PAPER

Low-Rank Representation with Graph Constraints for Robust Visual Tracking

Jieyan LIU[†], **Ao MA**[†], ***Nonmembers***, **Jingjing LI**[†a)], ***Member***, ***and*** **Ke LU**[†], ***Nonmember***

SUMMARY Subspace representation model is an important subset of visual tracking algorithms. Compared with models performed on the original data space, subspace representation model can effectively reduce the computational complexity, and filter out high dimensional noises. However, for some complicated situations, e.g., dramatic illumination changing, large area of occlusion and abrupt object drifting, traditional subspace representation models may fail to handle the visual tracking task. In this paper, we propose a novel subspace representation algorithm for robust visual tracking by using low-rank representation with graph constraints (LRGC). Low-rank representation has been well-known for its superiority of handling corrupted samples, and graph constraint is flexible to characterize sample relationship. In this paper, we aim to exploit benefits from both low-rank representation and graph constraint, and deploy it to handle challenging visual tracking problems. Specifically, we first propose a novel graph structure to characterize the relationship of target object in different observation states. Then we learn a subspace by jointly optimizing low-rank representation and graph embedding in a unified framework. Finally, the learned subspace is embedded into a Bayesian inference framework by using the dynamical model and the observation model. Experiments on several video benchmarks demonstrate that our algorithm performs better than traditional ones, especially in dynamically changing and drifting situations.

key words: *visual tracking, low-rank representation, graph embedding*

1. Introduction

In recent years, model based tracking algorithms have received significant attention due to their good performance of handling appearance variability of the target object, such as 3D models [1], integration of shape and color [2], foreground/background models [3], kernel-based filters [4] and subspace learning models [5], [6]. The common part of these algorithms is that all of them build or learn a model of the target object at first and then use it for tracking. To some extent, therefore, how to learn the initial model decides the tracking performance of each algorithm.

Subspace representation model is an important subset of visual tracking algorithms. Compared with models performing on the original data space, subspace representation model can effectively reduce the computational complexity due to its fast feature extraction. The subspace representation model provides a compact notion of the target rather than treating the target as an unclear sparse feature representation. It can also filter out high dimensional noises. The classical subspace tracking approach of Black and Jepson [7] was enhanced by incremental subspace updating in [6], [8]. Ho et al. [5] considered the general adaption problem as a subspace adaption problem, where the visual appearance variations at a short time period are represented as a linear subspace. Ross et al. [6] proposed an incremental visual tracker (IVT) with adaptive appearance model that aims to account for rigid appearance variations and deformable motions.

In various subspace learning methods, graph embedding [9] has been widely exploited for its superiority of preserving samples manifold structures. Recently, different kinds of approaches have been proposed to apply graph embedding for visual tracking [10], [11]. In order to preserve discriminative characteristics of target object after feature extraction, Zhang et al. [10] used some labeled foreground and background samples to construct the topological graph. Qiao et al. [11] focused on preserving the short distance among some labeled rotating head samples for face tracking. The essence of graph embedding is that two graphs, a within-cluster/class graph and a between-cluster/class graph, are constructed to preserve the relationship among samples [9], [12]. Before tracking starts, there may be some labeled samples in different states (e.g., different extents of rotation, different levels of occlusion and illumination conditions) available. If we can utilize the label information to design an optimal graph structure to extract the relationship among object samples, we can learn a subspace which can preserve the properties of the object distributions. In this paper, we classify samples into different clusters according to their states. In the tracking process, object samples obtained consecutively in a short time usually show high similarities when the object moves smoothly and regularly. In this situation, these object samples should be close to each other in the geometric structure. This neighbor relationship can be captured well by the within-cluster graph. The object may also change abruptly from one state to another state. In this case, there should also be a close connection between the object samples although they are in different states. This connection can be handled well by the between-cluster graph to degrade the drift problem.

The very assumption behind graph embedding used in visual tracking is that it can preserve the geometric manifold structure within the same state and among different states. Actually, the relationship among different samples not only can be formulated as geometric structure, it can also be ana-

Manuscript received October 14, 2016.
Manuscript revised January 26, 2017.
Manuscript publicized March 8, 2017.

[†]The authors are with the School of Compute Science & Engineering, University of Electronic Science & Technology of China, Chengdu, 611731, P.R. China.
a) E-mail: lijin117@yeah.net (Corresponding author)

DOI: 10.1587/transinf.2016EDP7422

lyzed from the view of feature selection. Specifically, in the tracking process, the object may have different modalities due to the change of pose, rotation, illumination or occlusion. However, the target objects obtained consecutively in a short time usually share a lot of features. If we can dig out the common factors shared among objects, we can obtain the latent low-dimensional space and employ it for the object tracking. Low-rank representation (LRR) [13] has been proven to be effective for revealing common latent factors from different appearances of one object. LRR also stands out for its superiority of robustness. Therefore, it is reasonable to introduce low-rank representation into our model to handle the challenging visual tracking task.

The "statistical properties" and "geometric structure" are two observations of the data from different viewpoints. The two viewpoints are not mutually exclusive and combining them normally could transcend the specific limitations of each perspective. Some influential work [14], [15] has demonstrated that the two properties are complementary and according experiments have shown the benefits of jointly optimizing them. Low-rank constraint is useful for feature selection and can technically uncover the shared features among different samples; while the graph allows the samples to follow the data manifold. Since the neighborhoods would be changed if we use a different feature space learned by feature selection method, we employ the graph embedding as a regularization to preserve the properties of the samples distributions. Finally, the main contributions of this work are listed as follows:

1) A novel graph structure is proposed under the framework of graph embedding to capture the relationship among target objects in the same observation state or different observation states.
2) Low-rank representation and graph embedding are jointly optimized in a unified framework to learn a robust subspace, and this subspace learning is incorporated into the Bayesian framework for robust visual tracking. Thus, the benefits from both feature selection and geometric structure preservation are exploited.

The rest of this paper is organized as follows. Section 2 presents the background information and related work. The propose method, low-rank representation with graph constraints for robust visual tracking, is detailed in Sect. 3. Experiments are reported in Sect. 4, and Sect. 5 is the conclusion.

2. Background Information and Related Work

In this section, we first introduce two related techniques, graph based subspace learning and low-rank representation. Then, we review some related work, especially tracking algorithms based on subspace model.

2.1 Graph Based Subspace Learning

Graph based subspace learning aims to learn a compact subspace where the data manifold structure can be captured by constructing appropriate graph. The graph is used to define the relationship among samples. From the view of spectral graph theory [16], the learned subspace should mostly preserve the topological structure of vertices in the graph. Consider the problem of mapping the weighted graph G to a line. Let $y = (y_1, y_2, \ldots, y_n)^T$ be such a map. The optimal y is given by minimizing [17]:

$$\begin{aligned} &\sum_{i,j=1}^{m} (y_i - y_j)^2 W_{ij} \\ &s.t.\ \ y^T D y = 1 \end{aligned} \tag{1}$$

where W is the weight matrix and D is a diagonal matrix whose entries are column (or row since W is symmetric) sums of W: $D_{ii} = \sum_{j \neq i} W_{ij}$, and $L = D - W$ is the Laplacian matrix of graph G [16]. In Eq. (1), the constraint $y^T D y = 1$ removes an arbitrary scaling factor in the embedding. Then the minimization problem reduces to find

$$y^* = \arg\min_{y^T D y = 1} y^T L y. \tag{2}$$

Suppose $y^T == p^T X$ is the linear projection from X to y, where p is a projection vector and $X = [x_1, \ldots, x_n]$ is a $d \times n$ matrix. Equation (2) can be rewritten as

$$\begin{aligned} &\min_p\ p^T X L X^T p \\ &s.t.\ \ p^T X D X^T p = 1. \end{aligned} \tag{3}$$

When P is the projection that maps the data points from the original space to a low-dimensional space, we can rewrite Eq. (3) to the following equation

$$\begin{aligned} &\min_P\ tr(P^T X L X^T P) \\ &s.t.\ \ P^T X D X^T P = I. \end{aligned} \tag{4}$$

Finally, the optimal P can be obtained by solving eigendecomposition

$$X L X^T P = \Lambda X D X^T P, \tag{5}$$

where Λ is a diagonal matrix whose diagonal elements are eigenvalues.

2.2 Low-Rank Representation

Low-Rank Representation (LRR) [13] has been proven to be effective for many machine learning problems, such as image classification [18], [19], subspace segmentation [13] and transfer learning [20]. A representative practical of LRR is the Robust PCA [21]. To discover the global subspace structures of data, LRR optimizes the following objective function:

$$\begin{aligned} &\min_{Z,E}\ rank(Z) + \lambda \|E\|_\ell \\ &s.t.\ X = AZ + E, \end{aligned} \tag{6}$$

where X is the data matrix and A is a dictionary that linearly

spans the data space. $rank(Z)$ is the rank of coefficients matrix Z. $\lambda > 0$ is a balanced parameter and $\|\cdot\|_\ell$ indicates certain regularization strategy, such as the ℓ_1, ℓ_2-norms, used for modeling the noise. By choosing an appropriate dictionary A, LRR can recover the underlying row space so as to reveal the true segmentation of data.

Recently, low-rank representation has been incorporated with subspace learning, e.g., Low-rank Transfer Subspace Learning (LTSL) [22], Supervised Regularization based Robust Subspace (SRRS) [18], Low-Rank Common Subspace (LRCS) [23] and Low-Rank Discriminate Embedding (LRDE) [19], aiming to find a more robust subspace with low-rank constraint.

2.3 Subspace Based Visual Tracking Algorithms

Subspace learning has been widely exploited in visual tracking tasks due to its good performance. Here we review some previous work reported in recent literatures. The earlier work [7] deployed classical subspace learning method Principal Component Analysis (PCA) [24] to tackle the visual tracking problem. However, PCA is not optimal to preserve the manifold structure of data samples. Qiao et al. [11] proposed a method to better preserve the manifold for dynamic visual tracking. To address the robustness of tracking algorithm, sparse representation [25], [26] and low-rank representation [27] were introduced into the visual tracking models. Graph models [25], [28] were also incorporated to preserve the relationship among samples. In the tracking process, most subspace models were combined with Bayesian framework to predict the location of the target object [6].

3. Low-Rank Representation with Graph Constraints for Robust Visual Tracking

In this section, we introduce the details about how we learn the robust subspace for visual tracking. Firstly, we give the details of how to construct an appropriate graph under the framework of graph embedding. Then, we formulate our objective as a low-rank optimization problem with graph constraints. Thirdly, we use the Alternating Direction Method of Multiplier (ADMM) [29] to solve the objective. In the last, we incorporate our model into the Bayesian framework to tackle visual tracking tasks.

3.1 Construct Graphs

Given a set $x_1, x_2, \cdots, x_h$ in $\mathcal{R}^d$, where x_i is the sample, h is the sample number and d is the feature dimension. The samples are classified into K different clusters, e.g., $C_1, C_2, \cdots, C_K$, according to the states. Samples with similar modalities are grouped into the same cluster. Figure 1 shows some clusters of 2 data sequences. Clearly, $h = \sum_{q=1,\cdots,K} n_q$, where n_q is the number of samples for cluster C_q.

In the tracking process, the target object is consecutively changing, that is, the features of the target object in current frame would usually show high similarity with that in the previous frame and that in the next frame. From the intuition and experience of human beings, we know that the most relevant factors of tracing in current frame are the target object state in the latest frames. If we say that global information (long-term memory) is predominant in recognition and classification tasks, we would say that active information (short-term memory) is important in tracking tasks. Therefore, it's significant to preserve the short-term memory for visual tracking. To address this problem, we use a cluster, denoted by C_{est}, to memorize the target objects obtained from the latest several frames in the tracking process. The size of C_{est} is limited to n_{est}, e.g., three or five, to ensure that it contains up to n_{est} target objects which are only obtained recently. C_{est} is updated every n_{est} observations obtained in the tracking process. Given data matrix $X = [X_1, X_2, \ldots, X_K, X_{est}]$, where data matrix $X_q (q = 1, 2, \ldots, K)$ corresponds to cluster C_q and X_{est} corresponds to cluster C_{est}, we aim to construct a graph which can reflect relationship among samples in the same state or different states. Suppose X contains n data points, we have $n = \sum_{q=1,2,\cdots,K} n_q + n_{est}$. Let $G = \{\mathcal{V}, W\}$ be the graph with vertex set $\mathcal{V}$ and similarity matrix W, where $\mathcal{V} = C_1 \bigcup C_2 \bigcup \cdots \bigcup C_K \bigcup C_{est}$ and W is an $n \times n$ matrix used to define the relationship between each pair of vertices. The graph structure is modeled as the combination of the within-cluster graph and the between-cluster graph, which are detailed in the next two subsections, respectively.

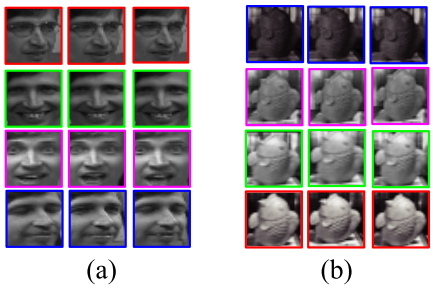


Fig. 1 Cluster samples. (a) Dudek sequence. Samples with different expressions or different extents of rotation (e.g., front face and profile) are grouped into different clusters. (b) Fish sequence. Samples with different lighting conditions are grouped into different clusters.

3.1.1 Construct Within-Cluster Graph

The within-cluster graph is constructed according to relationship among samples in each cluster. Let $G_w = \{\mathcal{V}, W_w\}$ be the within-cluster graph, where $\mathcal{V}$ is the vertex set and W_w is the $n \times n$ within-cluster matrix. Suppose x_i and x_j are any two vertices in $\mathcal{V}$, an edge is added between x_i and x_j if x_i and x_j are in the same cluster and x_j is one of x_i's k-nearest neighbors. Examples are shown in Fig. 2. The connection between vertices can reflect the transition of the modality in the same cluster. Thus this method can preserve the local manifold structure by keeping objects with similar modalities closer. We apply the heat kernel method to define

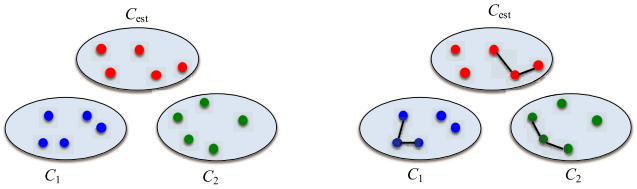


Fig. 2 The adjacency within-cluster graph. (a) The vertices of different colors belong to different clusters. (b) The adjacency within-cluster graph. We set $k = 2$ in nearest neighbor searching to preserve the local structure. Here we only take one vertex in each cluster as the example.

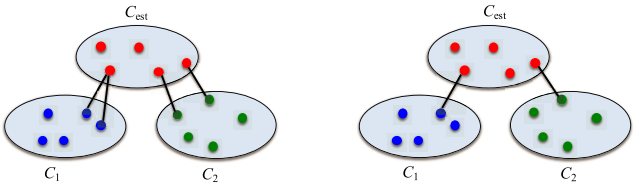


Fig. 3 The adjacency between-cluster graph. We connect C_{est} with each of other clusters to ensure that our model can handle complicated situations no matter which state the target object drift to. (a) When $k = 2$, the two-nearest pairs of neighbors between C_{est} and each of other clusters are selected, and each pair of neighbors are connected. (b) When $k = 1$, the nearest pair of neighbors between C_{est} and each of other clusters are selected, and each pair of neighbors are connected.

W_w. The element w_{ij}^w representing the similarity between x_i and x_j is defined as $w_{ij}^w = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$ if there is an edge between x_i and x_j, otherwise it is 0.

The within-cluster graph can be used to preserve the distribution of data points when the target object changes smoothly and regularly. However, in some cases, the target object may drift abruptly. To address this problem, we further construct the between-cluster graph as follows.

3.1.2 Construct Between-Cluster Graph

The between-cluster graph is used to preserve the distribution of samples in different states. Since samples in C_{est} are obtained recently and can reflect current state of the object, the between-cluster graph is constructed according to the relationship between C_{est} and other clusters. This method aims to set up connections between current state and other states, thus it can capture the abrupt changes of the object in the tracking process.

Let $G_b = \{\mathcal{V}, W_b\}$ be the between-cluster graph, where $\mathcal{V}$ is the vertex set and W_b is the $n \times n$ between-cluster matrix. To construct the between cluster graph G_b, we connect C_{est} and each of other clusters. Specifically, suppose x_i and x_j are any two vertices in $\mathcal{V}$, an edge is added between x_i and x_j if $x_i \in C_{est}$ and $x_j \in C_q$ $(q = 1, 2, \ldots, K)$ and $\{x_i, x_j\}$ is one of the k-nearest pairs of neighbors between C_{est} and C_q. Examples are shown in Fig. 3. In Fig. 3 (a), when k=2, the two-nearest pairs of neighbors between C_{est} and C_1 are selected and each pair of neighbors are connected. C_{est} and C_2 are connected in the same way. In Fig. 3 (b), when $k = 1$, the nearest pair of neighbors between C_{est} and C_1 are selected and connected. C_{est} and C_2 are also connected in the same way. Similarly, we apply the heat kernel method to define W_b. The element w_{ij}^b representing the similarity between x_i and x_j is defined as $w_{ij}^b = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$ if there is an edge between x_i and x_j, otherwise it is 0.

3.1.3 Graph Combination

The within-cluster graph G_w and between-cluster graph G_b are two complementary parts of graph G. Finally, we can obtain the graph structure by combining the two graphs. The similarity matrix W is the combination of W_w and W_b.

$$W = W_w + W_b. \tag{7}$$

3.2 Learning Subspace with Low-Rank Representation

Suppose the data matrix $X \in \mathcal{R}^{d\times n}$ consists of n data points in graph G, where d is the original data dimension. Given a complete basis matrix $A = [a_1, a_2, \cdots, a_m] \in \mathcal{R}^{d\times m}$, it can linearly spans the data space as

$$X = AZ, \tag{8}$$

where $Z \in \mathcal{R}^{m\times n}$ is the coefficient matrix. To achieve our goal of seeking a robust subspace $P \in \mathcal{R}^{d\times l}$ $(l \ll d)$which maps the n samples to the low-dimensional space, we first denote the projected low-dimensional data matrix as

$$\bar{X} = P^T X = P^T AZ. \tag{9}$$

Due to the fact that the samples belong to different clusters of the same class, the coefficient vectors corresponding to samples within the same cluster should be highly correlated, and therefore the coefficient matrix Z is expected to be low-rank. By incorporating the low-rank constraint and the subspace learning into a single problem, we formulate our problem as the following objective function:

$$\begin{aligned} &\min_{Z,P} rank(Z) + \lambda_1 tr(P^T XLX^T P) + \lambda_2 \|P^T AZ\|_F^2 \\ &s.t. P^T X = P^T AZ,\ P^T XDX^T P = I, \end{aligned} \tag{10}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are two trade-off parameters used to balance different parts of the objective and $L = D - W$ is the Laplacian matrix of graph G and $D_{ii} = \sum_{j\neq i} W_{ij}$. $tr(P^T XLX^T P)$ and $P^T XDX^T P = I$ are the typical forms of graph embedding as aforementioned in Eq. (4). The regularization term $\|P^T AZ\|_F^2$ is introduced to learn a more compact subspace. It should be noticed that the $rank(\cdot)$ minimization problem in objective Eq. (10) is difficult to solve. Fortunately, nuclear norm can be used as a good surrogate for the rank minimization problem [13]. Then, the objective becomes

$$\begin{aligned} &\min_{Z,P} \|Z\|_* + \lambda_1 tr(P^T XLX^T P) + \lambda_2 \|P^T AZ\|_F^2 \\ &s.t. P^T X = P^T AZ,\ P^T XDX^T P = I. \end{aligned} \tag{11}$$

We convert Eq. (11) to the equivalent Eq. (12) to make the optimization easier.

$$\begin{aligned} &\min_{Z,P} \|Z\|_* + \lambda_1\|X^TP - H\|_F^2 + \lambda_2\|P^TAZ\|_F^2 \\ &s.t. P^TX = P^TAZ, \end{aligned} \tag{12}$$

where H is a matrix whose rows are eigenvectors of the eigen-problem $WH = \Lambda DH$, and Λ is a diagonal matrix of which diagonal elements are eigenvalues. The proof of the equivalence of Eq. (11) and Eq. (12) can be found in [30].

In real world applications, the samples often mix with large amount of noise. To learn a robust subspace, we need to measure the noise and learn discriminative subspaces from the noiseless samples. Recently, [13] has shown that $l_{2,1}$-norm can be used to successfully model the sample-specific corruptions and other types of noise. Therefore, we introduce an additional term into our objective to model errors. Based on the above observations, we come up with the final objective function:

$$\begin{aligned} &\min_{Z,P,E} \|Z\|_* + \lambda_0\|E\|_{2,1} + \lambda_1\|X^TP - H\|_F^2 \\ &+\lambda_2\|P^TAZ\|_F^2 \\ &s.t. P^TX = P^TAZ + E,\ P^TP = I, \end{aligned} \tag{13}$$

where $\lambda_0 > 0$ is a trade-off parameters. The constraint $P^TP = I$ is introduced to learn a compact subspace.

3.3 Problem Optimization

To address the optimization problem of Eq. (13), we write it as the following equivalent problem by introducing a relax variable J:

$$\begin{aligned} &\min_{Z,P,E,J} \|J\|_* + \lambda_0\|E\|_{2,1} + \lambda_1\|X^TP - H\|_F^2 \\ &+\lambda_2\|P^TAZ\|_F^2 \\ &s.t. P^TX = P^TAZ + E,\ Z = J, \end{aligned} \tag{14}$$

Eq. (14) can be solved by Augmented Lagrangian Multiplier (ALM) [31]. Firstly, we transform Eq. (14) into the augmented Lagrangian function as follows:

$$\begin{aligned} &\|J\|_* + \lambda_0\|E\|_{2,1} + \lambda_1\|X^TP - H\|_F^2 + \lambda_2\|P^TAZ\|_F^2 \\ &+tr(Y^T(P^TX - P^TAZ - E)) + tr(U^T(Z - J)) \\ &+\frac{\mu}{2}(\|P^TX - P^TAZ - E\|_F^2 + \|Z - J\|_F^2) \end{aligned} \tag{15}$$

where $\mu > 0$ is a penalty parameter, Y and U are two Lagrange multipliers. To optimize the variables in Eq. (15), we use the Alternating Direction Method of Multiplier (ADMM) [29] since previous work [18] and [23] have demonstrated that ADMM works well in solving similar problems. By using ADMM, we can alternately update variables one by one with an iterative method. Specifically, once the target variable is determined, the others can be regarded as constants. For example, when we optimize P, the irrelevant terms in Eq. (15), e.g., $\|J\|_*$, $\lambda_0\|E\|_{2,1}$, $tr(U^T(Z - J))$ and $\|Z - J\|_F^2$ can be regarded as constants.

For clarity, we use subscript t to represent the t-th iteration and we optimize the variables in the $(t+1)$-th iteration as shown in Algorithm 1. For simplicity, we use the data matrix X itself as the dictionary A in Algorithm 1.

Algorithm 1. *Solving Eq. (15) by ADMM*

Input: data matrix X, λ_0, λ_1, λ_2.

Initialize: $J_0 = Z_0 = 0$, $E_0 = 0$, $Y = U = 0$,
$\mu_0 = 10^{-6}, \mu_{max} = 10^6, \rho = 1.2, \epsilon = 10^{-8}$.

Output: P, Z, E

1. Build the graph structure by using the method presented in 3.1
2. Initialize P by optimizing Eq. (5).
3. Learn H by solving the eigen-problem $WH = \Lambda DH$.

while not converged **do**

4. Update J_{t+1}:
$J_{t+1} = \arg\min_J \frac{1}{\mu_t}\|J\|_* + \frac{1}{2}\|J - Z_t - U_t/\mu_t\|_F^2$
5. Update E_{t+1}:
$E_{t+1} = \arg\min_E \frac{\lambda_0}{\mu_t}\|E\|_{2,1}$
$+\frac{1}{2}\|E - (P_t^TX - P_t^TAZ_t + Y_t/\mu_t)\|_F^2$
6. Update Z_{t+1}:
$Z_{t+1} = (2\lambda_2 + \mu_t)A^TP_tP_t^TA + \mu_tI)^{-1}$
$(\mu_t(A^TP_t(P_t^TX - E_t) + J_t) - U_t + A^TP_tY_t)$
7. Update P_{t+1}:
$P_{t+1} = (2\lambda_1XX^T + \mu_t((X - AZ_t)(X - AZ_t)^T)$
$+ 2\lambda_2AZ_tZ_t^TA^T)^{-1}((X - AZ_t)(\mu_tE^T - Y_t^T) + 2\lambda_1XH)$
$P_{t+1} \leftarrow orth(P_{t+1})$
8. Update the multipliers:
$Y_{t+1} = Y_t + \mu_t(P_{t+1}^TX - P_{t+1}^TAZ_{t+1} - E_{t+1})$
$U_{t+1} = U_t + \mu_t(Z_{t+1} - J_{t+1})$
9. Update the penalty parameter μ_{t+1}:
$\mu_{t+1} = min(\rho\mu_t, \mu_{max})$.
10. Check the convergence conditions:
$\|P_{t+1}^TX - P_{t+1}^TAZ_{t+1} - E_{t+1}\|_\infty < \epsilon$, and
$\|Z_{t+1} - J_{t+1}\|_\infty < \epsilon$, and
11. Update the iteration variable using $t = t + 1$.

End while.

3.4 Complexity Analysis

The main time consuming of Algorithm 1 depends on the following parts:

1) Nuclear norm calculation in step 4;

2) Matrix multiplication and inverse in step 6 and 7.

As aforementioned, X is a $d \times n$ matrix, where d is the original data dimension and n is the size of the dataset. The SVD computation in Step 4 takes $O(n^3)$. The cost of the inverse of an $n \times n$ matrix is $O(n^3)$. The general matrix multiplication of an $n \times n$ matrix takes $O(n^3)$. If there are γ multiplications, the total cost is $\gamma O(n^3)$. For simplicity, suppose the matrixes in step 6 and 7 are $n \times n$ matrixes, then the computation in step 6 and 7 each takes $(\gamma + 1)O(n^3)$. If the number of samples is large, the computation cost cannot be ignored. Fortunately, step 4 can be solved more efficiently by using the way proposed by [13] and the time complexity of matrix multiplication can be reduced to $O(n^{2.376})$ by using the method provided by [32]. Moreover, as the vertex set of graph G is composed by samples achieved currently and other samples with possible appearances, the number of

data points in X is fixed every time when the subspace is updated. Therefore, the time cost would not increase with the accumulation of observations in the tracking process. We clarify this in detail in Sect. 3.6.

3.5 Tracking Model

Bayesian framework has provided a robust and effective framework in many tracking algorithms [6], [10]. We introduce our low-rank graph embedding into Bayesian framework to learn a flexible and effective tracking algorithm. The goal is to find the best configuration of the target with a given observation.

Let $c_t = (x_t, y_t)$ be the coordinates of the center of the detection window with width w_t, height h_t and rotation angle θ_t for the target object in frame t. $\mathcal{X}_t = \{c_t, h_t, w_t, \theta_t\}$ is the state at time t, and $\mathcal{I}_{1:t}$ is the observation up to time t. The Bayesian tracking framework is then shown as follows:

$$p(\mathcal{X}_t|\mathcal{I}_{1:t}) \propto p(\mathcal{I}_t|\mathcal{X}_t) \int p(\mathcal{X}_t|\mathcal{X}_{t-1})p(\mathcal{X}_{t-1}|\mathcal{I}_{1:t-1})d\mathcal{X}_{t-1} \quad (16)$$

where $p(\mathcal{I}_t|\mathcal{X}_t)$ denotes the observation model that measures how much the target and observation at the proposed state coincide, and $p(\mathcal{X}_t|\mathcal{X}_{t-1})$ denotes the dynamical model between two states $\mathcal{X}_t$ and $\mathcal{X}_{t-1}$.

3.5.1 Dynamical Model

Each parameter in $\mathcal{X}_t$ is modeled independently by a Gaussian distribution around its counterpart in $\mathcal{X}_{t-1}$, and thus the motion between frames is itself an affine transformation. Specifically,

$$p(\mathcal{X}_t|\mathcal{X}_{t-1}) = N(\mathcal{X}_t; \mathcal{X}_{t-1}, \Sigma), \quad (17)$$

where Σ is a diagonal covariance matrix whose elements are the variances of the x, y position, height, width and rotation angle of the object.

3.5.2 Observation Model

The observation model is an important part of the Bayesian framework for visual tracking. Consider an image patch $\mathcal{I}_t$ predicated by $\mathcal{X}_t$, and $\mathcal{I}_t$ was generated from a subspace of the target spanned by P and centered at α. The probability of a sample being generated from this subspace is inversely proportional to the distance from the sample to the reference point (i.e. α) of the subspace. The probability can be computed by [6]:

$$p(\mathcal{I}_t|\mathcal{X}_t) = \mathcal{N}(\mathcal{I}_t; \alpha, PP^T + \varepsilon I). \quad (18)$$

where εI corresponds to the additive Gaussian noise in the observation process, and I is an identity matrix with proper size.

3.6 Tracking Procedures

In this section, we introduce our low-rank graph embedding into the Bayesian framework to complete the tracking task. The corresponding diagram of the tracking algorithm is shown in Fig. 4. We use f to indicate the number of observations accumulated in C_{est} in the tracking process. To learn the subspace for tracking, the graph is initialized when f =1 and C_{est} contains only one object (the first object). We now specify the steps of the proposed tracking algorithm as follows.

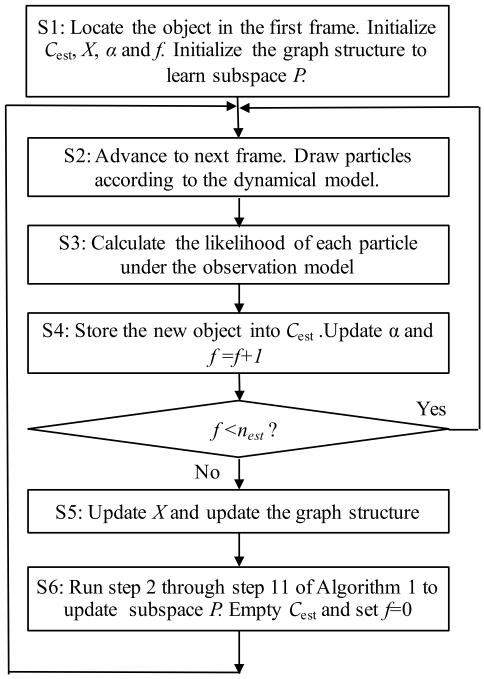


Fig. 4 Diagram of the tracking algorithm

S1: Locate the target object in the first frame, either manually or by using an automated detector, and use a single particle to indicate this location. Initialize the reference point α to be the appearance of the target in the first frame. Initialize C_{est} with the first target object and the effective number of objects in C_{est} is $f = 1$. Initialize $X = [X_1, X_2, \ldots, X_K, X_{est}]$, where X_q $(q = 1, 2, \ldots, K)$ corresponds to cluster C_q (obtained from training) and X_{est} corresponds to cluster C_{est}. Run the first step of Algorithm 1 to initialize the graph structure. Since there is only the first object in C_{est} at the beginning, W_w is initialized according to samples relationships in each cluster of $C_1, C_2, \cdots, C_K$, and W_b is initialized according to the relationship between the first target object and each cluster of $C_1, C_2, \cdots, C_K$. W is initialized by combining W_w and W_b. Run step 2 through step 11 of Algorithm 1 to learn subspace P.

S2: Advance to the next frame. Draw particles from the particle filter, according to the dynamical model.

S3: For each particle, extract the corresponding window from the current frame, and calculate its weight, which is its likelihood under the observation model.

S4: Store the image window corresponding to the most

Table 1 The description of the experimental sequences. Each video consists of 320 × 240 pixels gray-scale images recorded at 30 frames per second.

Sequence	Target type	# Frames	Description
Dudek	Human face	573	A person undergoing large pose, expression, appearance, and lighting changes, as well as partial occlusions.
David	Human face	462	A person moves from a dark to a bright area, undergoing large lighting and pose changes.
CarDark	Object	393	A vehicle moving in the night time with large illumination changes.
Fish	Object	476	A fish doll with lighting change and there exists large and unpredictable camera motion in the video.

likely particle as the new object into C_{est}, and update α to be the appearance of the object newly obtained. Increase the effective number of the objects in C_{est}, i.e., $f = f + 1$. If $f < n_{est}$, go to S2. Otherwise go to S5.

S5: Update X since n_{est} new objects have been accumulated in C_{est}. That is, the old X_{est} in X is replaced by the new X_{est}. Run the first step of Algorithm 1 to update the graph structure. Specifically, W_w is updated by updating the elements corresponding to the similaries between each pair of samples in C_{est}, and W_b is updated according to the relationship between C_{est} and each cluster of $C_1, C_2, \cdots, C_K$. W is obtained by combining W_w and W_b.

S6: Run step 2 through step 11 of Algorithm 1 to update subspace P. Empty C_{est} and set $f = 0$. Go to S2.

Just as we have discussed in Sect. 3.1 that the active information of the object states is predominant in tracking tasks. It can be seen from the above tracking procedure that after the graph is initialized in S1, each time when the graph is updated, C_{est} only contains the n_{est} newly obtained objects in the tracking process. In other word, the number of vertices in the graph is fixed each time when the graph is updated. Thus the computational cost would not increase and the subspace learning would not become slow with the accumulation of observations in the tracking process.

4. Experiments

In this section, we conduct several experiments to demonstrate the effectiveness of our algorithm. All experiments are carried out on a standard 2.50GHz computer with 8GB memory. Four representative video sequences, i.e., Dudek, David, CarDark and Fish[†], are used in our experiments. Table 1 shows the characteristics of these sequences. As there is no extra data of these sequences provided for training, for each video sequence, we divide it into several sub-sequences and each sub-sequence contains 30-50 consecutive frames. The sub-sequences are then separated into two sets, e.g., set $\mathcal{A}$ and set $\mathcal{B}$, according to the objects (samples) states (the definition of the samples states will be clarified in the next paragraph). Each set contains a few sub-sequences and both sets have similar distributions of the samples states. We first use the sub-sequences in $\mathcal{A}$ as the training sequences and use the sub-sequences in $\mathcal{B}$ as the testing sequences. Likewise, we then use the latter as the training sequences and use the former as the testing sequences. For each testing sub-sequence, we locate the object in the first frame and implement the tracking algorithm presented in Sect. 3.6. The final results on a video sequence are obtained by combining the test results of all sub-sequences in both sets.

We extract a frame every a few frames (e.g., five frames) from the training sequences to collect the training frames. The smallest rectangle containing the object is cropped from the selected frame and resized to 32 × 32 pixels to fit the default size of the tracking window. After being selected, the training samples are grouped into different clusters according to the samples states. The definition of the samples states of different video sequences may vary. Generally, it depends on the specific modalities, such as different poses, expressions, extents of rotation, levels of occlusion, lighting conditions, etc. For instances, the front face and the profile can be regarded as different states and an object in different lighting conditions can be regarded as in different states. To make clusters, K typical samples in K different states are selected from the training samples as the centers of K clusters. Then, the K-means algorithm is used to make clusters. Finally, we adjust the clustering results manually if it exists obviously incorrect clustering. If the number of samples in a cluster is restricted to n_q(n_q is set to 3 by default), we select only n_q samples randomly from each cluster as the cluster members. The number of clusters (K) varies for different sequences. We evaluate the results under different number of clusters and different number of samples in a cluster in Sect. 4.1. We also attempt to use the hierarchical clustering method to determine the number of clusters and make clusters in our future work.

The number of the nearest neighbors for the within-cluster graph is set to 2 and the number of the nearest pairs of neighbors for the between-cluster graph is set to 1. n_{est} is set to 5. The dimensionalities of the original space and the subspace are 1024 and 16, respectively.

4.1 Performance under Different Parameters

The center distance between the center of the tracked target and the center of the ground truth is used to measure the performance. Figure 5 shows the average center distance under different values of balanced parameters λ_0, λ_1 and λ_2 on Dudek sequence. The default values of these parameters are $\lambda_0 = 0.1$, $\lambda_1 = 1$ and $\lambda_2 = 1$. When we test the effect of a balanced parameter, the other two are set to their default values. It can be seen from Fig. 5 that the tracking result is insensitive to the variation of these parameters.

Figure 6 shows the results with different number of

[†]http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html

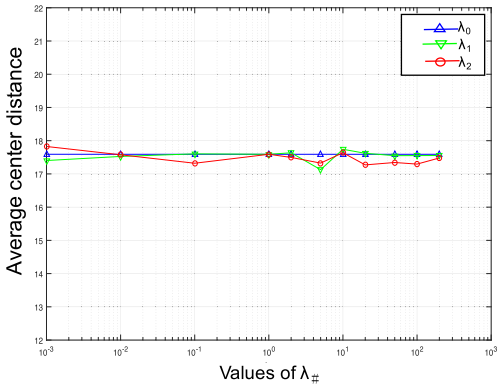


Fig. 5 Average center distance (in pixels) with different balanced parameters on Dudek sequence.

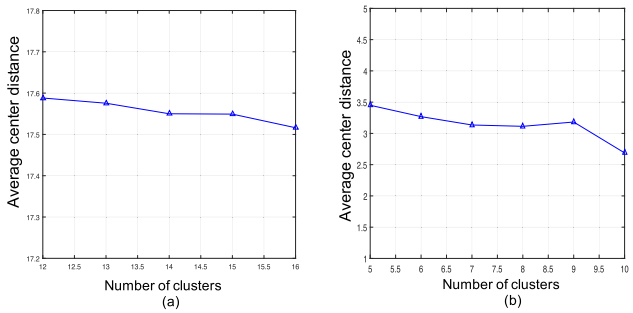


Fig. 6 Average center distance (in pixels) with different number of clusters (K) on (a) Dudek sequence and (b) Fish sequence. When the number of clusters changes, the number of samples for each cluster of $C_1, C_2, \cdots, C_K$ is set to 3 by default.

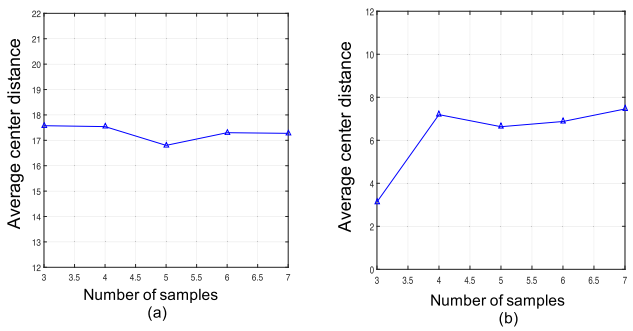


Fig. 7 Average center distance (in pixels) with different number of samples for each cluster of $C_1, C_2, \cdots, C_K$ on (a) Dudek sequence and (b) Fish sequence. When the number of samples in a cluster changes, the numbers of clusters are set to their default values (13 for Dudek and 6 for Fish).

clusters (K) on two video sequences. We can see from Fig. 6 that the optimal parameters are different among situations. Basically, the average center distance decreases with the increase of number of clusters, as more observation states of the target object can be captured and extracted by the graph. Figure 7 show the results with different number of samples in a cluster on two video sequences, respectively. In this experiment, the number of samples for each cluster of $C_1, C_2, \cdots, C_K$ is changed from 3 to 7 and n_{est} is set to 5 by default. The results show that the optimal number of samples for each cluster is different for different sequences.

Table 2 Running time (min) of all approaches on different sequences.

Sequence	IVT	KCF	CT	CLRST	MTT	ST	LRGC
Dudek	1.95	1.28	2.51	28.25	14.38	30.25	4.83
David	0.95	0.83	0.91	25.41	17.7	10.5	1.53
CarDark	0.83	0.65	0.76	9.5	4.36	13.65	1.28
Fish	1.01	0.86	0.99	25.76	15.51	19.97	1.51

For fair comparison, we use the compromise values as the default settings of these parameters in our approach to compare it with other related work in Sect. 4.2. For instances, K is set to 13 for Dudek sequence and 6 for Fish sequence respectively, and the number of samples for each cluster of $C_1, C_2, \cdots, C_K$ is set to 3 for all sequences.

4.2 Performance Comparison

To fully evaluate our approach, we compare it with 6 recently proposed algorithms. Incremental Visual Tracking (IVT) [6], Consistent Low-Rank Sparse Tracker (CLRST) [27], Multi-task Sparse Tracking (MTT) [26], Superpixel tracking (ST) [33], Kernelized Correlation Filters (KCF) [34] and Compressive Tracking (CT) [35]. IVT applies PCA to achieve and update the subspace online. CLRST represents particles as sparse linear combinations of dictionary templates to capture the low-rank structure of data. MTT formulates object tracking in a particle filter framework and deploys sparse learning to learn a robust subspace. ST is based on mean shift method. KCF formulate tracking as a ridge regression problem for correlation filter learning and apply HOG [36] in the visual model. CT uses an appearance model based on features extracted from the multi-scale image feature space with data-independent basis. For fair comparison, we use the source codes of all approaches with the default settings of the parameters provided by the authors. The initial parameters of the tracking window in the first frame of the testing sequence are same for all the approaches.

The running time of all approaches is shown in Table 2. It is worth noticing that the running time of all approaches in our experiment includes the time of saving the tracking result of each frame into the disk, thus the speeds of KCF and CT are not so fast as that are presented in [34]. We can see from Table 2 that the running time of IVT, KCF and CT is short and similar. The running time of our approach is close to that of these three approaches and it is much shorter than that of CLRST, MTT and ST. Our approach learns the subspace based on the samples achieved recently and other training samples. Thus the computation cost would not increase dramatically with the accumulation of observations.

We show the center distance (in pixels) between the tracking result and the ground truth over time for all approaches on 3 video sequences in Fig. 8. Some selected tracking results of all approaches are also shown in Fig. 9, Fig. 10, Fig. 11 and Fig. 12, respectively. Since the selected frames are the most representative or challenging ones in the video sequence, the results can reflect the global per-

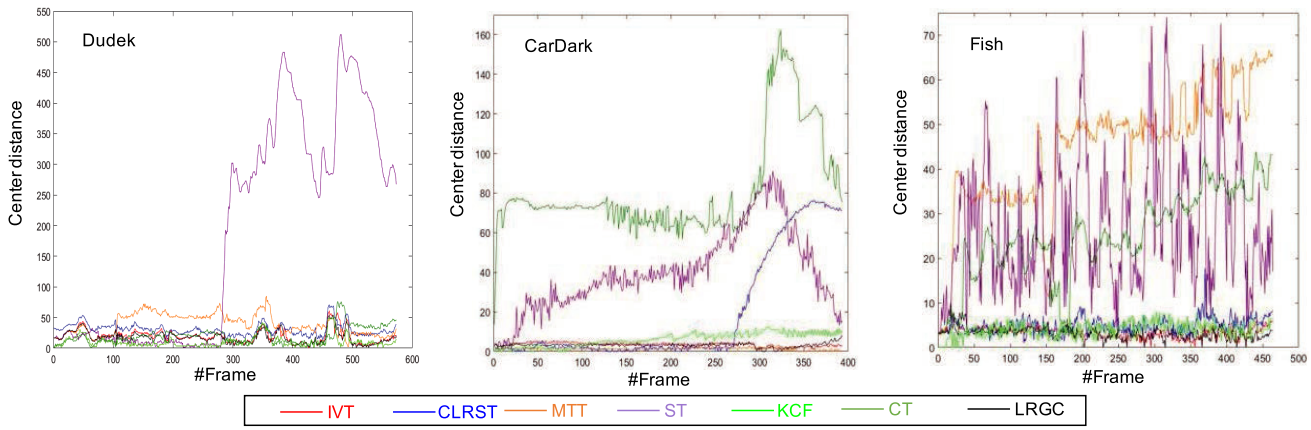


Fig. 8 Center distance (in pixels) between the center of the tracked target and the center of the ground truth over time for all approaches applied to 3 video sequences.

Fig. 9 Selected tracking results of (a) IVT, (b) CLRST, (c) MTT, (d) ST, (e) KCF, (f) CT and (g) our approach on Dudek sequence at frame #107, #184, #380, #470 and #570. The '×' in (d) indicates tracking failure, i.e., the tracking window is out of the image.

formance of different approaches. From the results, some discussions can be drawn as follows:

1) From the tracking results of faces in Fig. 9 and Fig. 10, we can see that our approach achieves good performances on both sequences. KCF perform good on

Fig. 10 Selected tracking results of (a) IVT, (b) CLRST, (c) MTT, (d) ST, (e) KCF, (f) CT and (g) our approach on David sequence at frame #31, #85, #120, #161 and #200.

Dudek sequence, but the locating rectangle is much bigger than the object on David sequence when the object moves abruptly, which indicates its ability of discriminating the object from the background becomes weaker when the object changes suddenly. CT works well in early stage when the object changes smoothly, but there are some biases on Dudek sequence and the locating rectangle is also much bigger than the object on both sequences when the object changes abruptly. IVT can successfully capture the smooth changes of object, and it can also handle small changes. However, there are still some biases when IVT tracks the profile of Dudek. The eigenbasis of IVT can capture the appearance details of the target in different poses, expressions, and with or without glasses, but the eigenbasis learned in IVT by PCA cannot preserve data manifold structure. Thus, when the target changes dramatically, e.g., from frontal face to profile, IVT may fail to effectively adapt to the changes. Since both CLRST and MTT deploy sparse learning to learn a robust subspace, their performances are also similar. ST works perfect in earlier stage. It uses an appearance model to distinguish the foreground target and the background, but it does not consider much about the change of the foreground target. Therefore, it fails when the foreground target changes abruptly.

2) Fig. 11 shows that both IVT and our approach perform well in dark environment. MTT also works well by dynamically updating particles. There are some biases in the tracking results (e.g., frame #123, #223, #280 and #355) of KCF. It seems that the tracking accuracy of KCF degrades when the object moves fast and the illumination changes frequently. Compared with IVT, MTT and our approach, the locating rectangle drawn by CLRST is much bigger, which means its discrimination ability is weak when features of the foreground and the

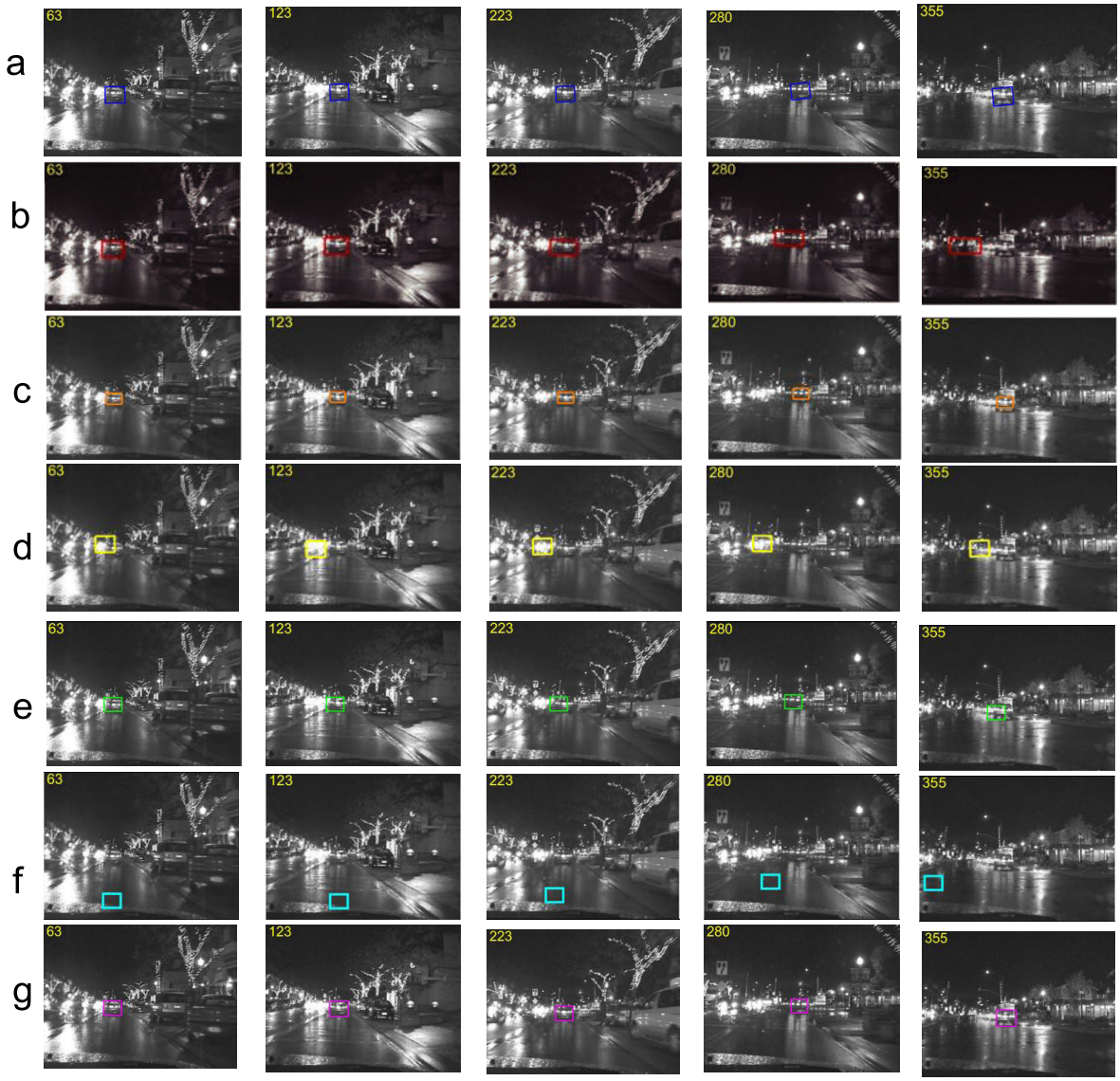


Fig. 11 Selected tracking results of (a) IVT, (b) CLRST, (c) MTT, (d) ST, (e) KCF, (f) CT and (g) our approach on CarDark sequence at frame #63, #123, #223, #280 and #355.

background are similar. In addition, it seems that ST is sensitive to target which has obvious difference with background. CT cannot capture the object on CarDark sequence and it seems that CT is not suitable for the environment with large illumination changes.

3) It can be seen from Fig. 12 that IVT, CLRST, KCF and our approach can steady handle the camera motion, but MTT, ST and CT perform comparatively bad when the differences between the foreground and the background are small.

4) IVT, KCF and CT runs fast, but the accuracy of tracking may degrade in some circumstances when the state of the object changes abruptly. The running time of our approach is close to that of these approaches. Moreover, our approach is robust to local changes of the target object by constructing a within-cluster graph to preserve the relationship of objects in similar modalities. It is also robust to abrupt changes of the target object by constructing a between-cluster graph to forcibly keep connections between current appearance (C_{est}) and other possible appearances. Besides, low-rank representation in our model helps to revel the common latent features shared by different appearances.

5) ST needs to generate the super-pixel regions in each tracking, thus it needs a higher cost of computation and storage and a relatively long time to complete the tracking task, while our approach is computational friendly by constructing a compact graph and projecting samples to a low-dimensional space.

Although our approach is robust to both local changes

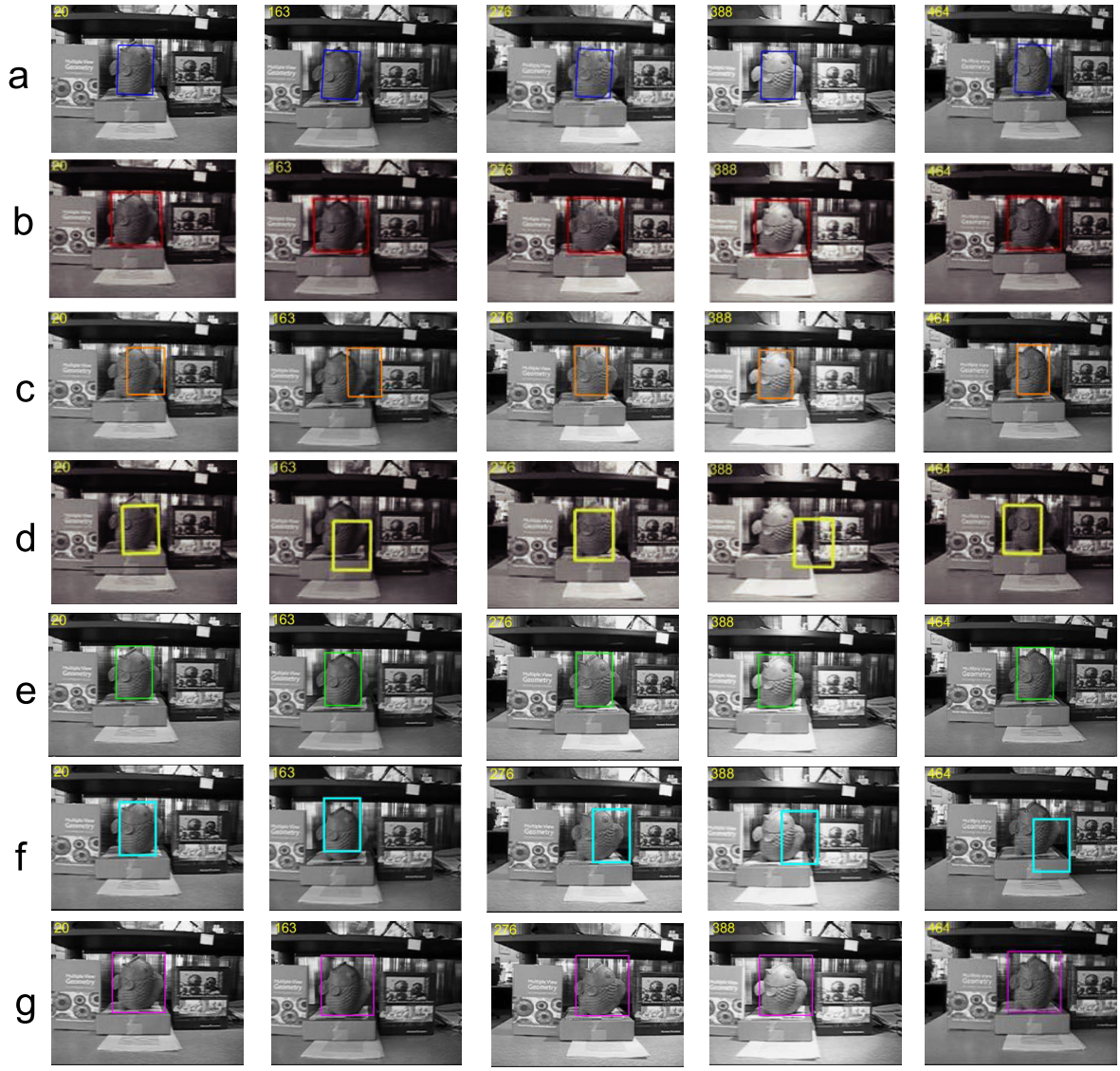


Fig. 12 Selected tracking results of (a) IVT, (b) CLRST, (c) MTT, (d) ST, (e) KCF, (f) CT and (g) our approach on Fish sequence at frame #20, #163, #276, #388 and #464.

and abrupt changes of the object in many circumstances, it needs to collect different observation states before tracking. This precondition may be not easy to satisfy in some circumstances. How to overcome this disadvantage is the work we need to do in the future.

5. Conclusion

This paper proposes a robust visual tracking algorithm by low-rank representation with graph constraints. It exploits the benefits from both feature selection and geometric structure preservation. Experimental evaluations on different faces and objects demonstrate that our approach is robust to both local changes and dramatically changes of the target object. Since our model needs some training samples to construct the graphs, in our future work, we will explore how to automatically generate training samples by only one sample, e.g., variation of pose and scale, shape deformation and occlusion.

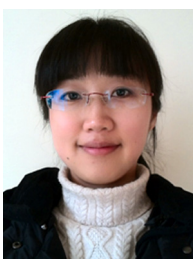
Acknowledgments

This work was supported in part by the National Science Foundation of China under Grant No. 61371183 and 61501096, the International Science and Technology Cooperation and Exchange Program of Sichuan Province under Grant No. 2016HH0020, the Fundamental Research Funds for the Central Universities under No. ZYGX2016J089, and the Applied Basic Research Program of Sichuan Province under Grant No. 2015JY0124.

References

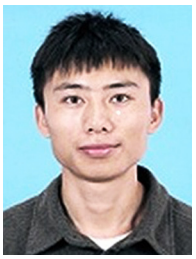
[1] M.L. Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration

of texture-mapped 3d models," IEEE Trans. Pattern Anal. Mach. Intell., vol.22, no.4, pp.322–336, 2000.
[2] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.232–237, 1998.
[3] M. Harville, "A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models," European Conference on Computer Vision, pp.543–560, 2002.
[4] B. Georgescu, D. Comaniciu, T.X. Han, and X.S. Zhou, "Multi-model component-based tracking using robust information fusion," International Workshop on Statistical Methods in Video Processing, pp.61–70, 2004.
[5] J. Ho, K.-C. Lee, M.-H. Yang, and D. Kriegman, "Visual tracking using learned linear subspaces," Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition., pp.782–789, 2004.
[6] D.A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," Int. J. Comput. Vision., vol.77, no.1-3, pp.125–141, 2008.
[7] M.J. Black and A.D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," Int. J. Comput. Vis., vol.26, no.1, pp.63–84, 1998.
[8] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, "Robust tracking using local sparse appearance model and k-selection," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1313–1320, 2011.
[9] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: a general framework for dimensionality reduction," IEEE Trans. Pattern Anal. Mach. Intell., vol.29, no.1, pp.40–51, 2007.
[10] X. Zhang, W. Hu, S. Maybank, and X. Li, "Graph based discriminative learning for robust and efficient object tracking," IEEE International Conference on Computer Vision, pp.1–8, 2007.
[11] H. Qiao, P. Zhang, B. Zhang, and S. Zheng, "Learning an intrinsic-variable preserving manifold for dynamic visual tracking," IEEE Trans. Syst., Man, Cybern., Part B (Cybernetics), vol.40, no.3, pp.868–880, 2010.
[12] D. Cai, X. He, and K. Zhou, "Locality sensitive discriminant analysis," International Joint Conference on Artificial Intelligence, pp.708–713, 2007.
[13] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," IEEE Trans. Pattern Anal. Mach. Intell., vol.35, no.1, pp.171–184, 2013.
[14] D. Cai, X. He, J. Han, and T.S. Huang, "Graph regularized nonnegative matrix factorization for data representation," Trans. Pattern Anal. Mach. Intell., vol.33, no.8, pp.1548–1560, 2011.
[15] X. Zhu and J. Lafferty, "Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning," International Conference on Machine Learning, pp.1052–1059, 2005.
[16] F.R. Chung, "Spectral graph theory," American Mathematical Society, vol.9, no.6, p.212, 1997.
[17] X. He and P. Niyogi, "Locality preserving projections," Advances in Neural Information Processing Systems, vol.16, no.1, pp.186–197, 2002.
[18] S. Li and Y. Fu, "Robust subspace discovery through supervised low-rank constraints.," SIAM International Conference on Data Mining, pp.163–171, 2014.
[19] J. Li, Y. Wu, J. Zhao, and K. Lu, "Low-rank discriminant embedding for multiview learning," IEEE Trans. Cybern., pp.1–14, 2016.
[20] Z. Ding, M. Shao, and Y. Fu, "Deep low-rank coding for transfer learning," Proc. International Joint Conference on Artificial Intelligence, pp.3453–3459, 2015.
[21] E.J. Candès, X.D. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," J. ACM (JACM), vol.58, no.3, p.1–37, 2011.
[22] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," Int. J. Comput. Vision., vol.109, no.1-2, pp.74–93, 2014.
[23] Z. Ding and Y. Fu, "Low-rank common subspace for multi-view learning," IEEE International Conference on Data Mining, pp.110–119, 2014.
[24] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," IEEE Trans. Pattern Anal. Mach. Intell., vol.19, no.7, pp.711–720, 1997.
[25] J. Zhao, J. Li, and K. Lu, "Robust visual tracking using sparse discriminative graph embedding," IEICE Trans. Inf. & Syst., vol.E98-D, no.4, pp.938–947, 2015.
[26] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2042–2049, 2012.
[27] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem, "Robust visual tracking via consistent low-rank sparse learning," Int. J. Comput. Vision., vol.111, no.2, pp.171–190, 2015.
[28] K. Lu, Z. Ding, and S. Ge, "Locally connected graph for visual tracking," Neurocomputing, vol.120, pp.45–53, 2013.
[29] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," Comput. Math. Appl., vol.2, no.1, pp.17–40, 1976.
[30] Q. Gu, Z. Li, and J. Han, "Joint feature selection and subspace learning," International Joint Conference on Artificial Intelligence, pp.1294–1299, 2011.
[31] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," Tech. Rep., UIUC, 2009.
[32] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," ACM symposium on Theory of computing, pp.1–6, 1987.
[33] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," International Conference on Computer Vision, pp.1323–1330, 2011.
[34] J.F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.3, pp.583–596, 2015.
[35] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," European Conference on Computer Vision. Lecture Notes in Computer Science, vol.7574, pp.864–877, 2012.
[36] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.886–893, 2005.



Jieyan Liu received her M.S. and Ph.D. degrees in Computer Application Technology from University of Electronic Science and Technology of China in 2004 and 2012, respectively. She is now a lecturer with the school of computer science and technology, University of Electronic Science and Technology of China. Her research interests include computer vision and pattern recognition.

Ao Ma received his B.S. degree in Computer Science in 2015 from the University of Electronic Science an Technology of China where he is studying for M.S. degree. He has a great interest in Machine Learning and Computer Vision.



Jingjing Li received his M.Sc. degree in Computer Science from University of Electronic Science and Technology of China in 2013. Now he is a Ph.D candidate in School of Computer Science and Technology, University of Electronic Science and Technology of China. He has a great interest in computer vision, especially video-based data analysis, and visual object tracking. He is a student member of IEEE and IEICE.

Ke Lu received his B.S. degree in thermal power engineering from Chongqing University, China in 1996. He obtained his M.Sc. and Ph.D degrees in Computer Application Technology from the University of Electronic Science and Technology of China, in 2003 and 2006, respectively. He is currently a professor in School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include pattern recognition, multimedia and computer vision.