Semi-Supervised Tensor-Based Graph Embedding Learning and Its Application to Visual Discriminant Tracking

Weiming Hu, Jin Gao, Junliang Xing, and Chao Zhang
(CAS Center for Excellence in Brain Science and Intelligence Technology, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190)
{wmhu, jin.gao, jlxing}@nlpr.ia.ac.cn; 861201030@qq.com

Stephen Maybank
(Department of Computer Science and Information Systems, Birkbeck College, Malet Street, London WC1E 7HX)
sjmaybank@dcs.bbk.ac.uk

Abstract: An appearance model adaptable to changes in object appearance is critical in visual object tracking. In this paper, we treat an image patch as a 2-order tensor which preserves the original image structure. We design two graphs for characterizing the intrinsic local geometrical structure of the tensor samples of the object and the background. Graph embedding is used to reduce the dimensions of the tensors while preserving the structure of the graphs. Then, a discriminant embedding space is constructed. We prove two propositions for finding the transformation matrices which are used to map the original tensor samples to the tensor-based graph embedding space. In order to encode more discriminant information in the embedding space, we propose a transfer-learning-based semi-supervised strategy to iteratively adjust the embedding space into which discriminative information obtained from earlier times is transferred. We apply the proposed semi-supervised tensor-based graph embedding learning algorithm to visual tracking. The new tracking algorithm captures an object's appearance characteristics during tracking and uses a particle filter to estimate the optimal object state. Experimental results on the CVPR 2013 benchmark dataset demonstrate the effectiveness of the proposed tracking algorithm.

Index terms: Discriminant tracking, Tensor samples, Semi-supervised learning, Graph embedding space

1. Introduction

Visual tracking [48, 49, 50, 51] is an essential component in many practical computer vision applications, such as visual surveillance, vehicle navigation, vision-based control, human computer interfaces, intelligent transportation, and augmented reality. Despite much effort resulting in many novel tracking algorithms, tracking generic objects remains challenging [31] because of the intrinsic and extrinsic appearance changes. The intrinsic appearance changes are caused by objects themselves, for example if they deform or rotate. The extrinsic appearance changes are associated with the environment of the objects, and the causes of such changes include partial occlusions, illumination changes, and cluttered or moving backgrounds. The effective modeling of object appearance variations plays a critical role in visual object tracking [1, 2, 3, 6, 8, 9, 13, 21]. Many tracking algorithms construct an adaptive appearance model for an object based on image patches collected in previous frames, and use this model to find the most likely image patch on the object in the current frame. Appearance

models can be categorized into generative models and discriminative models [45]. A generative model describes the distribution of the image patches of an object in previous frames. Tracking is reduced to a search for an optimal state that yields an object appearance most similar to the model. A discriminative model describes not only the image patches of the object but also some background image patches. The tracking is based on a binary classifier to distinguish the object from the background. Usually, the image patches corresponding to the tracking results are used as the labeled positive samples, and the image regions selected from the background are used as the labeled negative samples [6, 10]. Then, a classifier is trained in a supervised way using these labeled positive and negative samples. A number of new samples can be selected in each new frame, and these unlabeled samples are used in a semi-supervised way [6, 10] to improve the classifier.

In this paper, we propose a new discriminative tracking algorithm in which each image patch is represented by a 2-order tensor. The relations among object image patches and background image patches in the previous frames are represented by graphs. The 2-order tensor-based graph embedding is used to learn the discriminative subspace (the discriminative embedding space) for distinguishing the object image patches from the background image patches. A number of unlabeled image patches collected from the current frame are used to refine the discriminative embedding space.

1.1. Related work

In order to give the context of our work, we briefly review image-as-vector representation, image-as-feature representation, and image-as-matrix representation for image patches, together with semi-supervised strategies for discriminant tracking.

1) Image-as-vector representation: Many tracking algorithms [21, 35] adopt a holistic image-as-vector representation in which image patches are flattened to vectors. These algorithms include ℓ_1 minimization-based tracking algorithms [3, 17, 18, 33, 34], which exploit the sparse representation of image patches on the object. Kwon and Li [8, 9] constructed multiple basic appearance models using sparse principal component analysis (PCA) of a set of feature templates, such as global image-as-vector descriptors of hue, saturation, intensity, and edge information. These image-as-vector representation methods ignore the fact that an image is intrinsically a matrix or a 2-order tensor, and thus mask the underlying 2D structure of an image. This may lead to the loss of the discriminant information required for tracking.

2) Image-as-feature representation: There are algorithms that directly extract features from image patches and use the features to carry out tracking. For instance, Grabner et al. [5, 6] used Haar-like features, histograms of oriented gradients, and local binary patterns to obtain weak hypotheses for boosting-based tracking. In [2, 10] only Haar-like features were used, but great improvements were achieved by novel appearance models. Li et al. [12, 15] only used histograms of oriented gradients but applied novel appearance models to achieve good results. Adam et al. [1] robustly combined multiple patch votes with each image patch represented only by gray-scale

histogram features. Although image-as-feature representations, such as Haar-like featured-based ones, are appropriate for specifically designed discriminant learning methods, such as boosting, a lot of useful information is omitted from the features.

3) Image-as-matrix presentation: image-as-matrix representation, such as tensor representation, retains much more useful information because the original image structure is preserved. It usually models the relations between pixel rows and the relations between pixel columns while the image-as-vector representation usually only models the relations between pixels. Tensor-based subspace learning [7, 27, 29] has been applied to visual tracking [11, 13, 24, 25, 26]. Some tensor-based tracking algorithms [13, 24, 25] conducted PCA on the mode-*k* unfolding matrix. There are algorithms [11, 26] in which eigenvalue decomposition on the covariance [19] in the mode-*k* unfolding matrix was applied to carry out tracking tasks. Although these tensor-based tracking methods take into account the correlations between different dimensions of the object appearance which changes over time, they currently have the following limitations:

- The dimension reduction-based subspace learning methods used in [13, 24, 25] have the problem of subspace learning degradations [27] which may lead to loss of tracking. Although Yan et al. [27] proposed to rearrange pixels in the tensor to deal with subspace learning degradation, the time consuming pixel rearrangement is unsuitable for tracking applications.
- The current tensor-based tracking algorithms cannot fully detect the intrinsic local geometrical and discriminative structure of the collected image patches in tensor form.
- The current tensor-based tracking algorithms ignore the influence of the background and consequently suffer from distractions caused by background regions with appearances similar to the object.

Two-dimensional linear discriminant analysis (2DLDA) [30] was proposed to detect the discriminative structure of 2-order tensor samples. However, the intrinsic local geometrical structure of the samples cannot be detected, because 2DLDA does not take into account the variations in the samples within the same class.

Discriminant tracking utilizes the background information to improve object tracking. Avidan [46, 47] made important contributions to the research on discriminative tracking. In [46], Avidan trained online an ensemble of weak classifiers to distinguish between the object and the background. The resulting strong classifier was then used to label pixels in the next frame as either belonging to the object or the background. In [47], Avidan integrated the support vector machine classifier into an optic-flow-based tracker. In discriminant tracking, semi-supervised learning techniques can be applied to improve the classification of samples as the object or the background. Most semi-supervised improvement-based tracking algorithms [6, 10, 12] use all the unlabeled samples for training without selection. For example, Grabner et al. [6] weighted the unlabeled samples and then used all the unlabeled samples to train their feature selection-based boosting classifier. Li et al [10] developed an excellent semi-supervised CovBoost method for discriminant tracking. As not all the unlabeled samples are

useful for improving the classifier, it is required to select useful unlabeled samples and assign correct class labels to them. Bennett et al. [4] and Rosenberg et al. [20] proposed the classification margin improvement techniques [4, 20] to select the unlabeled samples with the highest classification confidences. These unlabeled samples were assigned with the class labels that are predicted by the current classifier. These techniques may increase the classification margin, but they do not provide any novel information to adjust the discriminative subspace. The margin improvement techniques are inappropriate for visual tracking.

1.2. Our work

Graph embedding for dimension reduction [22, 28] provides a new framework to handle the limitations in the image-as-matrix representations. In this paper, we propose a semi-supervised tensor-based graph embedding learning algorithm and apply it to visual discriminant tracking [38]. Fig. 1 shows the high level framework for our tracker. Our algorithm treats an image patch as a 2-order tensor. The labeled object and background tensor samples collected in previous frames and the unlabeled tensor samples collected in the current frame are used to train a tensor-based graph embedding space by our proposed semi-supervised graph embedding learning algorithm. This graph embedding algorithm is incorporated into the particle filtering-based tracking framework to track the object in the current frame. The tracking result in the current frame is used to update the set of the labeled samples.

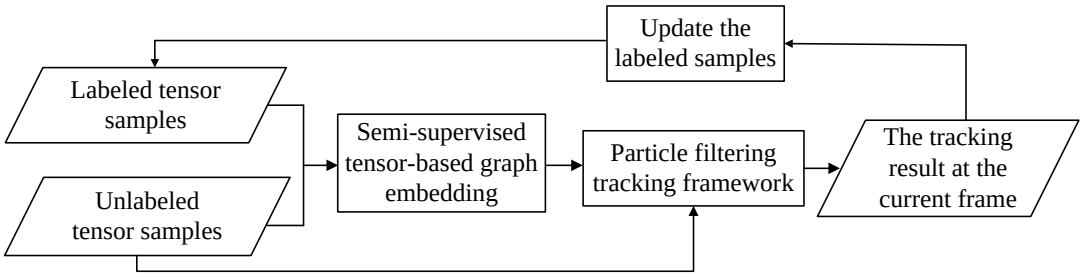


Fig. 1. The high level framework for our tracker.

In our 2-order tensor-based graph embedding learning algorithm, an intrinsic graph is designed to represent correlations between the object tensor samples and correlations between the background tensor samples. A penalty graph is designed to keep apart the object tensor samples and the background tensor samples. The framework of graph embedding for dimension reduction is used to find the transformation matrices for mapping the original tensor samples to the tensor-based graph embedding space. Due to the degradations in the tensor embedding space learning [27], the embedding may not contain enough discriminative information for reliable tracking. We propose to encode more discriminant information by improving the tensor-based discriminative embedding space using the available unlabeled tensor samples in a semi-supervised way. In discriminant tracking, the existing single transformation matrix-based method [36] improves the discriminant space only by adding to the objective function a constraint term for handling the unlabeled samples. This method is difficult to use intuitively and directly in the 2-order tensor-based graph embedding learning for which two transformation

matrices are required, because it is difficult to define an appropriate regularizer for handling the two associated transformation matrices. In this paper, we carry out a transfer-learning-based semi-supervised improvement in an iterative way. At each iteration, the unlabeled tensor samples which are most probably misclassified at the previous iteration are selected according to their similarities to the labeled samples which are collected at earlier tracking stages. The corrected class labels are assigned to the selected unlabeled samples and used to learn a new discriminative embedding space into which the information about the earlier changes in object appearance is transferred. The learned embedding spaces from different iterations are combined linearly to form a final adjusted embedding space.

The main contributions of our work are summarized as follows:

- We develop a 2-order tensor-based graph embedding learning algorithm. The intrinsic local geometrical and discriminative structures of the tensor samples are effectively represented.
- We propose a transfer-learning-based semi-supervised learning method to adjust the 2-order tensor graph embedding space. The semi-supervised technique selects the unlabeled tensor samples by comparing them with the samples with known classes collected at earlier times. Historical complementary information is transferred to adjust the discriminative embedding space.
- We incorporate the proposed semi-supervised 2-order tensor-based graph embedding learning procedure into a Bayesian inference framework. Then, a new visual discriminative tracker is formed to effectively capture the appearance changes and reliably separate a moving object from the background.

The remainder of the paper is organized as follows: Section 2 proposes our 2-order tensor-based graph embedding learning algorithm. Section 3 presents the transfer-learning-based semi-supervised improvement strategy. Section 4 describes our discriminant tracking algorithm. Section 5 demonstrates the experimental results. Section 6 summarizes the paper.

2. Tensor-Based Graph Embedding

In the following, we summarize the tensor operations, introduce the basic concept of tensor-based graph embedding, and propose our 2-order tensor-based graph embedding learning algorithm.

2.1. Tensor operations

A tensor [23] can be regarded as a multi-order "array" lying in multiple vector spaces. An *n*-order tensor is denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_k \times \ldots I_n}$, where I_k $(k = 1, 2, ..., n)$ is a positive integer. An element in the tensor is represented as $a_{i_1,\ldots,i_k,\ldots,i_n}$, where $1 \le i_k \le I_k$. The inner product of two *n*-order tensors $\mathcal{A}$ and $\mathcal{B}$ is defined as:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_n=1}^{I_n} a_{i_1,i_2,\ldots,i_n} b_{i_1,i_2,\ldots,i_n} \ . \tag{1}$$

The norm of $\mathcal{A}$ is $\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$, where for a 2-order tensor it is called the Frobenius norm and written as

$\|\mathcal{A}\|_F$. The distance between $\mathcal{A}$ and $\mathcal{B}$ is $\|\mathcal{A}-\mathcal{B}\|$.

Each order of a tensor is associated with a "mode". Along a mode *k*, a tensor is unfolded into a matrix $\mathbf{A}_{(k)} \in \mathbb{R}^{I_k \times (\prod_{i \neq k} I_i)}$ which consists of I_k-dimensional mode-*k* column vectors obtained by varying the *k*-th mode index i_k and keeping the indices of the other modes fixed. The inverse of mode-*k* unfolding is mode-*k* folding, which restores the original tensor $\mathcal{A}$ from $\mathbf{A}_{(k)}$. The mode-*k* product $\mathcal{A} \times_k \mathbf{M}$ of a tensor $\mathcal{A}$ and a matrix $\mathbf{M} \in \mathbb{R}^{J_k \times I_k}$ is a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \ldots \times I_{k-1} \times J_k \times I_{k+1} \times \ldots \times I_n}$ whose entries are:

$$c_{1,\ldots,i_{k-1},j,i_{k+1},\ldots,i_n} = \sum_{i=1}^{I_k} a_{i_1,\ldots,i_{k-1},i,i_{k+1},\ldots,i_n} M_{ji}, \; j = 1,\ldots,J_k \,. \tag{2}$$

The tensor $\mathcal{C}$ can also be obtained by matrix multiplication $\mathbf{C}_{(k)} = \mathbf{M}\mathbf{A}_{(k)}$ and then mode-*k* folding of $\mathbf{C}_{(k)}$. Given a tensor $\mathcal{A}$ and three matrices $\mathbf{\Gamma} \in \mathbb{R}^{J_n \times I_n}$, $\mathbf{\Psi} \in \mathbb{R}^{K_n \times J_n}$, and $\mathbf{Z} \in \mathbb{R}^{J_m \times I_m}$ ($n \neq m$), the following tensor mode-*n* product formulae hold $(\mathcal{A} \times_n \mathbf{\Gamma}) \times_m \mathbf{Z} = (\mathcal{A} \times_m \mathbf{Z}) \times_n \mathbf{\Gamma} = \mathcal{A} \times_n \mathbf{\Gamma} \times_m \mathbf{Z}$ and $(\mathcal{A} \times_n \mathbf{\Gamma}) \times_n \mathbf{\Psi} = \mathcal{A} \times_n (\mathbf{\Psi}\mathbf{\Gamma})$.

2.2. Tensor-based graph embedding

Let $\{\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}\}_{i=1,2,\ldots,N}$ be the set of *N* training samples in the *n*-order tensor form. An intrinsic graph $\mathcal{G}$ and a penalty graph $\mathcal{G}^p$, both of which have the vertex set $\{\mathcal{X}_i\}_{i=1,2,\ldots,N}$, are constructed to model the local geometrical and discriminative structure of tensor samples. Let $\mathbf{W}$ and $\mathbf{W}^p$ be the edge weight matrices of $\mathcal{G}$ and $\mathcal{G}^p$, respectively. The entry W_{ij} in W measures the similarity between vertices $\mathcal{X}_i$ and $\mathcal{X}_j$, and the entry W_{ij}^p in $\mathbf{W}^p$ measures their difference. The intrinsic graph describes desired statistical or geometric properties of the samples. The penalty graph characterizes statistical or geometric properties to be suppressed. The graphs $\mathcal{G}$ and $\mathcal{G}^p$ are designed according to the application [40].

The task of the tensor-based graph embedding [28] is to find an optimal low dimensional tensor representation for each vertex in a graph $\mathcal{G}$ such that the low dimensional tensor representations optimally characterize the similarities between the vertex pairs. The characteristics of the original intrinsic graph are preserved and at the same time the characteristics identified by the penalty graph are suppressed. Let $\{\mathbf{M}^k \in \mathbb{R}^{l_k \times I_k}\}_{k=1,2,\ldots,n, l_k < I_k}$ be a set of *n* transformation matrices which map the samples $\{\mathcal{X}_i\}_{i=1,2,\ldots,N}$ to *N* points $\{\mathcal{Y}_i \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_n}\}_{i=1,2,\ldots,N}$ in a lower dimensional tensor subspace. Namely $\mathcal{Y}_i = \mathcal{X}_i \times_1 \mathbf{M}^1 \times_2 \mathbf{M}^2 \ldots \times_n \mathbf{M}^n$. Then, an optimal transformation which preserves the graph structure **is** obtained by solving the following optimization:

$$\begin{aligned} \left\{\mathbf{M}^k\right\}_{k=1}^{n} &= \arg\min\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\mathcal{Y}_i - \mathcal{Y}_j\right\|^2 W_{ij}\right) \\ \text{subject to } & \sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\mathcal{Y}_i - \mathcal{Y}_j\right\|^2 W_{ij}^p = d \end{aligned} \tag{3}$$

where *d* is a constant.

2.3. Two order tensor-based graph embedding learning

In many applications, such as discriminant tracking, the samples can be expressed in the 2-order tensor form: $\{\boldsymbol{\mathcal{X}}_i \in \mathbb{R}^{I_1 \times I_2}\}_{i=1,2,...,N}$, and the sample set consists of labeled training samples and unlabeled samples. The unlabeled samples are usually given pseudo labels which are predicted by a classifier. Let "+1" indicate the labeled positive samples, and "–1" indicate the labeled negative samples. Let "+2" indicate the pseudo positive samples, and "–2" indicate the pseudo negative samples. Then, the class labels of the training samples are represented by $\{L_i, i=1,2,...,N\}$ $(L_i \in \{-2,-1,+1,+2\})$. Let n_c be the number of the samples with class $c \in \{-2,-1,+1,+2\}$. Then, $\sum_{c=-2}^{+2} n_c = N$.

The intrinsic graph in the graph embedding framework characterizes the intra-class compactness. The penalty graph characterizes the interclass separability. The distributions of samples, such as the background samples in tracking applications, are disordered, irregular, and multimodal. The global linking [36] between samples for defining the graph structure may not be effective for classes which are widely scattered. As a result, its ability to maximize the between-class scatters is limited. Local linking between samples for defining the graph structure is more appropriate to preserve the scattered sample distributions. In this paper, we design the two graphs, $\mathcal{G}$ and $\mathcal{G}^p$, to model the local geometrical and discriminative structure of the tensor samples.

2.3.1. Definition of graph structure

We define the affinity A_{ij} between samples *i* and *j* using the local scaling method in [32]. Without loss of generality, we assume that the data points in $\{\boldsymbol{\mathcal{X}}_i\}_{i=1}^{N}$ are ordered according to their labels $\{L_i\}$ ($L_i \in \{-2,-1,+1,+2\}$). When $L_i > 0$ and $L_j > 0$, A_{ij} is defined as:

$$A_{ij} = \exp\left(-\frac{\left\|\boldsymbol{\mathcal{X}}_i - \boldsymbol{\mathcal{X}}_j\right\|^2}{\sigma_i \sigma_j}\right) \tag{4}$$

where $\sigma_i = \left\|\boldsymbol{\mathcal{X}}_i - \boldsymbol{\mathcal{X}}_i^{(k)}\right\|$, and $\mathcal{X}_i^{(k)}$ is the *k*-th nearest neighbor of $\boldsymbol{\mathcal{X}}_i$ in $\{\boldsymbol{\mathcal{X}}_j\}_{j=n_{-2}+n_{-1}+1}^{N}$. The affinities for a positive or pseudo positive sample are defined for all the positive and pseudo positive samples. When $L_i < 0$ and $L_j < 0$, A_{ij} is defined as:

$$A_{ij} = \begin{cases} \exp\left(-\dfrac{\| \boldsymbol{\mathcal{X}}_i - \boldsymbol{\mathcal{X}}_j \|^2}{\sigma_i \sigma_j}\right), & \text{if } i \in N_k^+(j) \text{ or } j \in N_k^+(i) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $N_k^+(i)$ is the index set of the *k*-nearest neighbors of $\boldsymbol{\mathcal{X}}_i$ in $\{\boldsymbol{\mathcal{X}}_j\}_{j=1}^{n_{-2}+n_{-1}}$, $\sigma_i = \left\|\boldsymbol{\mathcal{X}}_i - \boldsymbol{\mathcal{X}}_i^{(k)}\right\|$, and $\boldsymbol{\mathcal{X}}_i^{(k)}$ is the *k*-th nearest neighbor of $\mathcal{X}_i$ in $\{\boldsymbol{\mathcal{X}}_j\}_{j=1}^{n_{-2}+n_{-1}}$. The affinities for a negative or pseudo negative sample are only

defined for its *k*-nearest negative or pseudo negative neighbors. This definition reflects the local spatial relation between negative and pseudo negative samples.

We construct the intrinsic graph $\mathcal{G}$ by defining the weights W_{ij} in $\mathbf{W}$ of $\mathcal{G}$. When $i=j$, $\left\|\mathcal{Y}_i-\mathcal{Y}_j\right\|=0$, and then the value of W_{ii} does not influence the optimization in (3). So, we only consider the definition of W_{ij} when $i\neq j$. The weights are defined as follows:

$$W_{ij}=\begin{cases}\dfrac{A_{ij}}{n_c}, & \text{if } L_i=L_j=c\\ \dfrac{A_{ij}}{\sqrt{n_{+1}n_{+2}}}, & \text{if } L_i>0,\ L_j>0,\ and\ L_i\neq L_j\\ \dfrac{A_{ij}}{\sqrt{n_{-1}n_{-2}}}, & \text{if } L_i<0,\ L_j<0,\ and\ L_i\neq L_j\\ 0, & \text{otherwise}\end{cases}. \tag{6}$$

When $L_iL_j>0$, the nearby data pairs (large values of A_{ij}) are assigned large positive weights, and the data pairs far apart (small values of A_{ij}) are assigned small positive weights. When $L_iL_j<0$, there is no linking between the samples.

We construct the penalty graph $\mathcal{G}^p$ by defining the elements $W_{ij}^p(i\neq j)$ of $\mathbf{W}^p$ of $\mathcal{G}^p$ as follows:

$$W_{ij}^p=\begin{cases}A_{ij}\left(\dfrac{1}{N}-\dfrac{1}{n_c}\right), & \text{if } L_i=L_j=c,\\ A_{ij}\left(\dfrac{1}{N}-\dfrac{1}{\sqrt{n_{+1}n_{+2}}}\right), & \text{if } L_i>0,\ L_j>0,\ and\ L_i\neq L_j\\ A_{ij}\left(\dfrac{1}{N}-\dfrac{1}{\sqrt{n_{-1}n_{-2}}}\right), & \text{if } L_i<0,\ L_j<0,\ and\ L_i\neq L_j\\ \dfrac{1}{N}, & \text{otherwise}\end{cases}. \tag{7}$$

When $L_iL_j>0$, the nearby data pairs are assigned more negative weights. When $L_iL_j<0$, the data pairs are assigned a positive weight $1/N$. As the penalty graph is used to separate the object from the background, we assign a positive weight to each pair of samples whose labels have different signs ("+" or "-") in order to increase separability between the object samples and the background samples. We assign a negative weight to each pair of samples whose labels have the same sign, in order to draw closer the samples with the same sign.

Due to the definition of A_{ij}, local spatial relations between tensor samples are included in the graph structure. The weights of the graphs are averaged by the number of the samples with the same label. The effect of the unlabeled samples is reflected by the terms $1/n_c$, $1/\sqrt{n_{+1}n_{+2}}$, and $1/(\sqrt{n_{-1}n_{-2}})$ in (6) and (7). In this way, non-uniformity of the number of the labeled positive (or negative) samples and the number of the pseudo positive (or negative) samples is dealt with.

2.3.2. Solution

He et al. [7] proposed a tensor subspace analysis method in which only an adjacency graph, rather than an intrinsic graph and a penalty graph, is used. We extend the derivation in [7] to the framework of graph embedding, to reduce (3) to a more tractable form.

Instantiation of (3) for 2-order tensors yields:

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} \| \mathbf{U}^T\mathcal{X}_i\mathbf{V} - \mathbf{U}^T\mathcal{X}_j\mathbf{V} \|_F^2 \, W_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} \| \mathbf{U}^T\mathcal{X}_i\mathbf{V} - \mathbf{U}^T\mathcal{X}_j\mathbf{V} \|_F^2 \, W_{ij}^p = d \end{aligned} \tag{8}$$

where $\mathbf{U}^T = \mathbf{M}^1$ and $\mathbf{V}^T = \mathbf{M}^2$. We derive (8) in the tensor form. In the 2-order tensor case, $\mathcal{Y}_i = \mathcal{X}_i \times_1 \mathbf{M}^1 \times_2 \mathbf{M}^2$. The mode-1 unfolding of the tensor $\mathcal{A}$ is $\mathbf{A}_{(1)} = \mathcal{A}$ and the mode-2 unfolding is $\mathbf{A}_{(2)} = \mathcal{A}^T$. Denote $\mathcal{X}_i \times_1 \mathbf{M}^1$ as tensor $\mathcal{T}$. Then, $\mathcal{T}$ can be computed by matrix multiplication $\mathbf{T}_{(1)} = \mathbf{M}^1\mathcal{X}_{i(1)} = \mathbf{U}^T\mathcal{X}_i$ and mode-1 folding of $\mathbf{T}_{(1)}$. Then, $\mathcal{T} = \mathbf{U}^T\mathcal{X}_i$. Tensor $\mathcal{Y}_i$ can be computed by matrix multiplication $\mathbf{Y}_{i(2)} = \mathbf{M}^2\mathbf{T}_{(2)} = \mathbf{V}^T\mathcal{T}^T$ and mode-2 folding of $\mathbf{Y}_{i(2)}$. Then, $\mathcal{Y}_i$ can be derived as:

$$\mathcal{Y}_i = (\mathbf{Y}_{i(2)})^T = (\mathbf{V}^T\mathcal{T}^T)^T = \mathcal{T}\mathbf{V} = \mathbf{U}^T\mathcal{X}_i\mathbf{V}. \tag{9}$$

Substitution of (9) into (3) yields (8).

Proposition 1: Let $\mathbf{D}$ and $\mathbf{D}^p$ be diagonal matrices whose diagonal elements are $D_{ii} = \sum_{j=1}^{N} W_{ij}$ and $D_{ii}^p = \sum_{j=1}^{N} W_{ij}^p$, respectively. The optimization problem in (8) can be reformulated as the following optimization problem:

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & \text{trace}(\mathbf{V}^T(\mathbf{D}_U - \mathbf{W}_U)\mathbf{V}) \\ \text{subject to} \quad & \text{trace}(\mathbf{V}^T(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{V}) = \frac{d}{2} \end{aligned} \tag{10}$$

where

$$\mathbf{D}_U = \sum_{i=1}^{N} D_{ii}\mathcal{X}_i^T\mathbf{U}\mathbf{U}^T\mathcal{X}_i, \mathbf{W}_U = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\mathcal{X}_i^T\mathbf{U}\mathbf{U}^T\mathcal{X}_j \tag{11}$$

$$\mathbf{D}_U^p = \sum_{i=1}^{N} D_{ii}^p\mathcal{X}_i^T\mathbf{U}\mathbf{U}^T\mathcal{X}_i, \mathbf{W}_U^p = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^p\mathcal{X}_i^T\mathbf{U}\mathbf{U}^T\mathcal{X}_j \tag{12}$$

Proof: Referring to [7], the following equations are obtained:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \| \mathbf{U}^T\mathcal{X}_i\mathbf{V} - \mathbf{U}^T\mathcal{X}_j\mathbf{V} \|_F^2 \, W_{ij} = \text{trace}\left(\mathbf{V}^T\left(\mathbf{D}_U - \mathbf{W}_U\right)\mathbf{V}\right) \tag{13}$$

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \| \mathbf{U}^T\mathcal{X}_i\mathbf{V} - \mathbf{U}^T\mathcal{X}_j\mathbf{V} \|_F^2 \, W_{ij}^p = \text{trace}\left(\mathbf{V}^T\left(\mathbf{D}_U^p - \mathbf{W}_U^p\right)\mathbf{V}\right) \tag{14}$$

Substitution of (13) and (14) into (8) reformulates the optimization problem in (8) as the optimization problem in

(10). In (10), **U** and **V** can be swapped. Similar to (11) and (12), we define $\mathbf{D}_V$, $\mathbf{W}_V$, $\mathbf{D}_V^p$, and $\mathbf{W}_V^p$.

Proposition 2: When **V** is fixed, the optimal **U** is composed of the generalized eigenvectors corresponding to the l_1 largest eigenvalues of the equation $(\mathbf{D}_V^p - \mathbf{W}_V^p)\mathbf{u} = \lambda(\mathbf{D}_V - \mathbf{W}_V)\mathbf{u}$, where **u** is a vector whose dimension equals the column vector dimension of **U**. Similarly, when **U** is fixed, the optimal **V** is composed of the generalized eigenvectors corresponding to the l_2 largest eigenvalues of the equation $(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{v} = \lambda(\mathbf{D}_U - \mathbf{W}_U)\mathbf{v}$, where **v** is a vector whose dimension equals the column vector dimension of **V**.

Proof: The Lagrangian format of the constrained optimization in (10) is:

$$\xi(\mathbf{V}) = \text{trace}(\mathbf{V}^T(\mathbf{D}_U - \mathbf{W}_U)\mathbf{V}) - \gamma\left(\text{trace}(\mathbf{V}^T(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{V}) - \frac{d}{2}\right). \tag{15}$$

Equation (15) corresponds to a minimization problem and the smallest eigenvalues are required for solving. We transform (15) to

$$g(\mathbf{V}) = \left(\text{trace}(\mathbf{V}^T(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{V}) - \frac{d}{2}\right) - \lambda\,\text{trace}(\mathbf{V}^T(\mathbf{D}_U - \mathbf{W}_U)\mathbf{V}). \tag{16}$$

Equation (16) corresponds to a maximization problem and the largest eigenvalues are required for solving. The partial derivative of (16) yields

$$\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} = \frac{\partial\left(\text{trace}(\mathbf{V}^T(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{V}) - \lambda\,\text{trace}(\mathbf{V}^T(\mathbf{D}_U - \mathbf{W}_U)\mathbf{V})\right)}{\partial \mathbf{V}} = \mathbf{0}. \tag{17}$$

It follows that

$$\mathbf{V}^T((\mathbf{D}_U^p - \mathbf{W}_U^p) + (\mathbf{D}_U^p - \mathbf{W}_U^p)^T) - \lambda\mathbf{V}^T((\mathbf{D}_U - \mathbf{W}_U) + (\mathbf{D}_U - \mathbf{W}_U)^T) = \mathbf{0}. \tag{18}$$

It is apparent that $(\mathbf{D}_U^p - \mathbf{W}_U^p) = (\mathbf{D}_U^p - \mathbf{W}_U^p)^T$ and $(\mathbf{D}_U - \mathbf{W}_U) = (\mathbf{D}_U - \mathbf{W}_U)^T$. Then

$$\mathbf{V}^T(\mathbf{D}_U^p - \mathbf{W}_U^p)^T = \lambda\mathbf{V}^T(\mathbf{D}_U - \mathbf{W}_U)^T \tag{19}$$

$$(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{V} = \lambda(\mathbf{D}_U - \mathbf{W}_U)\mathbf{V}. \tag{20}$$

Then, when **U** is fixed the optimal **V** consists of the l_2 generalized eigenvectors that correspond to the l_2 largest eigenvalues of the equation $(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{v} = \lambda(\mathbf{D}_U - \mathbf{W}_U)\mathbf{v}$.

The Laplacian matrix $\mathbf{D}_U^p - \mathbf{W}_U^p$ is $I_1 \times I_2$-dimensional. The corresponding Laplacian matrix for the image-as-vector representation-based tracking algorithms are $(I_1 \times I_2) \times (I_1 \times I_2)$ dimensional while the dimension of a vector obtained by flattening an image is $I_1 \times I_2$. Therefore, the number of the samples sufficient to learn an effective tensor subspace is much less than the number of the samples sufficient to learn an effective vector subspace.

2.3.3. Algorithm

We use the training dataset $\{\mathcal{X}_i, i = 1,2,...N\}$ associated with class labels $\{L_i, i = 1,2,...N\}$

($L_i \in \{-2,-1,+1,+2\}$), to acquire transformation matrices **U** and **V** which are used to project the original tensor samples to the tensor-based graph embedding space. It is difficult to obtain the optimal **U** and **V** simultaneously. As in [7], we estimate **U** and **V** iteratively. For a fixed **U**, we estimate the optimal **V** using Proposition 2. Then, with the estimated **V**, we update **U**. This iteration process is repeated a certain number of times. The algorithm is outlined below:

Step 1: Initially set **U** to the first l_1 columns of the identity matrix and set the iteration step *t* to 1.

Step 2: Calculate **W** and $\mathbf{W}^p$ from (6) and (7), respectively.

Step 3: for *t* = $1 \to T$ **do**

Calculate $\mathbf{D}_U$, $\mathbf{W}_U$, $\mathbf{D}_U^p$, and $\mathbf{W}_U^p$ from (11) and (12);

Compute $\mathbf{V}$ by solving the generalized eigenvector problem: $(\mathbf{D}_U^p - \mathbf{W}_U^p)\mathbf{v} = \lambda(\mathbf{D}_U - \mathbf{W}_U)\mathbf{v}$;

Calculate $\mathbf{D}_V$, $\mathbf{W}_V$, $\mathbf{D}_V^p$, and $\mathbf{W}_V^p$ by swapping **U** and **V** in (11) and (12);

Update $\mathbf{U}$ by solving the generalized eigenvector problem: $(\mathbf{D}_V^p - \mathbf{W}_V^p)\mathbf{u} = \lambda(\mathbf{D}_V - \mathbf{W}_V)\mathbf{u}$;

end for.

Step 4: Return **U** and **V**.

The number of iterations which ensures that the algorithm reaches a satisfactory result is determined empirically. While our graph embedding model updates the samples online, the graph embedding subspace is updated iteratively.

Based on the obtained **U** and **V**, a tensor classifier $h(\mathcal{X})$ is constructed using the Euclidean distance in the low dimensional tensor embedding space. The unlabeled samples with pseudo labels are combined with the labeled training samples to train the classifier $h(\mathcal{X})$ which labels $\mathcal{X}$ according to the weighted center positions $\mathbf{R}^+$ of the positive samples and $\mathbf{R}^-$ of the negative samples. The weighted center position $\mathbf{R}^+$ of the positive samples is defined as:

$$\mathbf{R}^+ = \mathbf{U}^T \left(\frac{\beta \sum_{i=1}^{N} \mathcal{X}_i \delta(L_i,+2) + (1-\beta) \sum_{i=1}^{N} \mathcal{X}_i \delta(L_i,+1)}{\beta \sum_{i=1}^{N} \delta(L_i,+2) + (1-\beta) \sum_{i=1}^{N} \delta(L_i,+1)} \right) \mathbf{V} \tag{21}$$

where $\beta (0.5 < \beta \le 1)$ is used to give more weight to the samples whose labels are "+2" than the samples whose labels are "+1", and $\delta(x,y)$ equals 1 if $x = y$, otherwise it equals 0. The weighted center position $\mathbf{R}^-$ of the negative samples is defined as:

$$\mathbf{R}^- = \mathbf{U}^T \left(\frac{\beta \sum_{i=1}^{N} \mathcal{X}_i \delta(L_i,-2) + (1-\beta) \sum_{i=1}^{N} \mathcal{X}_i \delta(L_i,-1)}{\beta \sum_{i=1}^{N} \delta(L_i,-2) + (1-\beta) \sum_{i=1}^{N} \delta(L_i,-1)} \right) \mathbf{V} . \tag{22}$$

We define a function $f(\mathcal{X})$:

$$f(\mathcal{X}) = \left\| \mathbf{U}^T \mathcal{X} \mathbf{V} - \mathbf{R}^- \right\| - \left\| \mathbf{U}^T \mathcal{X} \mathbf{V} - \mathbf{R}^+ \right\| + \pi_t \tag{23}$$

where π_t is chosen to ensure that for any training sample $\mathcal{X}_i$ $\text{sign}(f(\mathcal{X}_i)) = \text{sign}(L_i)$. Then, the classifier $h(\mathcal{X})$ is defined by $h(\mathcal{X}) = \text{sign}(f(\mathcal{X}))$. Two points ensure that samples are separable: 1) In general, the object samples are separable from the background samples. 2) In the penalty graph, more weights are defined to separate the object samples from the background samples.

3. Transfer-Learning-Based Semi-Supervised Improvement

We construct a semi-supervised method which uses the unlabeled data to adjust the learned discriminative embedding space. As in the previous discriminant trackers [6, 10], we use the image patches corresponding to the tracking results as the labeled positive samples, and use the image patches in the background as the labeled negative samples. Then, a discriminative embedding space is learned using these labeled positive and negative samples in the supervised way described in Section 2. We further select a number of useful unlabeled samples in the new frame to adjust the embedding space in a semi-supervised way.

We introduce transfer learning into the semi-supervised learning process. In the tracking application, if the sample set is updated frequently, then there can be a more rapid response to changes in appearance. However, errors may occur in the updates, and tracking drift may be induced by large variations in appearance over time. Stability of the tracker is reduced. If the sample set is updated slowly, the tracker is less affected by tracking errors and more robust to tracking drift, but its adaptability is reduced. To maintain both the adaptability and the stability of tracking, we construct two sample sets. One is called “the target set” [10] which only consists of the object and background samples collected from recent frames. The other is called “the auxiliary set” [10] which consists of the samples collected at earlier tracking stages. The target set is updated frequently, to ensure that it is adaptable to changes in appearance. The auxiliary set is updated slowly to ensure stability of tracking and robustness to tracker drift. The samples in the auxiliary set are treated as the source data and the samples in the target set are treated as the target data [52]. The changes in the environment ensure that the distribution of the samples is changed, i.e., the distribution of the source data may be quite different from the distribution of the target data. We transfer information in the source domain to the target domain [52]. First, we use only the samples in the target set to learn an initial tensor graph embedding space which retains the discriminative information about the changes in object appearance in recent frames. Then, using the auxiliary set, we select the unlabeled samples which are most probably misclassified by the current learned graph embedding space. Their labels are chosen according to their similarities to the samples in the auxiliary set. The selected samples are used to iteratively update the graph embedding tensor subspace. In this way, the discriminative information about the earlier changes in object appearance is transferred into the graph embedding space. This transfer-learning-based

semi-supervised improvement makes the appearance model more robust to tracking drift.

In the semi-supervised adjustment, samples with high similarity usually share the same label [41]. We do not use the projected tensor difference to measure the similarities between tensor samples. This is because the initial projection matrices in the iterative semi-supervised improvement process are not accurate enough for projecting tensor samples into a low-dimensional space. We also do not use the similarity measure based on the 2-order tensor distance in (4). This is because in contrast with the graph embedding learning process in which it is only necessary to compute the distances between recent samples, the transfer learning-based semi-supervised improvement requires the computation of the distances between the recent samples in the target set and the earlier samples in the auxiliary set. As the recent environment and earlier environments may be quite different, a simpler similarity measure based on the 2-order tensor distance is not effective enough for measuring similarities between recent samples and earlier samples. Instead, we develop a more accurate block division-based covariance matrix descriptor measuring the similarities between these samples. In the descriptor, the local relations between pixels in sample patches are modeled. The division of an image patch into non-overlapping blocks incorporates more local spatial information into the similarity measurement. The local information in the covariance distance is a supplementary to the holistic features used in the tensor-based graph embedding. In the following, we first describe the block-division-based similarity estimation, and then propose our transfer-learning-based semi-supervised improvement.

3.1. Block division-based similarity estimation

In each sample patch, a feature vector *f* for each pixel is defined as:

$$\mathbf{f} = \left(x, y, \phi, |\phi_x|, |\phi_y|, \sqrt{(\phi_x)^2 + (\phi_y)^2} \right) \tag{24}$$

where (x, y) are the pixel coordinates, ϕ is the intensity value of this pixel, and ϕ_x and ϕ_y are the first order intensity derivatives. We divide the patch into $m \times n$ blocks. Given a block (*p*, *q*), let Ω be the number of pixels in the block and let $\boldsymbol{\mu}$ be the mean of $\{\mathbf{f}_k\}_{k=1,2,...,\Omega}$. The image block (*p*, *q*) is represented using a covariance matrix $\mathbf{C}_{pq}$ [14, 37] which is obtained by:

$$\mathbf{C}^{pq} = \frac{1}{\Omega - 1} \sum_{k=1}^{\Omega} (\mathbf{f}_k - \boldsymbol{\mu})(\mathbf{f}_k - \boldsymbol{\mu})^T . \tag{25}$$

Given the symmetric positive definite matrix $\mathbf{C}^{pq}$, the SVD (singular value decomposition) for $\mathbf{C}^{pq}$ ($\mathbf{C}^{pq} = \mathbf{E}\boldsymbol{\Sigma}\mathbf{F}^T$) produces the orthogonal matrix $\mathbf{E}$ and the diagonal matrix $\boldsymbol{\Sigma} = Diag(\nu_1, \nu_2, ..., \nu_6)$ where $\{\nu_i\}_{i=1,2,...,6}$ are the eigenvalues of $\mathbf{C}^{pq}$. The matrix logarithm of $\mathbf{C}^{pq}$ is defined by:

$$\log(\mathbf{C}^{pq}) = \mathbf{E} \cdot Diag(\log(\nu_1), \log(\nu_2), ..., \log(\nu_6)) \cdot \mathbf{E}^T . \tag{26}$$

Due to the vector space structure of $\log(\mathbf{C}_{pq})$, it can be unfolded into a vector $\mathbf{o}_{pq}$. As $\log(\mathbf{C}_{pq})$ is a symmetric

matrix, only the elements of its upper triangular matrix are utilized in $\mathbf{o}_{pq}$. The off-diagonal elements in $\log(\mathbf{C}_{pq})$ are multiplied with $\sqrt{2}$ to ensure that the distance between any two symmetric matrices is equal to the distance between the corresponding unfolded vectors. The image patch is represented by the set of vectors $\{\mathbf{o}_{pq}\}_{p=1,2,...,m}^{q=1,2,...,n}$. The similarity between the block (p, q) in patch i and the block (p, q) in patch j is computed by:

$$s_{ij}^{pq} = \exp\left(-\frac{\left\|\mathbf{o}_i^{pq} - \mathbf{o}_j^{pq}\right\|^2}{\sigma_i^{pq}\sigma_j^{pq}}\right) \tag{27}$$

where $\sigma_i^{pq} = \left\|\mathbf{o}_i^{pq} - \mathbf{o}_{i^{(k)}}^{pq}\right\|$ and $i^{(k)}$ indicates the k-th nearest neighbor of patch i.

The blocks nearer to the center of an image patch are more informative, while the boundary blocks are prone to be influenced by the exterior of the image patch. So, we weight the blocks in a patch using the spatial global Gaussian filtering. The weight ω^{pq} for a block (p, q) is defined as:

$$\omega^{pq} = \exp\left(-\frac{(x_{pq} - x_o)^2 + (y_{pq} - y_o)^2}{2\sigma_{spatial}^2}\right) \tag{28}$$

where x_{pq} and y_{pq} are the positional coordinates of block (p, q), x_o and y_o are the positional coordinates of the center of the patch, and $\sigma_{spatial}$ is a scaling factor. Then, the similarity $S(\mathcal{X}_i, \mathcal{X}_j)$ between samples $\mathcal{X}_i$ and $\mathcal{X}_j$ is defined as:

$$S(\mathcal{X}_i, \mathcal{X}_j) = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}\omega^{pq}s_{ij}^{pq}}{\sum_{p=1}^{m}\sum_{q=1}^{n}\omega^{pq}}. \tag{29}$$

3.2. Semi-supervised improvement

Let $\{(\mathcal{X}_i, L_i)\}_{i=1}^{N_A}$ denote the auxiliary set of the tensor samples, where N_A is the number of the auxiliary samples, and L_i be the label, either "+1" or "−1", of sample $\mathcal{X}_i$. Let $\{(\mathcal{X}_i, L_i\}_{i=N_A+1}^{N_A+N_T}$ denote the target sample set, where N_T is the number of the samples in the target set. Let $\{\mathcal{X}_j\}_{j=1}^{N_U}$ denote the set of the unlabeled samples, where N_U is the number of the unlabeled samples.

We derive the transfer-learning-based semi-supervised adjustment algorithm in an iterative way. Let $h^{(0)}(\mathcal{X}) : \mathbb{R}^{I_1 \times I_2} \to \{-1, +1\}$ denote the tensor classifier that is obtained by the tensor-based graph embedding learning algorithm in Section 2.3.3 only using the labeled samples in the target set. Let $h^{(t)}(\mathcal{X}) : \mathbb{R}^{I_1 \times I_2} \to \{-1, +1\}$ denote the classifier that is obtained in the t-th iteration by the tensor-based graph embedding learning algorithm using the labeled samples in the target set and the samples with pseudo labels. It is used to improve $h^{(t-1)}(\mathcal{X})$. Let $H(\mathcal{X}) : \mathbb{R}^{I_1 \times I_2} \to \mathbb{R}$ denote the tensor classification model which is obtained by

linearly combining the classifiers obtained in the previous $\hat{T}$ iterations:

$$H(\boldsymbol{\mathcal{X}}) = \alpha_0 h^{(0)}(\boldsymbol{\mathcal{X}}) + \sum_{t=1}^{\hat{T}} \alpha_t h^{(t)}(\boldsymbol{\mathcal{X}}) \tag{30}$$

where α_0 and $\{\alpha_t\}$ are the combination weights. While $h(\boldsymbol{\mathcal{X}})$ outputs the sign for labeling $\boldsymbol{\mathcal{X}}$, the ensemble classifier $H(\boldsymbol{\mathcal{X}})$ outputs a real value which represents the weight for labeling $\boldsymbol{\mathcal{X}}$ and whose sign corresponds to the label of $\boldsymbol{\mathcal{X}}$. At the $(\hat{T}+1)$-*th* iteration, our goal is to find a new tensor classifier $h(\boldsymbol{\mathcal{X}})$ together with the combination weight α, satisfying the following optimization:

$$\begin{aligned} \underset{h(\boldsymbol{\mathcal{X}}),\alpha}{\text{Minimize}} \quad & F = \sum_{i=1}^{N_A}\sum_{j=1}^{N_U} S_{ij} \exp\left(-2L_i\left(H(\boldsymbol{\mathcal{X}}_j) + \alpha h(\boldsymbol{\mathcal{X}}_j)\right)\right) \\ & + \eta \sum_{j_1=1}^{N_U}\sum_{j_2=1}^{N_U} S_{j_1 j_2} \exp\left(H(\boldsymbol{\mathcal{X}}_{j_1}) - H(\boldsymbol{\mathcal{X}}_{j_2})\right)\exp\left(\alpha(h(\boldsymbol{\mathcal{X}}_{j_1}) - h(\boldsymbol{\mathcal{X}}_{j_2}))\right) \\ \text{Subject to} \quad & h(\boldsymbol{\mathcal{X}}_i) = L_i, i = 1,\ldots N_A \end{aligned} \tag{31}$$

where η weights the contribution of the inconsistencies among the unlabeled data. The usual value of η is N_A / N_U. The similarity S_{ij} between a sample in the auxiliary set and an unlabeled sample and the similarity $S_{j_1 j_2}$ between two unlabeled samples j_1 and j_2 are computed using the block division-based covariance matrix descriptor in Section 3.1. The first term in the objective function in (31) measures the inconsistencies between the labeled samples in the auxiliary set and the unlabeled samples: for an unlabeled sample *j* which is similar to a labeled sample *i* (i.e., S_{ij} is large), it decreases the objective function if sample *j* shares the same label with sample *i* (i.e., $h(\boldsymbol{\mathcal{X}}_j) = L_i$). The second term in the objective function measures the inconsistencies among the unlabeled samples: it decreases the objective function if a pair of similar unlabeled samples j_1 and j_2 (i.e., $S_{j_1 j_2}$ is large) share the same label ($h(\boldsymbol{\mathcal{X}}_{j_1}) = h(\boldsymbol{\mathcal{X}}_{j_2})$). Although there is the product of the labels in the first term, the component L_i in the product is 1 or -1, and the product is equal to $H(\boldsymbol{\mathcal{X}}_j) + \alpha h(\boldsymbol{\mathcal{X}}_j)$ or $-(H(\boldsymbol{\mathcal{X}}_j) + \alpha h(\boldsymbol{\mathcal{X}}_j))$. Then, the first term and the second term in the objective function in (31) finally have similar forms.

On regrouping the terms in the objective function *F* in (31), the objective function is rewritten as follows:

$$F = \sum_{j=1}^{N_U}\left(P_j e^{-2\alpha h(\boldsymbol{\mathcal{X}}_j)} + Q_j e^{2\alpha h(\boldsymbol{\mathcal{X}}_j)}\right) \tag{32}$$

where

$$P_j = \sum_{i=1}^{N_A} S_{ij} e^{-2H(\boldsymbol{\mathcal{X}}_j)} \delta(L_i, 1) + \frac{\eta}{2}\sum_{j_*=1}^{N_U} S_{jj_*} e^{(H(\boldsymbol{\mathcal{X}}_{j_*}) - H(\boldsymbol{\mathcal{X}}_j))} \tag{33}$$

$$Q_j = \sum_{i=1}^{N_A} S_{ij} e^{2H(\boldsymbol{\mathcal{X}}_j)} \delta(L_i, -1) + \frac{\eta}{2}\sum_{j_*=1}^{N_U} S_{jj_*} e^{(H(\boldsymbol{\mathcal{X}}_j) - H(\boldsymbol{\mathcal{X}}_{j_*}))} \tag{34}$$

where $\delta(.)$ is defined as in (21). According to [16], *F* is minimized when only the unlabeled samples with

maximum values of $|P_j - Q_j|$ are classified as $h(\mathcal{X}_j) = \text{sign}(P_j - Q_j)$. So, we choose these unlabeled samples, and label them with $2\text{sign}(P_j - Q_j)$ ("-2" or "+2"). Then, they are combined with the labeled samples $\{\mathcal{X}_i\}_{i=N_A+1}^{N_A+N_T}$ in the target set to learn a new discriminative embedding space which yields a new tensor classifier $h(\mathcal{X})$. By minimizing F [16], the optimal α is derived as:

$$\alpha = \frac{1}{4}\ln\left(\frac{\sum_{j=1}^{N_U} P_j \delta(h(\mathcal{X}_j),1) + \sum_{j=1}^{N_U} Q_j \delta(h(\mathcal{X}_j),-1)}{\sum_{j=1}^{N_U} P_j \delta(h(\mathcal{X}_j),-1) + \sum_{j=1}^{N_U} Q_j \delta(h(\mathcal{X}_j),1)}\right). \tag{35}$$

Then, the improved tensor classification model $H(\cdot)$ is obtained.

The procedure of the semi-supervised improvement algorithm is outlined as follows:

Step 1: Compute the similarities between the unlabeled samples and the similarities between the auxiliary samples and the unlabeled samples.

Step 2: Train $h^{(0)}(\mathcal{X})$ by the tensor-based graph embedding algorithm only using $\{\mathcal{X}_i\}_{i=N_A+1}^{N_A+N_T}$.

Step 3: Compute α_0 using (33), (34), and (35).

Step 4: Initialize $H(\mathcal{X}) = \alpha_0 h^{(0)}(\mathcal{X})$;

Step 5: for $t = 1 \to \hat{T}$ **do**

Compute P_j and Q_j for every unlabeled sample *j* using (33) and (34);

Find the unlabeled samples with maximal values of $|P_j - Q_j|$, and label them with $2\text{sign}(P_j - Q_j)$;

Combine these chosen samples with $\{\mathcal{X}_i\}_{i=N_A+1}^{N_A+N_T}$ to train a new classifier $h^{(t)}(\mathcal{X})$ using the tensor-based graph embedding learning algorithm;

Compute α_t using (35);

adjust the tensor classification model by $H(\mathcal{X}) \leftarrow H(\mathcal{X}) + \alpha_t h^{(t)}(\mathcal{X})$;

end for

Step 6: Return $H(\mathcal{X})$.

From (33) and (34), it is seen that, if an unlabeled sample $\mathcal{X}_j$ is highly similar to the positive samples in the auxiliary set, but wrongly predicted to have the negative label by the current tensor-based graph embedding classifier, then this unlabeled sample has a large P_j and a small Q_j. We choose this unlabeled sample and label it with $2\text{sign}(P_j - Q_j)$. Likewise, we choose the top few most "mis-predicted" samples for improving the tensor-based ensemble classifier $H(\mathcal{X})$. These chosen unlabeled samples are useful to encode discriminant

information at earlier time for compensating for the loss of discriminant information in the tensor-based graph embedding space.

4. Visual Tracking

We apply the proposed transfer-learning-based semi-supervised tensor-based graph embedding learning algorithm to appearance-based object tracking. We incorporate it into the flexible and effective Bayesian inference tracking framework [21] to generate a new tracking algorithm. Fig. 2, an extension of Fig. 1, shows an overview of the proposed tracker. The main modules in the proposed tracking algorithm include sampling the positive and negative samples in the target set, collecting the labeled samples in the auxiliary set, sampling the unlabeled samples, and combining the transfer-learning-based semi-supervised tensor graph embedding algorithm with the particle filtering framework.

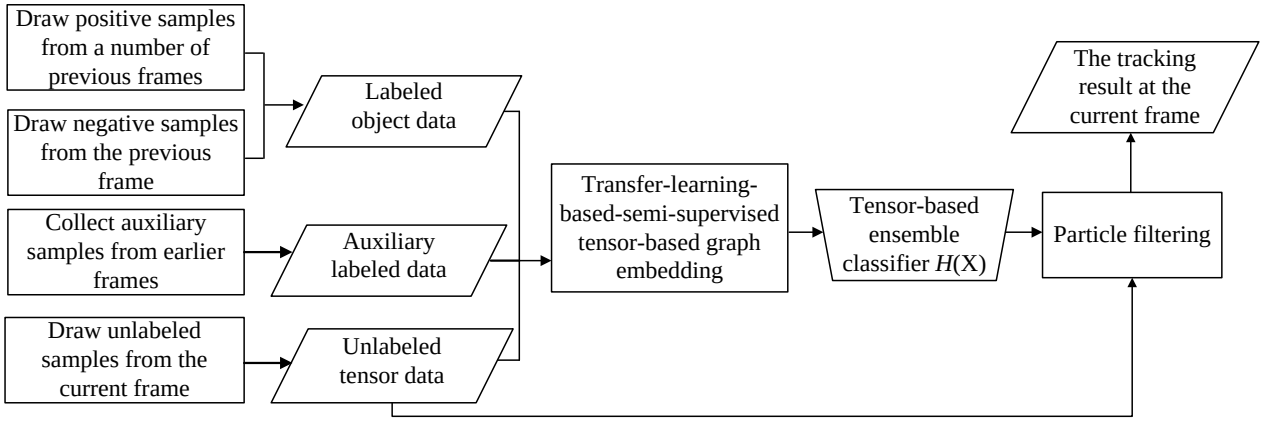


Fig. 2. An overview of the proposed tracker.

The online update of our tracker is carried out by updating the samples in the target set and the auxiliary set. The tracking results at a number of previous frames are used to generate positive samples in the target set. The variations of these samples are mainly caused by the object appearance variations over time. We draw the negative samples in the target set from the image region near to and surrounding the image patch indicated by the tracking result at the previous frame t-1, to ensure that these negative samples do not lie too far or too close to the object. A dense sampling method is used: a window slides in the surrounding region with high overlap ratio to generate a large number of image patches. A subset of the patches is randomly selected and the elements are labeled as negative. The auxiliary set of the labeled samples is collected from the earlier frames, i.e., we choose a positive sample and some negative samples every several frames for the auxiliary set.

An object's state is represented using an affine parameter vector $\boldsymbol{\eta}$. A Gaussian distribution $G(\boldsymbol{\eta}_{t-1}, \boldsymbol{\Sigma})$ is used to model the state transition distribution corresponding to the dynamic model in the particle filtering. The mean vector of the Gaussian distribution is the affine parameter vector $\boldsymbol{\eta}_{t-1}$ estimated at the previous frame t-1, and the covariance matrix $\boldsymbol{\Sigma}$ is set empirically. We simply consider the object state information in 2D (x, y)

translation and scaling in order to fairly compare our results with other tracking algorithms. The particles at the current frame *t* are drawn from the state transition distribution $G(\boldsymbol{\eta}_{t-1}, \boldsymbol{\Sigma})$. The image patches determined by the state vectors of these particles are normalized to tensors $\{\boldsymbol{\mathcal{X}}_j\}_{j=1}^{N_U}$ which are used as the unlabeled tensor samples.

We apply the proposed semi-supervised tensor-based graph embedding learning algorithm to the labeled samples in the target set, the labeled samples in the auxiliary set, and the unlabeled samples. Then, the ensemble classifier $H(\boldsymbol{\mathcal{X}})$ is obtained. Corresponding to $H(\boldsymbol{\mathcal{X}})$, the weight $E(\boldsymbol{\eta}_j)$ of each particle $\boldsymbol{\eta}_j$ (j=1,2,..,N_U) in the particle filtering is defined as:

$$E(\boldsymbol{\eta}_j) = \alpha_0 f^{(0)}(\boldsymbol{\mathcal{X}}_j) + \sum_{t=1}^{\hat{T}} \alpha_t f^{(t)}(\boldsymbol{\mathcal{X}}_j) \tag{36}$$

where $\boldsymbol{\mathcal{X}}_j$ is the tensor corresponding to particle $\boldsymbol{\eta}_j$. Maximum a posterior (MAP) estimation is used to estimate the tracking result $\boldsymbol{\eta}_t$ at the current frame *t*:

$$\boldsymbol{\eta}_t = \arg\max_j E(\boldsymbol{\eta}_j). \tag{37}$$

The global tracking algorithm is summarized as follows:

Step 1: The positive tensor samples in the target set are updated using the tracking results at the previous frame.

Step 2: The negative tensor samples in the target set are drawn from the image region near to and surrounding the patch indicated by the tracking result $\boldsymbol{\eta}_{t-1}$ at the previous frame *t*-1.

Step 3: The samples in the auxiliary set are updated once every several frames.

Step 4: The dynamic model $G(\boldsymbol{\eta}_{t-1}, \boldsymbol{\Sigma})$ is used to produce a number of particles at the current frame *t*. The tensors corresponding to these particles are used as the unlabeled tensor samples.

Step 5: The transfer-learning-based semi-supervised tensor graph embedding algorithm is applied to the labeled samples in the auxiliary set, the labeled samples in the target set, and the unlabeled samples. A final graph embedding space is then obtained iteratively.

Step 6: The weight of each particle $\boldsymbol{\eta}_j$ is evaluated using $E(\boldsymbol{\eta}_j)$.

Step 7: The particle with the largest weight is taken as the tracking result, as shown in (37).

To simplify the complexity analysis of the algorithm, we assume that $l_1 = l_2 = l$, and $I_1 = I_2 = I$. The main computational cost of the proposed tracker is the calculations of $\mathbf{W}_U$, $\mathbf{W}_V$, $\mathbf{W}_U^p$, and $\mathbf{W}_V^p$, the generalized eigenvalue decomposition, and calculation of P_j and Q_j. Each of the calculations of $\mathbf{W}_U$, $\mathbf{W}_V$, $\mathbf{W}_U^p$, and $\mathbf{W}_V^p$ requires $\mathcal{O}(N^2(I^2 + 2I^2 l))$ floating-point multiplications. The sparseness of $\mathbf{W}$ and $\mathbf{W}^p$ reduces the computational cost. The generalized eigenvalue decomposition requires $\mathcal{O}(I^3)$ floating-point multiplications.

Calculation of P_j and Q_j for each unlabeled sample *j* requires $\mathcal{O}(N_A + N_U)$ floating-point multiplications. So, the computational complexity of the proposed tracker is $\mathcal{O}(\hat{T}(N_A + N_U)N_U + (\hat{T}+1)T(N^2(I^2 + 2I^2 l) + I^3))$. As $\hat{T}, T, I, l << N < N_U$, The computational complexity approximates $\mathcal{O}((N_A + N_U)N_U + N^2)$.

5. Experimental Results

There are many comparison results in the conference version of the work [38]. In this paper, we only present the new experimental comparison results focusing on the benchmark tracking dataset [39] published in CVPR 2013. In this dataset, there are 51 fully annotated sequences containing more than 25000 frames. These sequences were annotated with the following 11 attributes: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter, and low resolution. The providers of the benchmark evaluated 29 state-of-the-art tracking algorithms and released their results on this dataset. The evaluation of the results is based on the following criteria:

- the center location error (CLE) between the center of the predicted bounding box and the center of the ground truth bounding box at each frame;
- the visual overlap ratio (VOR) between the predicted bounding box B_p and the ground truth bounding box B_g:

$$\text{VOR} = \frac{area(B_p \cap B_g)}{area(B_p \cup B_g)}. \tag{38}$$

The benchmark used the precision plots based on the location error metric and the success plots based on the overlap rate metric to evaluate performances of different tracking algorithms:

- Precision plot: the x-coordinate of a point in the plot is a location error threshold, and the y-coordinate of the point is the proportion of a video's frames in which the center location error is less than the threshold. These points form a curve as the threshold is gradually increased.
- Success plot: the x-coordinate of a point in the plot is an overlap ratio threshold, and the y-coordinate of the point is the proportion of a video's frames in which the overlap ratio is larger than the threshold.

The following representative precision and representative success rate were used to summarize the overall tracking performance:

- The y-coordinate of the curve in the precision plot when the x-coordinate is 20 pixels
- The y-coordinate of the curve in the success plot when the x-coordinate is 0.5.

In the experiments, the first positive sample was the initial bounding box at the first frame supplied on the benchmark. Eight positive samples were constructed by perturbing a few pixels in four possible directions at the corner points of the first sample at the first frame. These nine samples were used as initial positive samples. The number of particles, i.e., the number N_U of the unlabeled samples, was set to 600. The parameter *k* for the

k-nearest neighbors was empirically chosen as 7 according to [32]. The number n_{+1} of the positive samples and the number n_{-1} of the negative samples in the target set were set to 50 and 100, respectively. The scaling factor $\sigma_{spatial}$ in (28) was set to 2.9. A positive sample and four negative samples were chosen every six frames for the auxiliary set. The number of the positive samples and the number of the negative samples in the auxiliary dataset were set to 20 and 80, respectively. We found 60 unlabeled samples with maximum values of $|P_j - Q_j|$ and used them to adjust the ensemble classifier $H(\mathcal{X})$. All the image patches corresponding to the samples were normalized to templates of size 32×32, making the dimensions of the 2-order tensors unchanged. Each image patch was divided into 4*4 blocks. The number of columns of both **U** and **V** in the tensor-based graph embedding learning algorithm were set to 2, i.e., $l_1 = l_2 = 2$. The number *T* of iterations for estimating **U** and **V** was set to 20 for solving $h^0(\mathcal{X})$ and to 5 for solving $h^t(\mathcal{X})$, $t \geq 1$. The weight β in (21) was set to 0.8. The parameter $\hat{T}$ in the semi-supervised adjustment algorithm was set to 4 for a compromise between tracking speed and accuracy, i.e., 5 tensor-based ensemble classifiers are used in the final model. The above parameter settings remained the same in all the experiments.

We evaluated the performance of our tracking algorithm on the entire dataset and on the sub-sets with different annotated attributes. We first validated the main properties of our tracker, such as image-as-matrix representation and semi-supervised improvement. Then, we compared our tracker with state-of-the-art trackers.

5.1. Illustration of properties of our tracking algorithm

Our framework for tracking is a combination of multiple strategies whose coordination ensures the final tracking performance. To illustrate the effectiveness of the strategies of the image-as-matrix representation, the graph embedding for discriminative representation of tensors, the semi-supervised improvement (ensemble learning), the transfer learning, and the spatial Gaussian filtering in our tracking algorithm, we compared our Tensor-based Discriminant Tracking algorithm with Transfer-learning-based Semi-Supervised Improvement (TrSSI-TDT) with the following variants:

- **TrSSI-VDT:** This is the image-as-Vector representation-based Discriminant Tracking algorithm with transfer-learning-based semi-supervised improvement, i.e., the image-as-tensor representation is replaced with the image-as-vector representation. It uses the traditional vector-based graph embedding with only one transformation matrix in [28] to learn the graph embedding. The classifier $h(\mathcal{X})$ is constructed using this transformation matrix. Then, the transfer-learning-based semi-supervised improvement in Section 3 is applied in the vector representation.
- **TrSSI-2DLDA:** This is the tracking algorithm obtained by replacing the 2D tensor discriminant graph embedding in TrSSI-TDT with the 2D linear discriminant analysis (LDA).
- **TDT:** This is the tensor-based discriminant tracking algorithm without semi-supervised improvement,

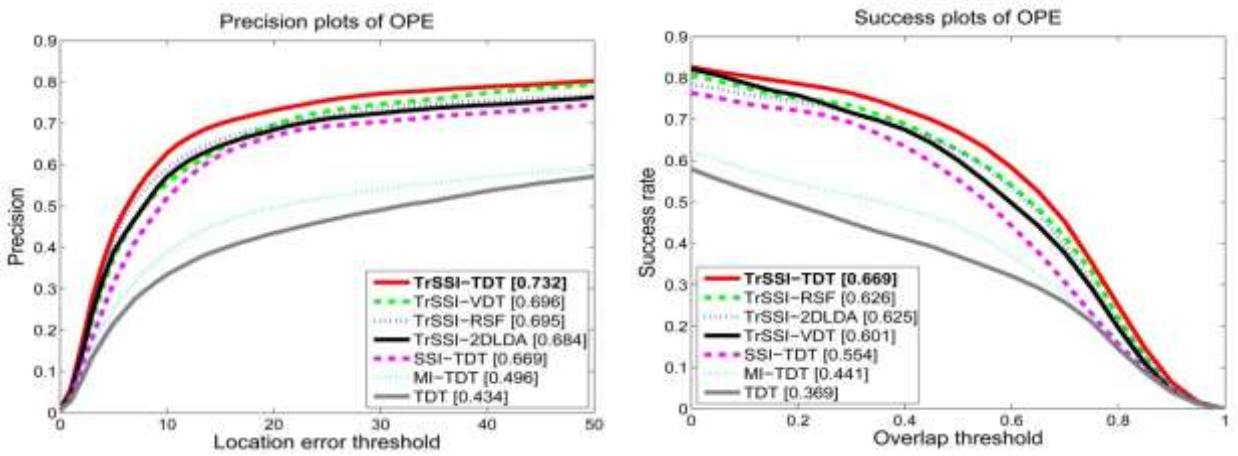
i.e., the transfer-learning-based semi-supervised improvement (TrSSI) is removed from TrSST-VDT.

- **MI-TDT:** This is the tensor-based discriminant tracking algorithm combined with the Margin Improvement technique [4], which adjusts the classification margin in a semi-supervised way. Namely, it is obtained by replacing TrSSI in TrSSI-TDT with MI in [4].
- **SSI-TDT:** This is the algorithm obtained by replacing the auxiliary sample set with the target sample set, i.e., transfer learning is not used in the semi-supervised improvement. This algorithm is just the one in our conference version of the work [38].
- **TrSSI-RSF:** This algorithm is obtained by Removing the Spatial Filtering in (28) from TrSSI-TDT.

Fig. 3. Tracking results of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF.

Fig. 3 shows some examples of the tracking results of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF on the benchmark dataset. Fig. 4 shows the precision plots and the success

plots of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF on all the sequences in the benchmark dataset. From these figures, the following useful points are revealed:



(a) (b)

Fig. 4. The precision plots and the success plots of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF on all the sequences: each decimal number in the legend is the representative precision or the representative success rate of the corresponding tracker: (a) the precision plots and (b) the success plots.

- The transfer-learning-based semi-supervised improvement (TrSSI), semi-supervised improvement (SSI), and margin improvement (MI) enhance TDT, and the enhancement from SSI is higher than from MI. This indicates that TrSSI and SSI effectively adjust the graph embedding subspace.
- Our TrSSI-TDT obtains more accurate results than TrSSI-2DLDA. This illustrates the effectiveness of the graph embedding for a discriminative representation for 2D tensors.
- TrSSI-TDT yields more accurate results than TrSSI-VDT. As shown in the sixth and seventh examples in Fig. 3, when occlusion occurs, TrSSI-TDT still successfully tracks the objects, but TrSSI-VDT loses the track. This indicates that the 2-order tensor (image-as-matrix) representation which uses the projections **U** and **V** retains more useful structure information than image-as-vector representation which uses one projection.
- TrSSI-TDT yields more accurate results than SSI-TDT. This indicates that historical information is effectively transferred into our object appearance model. This ensures that our tracker keeps the diversity of object appearance and avoids tracking drift after large changes in appearance, caused by large occlusions or serious pose or scale variations, etc.
- TrSSI-TDT yields more accurate results than TrSSI-RSF. This indicates the effectiveness of the spatial filtering in our appearance model.
- TrSSI-VDT, in which the tensor representation is not used, yields slightly more accurate results than SSI-TDT in which transfer leaning is not used. This indicates that transfer leaning may, overall, be more important than tensor representation for improving the tracking accuracy.

The proposed transfer-learning-based semi-supervised improvement can be used for reference for other tracking algorithms for improving the tracking performance.

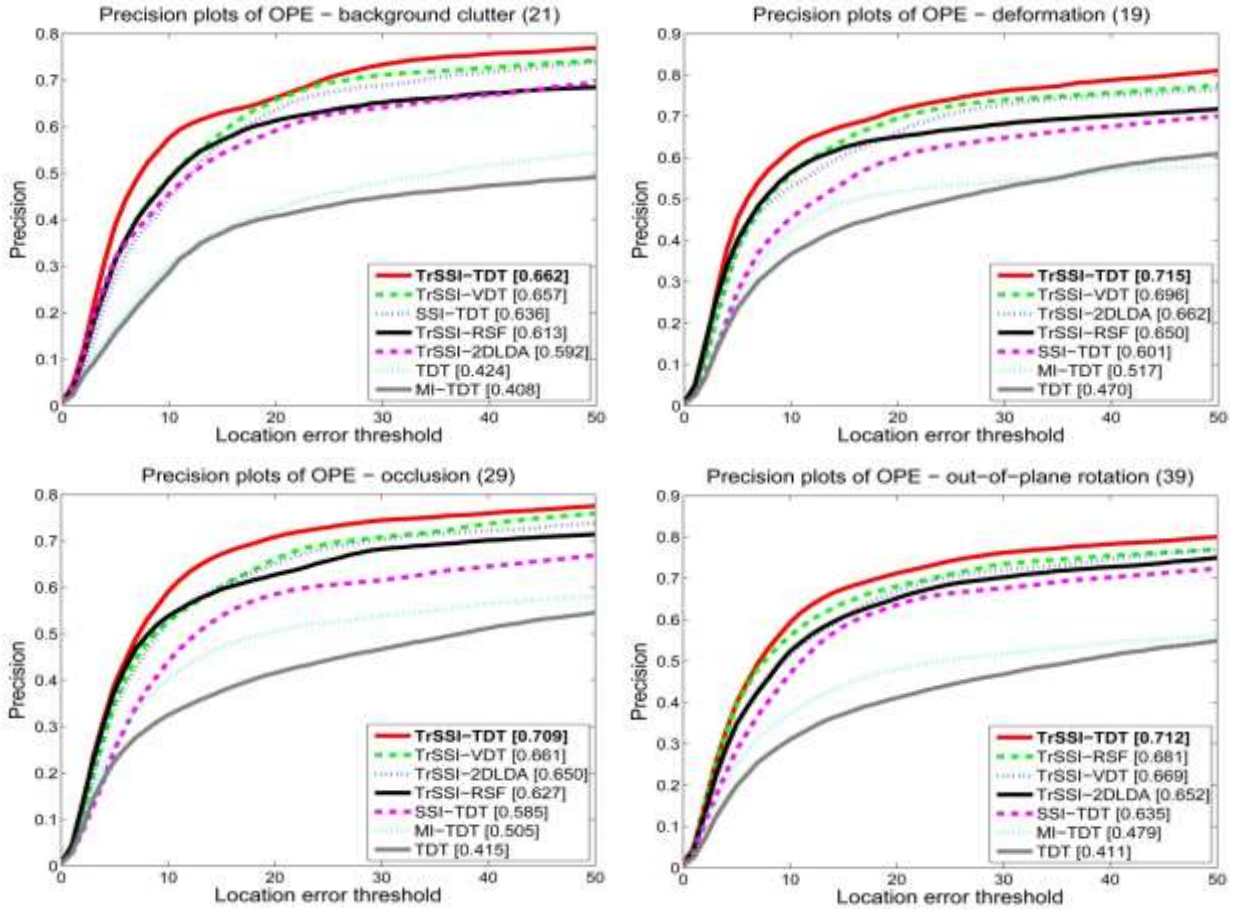


Fig. 5. The precision plots of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF for different attributes.

Figs. 5 and 6 show, respectively, the precision plots and the success plots of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF for annotated attributes, respectively. Due to the space limitation, only the plots for the attributes of occlusion, deformation, out-of-plane rotation, and background clutter are shown here. The plots for all the 11 annotated attributes are shown in the supplemental file which is published electronically. The following useful points are exhibited:

- TrSSI always improves the tracking results except for low resolution videos. TDT and MI-TDT are not robust against large changes in illumination (such as in videos Car4, Cardark, Coke, David, and singer2), occlusions (such as in videos Coke, David3, and Lemming), and background clutter (such as in videos Football, Freeman4, and Lemming), etc. TrSSI-TDT can effectively handle large occlusions (such as in videos Coke, David3, Freeman4, and Lemming). However, the variants lose the track to some extent.
- For the videos with the attributes of occlusion, deformation, and out-of-plane rotation, transfer learning contributes more than tensor representation, because the results of TrSSI-VDT are more accurate than

the results of SSI-TDT. For the videos with the attributes of background clutter, in-plane rotation, and illumination variation, transfer learning contributes as much as tensor representation and combination of transfer learning and tensor representation into the semi-supervised adjustment produces a large improvement in the results of our TrSSI-TDT.

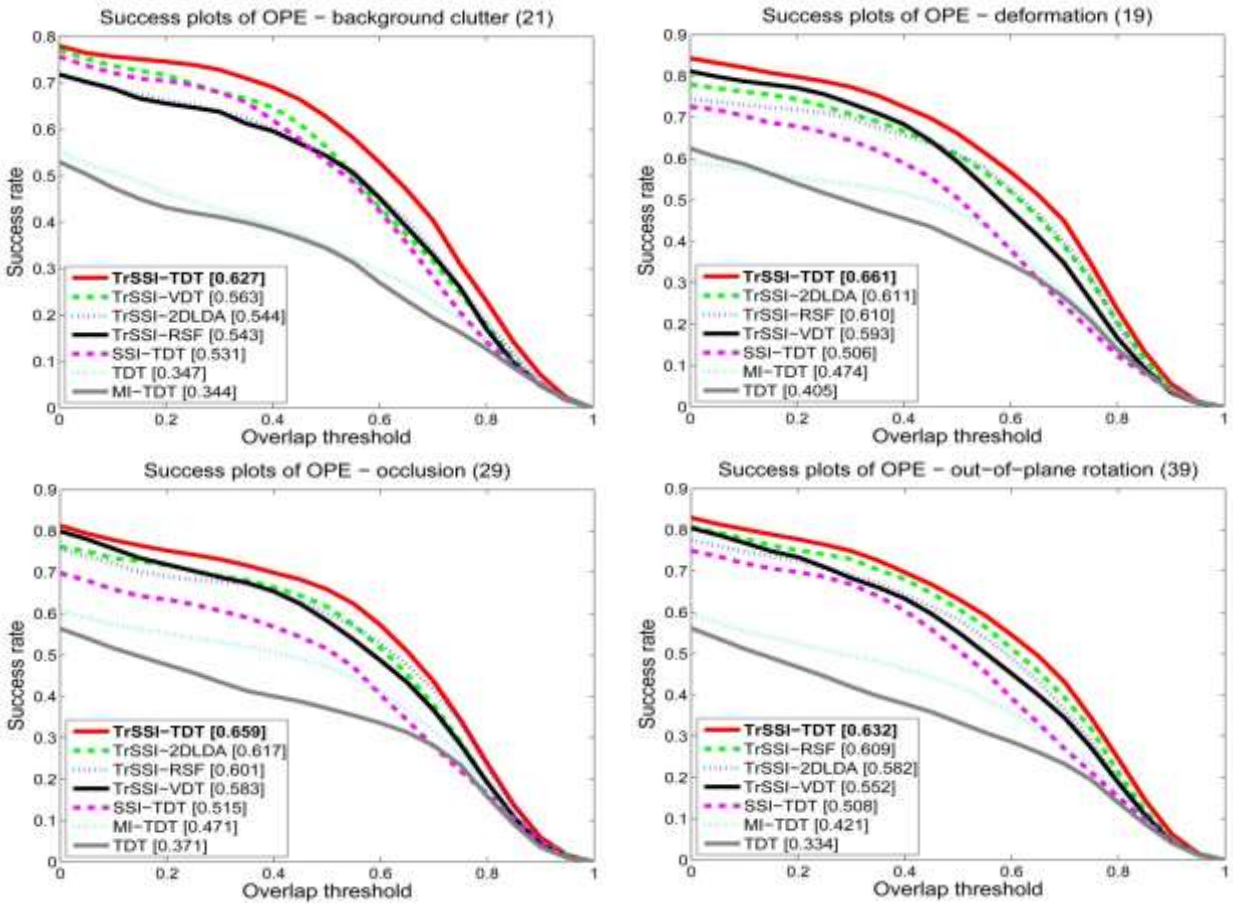


Fig. 6. The success plots of TrSSI-TDT, TrSSI-2DLDA, TrSSI-VDT, TDT, MI-TDT, SSI-TDT, and TrSSI-RSF for different attributes.

5.2. Comparison with competing trackers

To show the effectiveness of TrSSI-TDT, we conducted a comparison between TrSSI-TDT and the 29 state-of-the-art trackers whose results were released on the CVPR 2013 benchmark dataset.

Fig. 7 shows some tracking results of TrSSI-TDT and the top 10 ranked competing trackers on the dataset. Fig. 8 shows the precision plots and the success plots of the top 10 ranked trackers among TrSSI-TDT and the 29 competing trackers on all the sequences in the benchmark dataset. The following useful points are seen:

- In the precision plots, when the location error threshold lies within a large interval [5, 50], our TrSSI-TDT yields the highest precision compared with the competing algorithms. In the success rate plots, when the overlap rate threshold lies within a large interval (0, 0.8), TrSSI-TDT yields the highest success rates. TrSSI-TDT' s representative precision is larger, by 7.6%, than the representative precision of Struck [42] which is the top tracker ranked by the representative precision among all the

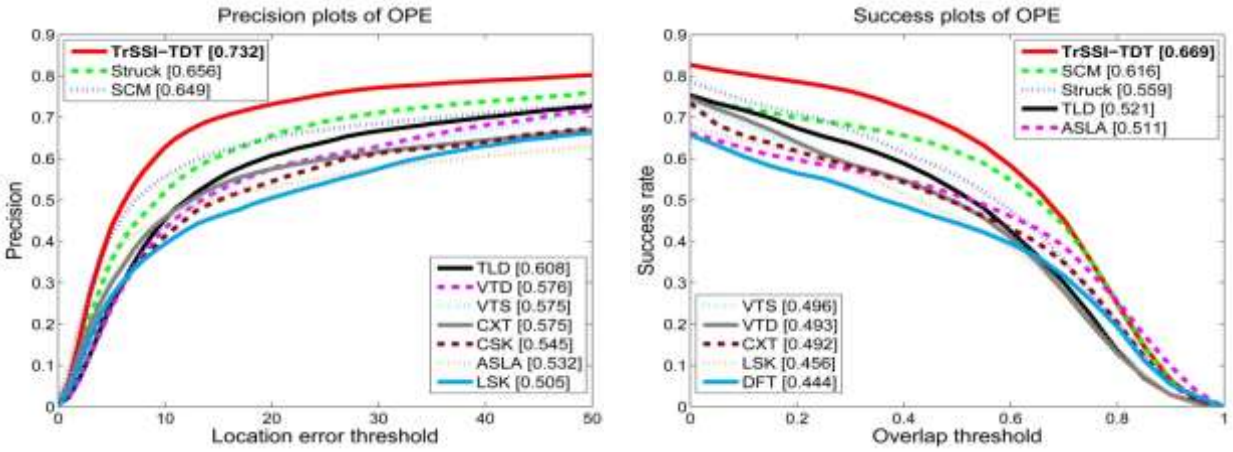
competing trackers. TrSSI-TDT's representative success rate is larger, by 5.3%, than the representative success rate of SCM [43] which is the top tracker ranked by the representative success rate among all the competing trackers.

- Even the TrSSI-VDT and the SSI-TDT yield performances comparable to Struck and SCM. This further indicates the effectiveness of the transfer learning and semi-supervised improvement in our algorithm.

Fig. 7. Tracking results of TrSSI-TDT and the top 10 ranked competing trackers.

Figs. 9 and 10 show, respectively, the precision plots and the success plots of the top 10 ranked trackers among TrSSI-TDT and the 29 competing trackers for the annotated attributes of occlusion, deformation, out-of-plane rotation, and background clutter. The plots for all the 11 annotated attributes are shown in the

supplemental file. The following points are seen:



(a) (b)

Fig. 8. The precision plots and the success plots of the top 10 ranked trackers among TrSSI-TDT and the 29 competing trackers on all the sequences: (a) the precision plots; (b) the success plots.

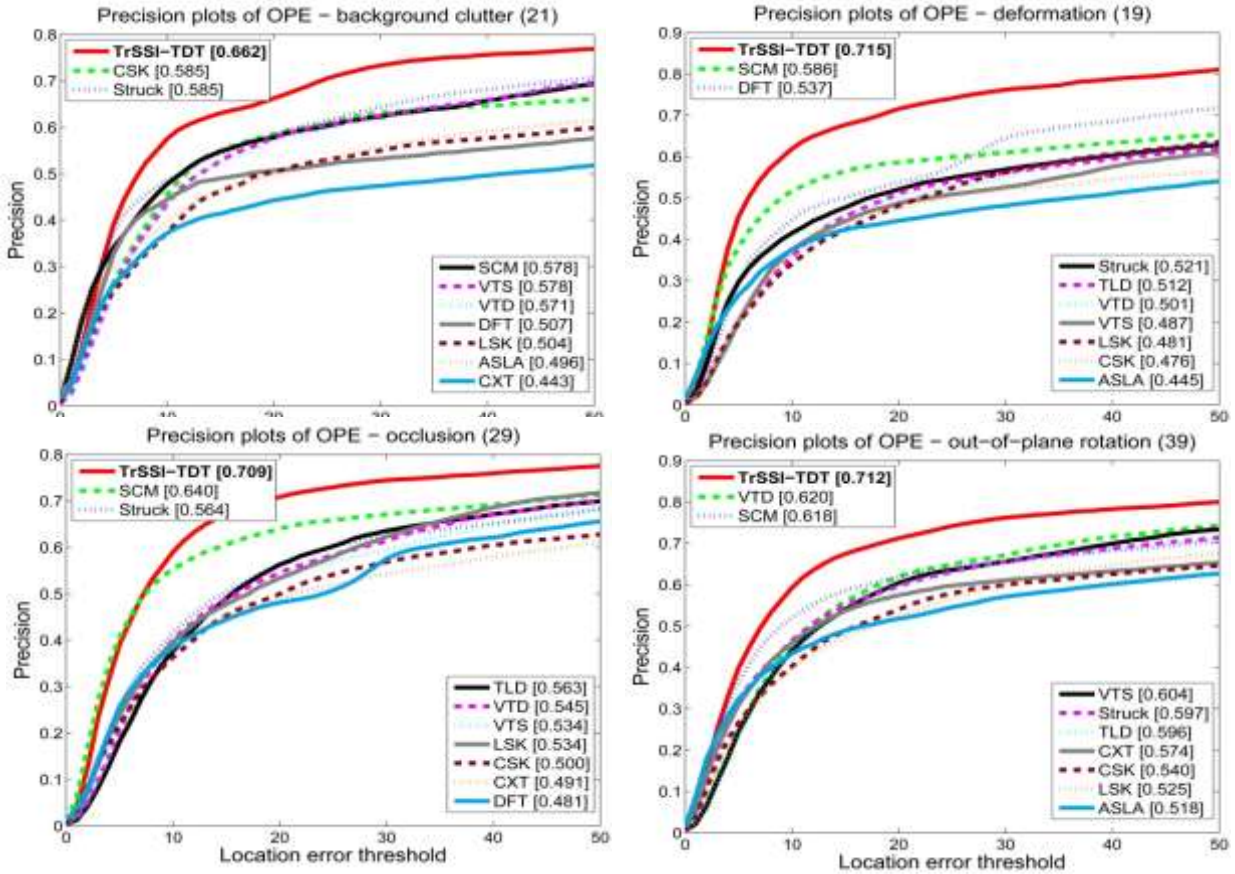


Fig. 9. The precision plots of the top 10 ranked trackers among TrSSI-TDT and the 29 competing trackers for different attributes.

- Our TrSSI-TDT yields the results of top one for 6 attributes, top-2 for 8 attributes, and top -4 for 10 attributes.
- Our TrSSI-TDT clearly outperforms the competing trackers for the following attributes which pose frequent challenges in tracking: occlusions (such as in the videos David3, Jogging-1, and Woman), illumination variation (such as in the videos Coke, David, and Singer2), deformation (such as in the

videos Basketball, Crossing, and Subway), and background clutter (such as in the videos Football, Freeman4, and Deer).

- TrSSI-TDT performs reasonably well on the videos with the attributes of fast motion and out-of-view, as TrSSI-TDT does not use a specific motion model with re-detection such as in TLD (tracking-learning-detection) [44].

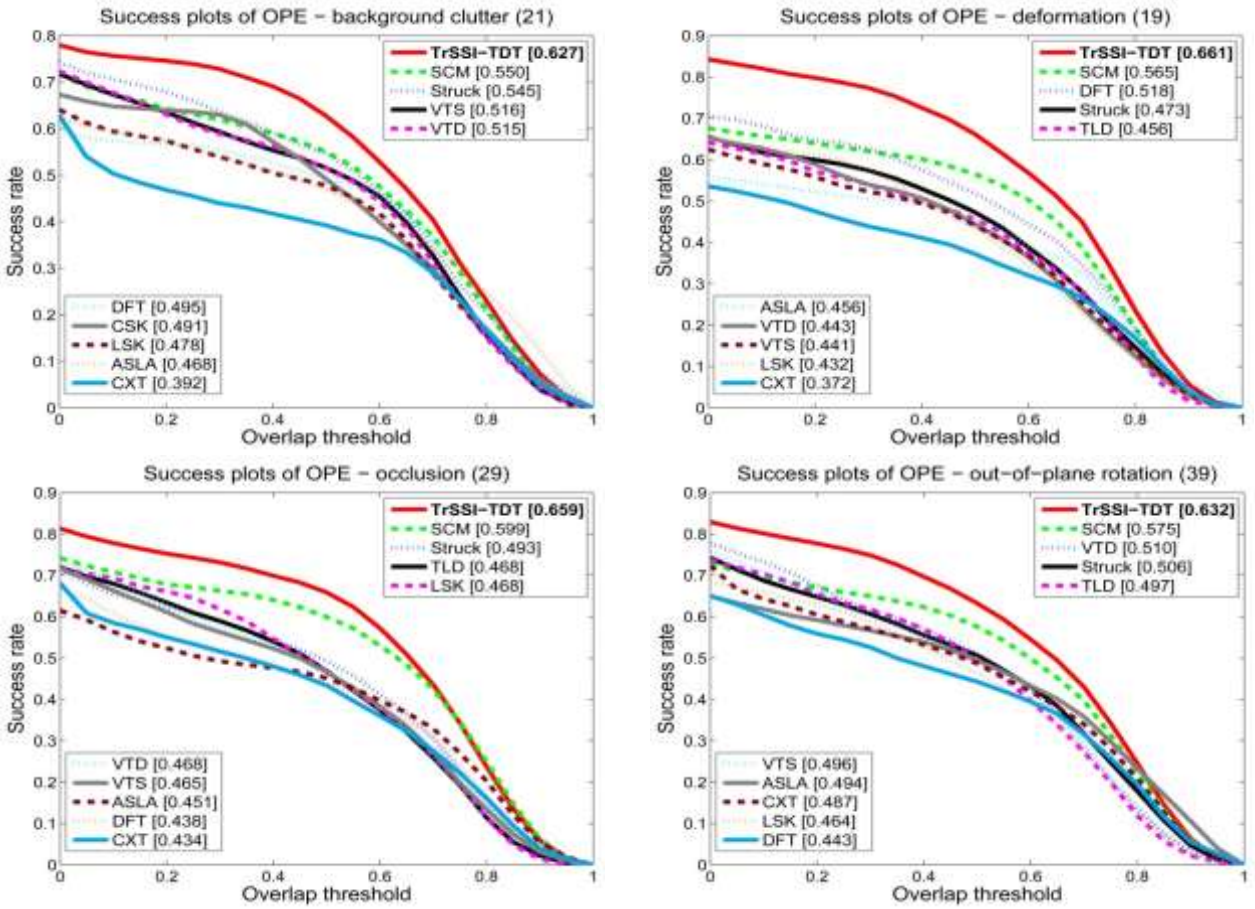


Fig. 10. The success plots of the top 10 ranked trackers among TrSSI-TDT and the 29 competing trackers for different attributes.

We also compared our tracker with the very recent trackers in [53, 54]. On the benchmark dataset, the representative precision of the tracker in [53] is 0.730, and its representative success rate is 0.551. The representative precision of the tracker in [54] is 0.649, and its representative success rate is 0.484. Overall, the results of our tracker are better than the results in [53, 54].

6. Conclusion

In this paper, we have proposed an effective and robust discriminant tracking algorithm based on the proposed transfer-learning-based semi-supervised 2-order tensor graph embedding algorithm. The effectiveness of our work is attributed to:

- the specially designed two graphs for modeling the local geometrical and discriminative structure of the 2-order tensor samples;
- the learning of the 2D tensor-based graph embedding space

- the transfer-learning-based semi-supervised adjustment technique.

This 2-order tensor representation-based algorithm retains more discriminant information than image-as-vector representation-based algorithms. The transfer-learning-based semi-supervised adjustment technique effectively transfers discriminant information obtained from earlier times into the discriminative embedding space. This makes the proposed tracking algorithm able to address the challenges caused by heavy occlusion and large pose variations, etc. Experimental comparisons on the CVPR 2013 benchmark tracking dataset have demonstrated the effectiveness and robustness of the proposed tracking algorithm.

In our future work, we will extend our image-as-matrix representation to higher-order tensor representation, e.g., 3-order tensor representation, with a feature vector for each pixel.

References

1. A. Adam, E. Rivlin, and I. Shimshoni, "Robust Fragments-Based Tracking Using the Integral Histogram," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 798-805, June 2006.
2. B. Babenko, M.-H. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619-1632, 2011.
3. C. Bao, Y. Wu, H. Ling, and H. Ji, "Real Time Robust l_1 Tracker Using Accelerated Proximal Gradient Approach," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1830-1837, June 2012.
4. K.P. Bennett, A. Demiriz, and R. Maclin,"Exploiting Unlabeled Data in Ensemble Methods," *in Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 289-296, 2002.
5. H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-line Boosting," *in Proc. of British Machine Vision Conference*, vol. 1, pp. 47-56, Sep. 2006.
6. H. Grabner, C. Leistner, and H. Bischof, "Semi-Supervised On-line Boosting for Robust Tracking," *in Proc. of European Conference on Computer Vision*, pp. 234-247, Oct. 2008.
7. X. He, D. Cai, and P. Niyogi, "Tensor Subspace Analysis," *in Proc. of Annual Conference on Neural Information Processing Systems*, pp. 499-506, 2006.
8. J. Kwon and K. Lee, "Visual Tracking Decomposition," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1269-1276, June 2010.
9. J. Kwon and K. Lee, "Tracking by Sampling Trackers," *in Proc. of IEEE International Conference on Computer Vision,* pp. 1195-1202, Nov. 2011.
10. G. Li, L. Qin, Q. Huang, J. Pang, and S. Jiang, "Treat Samples Differently: Object Tracking with Semi-Supervised Online Covboost," *in Proc. of IEEE International Conference on Computer Vision,* pp. 627-634, Nov. 2011.
11. P. Li and Q. Sun, "Tensor-Based Covariance Matrices for Object Tracking," *in Proc. of IEEE International Conference on Computer Vision Workshops,* pp. 1681-1688, Nov. 2011.
12. X. Li, A. Dick, H. Wang, C. Shen, and A. van den Hengel, "Graph Mode-Based Contextual Kernels for Robust SVM Tracking," *in Proc. of IEEE International Conference on Computer Vision,* pp. 1156-1163, Nov. 2011.
13. X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo, "Robust Visual Tracking Based on Incremental Tensor Subspace Learning," *in Proc. of IEEE International Conference on Computer Vision,* pp. 1-8, Oct. 2007.
14. X. Li, W. Hu, Z. Zhang, M. Zhu, and J. Cheng, "Visual Tracking via Incremental Log-Euclidean Riemannian Subspace Learning," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, June 2008.
15. X. Li, C. Shen, Q. Shi, A. Dick, and A. van den Hengel, "Non-Sparse Linear Representations for Visual Tracking with Online Reservoir Metric Learning," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1760-1767, June 2012.
16. P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu,"Semiboost: Boosting for Semi-Supervised Learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2000-2014, Nov. 2009.
17. X. Mei and H. Ling, "Robust Visual Tracking Using l_1 Minimization," *in Proc. of IEEE International Conference on Computer Vision,* pp. 1436-1443, Sept.- Oct. 2009.
18. X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Minimum Error Bounded Efficient l_1 Tracker with Occlusion Detection," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1257-1264, June 2011.
19. F. Porikli, O. Tuzel, and P. Meer, "Covariance Tracking Using Model Update Based on Lie Algebra," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 728-735, June 2006.
20. C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-Supervised Self-Training of Object Detection Models," *in Proc. of IEEE Workshop on Applications of Computer Vision*, pp. 29-36, Jan. 2005.

21. D.A. Ross, J. Lim, R. Lin, and M. Yang, "Incremental Learning for Robust Visual Tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125-141, May 2008.
22. M. Sugiyama, "Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis," *The Journal of Machine Learning Research*, vol. 8, pp. 1027-1061, May 2007.
23. M.A.O. Vasilescu and D. Terzopoulos, "Tensor Textures: Multilinear Image-Based Rendering," *ACM Trans. on Graphics (ACM SIGGRAPH)*, vol. 23, no. 3, pp. 336-342, Aug. 2004.
24. Q. Wang, F. Chen, and W. Xu, "Tracking by Third-Order Tensor Representation," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 385-396, April 2011.
25. J. Wen, X. Gao, X. Li, and D. Tao, "Incremental Learning of Weighted Tensor Subspace for Visual Tracking," *in Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pp. 3688-3693, Oct. 2009.
26. Y. Wu, J. Cheng, J. Wang, and H. Lu, "Real-Time Visual Tracking via Incremental Covariance Tensor Learning," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1631-1638, Sept.-Oct. 2009.
27. S. Yan, D. Xu, S. Lin, T.S. Huang, and S.F. Chang, "Element Rearrangement for Tensor-Based Subspace Learning," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, June 2007.
28. S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph Embedding and Extensions: a General Framework for Dimensionality Reduction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40-51, Jan. 2007.
29. J. Ye, "Generalized Low Rank Approximations of Matrices," *Machine Learning*, vol. 61, no. 1-3, pp. 167-191, Nov. 2005.
30. J. Ye, R. Janardan, and Q. Li, "Two-Dimensional Linear Discriminant Analysis," *in Proc. of Annual Conference on Neural Information Processing Systems*, vol. 17, pp. 1569-1576, Dec. 2005.
31. A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: a Survey," *ACM Computing Surveys*, vol. 38, no. 4, Article no. 13, 2006.
32. L. Zelnik-Manor and P. Perona, "Self-Tuning Spectral Clustering," *in Proc. of Annual Conference on Neural Information Processing Systems*, pp. 1601-1608, Dec. 2005.
33. T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Low-Rank Sparse Learning For Robust Visual Tracking," *in Proc. of European Conference on Computer Vision*, pp. 470-484, Oct. 2012.
34. T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust Visual Tracking via Multitask Sparse Learning," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2042-2049, June 2012.
35. X. Zhang, W. Hu, S. Maybank, and X. Li, "Graph Based Discriminative Learning for Robust and Efficient Object Tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1-8, Oct. 2007.
36. M. Sugiyama, "Local Fisher Discriminant Analysis for Supervised Dimensionality Reduction," *in Proc. of International Conference on Machine Learning*, pp. 905-912, 2006.
37. O. Tuzel, F. Porikli, and P. Meer, "Region Covariance: a Fast Descriptor for Detection and Classification," *in Proc. of European Conference on Computer Vision*, vol. 2, pp. 589-600, 2006.
38. J. Gao, J.L. Xing, W.M. Hu, and S. Maybank, "Discriminant Tracking Using Tensor Representation with Semi-supervised Improvement," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1569-1576, Dec. 2013.
39. Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: a Benchmark," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411-2418, 2013.
40. D. Cai, X.F. He, J.W. Han, and T.S. Huang, "Graph Regularized Nonnegative Matrix Factorization for Data Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548-1560, 2011.
41. X.F. He, M. Ji, and H.J. Bao, "A Unified Active and Semi-Supervised Learning Framework for Image Compression," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 65-72, 2009.
42. S. Hare, A. Saffari, and P. Torr, "Struck: Structured Output Tracking with Kernels," *in Proc. of IEEE International Conference on Computer Vision*, pp. 263-270, Nov. 2011.
43. W. Zhong, H. Lu, and M.-H. Yang, "Robust Object Tracking via Sparsity-Based Collaborative Model," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1838-1845, June 2012.
44. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409-1422, July 2012.
45. F. Pernici and A.D. Bimbo, "Object Tracking by Oversampling Local Features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2538-2551, Dec. 2014.
46. S. Avidan, "Ensemble Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261-271, Feb. 2007.
47. S. Avidan, "Support Vector Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064-1072, Aug. 2004.
48. S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally Orderless Tracking," *International Journal of Computer Vision*, vol. 111, no. 2, pp. 213-228, 2015.
49. S. Oron, A. Bar-Hillel, and S. Avidan, "Extended Lucas-Kanade Tracking," *in Proc. of European Conference on Computer Vision*, vol. 5, pp. 142-156, 2014.
50. S. Oron, A. Bar-Hillel, and S. Avidan, "Real-Time Tracking-with-Detection for Coping with Viewpoint Change," *Machine*

Vision and Applications, vol. 26, no. 4, pp. 507-518, 2015.
51. S. Dekel, N.A. Sochen, and S. Avidan, "Incremental Level Set Tracking," *Innovations for Shape Analysis, Models and Algorithms*, Springer 2013, ISBN 978-3-642-34140-3, pp. 407-420, 2013.
52. S.J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
53. J.F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583 - 596, 2015.
54. T. Zhang, S. Liu C. Xu, S. Yan, B. Ghanem, N. Ahuja1, and M.-H. Yang, "Structural Sparse Tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 150-158, June 2015.

Weiming Hu received the Ph.D. degree from the department of computer science and engineering, Zhejiang University in 1998. From April 1998 to March 2000, he was a postdoctoral research fellow with the Institute of Computer Science and Technology, Peking University. Now he is a professor in the Institute of Automation, Chinese Academy of Sciences. His research interests are in visual motion analysis and recognition of web objectionable information.

Jin Gao Jin Gao received his Bachelor's degree from the Beihang University, Beijing, China in 2010 and the Ph.D. degree from the Institute of Automation, University of Chinese Academy of Sciences (CAS), in 2015. Now he is an assistant professor with the National Laboratory of Pattern Recognition, Institute of Automation, CAS. His research interests include visual tracking, augmented reality, semi-supervised learning.

Junliang Xing received the B.S. degree in computer science and mathematics from Xi'an Jiaotong University, Shaanxi, China, in 2007, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2012. He is currently an assistant professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests mainly focus on computer vision problems related to faces and humans.

Chao Zhang received his B.S. degree in school of traffic and transportation from Beijing Jiaotong University in 2014. He is currently a master's degree student in the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University. His research interests include visual tracking, computer vision, and machine learning.

Stephen Maybank received a BA in Mathematics from King's college Cambridge in 1976 and a PhD in computer science from Birkbeck college, University of London in 1988. Now he is a professor in the School of Computer Science and Information Systems, Birkbeck College. His research interests include the geometry of multiple images, camera calibration, visual surveillance etc.