to have similar representations). To incorporate occlusion and noise handling, we allow for sparse error **E** to contaminate **Z**. **Z** is computed by solving a low-rank, sparse representation problem, whose solution is obtained after a sequence of closed form update steps made possible by the Inexact Augmented Lagrange Multiplier (IALM) method. To account for viewpoint, appearance, and illumination changes and to alleviate tracking drift, we update **D** adaptively via a sequence of template replacement and reweighting steps.

Contributions: Compared with existing approaches, the contributions of this work are the following. **(1)** We formulate object tracking as a sparse and low-rank representation problem, which provides a new perspective on robust visual tracking. This is done by primarily taking advantage of the relationship between particle appearances and jointly representing them w.r.t. a dictionary of templates, which is dynamically updated. To the best of our knowledge, this is the first work to exploit the low-rank nature inherent to object tracking. **(2)** Although we compute particle representations jointly, we solve the sparse and low-rank representation problem efficiently through a sequence of closed form updates, which make LRST computationally attractive (linear in the number of particles and template size).

The rest of the paper is organized as follows. In Section 2, we summarize the work most related to ours. The particle filter algorithm is birefly reviewed in Section 3. The proposed LRST method is described in detail in Section 4. Experimental results are reported and analyzed in Section 5. We conclude in Section 6.

2 Related Work

Object tracking boasts of an extensive literature. Here, we review previous work most relevant to this paper and refer to [2] for a more thorough review. In general, tracking methods can be categorized into two groups: generative and discriminative. Generative tracking methods adopt an appearance model to describe the target observations, and the aim is to search for the target location that has the most similar appearance to the model. Examples of generative methods are eigentracker [6], mean shift tracker [7], and incremental tracker [8]. Discriminative tracking methods view the object tracking as a binary classification problem, which seeks the target location that can best separate the target from the background. Examples of discriminative methods are on-line boosting [9], ensemble tracking [10], online multi-view forests for tracking [11], and online multiple instance learning tracking [12].

Over the last few years, particle filters (also known as condensation or sequential Monte Carlo models) have proven to be powerful tools for object tracking [13]. The strength of these methods lies in their simplicity, flexibility, and systematic treatment of nonlinearity and non-Gaussianity. While the use of more particle samples can improve track robustness, the computational load of

particle filter trackers tends to increase linearly with the number of particles. Consequently, researchers have proposed methods to speed up the particle filter framework, such as, the coarse-to-fine strategy in [14].

Recently, sparse representation has been successfully introduced to object tracking [1,3,5,15,16] based on the particle filter framework. In [1], a tracking candidate is represented as a sparse linear combination of object and trivial templates. Sparse representation is computed by solving an ℓ_1 minimization problem, which addresses the inverse intensity pattern problem during tracking. In [3], dynamic group sparsity is integrated into the tracking problem and very high dimensional image features are used to improve tracking robustness. In [5], dimensionality reduction and a customized orthogonal matching pursuit algorithm are adopted to accelerate the L_1 tracker [1]. Our proposed method is inspired by these works. To improve efficiency and to capitalize on the interdependence between particle sample appearances, we employ a sparse, low-rank target representation, whose templates can be updated dynamically to capture changes in the target's appearance. Our work is also motivated by recent advances in low-rank representation and its recently discovered applications in computer vision, such as robust face recognition [17], subspace clustering [18], background subtraction [19].

3 Particle Filter

The particle filter [13] is a Bayesian sequential importance sampling technique for estimating the posterior distribution of state variables characterizing a dynamic system. It provides a convenient framework for estimating and propagating the posterior probability density function of state variables regardless of the underlying distribution through a sequence of prediction and update steps. Let $\mathbf{s}_t$ and $\mathbf{y}_t$ denote the state variable describing the parameters of an object at time t (e.g. location or motion parameters) and its observation respectively. In the particle filter framework, the posterior $p(\mathbf{s}_t|\mathbf{y}_{1:t})$ is approximated by a finite set of n samples $\left\{\mathbf{s}_t^i\right\}_{i=1}^n$ (called particles) with importance weights w_i. The particle samples $\mathbf{s}_t^i$ are drawn from an importance distribution $q(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{y}_{1:t})$, which for simplicity is set to the state transitional probability $p(\mathbf{s}_t|\mathbf{s}_{t-1})$. In this case, the importance weight of particle i is updated by the observation likelihood as: $w_t^i = w_{t-1}^i p(\mathbf{y}_t|\mathbf{s}_t^i)$.

Particle filters have been used extensively in object tracking [2]. In this paper, we also employ particle filters to track the target object. Similar to [1], we assume an affine motion model between consecutive frames. Therefore, the state variable $\mathbf{s}_t$ consists of the six parameters of the affine transformation (2D linear transformation and a 2D translation). By applying an affine transformation using $\mathbf{s}_t$ as parameters, we crop the region of interest $\mathbf{y}_t$ from the image and normalize it to the size of the target templates in our dictionary. The state transition distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is modeled to be Gaussian with the the dimensions of $\mathbf{s}_t$ assumed independent. The observation model $p(\mathbf{y}_t|\mathbf{s}_t)$ reflects the similarity between a target candidate (particle) and dictionary templates. In this paper,

$p(\mathbf{y}_t|\mathbf{s}_t)$ is computed as a function of the difference between the low-rank representation of the target corresponding to object templates in the dictionary and its representation corresponding to the background templates.

4 Low-Rank Sparse Tracker (LRST)

In this section, we give a detailed description of our particle filter based tracking method, denoted as the Low-Rank Sparse Tracker (LRST).

4.1 Low Rank Sparse Representation of a Tracking Target

In our particle filter based tracking method, particles are randomly sampled around the current state of the tracked object according to a zero-mean Gaussian distribution. In the t^{th} frame, we consider n particle samples, whose observations (pixel color values) are denoted in matrix form as: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$. Each column of $\mathbf{Z}$ is a particle in $\mathbb{R}^d$. In the noiseless case, each particle $\mathbf{x}_i$ is represented as a linear combination of templates that form a dictionary $\mathbf{D}_t = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_m]$, such that $\mathbf{X} = \mathbf{D}_t\mathbf{Z}$. Here, the columns of $\mathbf{Z}$ denote the representation of the particles with respect to $\mathbf{D}_t$. The dictionary columns comprise the templates that will be used to represent each particle. These templates include visual observations of the tracked object and the background (non-object) possibly under a variety of appearance changes. Since our representation is constructed at the pixel level, misalignment between dictionary templates and particles might lead to degraded performance. To alleviate this problem, one of two strategies can be employed. **(i)** $\mathbf{D}_t$ can be constructed from a dense sampling of the object and the background, which includes transformed versions of both. **(ii)** Columns of $\mathbf{X}$ can be aligned to columns of $\mathbf{D}_t$ as in [17]. In this paper, we employ the first strategy, which leads to a larger m but a lower overall computational cost. We denote $\mathbf{D}_t$ with a subscript because its templates will be progressively updated to incorporate variations in object appearance due to changes in illumination, viewpoint, etc. How to update $\mathbf{D}_t$ systematically will be introduced later.

We base the formulation of our tracking method on the following observations. **(a)** Because particles are densely sampled around the current object state, most of them will have similar representations with respect to $\mathbf{D}_t$. Therefore, the resulting representation matrix $\mathbf{Z}$ is expected to be low-rank, even for large values of m and n. **(b)** Inspired by the L_1 tracker [1], a good target candidate (particle) $\mathbf{x}_i$ has only a limited number of nonzero coefficients in its corresponding representation $\mathbf{z}_i$. In other words, only a few dictionary templates are required to reliably represent a particle. **(c)** In many visual tracking scenarios, target objects are often corrupted by noise or partially occluded. As in [1], this noise can be modeled as sparse additive noise that can take on large values anywhere in its support. We combine **(a)-(c)** to obtain the problem in Eq (1), where $\mathbf{E}$ is the error due to noise and/or occlusion and λ_1, λ_2 and λ_3 are three parameters to balance the importance of each term. We set $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in our experiments. The solution to Eq (1) is described in Section 4.3.

$$\min_{\mathbf{Z},\mathbf{E}} \lambda_1 \|\mathbf{Z}\|_* + \lambda_2 \|\mathbf{Z}\|_{1,1} + \lambda_3 \|\mathbf{E}\|_{1,1} \quad (1)$$
$$\text{such that: } \mathbf{X} = \mathbf{D}_t\mathbf{Z} + \mathbf{E}$$

Low-Rank Representation $\|\mathbf{Z}\|_*$: We seek to minimize the rank of the representations of all particles *together*. Since directly minimizing matrix rank is NP-hard in general, we resort to minimizing its convex envelope – its nuclear (or trace) norm $\|\mathbf{Z}\|_*$. As compared to the L_1 tracker, the particles at instance t are represented jointly and not separately. Joint representation capitalizes on the correlations between particles to provide a more robust and computationally efficient solution. Instead of solving n separate ℓ_1 minimization problems (in the case of L_1 tracking), we consider a single rank minimization problem that is solved by a sequence of closed form update operations. To the best of our knowledge, this work is the first to exploit the low-rank nature of particle representations in a tracking setting.

Sparse Representation $\|\mathbf{Z}\|_{1,1}$: Templates in $\mathbf{D}_t$ showcase the possible variations in appearance of the target object and background, but only a sparse number of these templates is required to reliably represent each particle [1]. This sparse scheme is free of model training, robust to sparse noise, and has proven useful in discriminating between samples of different classes [20].

Reconstruction Error $\|\mathbf{E}\|_{1,1}$: For robustness against sparse significant errors (e.g. due to occlusion), we seek to minimize the ℓ_1 norm of each column of $\mathbf{E}$. This sparse error assumption has been widely adopted in tracking [1] and other applications [20]. Unlike the L_1 tracker [1] that incorporates sparse error by augmenting $\mathbf{D}_t$ with a large number ($2d$) of "trivial" templates, we compute $\mathbf{E} \in \mathbb{R}^{d\times n}$ directly. Moreover, the values and support of columns in $\mathbf{E}$ are quite informative, since they can indicate the presence of occlusion (large values but sparse support) and whether a candidate particle is sampled from the background (large values with non-sparse support).

Adaptive Dictionary: $\mathbf{D}_t$ is initialized by sampling image patches around the initial target position. For accurate tracking, the dictionary must be updated in successive frames to model changing target appearance. Furthermore, to alleviate the problem of target drift in tracking, we augment $\mathbf{D}_t$ with templates representative of the background, such that $\mathbf{D}_t = [\mathbf{D}_O \quad \mathbf{D}_B]$, where $\mathbf{D}_O$ and $\mathbf{D}_B$ represent the target object and background templates respectively. Thus, a particle $\mathbf{z}_k$ is composed of an object representation $\mathbf{z}_k^O$ and a background representation $\mathbf{z}_k^B$. In this paper, the tracking result $\mathbf{y}_t$ at instance t is the particle $\mathbf{x}_i$ such that $i = \arg\max_{k=1,\ldots,n} \left(\|\mathbf{z}_k^O\|_1 - \|\mathbf{z}_k^B\|_1\right)$. This encourages the tracking result to be represented well by the object and *not* the background templates. We also exploit this discriminative information to design a systematic procedure for updating $\mathbf{D}_t$. To each object template in $\mathbf{D}_O$, we allocate a weight ω_i that is indicative of how representative the template is. In fact, the more a template is used to represent tracking results, the higher its weight is. If particles are sufficiently represented (up to a predefined threshold) by the dictionary, then

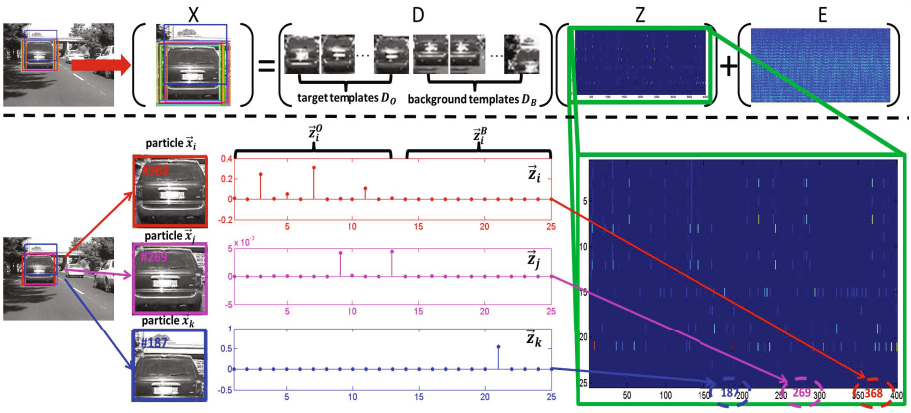

Fig. 1. Schematic example of our LRST algorithm. The representation $\mathbf{Z}$ of sample particles $\mathbf{X}$ w.r.t. dictionary $\mathbf{D}$ (set of object and background templates) is learned by solving Eq (1). Notice that $\mathbf{Z}$ is sparse (i.e. few dictionary templates are used) and low-rank (i.e. dictionary templates are reused for representation). The particle $\mathbf{x}_i$ is selected among all other particles as the tracking result, since $\mathbf{x}_i$ is represented the best by the object templates only.

there is no need to update it. Otherwise, the current tracking result replaces the object template that has the smallest weight. The weight of this new template is set to the median of the current normalized weight vector $\boldsymbol{\omega}$. $\mathbf{D}_B$, on the other hand, is updated at every frame by resampling patches at a sufficient distance from the current tracking result.

4.2 Discussion

As shown in Eq (1), we propose a generic formulation for robust object tracking using low-rank sparse representation. With different values of λ_1 and λ_2, different object trackers are obtained. When $\lambda_1 = 0$, it is a Sparse Tracker (ST), and when $\lambda_2 = 0$, it is a Low Rank Tracker (LRT). if $\lambda_1 \neq 0$ and $\lambda_2 \neq 0$, we denote it as a Low Rank Sparse Tracker (LRST). In a word, the popular L_1 tracker [1] and the proposed LRT, ST are three special case trackers of the more general LRST. In Fig. 1, we present an example of how our tracker works. Given all particles $\mathbf{X}$ (sampled around the tracked car) and based on a dictionary $\mathbf{D} = [\mathbf{D_O}\ \mathbf{D_B}]$, we learn the representation matrix $\mathbf{Z}$ by solving Eq (1). Note that smaller values are darker in color. Clearly, $\mathbf{Z}$ is sparse (small number of templates used) and low-rank (templates are reused among particles). The particle $\mathbf{x}_i$ is chosen as the current tracking result $\mathbf{y}_t$ because its representation difference $\left(\|\mathbf{z}_i^O\|_1 - \|\mathbf{z}_i^B\|_1\right)$ is largest among all particles. Since particle $\mathbf{x}_j$ is a misaligned version of the car, it is not represented well by $\mathbf{D}_O$ (i.e. $\mathbf{z}_j^O$ has small values). Particle $\mathbf{x}_k$ is represented well by $\mathbf{D}_B$ (i.e. $\mathbf{z}_k^B$ has large values). This precludes the tracker from drifting into the background.

4.3 Solving Eq (1)

Unlike many other works that only focus on one of the two convex and non-smooth regularizers (sparse $\|.\|_1$ regularizer or low-rank $\|.\|_*$ regularizer), the cost function in Eq (1) combines both. In order to handle these two regularizers independently, we introduce two slack variables and add two equality constraints as in Eq (2).

$$\min_{\mathbf{Z}_{1-3},\mathbf{E}} \quad \lambda_1 \|\mathbf{Z}_1\|_* + \lambda_2 \|\mathbf{Z}_2\|_{1,1} + \lambda_3 \|\mathbf{E}\|_{1,1} \tag{2}$$
$$\text{such that: } \begin{cases} \mathbf{X} = \mathbf{D}\mathbf{Z}_3 + \mathbf{E} \\ \mathbf{Z}_3 = \mathbf{Z}_1 \\ \mathbf{Z}_3 = \mathbf{Z}_2 \end{cases}$$

This transformed problem can be minimized using the conventional Inexact Augmented Lagrange Multiplier (IALM) method that has attractive quadratic convergence properties and is extensively used in matrix rank minimization problems [17]. IALM is an iterative method that augments the traditional Lagrangian function with quadratic penalty terms. This allows closed form updates for each of the unknown variables. The updates are closed form due to the identities in Eq (3,4), where $\mathcal{S}_\lambda(\mathbf{A}_{ij}) = \text{sign}(\mathbf{A}_{ij}) \max(0, |\mathbf{A}_{ij}| - \lambda)$ is called the soft-thresholding operator and $\mathcal{J}_\lambda(\mathbf{A}) = \mathbf{U}_A \mathcal{S}_\lambda(\mathbf{\Sigma}_A) \mathbf{V}_A^T$ is the singular value soft-thresholding operator. Due to space limitations, we do not elaborate on the optimization details here but do so in the **supplementary material**.

$$\mathbf{X}^* = \arg\min \quad \|\mathbf{X} - \mathbf{A}\|_F^2 + 2\lambda\|\mathbf{X}\|_{1,1} = \mathcal{S}_\lambda(\mathbf{A}) \tag{3}$$

$$\mathbf{X}^* = \arg\min \quad \|\mathbf{X} - \mathbf{A}\|_F^2 + 2\lambda\|\mathbf{X}\|_* = \mathcal{J}_\lambda(\mathbf{A}) \tag{4}$$

The computational bottleneck of LRST lies in the SVD of matrix $\mathbf{Z}$. Since $\mathbf{Z}$ is low-rank and rectangular, its SVD can be computed efficiently with time complexity $\mathcal{O}(mnr)$, where r is its rank such that $r \leq \sqrt{\min(m,n)}$. Because $m \ll \min(n,d)$ in the majority of cases, the overall computational complexity (per frame) of LRST is $\mathcal{O}(nd)$, which is linear in both the number of particles and the template size. This complexity is on par with that of other fast particle-based tracking algorithms. In comparison, the computational complexity of the L_1 tracker [1], which uses a sparse linear representation similar to LRST, is at least $\mathcal{O}\left(nd^2\right)$, since the number of dictionary templates (object and trivial) is $(m + 2d)$ and n Lasso problems are solved separately. Clearly, LRST is more computationally attractive than L_1 tracker. In fact, our results show that LRST is two orders of magnitude faster than L_1 tracker in general. For example, when $m = 25$, $n = 400$, and $d = 32 \times 32$, the average per-frame run-time for LRST and L_1 trackers are 3.0 and 340 seconds respectively. Moreover, increasing m and d will improve performance without much added computational cost.

5 Experimental Results

In this section, we present experimental results that validate the effectiveness and efficiency of our LRST method. We also conduct a thorough comparison between LRST and state-of-the-art tracking methods where applicable.

5.1 Datasets and Baselines

To evaluate our trackers (ST, LRT and LRST), we compile a set of 15 challenging tracking sequences (e.g. *car4*, *david indoor*, and *soccer* sequences) that are publicly available online[1]. Due to space constraints, we will only show results on 10 of these sequences, leaving the rest for the **supplementary material**. These videos are recorded in indoor and outdoor environments and include challenging appearance variations due to changes in pose, illumination, scale, and the presence of occlusion. We compare our trackers against 6 recent and state-of-the-art visual trackers denoted as: VTD [21], L_1 [1], IVT [8], MIL [12], Frag (Fragments-based tracker) [22], and OAB [9]. We implemented these trackers using publicly available source codes or binaries provided by the authors themselves. They were initialized using their default parameters for all video sequences.

5.2 Implementation Details

All our experiments are done using MATLAB R2008b on a 2.66GHZ Intel Core2 Duo PC with 6.0GB RAM. The template size d is set to half the size of the target initialization in the first frame. So, d is usually in the order of several hundred (or a few thousand) pixels. For all our experiments, we model $p(\mathbf{s}_t|\mathbf{s}_{t-1}) \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}))$, where $\boldsymbol{\sigma} = [0.005, 0.0005, 0.0005, 0.005, 4, 4]^T$. We set the number of particles $n = 400$, the total number of templates $m = 25$. In all cases, the initial position of the target is selected manually. In Sections 5.3 and 5.4, we give a qualitative and quantitative analysis of the proposed trackers (ST, LRT and LRST), as well as compare it against the 6 baseline methods. Our experiments show that LRST produces more robust and accurate tracks. Tracking results are made available in the **supplementary material**.

5.3 Qualitative Comparison

The *car4* sequence was captured in an open road scenario. Tracking results at frames {50, 187, 204, 380, 530, 641} for all 9 methods are shown in Fig. 2(a). The different tracking methods are color-coded. OAB, Frag, and VTD start to drift from the target at frame 187, while MIL starts to show some target drifting at frame 200 and finally loses the target at frame 300. L_1 and ST can track the target when illumination changes. However, there is a little shift compared with ground truth. The target is successfully tracked by other trackers (IVT, LRT

[1] `vision.ucsd.edu/~bbabenko/project_miltrack.shtml; www.cs.toronto.edu/ dross/ivt/; cv.snu.ac.kr/research/~vtd/`

and LRST) throughout the entire sequence even though the drastic illumination changes. These results show the sparsity and low-rank property are useful for robust object tracking.

In the *david* sequence, a moving face is tracked. The tracking results at frames {320, 430, 460, 500, 650, 690} are shown in Fig. 2(b). Frag and VTD fail around frames 430 and 460 respectively. OAB starts to drift at frame 550. MIL and L_1 adequately track the face, but experience target drift, especially at frames 690 and 500 respectively. The IVT, ST, LRT and LRST trackers track the moving face accurately throughout the sequence.

Results on the *faceocc2* sequence are shown in Fig. 2(c). Most trackers start drifting from the man's face when it is almost fully occluded by the book. Because the LRST method explicitly handles partial occlusions, updates the object dictionary progressively, and continuously incorporates background templates, it handles the appearance changes in this sequence very well and continues tracking the target during and after the occlusion.

Fig. 2(d) shows tracking results for the *girl* sequence. Performance on this sequence exemplifies the robustness of LRST to occlusion (complete occlusion of the girl's face as she swivels in the chair) and large pose change (the girl's face undergoes extensive 3D rotation about multiple axes). Only ST, LRT, LRST and L_1 trackers are capable of tracking the target during the entire sequence. Other trackers experience drift at different instances: Frag at frame 248, OAB and IVT at frame 436, and VTD at frame 477.

In the *shaking* sequence, the tracked object is subject to changes in illumination and pose. Even though the stage lighting is drastically changed and the object appearance is severely varied due to head shaking, our method successfully tracks the object (refer to Fig. 2(e)). Since our trackers (ST, LRT and LRST) perform adaptive template updating, they effectively handle the inherent pose variations. However, other methods (OAB, IVT, L_1 , and Frag) fail to track the object when these changes occur. VTD and MIL can track the object quite well except for some errors around frame 60.

The *football* sequence includes severe background clutter, which is similar in appearance to the target to be tracked. The OAB, L_1, and ST methods drift at frame 100, 246 and 271, respectively. For the other methods, tracking drifts from the intended object (helmet) to other similar looking objects in the vicinity. This is especially the case when the two football players collide at frame 362 (refer to Fig. 2(f)). The proposed LRT and LRST method overcome this problem and successfully track the target with the help of background information and the low-rank robustness property.

The *singer1(l)* and *skating1* sequences contain abrupt object motion with significant illumination and scale changes, which cause most of the trackers to drift as shown in Fig. 2(g-h). ST, LRT, LRST and VTD work well.

Results on the *soccer* sequence are shown in Fig. 2(i). They demonstrate how our proposed low-rank sparse method outperforms the state-of-the-art trackers when the target is severely occluded by other objects. LRST accurately tracks the player's face despite scale and pose changes as well as occlusion/noise from

Fig. 2. Tracking results (color-coded bounding boxes) of 9 tracking methods. Frame numbers are overlayed in red.

the confetti raining around him. Other methods (IVT, L_1 , OAB, MIL, and Frag) fail to track the object reliably. ST, LRT and VTD can track the target in this sequence with some drift.

The *trellis70* sequence is captured in an outdoor environment where lighting conditions change drastically. The video is acquired when a person walks underneath a trellis covered by vines. As shown in Fig. 2(j), the cast shadow changes the appearance of the target face drastically. Furthermore, the combined effects of pose and lighting variations along with a low frame rate make the tracking task extremely difficult. Nevertheless, LRST and LRT trackers follow the target accurately and robustly, while the other tracking methods perform below par in this case. VTD and Frag fail at frame 185. The L_1 and ST trackers start drifting at frame 288 and 327 respectively, while MIL and OAB fail at frame 327. IVT starts drifting at frame 330.

5.4 Quantitative Comparison

To give a fair quantitative comparison between the 9 trackers, we obtain manually labeled ground truth tracks for all sequences. To evaluate a tracker, we compute the distance (in pixels) between the center of the tracking result to that of the ground truth, averaged over all frames as in [1,12]. The smaller this distance the better the tracking algorithm is. Most of the ground truth can be downloaded with the video sequence.

In Fig. 3, we show the mean center distance of all 9 algorithms on 10 sequences. Except for the *football* and *faceocc2* sequences, in which our proposed trackers obtain similar result as VTD and IVT, our algorithms do outperform the other methods, and in several cases by a significant amount. Both Frag and L_1 trackers perform well against partial occlusion but tend to fail in the presence of severe illumination and pose changes. IVT performance is hardly affected by most changes in appearance except for severe illumination changes. OAB performance is affected by background clutter and easily drifts. Similar to the previous trackers, MIL is robust to most changes in appearance except severe illumination change, which causes the tracking to drift away from the tracker into the background. VTD is the most robust of the 6 methods we compare against, yet it tends to drift due to illumination changes, severe occlusions, and pose changes.

Based on the results in Fig. 3, we analyze the importance of each term in Eq (1) and the corresponding 3 trackers (ST, LRT and LRST). Results from the L_1 and ST trackers show that representation sparsity is very effective for robust object tracking and is the primary reason why they perform so well against many state-of-the-art trackers. Compared with ST, LRT obtains much better results, which demonstrates the effectiveness of the low-rank property in object tracking. This is due to the fact that the low-rank property mines the correlations among particle appearances, thus, making the learned representations more robust. Compared with ST and LRT, LRST obtains the best performance, which shows that the combination of the low-rank and sparsity properties is more effective than either of them taken separately. In fact, the rank of matrix $\mathbf{Z}$ is iteratively reduced

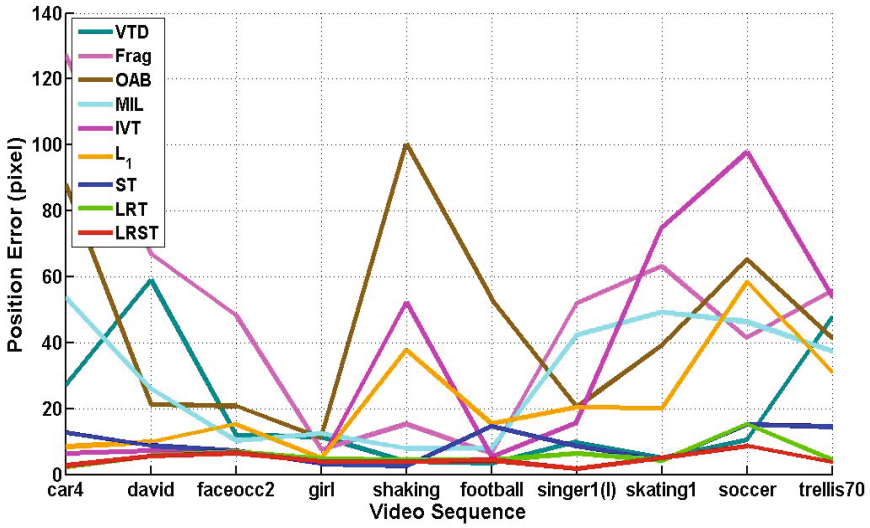

Fig. 3. Mean distances of 9 different trackers on 10 different video sequences. On average, the proposed trackers (ST, LRT and LRST) outperform the other 6 state-of-the-art trackers. For each sequence, the smallest and second smallest distances are denoted in red and blue respectively.

in LRT and LRST. As shown in Fig. 1, the rank of the learned representation $\mathbf{Z}$ is 10, which is reduced from 25.

Now, we compare the performance of the LRST, ST and L_1 trackers, which use the sparsity property. Based on the results in Fig. 3, LRST outperforms ST and L_1. That is because ST and L_1 trackers learn particle representations separately, while LRST capitalizes on the dependencies among different particles to obtain a more robust joint representation. Our results demonstrate that it is useful for visual tracking to mine relationships between particle appearances. Moreover, in theory, the L_1 tracker is a special case of our LRST framework, and it should produce the same results as ST. However, this is not reflected in our empirical results due to three reasons. **(a)** The L_1 tracker is forced to adopt a smaller template size ($d = 12 \times 15$) due to its high computational cost $\mathcal{O}(nd^2)$. A larger d leads to a richer representation and improved tracking performance. As mentioned earlier, ST methods set d to half the size of the initial bounding box, which is generally more than 600 pixels. In addition, ST uses background information for more robust tracking, while L_1 does not. **(b)** In the public MATLAB implementation of L_1, the dictionary weights are used not only to update the target templates but also to multiply the templates themselves, which leads to an artificially sparser representation. For ST, the weights are only used to update the target templates. In addition, ST uses a more efficient solver (i.e. IALM algorithm [23]) to learn particle representations. **(c)** Since the L_1 and ST trackers both adopt the particle filter framework, they tend to randomly sample different particles from the same frame.

6 Conclusion

In this paper, we propose a new particle-filter based tracking algorithm that exploits the low-rank nature of particle representations. We model tracking as a low-rank sparse learning problem at the level of particles and solve it using an efficient IALM approach. For further robustness against tracking drift and significant changes in target appearance, we introduce background templates into the representation and an online procedure that adaptively updates the templates. We extensively analyze the performance of our tracker on 15 challenging videos and show it outperforming 6 recent and state-of-the-art tracking methods.

Acknowledgment. This study is supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapores Agency for Science, Technology and Research (A*STAR).

References

1. Mei, X., Ling, H.: Robust Visual Tracking and Vehicle Classification via Sparse Representation. TPAMI 33, 2259–2272 (2011)
2. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys 38(4), Article 13 (December 2006)
3. Liu, B., Yang, L., Huang, J., Meer, P., Gong, L., Kulikowski, C.: Robust and Fast Collaborative Tracking with Two Stage Sparse Optimization. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 624–637. Springer, Heidelberg (2010)
4. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. IJCV 29, 5–28 (1998)
5. Li, H., Shen, C., Shi, Q.: Real-time visual tracking with compressed sensing. In: CVPR (2011)
6. Black, M.J., Jepson, A.D.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. IJCV 26, 63–84 (1998)
7. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-Based Object Tracking. TPAMI 25, 564–575 (2003)
8. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental Learning for Robust Visual Tracking. IJCV 77, 125–141 (2008)
9. Grabner, H., Grabner, M., Bischof, H.: Real-Time Tracking via On-line Boosting. In: BMVC (2006)
10. Avidan, S.: Ensemble tracking. In: CVPR, pp. 494–501 (2005)
11. Leistner, C., Godec, M., Saffari, A., Bischof, H.: On-Line Multi-view Forests for Tracking. In: Goesele, M., Roth, S., Kuijper, A., Schiele, B., Schindler, K. (eds.) DAGM 2010. LNCS, vol. 6376, pp. 493–502. Springer, Heidelberg (2010)
12. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR (2009)
13. Doucet, A., De Freitas, N., Gordon, N.: Sequential monte carlo methods in practice. Springer, New York (2001)
14. Yang, C., Duraiswami, R., Davis, L.: Fast multiple object tracking via a hierarchical particle filter. In: ICCV (2005)

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, pp. 2042–2049 (2012)
16. Zhang, T., Ghanem, B., Liu, S., Ahuja, N.: Robust visual tracking via structured multi-task sparse learning. To appear in IJCV (2012)
17. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: RASL: Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2010)
18. Liu, G., Lin, Z., Yu, Y.: Robust subspace segmentation by low-rank representation. In: ICML (2010)
19. Candès, E., Li, X., Ma, Y., Wright, J.: Robust Principal Component Analysis? Journal of the ACM 58 (2011)
20. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. TPAMI 31, 210–227 (2009)
21. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: CVPR, pp. 1269–1276 (2010)
22. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR, pp. 798–805 (2006)
23. Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., Ma, Y.: Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. Technical Report UILU-ENG-09-2214, UIUC (August 2009)