Single and Multiple Object Tracking Using a Multi-Feature Joint Sparse Representation

Weiming Hu, Wei Li, and Xiaoqin Zhang
(National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190)
{wmhu, weili, xqzhang}@nlpr.ia.ac.cn

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

Abstract: In this paper, we propose a tracking algorithm based on a multi-feature joint sparse representation. The templates for the sparse representation can include pixel values, textures, and edges. In the multi-feature joint optimization, noise or occlusion is dealt with using a set of trivial templates. A sparse weight constraint is introduced to dynamically select the relevant templates from the full set of templates. A variance ratio measure is adopted to adaptively adjust the weights of different features. The multi-feature template set is updated adaptively. We further propose an algorithm for tracking multi-objects with occlusion handling based on the multi-feature joint sparse reconstruction. The observation model based on sparse reconstruction automatically focuses on the visible parts of an occluded object by using the information in the trivial templates. The multi-object tracking is simplified into a joint Bayesian inference. The experimental results show the superiority of our algorithm over several state-of-the-art tracking algorithms.

Index terms: Visual object tracking, Tracking multi-objects under occlusions, Multi-feature joint sparse representation

1. Introduction

The task of visual object tracking is to infer moving objects' states, e.g. location, scale, or velocity, from the observations in a video. Visual object tracking is a key component in numerous applications, such as visual surveillance, vision-based control, human-computer interfaces, intelligent transportation, and augmented reality. An appearance model is essential for tracking objects. Changes in illumination, viewpoint and pose, occlusions, and background clutter make it difficult to construct a robust appearance model which is capable of adapting to changes in the environment. Many appearance models [1, 2, 3, 4] have been proposed for object tracking. These include models based on histograms, kernel density estimates, Gaussian mixture models (GMMs), conditional random fields, subspaces, and discriminative classification.

Sparse representation-based object appearance models [5, 6] have only recently been applied to visual tracking. Mei et al. [5, 45] proposed a sparse approximation-based tracking algorithm using the L_1-regularized minimization. An observation in a frame is sparsely represented in the space spanned by object templates and trivial templates. The object templates are obtained from the previous frames. The trivial templates account for the pixels contaminated by occlusion or noise. The conventional sparse representation-based tracking only uses the pixel gray levels in the sequence of frames to construct the template set. The pixel gray level is not always representative enough to distinguish the object from the background or from other objects. It is necessary to construct a new and more powerful sparse representation-based appearance model which can combine useful features, such as colors, textures, or edges. In addition, it is still challenging to use a sparse representation-based appearance model to track multiple objects under occlusions.

In this paper, we propose a tracking algorithm based on a multi-feature joint sparse representation which is used to fuse the templates constructed from the different image features. A few templates, that are most representative for reconstructing each candidate observation, are obtained by determining the reconstruction coefficients of the templates of the multi-features. The observation with the smallest reconstruction error is chosen as the tracking result under the particle filtering framework. We further extend this multi-feature joint sparse reconstruction-based algorithm to multi-object tracking with occlusion handling. The location and size of the occluded parts are inferred using the trivial templates. The task of multi-object tracking with occlusion handling is reduced to a simple joint state Bayesian inference.

The main contributions of our work are as follows:

- A multi-feature joint sparse representation-based appearance model is proposed for object tracking. The variance ratio measure in [13] is introduced to estimate different features' weights for calculating the sparse reconstruction error. The templates of the objects are updated adaptively using the tracking results to keep the representative templates in the tracking process.
- We use a sparse weight constraint in the joint sparse representation to dynamically select the templates from the template set and estimate the coefficients of the templates. A relatively large pool of templates is kept in the template set. An accelerated proximal gradient (APG) algorithm is introduced to handle the multi-feature sparse representation task.
- A multi-feature joint sparse representation-based algorithm is proposed for tracking multi-objects through occlusions. Implicit and explicit methods for handling occlusions are proposed to determine the un-occluded parts of an object. The sparse reconstruction selects the visible parts of the object for matching with the templates. A cross iteration-based method is proposed to reduce the computational complexity of finding the optimal state in the joint state space of multi-objects.

We test our algorithms on different video sequences involving heavy occlusions, large changes in appearance and pose, and noisy backgrounds. With the above contributions, our work significantly extends the baseline [5].

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 summarizes the sparse representation-based appearance model. Section 4 proposes our multi-feature joint sparse representation-based appearance model. Section 5 describes the Bayesian state inference for single object tracking. Section 6 presents our algorithm for tracking multi-objects with occlusion handling. Section 7 shows the experimental results. Section 8 concludes the paper.

2. Related Work

In order to give the broad context, we briefly review the related work on appearance model-based tracking and multi-object tracking.

2.1. Appearance model-based tracking

In terms of the construction of appearance models, tracking algorithms can be categorized into generative model-based [1] and discriminative model-based [2]. The generative model-based methods [3, 4] describe the visual observations of a moving object, and then tracking is reduced to a search for an optimal state that yields an object appearance most similar to the appearance model. In the discriminative models, tracking is viewed as a binary classification based on an optimal decision boundary which distinguishes the object from the background.

2.1.1. Generative models

The useful methods for constructing the generative appearance models include color histograms, GMMs **(**Gaussian mixture models), subspace learning, and manifold learning.

Color histograms [37, 38] of image regions are widely used to construct generative appearance models because of the simplicity with which they are computed and their robustness to region scaling, rotation, and shape variations. Their limitation is that they usually ignore the spatial distribution of pixel values.

GMM-based appearance models use a mixture of weighted Gaussian distributions to learn a statistical model for colors. Jepson et al. [7] modeled the object appearance as a mixture of a stable component, a transient component, and an outlier component. Zhou et al. [8] modeled an object appearance using an adaptive mixture of Gaussians. Yu et al. [9] proposed a spatial appearance model which captures the properties of both local appearance changes and global spatial information about pixel values. Wang et al. [10] used a spatial-color mixture of Gaussians to model object appearance. On one hand, GMMs can be applied to individual pixels, and then the correlations between the values of nearby pixels are ignored. On the other hand, GMMs can be applied to object image regions' convolution outputs, and then the local structure of image intensity can be explicitly captured. The limitations of GMMs-based appearance models are that it is necessary but difficult to manually set the number of Gaussians and the updating of the GMMs-based appearance models requires a number of thresholds which have to be set manually, making automatic updating difficult.

In contrast to color histograms and GMMs-based appearance models which are pixel-based and thus sensitive to global appearance changes and noise, linear subspace learning-based appearance models [39] consider the image as a whole and then represent the image as a vector. Lim et al. [40] presented a human tracking algorithm based on a nonlinear dimension reduction technique. Ho et al. [41] presented a visual tracking algorithm based on linear subspace learning. Ross et al. [4] proposed a generalized tracking algorithm based on an incremental vector subspace learning method. Wu et al. [11] used an online tensor decomposition method to capture spatial layout information about appearances for tracking. Kwon and Lee [18] decomposed an observation model into multiple basic observation models that are constructed using the sparse principal component analysis (PCA). Nguyen et al. [35] proposed a probabilistic PCA-based method to model images using a low-dimensional joint Gaussian distribution between pixels. The second-order statistics among pixels were included in the covariance matrix of the weighted principal components. Subspace learning-based methods have the following limitations:

- When the appearance of an object has large changes, the correspondence of pixels between the object and the subspace is not accurate and samples may lie beyond the range of the linear subspace. This can cause tracking failure. Kernel methods can be used to describe the nonlinear distribution of samples, but it is difficult to select appropriate kernel functions in this application.
- The reconstruction error for subspace learning-based appearance models is usually based on the sum of squared differences. This makes the appearance models sensitive to image noise and partial occlusion.

Nonlinear manifold learning methods usually construct appearance models which depend on local features computed from the values of nearby pixels. Porikli *et al.* [12] proposed a Riemannian metric-based object tracking method in which object appearances were represented using the covariance matrix of image features. Tuzel *et al.* [43] proposed an algorithm for detecting people by classification on Riemannian manifolds. The limitations of these image covariance matrix-based appearance models are that they do not directly model occlusion and noise, and direct spatial information about position relations between pixels is partially lost.

2.1.2. Discriminative models

Discriminative models select different discriminative features and construct a classifier in the feature space to distinguish between the features from the object and the features from the background. Avidan [42] constructed offline a support vector machine (SVM)-based classifier which was incorporated into an optical flow-based tracker. Collins et al. [13] developed an efficient online feature ranking mechanism which was embedded in a tracking system. Avidan [14] combined several weak classifiers into a strong classifier to label pixels as object or background. Grabner et al. [15, 16] adopted online boosting to select discriminative local features for tracking. Saffari et al. [17] introduced a random forest algorithm, in which the decision trees were built online, to select features for tracking. Zhang et al. [51] proposed a tracking algorithm with an appearance model based on features extracted from the multi-scale image feature space. Samples of foreground objects and the background were compressed using the same sparse measurement matrix. The limitation of the discriminative models is that they are dependent on the selection of suitable positive and negative samples which are used to update the classifier. An unsuitable selection may cause tracker drift.

In summary, different appearance models, no matter whether they are generative or discriminative, have their own merits and limitations. It is interesting to develop a new robust tracking algorithm which effectively fuses multiple appearance cues and directly models occlusions and noise.

2.2. Multi-object tracking with occlusion handling

For multi-object tracking [48, 49, 50], if close objects have very similar appearances and they are tracked separately using a single object tracking algorithm, then the different trackers may lock onto the same object. To carry out multiple object tracking, a joint posterior or a joint likelihood for all the objects is usually maintained to explicitly model the interactions between objects. However, the computation of the joint state space is very expensive. Multiple object tracking is usually transformed into a constrained optimization problem by defining a one-to-one mapping constraint to simplify the computation. Then, optimization methods, such as the Hungarian algorithm or the network flow algorithm, can be adopted to handle multi-object tracking problem.

In multi-object tracking, occlusion handling based on high level associations is crucial. The usual way to tackle occlusions is to explicitly model the occlusion relations between different objects. Different models have been used. Rasmussen and Hager [30] deduced occlusion relations by analyzing the affiliation of each pixel to the different objects. Elgammal and Davis [31] segmented people under occlusion by incorporating the occlusion relations of the different depth layers into a likelihood function. Wu et al. [32] applied a Bayesian network to track two faces through occlusion in which an extra hidden process for the occlusion representation was introduced. Sudderth et al. [33] proposed a hand tracking algorithm in which the occlusion relations were inferred by using nonparametric belief propagation. The occlusion reasoning is complex, and it is very difficult to deduce the occlusion relations. If the deduced occlusion relations are not correct, then tracking may fail. In contrast to the explicit modeling of occlusion relations, an alternative is to implicitly handle occlusions by adopting particular rules. MacCormick and Blake [34] developed a data association filter based on a probabilistic exclusion principle to infer the object states during occlusions. Nguyen et al. [35] used the spatiotemporal context of each object to maintain the correct identity of the object during occlusions. Yang et al. [36] tracked multi-objects by finding the Nash equilibrium in a game. The limitation of the implicit methods is that it is difficult to define effective rules for occlusion handling. It is necessary to develop a multi-object tracking algorithm which has **a** high accuracy of occlusion handling while keeping high robustness and which does not rely on particular rules for finding occlusions.

3. Sparse Representation-Based Appearance Model

Traditional sparse representation-based tracking methods [5, 27, 28, 29] are applied directly to the intensity images, and trivial templates are used to model occlusion and noise. The global appearance of an object under different conditions can be well approximated by a low dimensional subspace spanned by a set of object templates. Therefore, during tracking, an appearance candidate of an object can be approximated by linear combinations of a set of templates which are selected from the tracking results in the previous frames. A tracking result is an image patch which is normalized to a particular size and stacked to produce a μ -dimensional vector, where μ equals to the number of the pixels in the normalized patch. Let $\{\mathbf{h}_i \in \mathbb{R}^{\mu}\}_{i=1}^{n}$ be a set of *n* templates, i.e., $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n] \in \mathbb{R}^{\mu \times n}$. An appearance candidate $\mathbf{m} \in \mathbb{R}^{\mu}$ of the object in a given image is approximated by a linear combination of $\{\mathbf{h}_i\}_{i=1}^{n}$ [6]:

$$\mathbf{m} = w_1\mathbf{h}_1 + w_2\mathbf{h}_2 + ... + w_n\mathbf{h}_n + \boldsymbol{\varepsilon} = \mathbf{Hw} + \boldsymbol{\varepsilon} \quad (1)$$

where $\mathbf{w} = [w_1, w_2, ..., w_n]^T \in \mathbb{R}^n$ is a coefficient vector and **ε** is a noise term. If the appearance of the object is badly affected by noise or occlusion, then the components of **ε** may be very large. A set of μ trivial templates $\{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, ..., \boldsymbol{\tau}_{\mu}\}$ is defined to encode the values of **ε**. In the *i*-th trivial template, the *i*-th pixel value is nonzero and all the other pixel values are 0. Each trivial template i also is stacked to produce a μ-dimensional vector, i.e., $\boldsymbol{\tau}_i \in \mathbb{R}^{\mu}$. Let $\mathbf{T} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, ..., \boldsymbol{\tau}_\mu] \in \mathbb{R}^{\mu \times u}$. The discrepancy $\boldsymbol{\varepsilon}$ is encoded using a linear combination of the trivial templates:

$$\boldsymbol{\varepsilon} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, ..., \boldsymbol{\tau}_\mu][e_1, e_2, ..., e_\mu]^T = \mathbf{Te} \tag{2}$$

where e_i is the coefficient of the i-th trivial template and $\mathbf{e} = [e_1, e_2, ..., e_\mu]^T \in \mathbb{R}^{u}$. The composition of the object templates and the trivial templates for a candidate observation is shown in Fig. 1, where a trivial template is shown as a dark square which contains a single white dot corresponding to the nonzero value. The appearance $\mathbf{m}$ in (1) is rewritten as:

$$\mathbf{m} = [\mathbf{H} \quad \mathbf{T}] \begin{bmatrix} \mathbf{w} \\ \mathbf{e} \end{bmatrix} = \mathbf{Bf} \tag{3}$$

where $\mathbf{B} = [\mathbf{H} \quad \mathbf{T}] \in \mathbb{R}^{\mu \times (n+\mu)}$ and $\mathbf{f} = \begin{bmatrix} \mathbf{w}^T & \mathbf{e}^T \end{bmatrix}^T \in \mathbb{R}^{(n+\mu)}$.

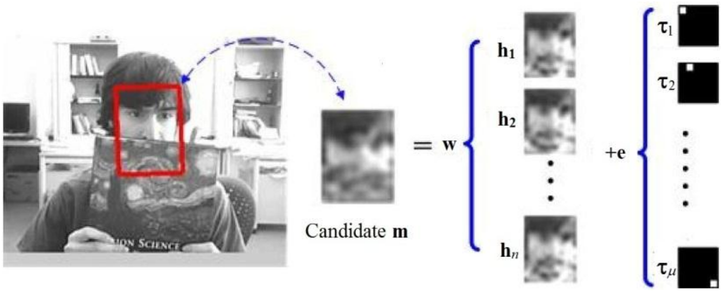


Fig. 1. An illustration of representation with object templates and trivial templates for a candidate observation.

Videos are highly redundant data. Much information, such as the appearance of an object in consecutive frames, is repetitive. It is intuitive that the appearance of a tracked object in a video can be sparsely represented by its appearances in the previous frames. So, an observation $\mathbf{m}$ can be described by a linear combination of object templates $\mathbf{H}$ and trivial templates $\mathbf{T}$, with the vector $\mathbf{f}$ constrained to be sparse. As a basic strategy in sparse representation, the sparseness of $\mathbf{f}$ is described by the L_1 norm of $\mathbf{f}$, which is defined as $\|\mathbf{f}\|_1 = \sum_i |f_i|$ [5, 26]. Then, the optimal coefficient vector $\mathbf{f}$ can be found by minimizing the square of the reconstruction error for $\mathbf{m}$ with an L_1 regularization. Let $\|\cdot\|_2$ be the L_2 norm which is defined as $\|\mathbf{y}\|_2 = \sqrt{\sum_i y_i^2}$ for a vector $\mathbf{y}$. Then, the optimization is formulated by:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \left(\|\mathbf{m} - \mathbf{Bf}\|_2^2 + \lambda \|\mathbf{f}\|_1 \right) \tag{4}$$

where λ is a regularization factor and the L_2 norm is used to specify the reconstruction error. The minimization in (4) can be carried out efficiently by linear programming [19].

In (4), there is tradeoff between the reconstruction error and the sparseness. When the occlusion is large, to decrease the reconstruction error, the combination of trivial templates may no longer be sparse in spite of the sparseness regularization. But, the optimal observation corresponding to the tracking result can obtain less dense representation of the trivial templates, i.e., more un-occluded pixels can take part in estimating the reconstruction error. Therefore, introduction of trivial templates makes the sparse representation more robust to dense occlusions [5, 6].

4. Multi-Feature Joint Sparse Representation-Based Appearance Model

In contrast with the traditional intensity images-based sparse representation appearance models for tracking, we propose a multi-feature joint sparse representation-based appearance model which fuses multiple cues, such as hue, saturation, intensity, the gradient-based edge template, and the texture feature obtained by Gabor filtering [13], in order to achieve more robust tracking. According to the intuition that if object appearances are similar all their features are similar, once an object template is selected for sparse representation of an observation, all the features of the template should be valid for the sparse representation. So, joint sparse representation of multi-features can effectively combine multi-features. For each feature, the sparse representation, together with its occlusion or noise handling, are managed in the same way that the traditional sparse representation-based methods manage the intensity image. All feature templates are fused using the $L_{2,1}$ norm-based joint sparse representation which is a standard framework for information fusion, with a sound theoretical basis [52, 53] and a simple implementation.

4.1. Multi-feature joint sparse representation

We extract *K* different types of feature from each object appearance template to form *K* feature templates. Each type of feature corresponds to one feature template. The dimension of each feature is equal to the number of pixels in an object appearance template. For each feature *k*, the feature template set $\mathbf{H}^k$ is $\mathbf{H}^k = \{\mathbf{h}_1^k, \mathbf{h}_2^k, ..., \mathbf{h}_n^k\}$, where *n* is the number of the object appearance templates. The *k*-th feature vector $\mathbf{m}^k$ of a candidate observation is represented using the following equation:

$$\mathbf{m}^k = \sum_{i=1}^{n} w_i^k \mathbf{h}_i^k + \boldsymbol{\varepsilon}^k \tag{5}$$

where $\mathbf{w}^k = [w_1^k, ..., w_n^k]^T$ is the reconstruction coefficient vector for the *k*-th feature, and $\boldsymbol{\varepsilon}^k$ is the residual term. The $\{\mathbf{w}^k\}_{k=1}^K$ are found using the least square regressions with the $L_{2,1}$ mixed-norm regularization [20, 21]:

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \left(\frac{1}{2} \sum_{k=1}^{K} \left\| \mathbf{m}^k - \sum_{i=1}^{n} w_i^k \mathbf{h}_i^k \right\|_2^2 + \lambda \|\mathbf{W}\|_{2,1} \right) \tag{6}$$

where $\mathbf{W} = [\mathbf{w}^1, \mathbf{w}^2, ..., \mathbf{w}^K] \in \mathbb{R}^{n \times K}$. The $L_{2,1}$-mixed norm of **W** is:

$$\|\mathbf{W}\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{k=1}^{K} (w_i^k)^2} = \sum_{i=1}^{n} \|\mathbf{w}_i\|_2 \tag{7}$$

where $\mathbf{w}_i = [w_i^1, w_i^2, ...w_i^K]^T \in \mathbb{R}^K$. The $L_{2,1}$ mixed-norm regularization includes the L_2 norm of the coefficient vector $\mathbf{w}_i$ for each object appearance template *i* and the L_1 norm of the vector of the L_2 norm values for all the object appearance templates. The $L_{2,1}$ mixed-norm guarantees joint sparse representation, because:

- The L_1 norm in the $L_{2,1}$ mixed-norm ensures that the object appearance templates chosen to represent an observation are as few as possible;

- The L_2 norm in the $L_{2,1}$ norm ensures that when $\sum_{k=1}^{K}\left(w_i^k\right)^2=0$, the coefficients $\{w_i^k\}_{k=1}^{K}$ of all the K feature templates for object appearance template i are equal to 0, i.e. when the object appearance template i is not chosen to represent an observation, all the feature templates of the object appearance template i should also not be chosen to represent the observation.

4.2. Multi-feature joint sparse representation-based appearance model

If templates dissimilar to the observation are selected as nonzero entries for the sparse representation, then there is an increased probability that a background observation is erroneously chosen as the tracking result. To make sure that the observation lies in the span of the representative templates, we define a weight for the coefficient of each template so as to favor the selection of templates which are similar to the observation. As the tracking result in the current frame is similar to the tracking result in the previous frame, we weight the object template coefficients using the Euclidean distances from the tracking result in the previous frame to the feature templates. We define a weight matrix $\mathbf{D}_i \in \mathbb{R}^{K\times K}$ for object template i as:

$$\mathbf{D}_i = diag\left(\left\|\tilde{\mathbf{m}}^1-\mathbf{h}_i^1\right\|,\left\|\tilde{\mathbf{m}}^2-\mathbf{h}_i^2\right\|,\ldots,\left\|\tilde{\mathbf{m}}^k-\mathbf{h}_i^k\right\|,\ldots,\left\|\tilde{\mathbf{m}}^K-\mathbf{h}_i^K\right\|\right) \tag{8}$$

where $\mathbf{h}_i^k$ is the k-th feature’s template of the i-th appearance template and $\tilde{\mathbf{m}}^k$ is the k-feature vector of the tracking result in the previous frame. The trivial templates are added into (6) as specified in (3). Let $\mathbf{f}^k=[(\mathbf{w}^k)^T,(\mathbf{e}^k)^T]^T$ ($k=1,\ldots,K$) and $\mathbf{F}=[\mathbf{f}^1,\mathbf{f}^2,\ldots,\mathbf{f}^K]\in\mathbb{R}^{(n+\mu)\times K}$. Let $\mathbf{e}_j=[e_j^1,e_j^2,\ldots,e_j^K]^T\in\mathbb{R}^K$ ($j=1,2,\ldots,\mu$). Then, Equation (6) becomes:

$$\hat{\mathbf{F}}=\min_{\mathbf{F}}\left(\frac{1}{2}\sum_{k=1}^{K}\left\|\mathbf{m}^k-\mathbf{B}^k\mathbf{f}^k\right\|_2^2+\lambda\left(\sum_{i=1}^{n}\left\|\mathbf{D}_i\mathbf{w}_i\right\|_2+\sum_{j=1}^{\mu}\left\|\mathbf{e}_j\right\|_2\right)\right), \tag{9}$$

where the weight matrix $\mathbf{D}_i$ acts on the feature templates but not on the trivial templates. It is seen that by using (8) information about previous tracks is incorporated into the optimization objective.

In the traditional sparse representation-based tracking [5] in which the coefficients of the templates are not weighted, the size of the set of object templates is usually required to be small in order to control the computational complexity for handling the sparse optimization problem. This excludes many templates which would be useful for the subsequent tracking. Inspired by locality-constrained linear coding [22], we introduce a threshold to constrain the coefficients in order to enlarge the size of the set of the templates without incurring a large computational cost for solving the sparse optimization. The weight matrix $\mathbf{D}_i$ in (8) is constrained and redefined as:

$$D_i^k=\begin{cases}\dfrac{\left\|\tilde{\mathbf{m}}^k-\mathbf{h}_i^k\right\|}{\max\limits_i\left\|\tilde{\mathbf{m}}^k-\mathbf{h}_i^k\right\|} & if\left\|\tilde{\mathbf{m}}^k-\mathbf{h}_i^k\right\|\le\epsilon\\ \infty & \text{otherwise}\end{cases}. \tag{10}$$

where ϵ is a threshold. This equation ensures that D_i^k grows linearly with the difference $\left\|\tilde{\mathbf{m}}^k-\mathbf{h}_i^k\right\|$, until the difference exceeds the threshold ϵ [46]. Beyond the threshold, D_i^k is set as infinitely large, which forces the coefficient of the template to 0. In this way, many more templates can be retained in the template set. Using the weights in (10) to simplify the search for a sparse representation is reasonable, as the aim in a sparse representation is to choose as few templates as possible.

An approximated accelerated proximal gradient (APG) algorithm [20, 23] is used to efficiently solve the $L_{2,1}$ norm-based optimization in (9). Given the feature template sets $\{\mathbf{H}^k\}_{k=1}^K$, the algorithm consists of alternate and iterative updating of the coefficient matrix $\mathbf{F}^t = [f_i^{k,t}]_{i=1,\dots,n+\mu}^{k=1,\dots,K}$ and an aggregation matrix $\mathbf{V}^t = [v_i^{k,t}]_{i=1,\dots,n+\mu}^{k=1,\dots,K}$, where *t* indexes an iteration. Each iteration *t*+1 consists of the following two steps:

- a generalized gradient mapping step to update the matrix $\mathbf{F}^{t+1}$ using the aggregation matrix $\mathbf{V}^t$ obtained in the previous iteration *t*.
- an aggregation step to update $\mathbf{V}^{t+1}$ by linearly combining $\mathbf{F}^{t+1}$ and $\mathbf{F}^t$.

In the generalized gradient mapping step, given $\mathbf{V}^t$, $\mathbf{F}^{t+1}$ is updated using the following two formulae:

$$\mathbf{f}^{k,t+1} \leftarrow \tilde{\mathbf{d}}^k \odot (\boldsymbol{v}^{k,t} - \alpha \boldsymbol{\beta}^{k,t}), \quad k = 1, 2, \dots, K \tag{11}$$

$$\mathbf{f}_i^{t+1} \leftarrow \max\left\{1 - \frac{\alpha\lambda}{\left\|\mathbf{f}_i^{t+1}\right\|_2}, 0\right\} \mathbf{f}_i^{t+1}, \quad i = 1, 2, \dots, n \tag{12}$$

where

$$\boldsymbol{\beta}^{k,t} \leftarrow -(\mathbf{B}^k)^T \mathbf{m}^k + (\mathbf{B}^k)^T (\mathbf{B}^k) \mathbf{v}^{k,t} \tag{13}$$

α is the step size, $\tilde{\mathbf{d}}^k = [1/D_1^k, 1/D_2^k, \dots, 1/D_n^k, 1, \dots, 1]^T \in \mathbb{R}^{n+\mu}$ to ensure that only the coefficients of the object templates are affected by the weight constraint, and "$\odot$" denotes the element-wise product of two vectors. In the aggregation step, the aggregation matrix $\mathbf{V}^{t+1}$ is updated by linearly combining $\mathbf{F}^{t+1}$ and $\mathbf{F}^t$:

$$\mathbf{V}^{t+1} \leftarrow \mathbf{F}^{t+1} + \frac{\gamma_{t+1}(1-\gamma_t)}{\gamma_t}(\mathbf{F}^{t+1} - \mathbf{F}^t) \tag{14}$$

where γ_t is usually set as $\gamma_t = 2/(2+t)$. Table 1 summaries the optimization procedure of the multi-feature joint sparse representation for an observation.

Table 1: The multi-feature joint sparse representation for an observation

```
Input: templates $\{\mathbf{H}^k\}_{k=1}^K$, an observation $\{\mathbf{m}^k\}_{k=1}^K$, the regularization parameter $\lambda$, and the step size value $\alpha$.
Initialization: Initialize $\mathbf{f}^{k,0}$ and $v^{k,0}$; Set $\gamma_0 = 1$ and $t \leftarrow 0$.
Repeat {Main loop}
    $\mathbf{f}^{k,t+1} \leftarrow \tilde{\mathbf{d}}^k \odot (\mathbf{v}^{k,t} - \alpha(-(\mathbf{B}^k)^T \mathbf{m}^k + (\mathbf{B}^k)^T (\mathbf{B}^k) \mathbf{v}^{k,t}))$, $k$ =1, 2, ..., $K$
    $\mathbf{f}_i^{t+1} \leftarrow \max\left\{1 - \frac{\alpha\lambda}{\|\mathbf{f}_i^{t+1}\|_2}, 0\right\} \mathbf{f}_i^{t+1}$, $i$=1,2, …, $n$
    $\gamma_t = \frac{2}{t+2}$
    $\mathbf{V}^{t+1} \leftarrow \mathbf{F}^{t+1} + \frac{\gamma_{t+1}(1-\gamma_t)}{\gamma_t}(\mathbf{F}^{t+1} - \mathbf{F}^t)$
        t ← t +1,
until convergence is reached
Output: F.
```

Using the optimal coefficients $\mathbf{w}^k$ for each feature *k* obtained by the algorithm shown in Table 1, the *k*-th feature vector $\mathbf{m}^k$ of the candidate sample $\mathbf{M}=[\mathbf{m}^1,..,\mathbf{m}^k,...,\mathbf{m}^K]$ is reconstructed by $\mathbf{H}^k\mathbf{w}^k$. Then, the reconstruction error *R*(**M**) of **M** is accumulated over all the *K* features:

$$R(\mathbf{M})=\sum_{k=1}^{K}\theta^k\left\|\mathbf{m}^k-\mathbf{H}^k\mathbf{w}^k\right\|_2^2 \tag{15}$$

where $\{\theta^k\}_{k=1}^K$ are the weights that measure the influences of different features on the reconstruction error. Usually, θ^k is set as 1/*K* for simplicity, meaning that different features have the same influence on the reconstruction error.

4.3. Feature weight adaptation

We introduce a variance ratio [13] to adaptively adjust the weights $\{\theta^k\}$ for each feature *k*, in order to give a larger weight to a feature which has more ability to distinguish between the object and the background. We determine this ability for each feature using the pixels within the object region and the background pixels lying inside a rectangle surrounding the object region.

For each feature, two histograms are used to estimate the distributions of the values of the object pixels and the values of the particular background pixels, respectively. Let $\{p(i)\}_{i=1,...,O}$ and $\{q(i)\}_{i=1,...,O}$ be the L_1 normalized histograms of the values of the object pixels and the values of the background pixels, respectively, where *O* is the number of bins and *i* indexes a bin. The log likelihood $l(i)$ for the *i*-th bin is computed by:

$$l(i)=\log\frac{\max(p(i),\xi)}{\max(q(i),\xi)} \tag{16}$$

where ξ>0 is a small value which is used to ensure that the numerator and the denominator are not equal to zero. As the variance $\upsilon(x)$ of a random variable *x* equals to the expectation of x^2 minus the square of the expectation of *x*: $\upsilon(x)=E(x^2)-(E(x))^2$, the variance $\upsilon(\mathbf{l},\mathbf{p})$ of $\{l(i)\}_{i=1,...,O}$ with respect to the histogram $\{p(i)\}_{i=1,...,O}$ is computed by:

$$\upsilon(\mathbf{l},\mathbf{p})=E(l^2(i))-(E(l(i)))^2=\sum_{i=1}^{O}\left(p(i)l^2(i)\right)-\left(\sum_{i=1}^{O}p(i)l(i)\right)^2. \tag{17}$$

Similarly, we compute the variance $\upsilon(\mathbf{l},\mathbf{q})$ of $\{l(i)\}_{i=1,...,O}$ relative to $\{q(i)\}_{i=1,..,O}$ and the variance $\upsilon(\mathbf{l},(\mathbf{p}+\mathbf{q})/2)$ of $\{l(i)\}_{i=1,...,O}$ relative to $\left\{\left(p(i)+q(i)\right)/2\right\}_{i=1,..,O}$. The variance ratio $ƛ$ of the log likelihood function is defined as:

$$ƛ(\mathbf{l},\mathbf{p},\mathbf{q})=\frac{\upsilon(\mathbf{l},(\mathbf{p}+\mathbf{q})/2)}{\upsilon(\mathbf{l},\mathbf{p})+\upsilon(\mathbf{l},\mathbf{q})}. \tag{18}$$

The denominator of (18) is the sum of the within class variances for the object and background classes, and the numerator is the total variance of the data from both the object and the background. The intuition behind the variance ratio is that log likelihood values of pixels on both the object and the background are expected to be tightly clustered, and the two clusters are expected to be spread apart as much as possible [13]. So, this variance ratio for the feature can evaluate the feature's ability to distinguish between the object and the background.

The weight θ^k for each feature k is set to the normalized variance ratio of the corresponding feature:

$$\theta^k = \frac{\bar{\lambda}(\mathbf{l}^k, \mathbf{p}^k, \mathbf{q}^k)}{\sum_{j=1}^{K} \bar{\lambda}(\mathbf{l}^j, \mathbf{p}^j, \mathbf{q}^j)} . \tag{19}$$

4.4. Updating of object templates

During tracking, the template set is updated online using the most recent tracking results to capture changes in the object appearance and keep more representative template in the template set, in order that tracking can be carried out effectively even when the size of the template set is not large. Our adaptive updating strategy essentially follows the one proposed in [5]. The differences between our updating strategy and the strategy in [5] and the justifications of the differences are summarized below:

- The templates obtained in the first few frames are kept fixed during the whole tracking process. This can reduce the possibility of tracking drift, as the first templates are usually reliable. In [5], the templates obtained in the first frame were treated equally with other templates and could be replaced.
- The weight φ of each object appearance template i is updated using the coefficients of all the K features: $\varphi \leftarrow \varphi \sum_{k=1}^{K} \exp(w_i^k)$. The correlations between different features in each object appearance in the joint sparse representation are reflected in the weight φ, in contrast to [5] in which only the gray feature is used for the template weight updating.
- We add a forgetting factor in the estimation of the weight of each object appearance template. Namely, the weight φ is further updated by $\varphi \leftarrow \varphi \delta^{G-g}$, where $\delta \in [0,1]$ is a forgetting factor, G is the current time, and $g \in [1, G]$ is the time of the template being added into the template set. Then, the weight assigned to the object appearance template decreases over time and captures the most recent changes. This forgetting factor is not used in [5].
- The difference between the tracking result **M** and an object appearance template i is estimated by:

$$\sqrt{\sum_{k=1}^{K} \theta^k \left\| \mathbf{m}^k - \mathbf{h}_i^k \right\|_2^2} \tag{20}$$

If the difference between the new tracking result and the object template with the lowest weight in the current template set is larger than a threshold, then this object template is replaced with the new tracking result to reflect the changes in appearance. The new template is initialized to have the median weight of the current templates. In [5], the difference between an observation and a template is defined using the cosine of the angle between the feature vectors for the observation and the template. The cosine cannot be directly applied to the estimation of similarity between vector sets, each of which consists of multi-feature vectors with different weights. So, we use, instead, (20) to estimate the similarity between the sets of feature vectors.

- Occlusions and noise are considered in our template updating strategy. If the proportion of an object's pixels which are not badly affected by noise or occlusion in the object, as determined using the trivial templates, are larger than a threshold, then the appearance model of the object is updated, otherwise the template updating is not carried out. This avoids updating affected by large occlusions or noise.

4.5. Remark

Using the weights in (10), the standard size to which each appearance is normalized, the APG algorithm, and the standard size of the template set make it practical to solve the sparse coding for each frame.

5. Bayesian State Inference for Single Object Tracking

The object is localized using a rectangular window and its state at frame *t* is represented using a six-dimensional affine parameter vector $\mathbf{z}_t = [x_t^1, x_t^2, r, c, a, \phi]$, where x_t^1 and x_t^2 are the 2D translation parameters and (r, c, a, ϕ) are the deformation parameters, corresponding to the rotation angle, the scale, the aspect ratio, and the skew direction, respectively. Given an observation $\mathbf{M}_t = [\mathbf{m}_t^1, ...\mathbf{m}_t^K] \in \mathbb{R}^{\mu \times K}$, the task of tracking is to infer the state $\mathbf{z}_t$ of the object. This can be represented as a Bayesian posterior probability inference [24]:

$$p(\mathbf{z}_t \mid \mathbf{M}_t) \propto p(\mathbf{M}_t \mid \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p(\mathbf{z}_{t-1} \mid \mathbf{M}_{t-1}) d\mathbf{z}_{t-1} \tag{21}$$

where $p(\mathbf{M}_t \mid \mathbf{z}_t)$ is the observation model and $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ is the dynamic model. A particle filter [24] approximates the posterior probability density using a set of weighted particles.

The observation model $p(\mathbf{M}_t \mid \mathbf{z}_t)$, which reflects the similarity between $\mathbf{M}_t$ and the template set, is estimated by:

$$p(\mathbf{M}_t \mid \mathbf{z}_t) \propto \exp\left\{ -\eta \sum_{k=1}^{K} \theta^k \left\| \mathbf{m}_t^k - \mathbf{H}^k \mathbf{w}^k \right\|_2^2 \right\} \tag{22}$$

where η is a scaling factor controlling the shape of the Gaussian function.

A Gaussian distribution $G(\mathbf{z}_{t-1}, \Sigma)$ is used to model the state transition distribution corresponding to the dynamic model, where the mean vector $\mathbf{z}_{t-1}$ of the Gaussian distribution is the affine parameter vector estimated at the previous frame *t*-1, and the covariance matrix Σ is set empirically. The particles at the current time *t* are drawn from the state transition distribution $G(\mathbf{z}_{t-1}, \Sigma)$. During the sampling, if there is a negative value in the quantities which are strictly positive, then the sample is ignored.

The weight of each particle $\mathbf{z}_t$ is evaluated using $p(\mathbf{M}_t \mid \mathbf{z}_t)$. Maximum a posterior (MAP) estimation is used to estimate the state of the object at each frame: Observations associated with the particles are found, and the observation which is most similar to the appearance model is taken as the tracking result. Table 2 summarizes the implementation of the tracking algorithm.

Table 2: The tracking Algorithm

1: Input the set of the templates $\{\mathbf{H}^k\}_{k=1}^K$ which has been updated at the previous frame *t*-1, and the state $\mathbf{z}_{t-1}$ of the object at frame *t*-1.

2: Use the dynamic model $G(\mathbf{z}_{t-1},\Sigma)$ to produce a number of particles at the current frame *t*.

3: Use the algorithm shown in Table 1 to estimate the multi-feature joint sparse representation for each particle $\mathbf{z}_t$ at the current frame *t*.

4: Evaluate the weight of each particle $\mathbf{z}_t$ using $p(\mathbf{M}_t \mid \mathbf{z}_t)$ in (22).

5: Take the observation specified by the particle with the largest weight as the tracking result.

6: The set of the templates is updated using the method in Section 4.4.

7. Output the tracking result at the current frame and the new set of templates.

6. Multi-Object Tracking with Occlusion Handling

Our algorithm for multi-object tracking with occlusion handling is an extension of our single object tracking algorithm. We tackle non-severe occlusions and severe (or complete) occlusions using different strategies for defining the observation model. This is because when severe or complete occlusion happens, there are not enough visible parts of the object to provide reliable matching between the observation and the appearance model. The dynamical model, which is a prior model of the object's motion, is unchanged during the whole tracking process. The templates of each object are updated using the strategy in Section 4.4. For simplicity, we focus on describing handling of occlusions between two objects. The method is easily extended to more objects.

6.1. Observation model for non-severely occluded object

When there is no severe occlusion, we use the trivial templates in the appearance model to construct the observation model for each occluded object.

When occlusion occurs between two objects A and B, the observation of one object may be dependent on the state of the other object. So, the observation model of object A can be represented by $p(\mathbf{M}_t^A \mid \mathbf{z}_t^A, \mathbf{z}_t^B)$. To represent the visible parts of the object, we define a matrix $[\vartheta_i^k]_{i=1,..,\mu}^{k=1,...,K}$ such that ϑ_i^k is set to 0, if the *i*-th pixel is considered to be occluded, otherwise set to 1. Let Γ be the number of the elements whose values are 1 in ϑ. In order to accurately estimate the similarity between the observation and the appearance model, we define the observation model based on the visible parts of an object as follows:

$$p(\mathbf{M}_t^A \mid \mathbf{z}_t^A, \mathbf{z}_t^B) = \frac{1}{\Omega}\exp\left(-\frac{\zeta}{\Gamma}\sum_{k=1}^{K}\theta^k \left\|(\mathbf{m}_t^k - \mathbf{H}^k\mathbf{w}^k)\odot\boldsymbol{\vartheta}^k\right\|_2^2\right) \tag{23}$$

where ζ is a scaling factor, and

$$\begin{aligned}\Omega &= \int \exp\left(-\frac{\zeta}{\Gamma}\sum_{k=1}^{K}\theta^k \left\|(\mathbf{m}_t^k - \mathbf{H}^k\mathbf{w}^k)\odot\boldsymbol{\vartheta}^k\right\|_2^2\right) d(\mathbf{M}_t^A) \\ &= \prod_{k=1}^{K}\int \exp\left(-\frac{\zeta}{\Gamma}\theta^k \left\|(\mathbf{m}_t^k - \mathbf{H}^k\mathbf{w}^k)\odot\boldsymbol{\vartheta}^k\right\|_2^2\right) d(\mathbf{m}_t^k)\end{aligned}. \tag{24}$$

We explain two points about (23) and (24):

- Given an observation $\mathbf{M}_t$, the coefficients of the templates are recovered using the multi-feature joint sparse representation algorithm, and the identifier variables $\{\vartheta\}$ are determined using the recovered coefficients of the trivial templates. So, the identifier variables as well as the coefficients of the templates can be treated as functions of $\mathbf{M}_t$. Then, Ω in (24) is a constant for a given frame.

- We use Γ to handle changes in the extent of the visible parts of the object in different frames, i.e., take into account the area of the occluded portion of the object.

We propose two ways for estimating the matrix $[\vartheta_i^k]_{i=1,...,\mu}^{k=1,...,K}$. The first way is implicit, and does not involve reasoning about the occlusion relations between objects. The second way depends on explicit reasoning about occlusion relations.

- In the implicit way, the matrix $[\vartheta_i^k]_{i=1,..,\mu}^{k=1,...,K}$ is estimated simply using the values of the recovered feature trivial template coefficients $[e_i^k]_{i=1,..,\mu}^{k=1,...,K}$, as these values include the information about the position and size of the occluded parts of the object. If a value in $[e_i^k]_{i=1,..,\mu}^{k=1,...,K}$ is larger than a threshold, it is considered that the corresponding pixel is occluded and the corresponding element in $[\vartheta_i^k]_{i=1,..,\mu}^{k=1,...,K}$ is set to 0, otherwise set to 1. The merit of this implicit way is that it is simple, and effective in most cases. Its limitation is that when objects which are easily confused with one another also occlude one another, the occlusion information in the trivial templates may not be very accurate.
- In the explicit way, we use the information in the trivial templates to determine object occlusion relations. Given the states of two objects, the observations corresponding to the states are specified. The optimal coefficients of the templates and the trivial templates of the two objects are estimated, and the pixels in the overlapped region between the two observations are found. Within the overlapped region, the number of the pixels, for which the coefficients of the trivial templates are larger than the predefined threshold, is calculated for each object. It is determined that the object with the greater number of such pixels is occluded by the other object. In the matrix $[\vartheta_i^k]_{i=1,..,\mu}^{k=1,...,K}$ for the occluded object, all the elements which correspond to the pixels within the overlapped region are set to 0. The merit of this explicit way is that it can correct the errors which arise when objects which are easily confused with one another also occlude one another. Its limitation is that it depends on the stability of the occlusion relation reasoning.

6.2. Multi-object tracking with non-severe occlusion handling

Let $\mathbf{z}_t^{\mathrm{J}} = [(\mathbf{z}_t^{\mathrm{A}})^T, (\mathbf{z}_t^{\mathrm{B}})^T]^T$ be the joint state of objects A and B at frame *t*. Given the observation $\mathbf{M}_t$, inferring $\mathbf{z}_t^{\mathrm{J}}$ with occlusion handling is represented as a Bayesian posterior estimation:

$$p(\mathbf{z}_t^{\mathrm{J}}, \omega_t \mid \mathbf{M}_t) \propto p(\mathbf{M}_t \mid \mathbf{z}_t^{\mathrm{J}}, \omega_t) \int p(\mathbf{z}_t^{\mathrm{J}} \mid \mathbf{z}_{t-1}^{\mathrm{J}}) p(\mathbf{z}_{t-1}^{\mathrm{J}} \mid \omega_{t-1}, \mathbf{M}_{t-1}) d\mathbf{z}_{t-1}^{\mathrm{J}} \quad (25)$$

where ω_t is the occlusion relation between A and B.

In our algorithm, the occluded and visible parts of an object are automatically explored during the sparse reconstruction based on the $L_{2,1}$ norm. By using the observation model in (23) to measure the similarity between the observation $\mathbf{M}_t$ warped by the object state $\mathbf{z}_t^{\mathrm{J}}$ and the templates, the Bayesian posterior inference for tracking objects A and B can be simplified in the following way:

$$p(\mathbf{z}_t^{\mathrm{J}} \mid \mathbf{M}_t) \propto p(\mathbf{M}_t \mid \mathbf{z}_t^{\mathrm{J}}) \int p(\mathbf{z}_t^{\mathrm{J}} \mid \mathbf{z}_{t-1}^{\mathrm{J}}) p(\mathbf{z}_{t-1}^{\mathrm{J}} \mid \mathbf{M}_{t-1}) d(\mathbf{z}_{t-1}^{\mathrm{J}}) . \tag{26}$$

With this simplified inference expression, the posterior probability $p(\mathbf{z}_t^{\mathrm{J}} \mid \mathbf{M}_t)$ can be approximated using a set of weighted particles in the joint state space of A and B. Given a set of particles $\{\mathbf{z}_t^{\mathrm{A},i}, \mathbf{z}_t^{\mathrm{B},i}\}$ generated from the transition model $p(\mathbf{z}_t^{\mathrm{A}} \mid \mathbf{z}_{t-1}^{\mathrm{A}})$ and $p(\mathbf{z}_t^{\mathrm{B}} \mid \mathbf{z}_{t-1}^{\mathrm{B}})$, the weight of each particle *i* is evaluated using the observation likelihood $p(\mathbf{M}_t \mid \mathbf{z}_t^{\mathrm{J}})$ which is estimated as follows:

$$p(\mathbf{M}_t \mid \mathbf{z}_t^{\mathrm{J}}) = p(\mathbf{M}_t^{\mathrm{A}} \mid \mathbf{z}_t^{\mathrm{A}}, \mathbf{z}_t^{\mathrm{B}}) p(\mathbf{M}_t^{\mathrm{B}} \mid \mathbf{z}_t^{\mathrm{B}}, \mathbf{z}_t^{\mathrm{A}}) . \tag{27}$$

The optimal state $\mathbf{z}_t^{\mathrm{J}}$ corresponds to the particle which has the maximal weight.

If we estimate the weights of all the particles in the joint state space for multi-objects to find the optimal state of the objects, the time complexity is $O(N^{\Upsilon})$, where *N* is the number of particles per object and Υ is the number of objects. When the number of the tracked objects is more than two, the calculation is too time consuming. To increase the efficiency, a cross iteration is adopted. Let $\mathbf{z}_{t,s}^{\mathrm{A}}$ and $\mathbf{z}_{t,s}^{\mathrm{B}}$ be the estimated optimal states for the objects A and B at the *s*-th iteration. The optimal states of objects A and B at the next iteration are estimated using the following formulae:

$$\hat{\mathbf{z}}_{t,s+1}^{\mathrm{A}} = \arg\max_{\mathbf{z}_t^{\mathrm{A}}} p(\mathbf{M}_t^{\mathrm{A}} \mid \mathbf{z}_{t,s}^{\mathrm{A}}, \mathbf{z}_{t,s}^{\mathrm{B}}) p(\mathbf{M}_t^{\mathrm{B}} \mid \mathbf{z}_{t,s}^{\mathrm{B}}, \mathbf{z}_{t,s}^{\mathrm{A}}) \tag{28}$$

$$\hat{\mathbf{z}}_{t,s+1}^{\mathrm{B}} = \arg\max_{\mathbf{z}_t^{\mathrm{B}}} p(\mathbf{M}_t^{\mathrm{A}} \mid \hat{\mathbf{z}}_{t,s+1}^{\mathrm{A}}, \mathbf{z}_{t,s}^{\mathrm{B}}) p(\mathbf{M}_t^{\mathrm{B}} \mid \mathbf{z}_{t,s}^{\mathrm{B}}, \hat{\mathbf{z}}_{t,s+1}^{\mathrm{A}}) . \tag{29}$$

The cross-iteration of (28) and (29) continues until convergence. The time complexity of the algorithm decreases to $O(N\Upsilon)$ from $O(N^{\Upsilon})$. This ensures that when the number of objects increases, the runtime does not quickly increase.

6.3. Handling of severe occlusions

When an object is severely occluded (for example more than 70% percent of the object is occluded), there are not enough visible parts of the object to provide reliable matching between the observation and the appearance model. If complete occlusion occurs, it is impossible to evaluate the observations using the appearance model. In the case of severe or complete occlusions, object motion information in the previous frames is more reliable. In these circumstances, we construct a new observation model $p(\mathbf{M}_t \mid \mathbf{z}_t)$, which takes the motion velocity constraint into consideration, to estimate the position of the object.

Let $\vec{\mathbf{v}}_{t-1}^{i} = \mathbf{x}_{t-1}^{i} - \mathbf{x}_{t-2}^{i}$ and $\vec{\mathbf{v}}_{t}^{i} = \mathbf{x}_{t}^{i} - \mathbf{x}_{t-1}^{i}$ be the motion vectors of object *i* in two consecutive frames *t*-1 and *t*, where $\mathbf{x}_t^{i}$ is the position of object *i* at frame *t*. The change in object motion velocities between two consecutive frames is usually very small. So, a particle which has smaller changes in motion velocity is given a larger weight. Then, the likelihood function is defined as follows:

$$p(\mathbf{M}_t \mid \mathbf{z}_t) \propto \exp(-\Theta^{v}_{t,t-1})\exp(-\left\|\vec{\mathbf{v}}^{i}_{t} - \vec{\mathbf{v}}^{i}_{t-1}\right\|_2) \tag{30}$$

where $\Theta^{v}_{t,t-1}$ is the angle between $\vec{\mathbf{v}}^{i}_{t-1}$ and $\vec{\mathbf{v}}^{i}_{t}$.

6.4. Extension to more than two object tracking

For $\Upsilon\,(\Upsilon \geq 3)$ objects, the observation model for each object *r* is represented as $p(\mathbf{M}^r_t \mid \mathbf{z}^1_t, \mathbf{z}^2_t, ..., \mathbf{z}^r_t, ..., \mathbf{z}^{\Upsilon}_t)$. It is estimated using the right-hand side of (23). The key problem is to determine the matrix $[\vartheta^k_i]^{k=1,...,K}_{i=1,..,\mu}$ for object *r*. In the implicit way, the matrix $[\vartheta^k_i]^{k=1,...,K}_{i=1,..,\mu}$ is estimated simply using the values of the recovered feature trivial template coefficients $[e^k_i]^{k=1,...,K}_{i=1,..,\mu}$, as in the case of two objects. In the explicit way, for each pair of objects which includes object *r*, the information in the trivial templates is used to determine their occlusion relations, and then the matrix $[\vartheta^k_i]^{k=1,...,K}_{i=1,..,\mu}$ for object *r* is updated if object *r* is occluded. After all such pairs of objects are considered, the matrix $[\vartheta^k_i]^{k=1,...,K}_{i=1,..,\mu}$ is determined. Correspondingly, the observation likelihood $p(\mathbf{M}_t \mid \mathbf{z}^{\mathrm{J}}_t)$ becomes:

$$p(\mathbf{M}_t \mid \mathbf{z}^{\mathrm{J}}_t) = \prod_{r=1}^{\Upsilon} p(\mathbf{M}^r_t \mid \mathbf{z}^1_t, ..., \mathbf{z}^r_t, ..., \mathbf{z}^{\Upsilon}_t) \tag{31}$$

where $\mathbf{z}^{\mathrm{J}}_t$ is the joint state of the Υ objects. The formula for estimating the optimal state of object *r* at the next iteration becomes:

$$\hat{\mathbf{z}}^r_{t,s+1} = \arg\max_{\mathbf{z}^r_t} \prod_{r=1}^{\Upsilon} p(\mathbf{M}^r_t \mid \hat{\mathbf{z}}^1_{t,s+1}, .., \hat{\mathbf{z}}^{r-1}_{t,s+1}, \mathbf{z}^r_{t,s}, ..., \mathbf{z}^{\Upsilon}_{t,s})\,. \tag{32}$$

6.5. Combination with single object tracking

When there are no occlusions in the previous frame, the extent of any occlusion in the current frame is not large and the single object tracking algorithm is robust to track the objects. So, in order to reduce the runtime, in the current frame the multi-object tracking algorithm is only used to track the objects which occluded each other in the previous frame.

In the instances in which objects are relatively dense or easily confused with one another, even if the objects are not overlapped, their trackers may be absorbed onto the same object. We heuristically handle these instances. When there is no occlusion between two objects A and B at the previous frame and suddenly there is severe occlusion between them at the current frame, the appearance similarity between objects A and B at the previous frame is checked. If the similarity is large, then it is determined that the two trackers are absorbed onto one object. For object A or B, the particle whose corresponding observation is second to most similar to the appearance model of the object is found. If the similarity between the second to most similar observation of object A and the appearance model of object A is larger than the similarity between the second to most similar observation of object B and the appearance model of object B, then the tracking result of object A in the current frame is changed to the second to most similar observation of object A, otherwise the tracking result of object B is changed to the second-most similar observation of object B.

7. Experiments

We tested our algorithms on various publicly available sequences. Comparisons with several state-of-the-art algorithms were made. Both qualitative analysis and quantitative evaluations were presented to show the effectiveness of our algorithms.

The parameters were set empirically according to their properties or by referring to their settings in the previous work. To maintain the balance between the reconstruction and the sparseness, the regularization parameter λ in (9) was set to 0.01. To maintain the balance between the accuracy of the coefficients and the number of APG iterations, the step size α in Table 1 was set to 0.5. To maintain the balance between the accuracy and the speed of tracking, the number of the object appearance templates in the template set was set to 60, and the number of particles was set to 420. The threshold ϵ in the weight constraint was set to 0.1 in order that the coefficients of most of the templates in the template set can be directly forced to 0. The forgetting factor δ for template updating was set to 0.95 which is a value less than but close to 1. The variances of the affine parameters were set to 5^2, 5^2, 0, 0, 0, and 0 which were also used by the competing algorithms in [18] and [25] to ensure fair follow-up comparison. The threshold for determining whether the tracking result was used to update the template set was set to 70%, to ensure that when most parts of an object were not badly affected by noise or occlusion, the tracking result was used to update the template set. The threshold used to determine whether a pixel is occluded was set to a small value, 0.05. The threshold used to determine whether an object appearance template is replaced with the tracking result was set to 0.3. The scaling factor η in (22) was set to 30. The scaling factor ζ in (23) was set to 6000. For color image sequences, we used the hue, saturation, intensity, the gradient-based edge template, and the texture feature obtained by Gabor filtering. For gray image sequences, only the intensity, the edge template, and the texture feature were adopted. The number of possible Gabor features is very high [47]. To increase the efficiency of the tracking algorithm, one Gabor filter was chosen for one sequence. Given a Gabor filter with fixed parameter values, its feature vector has a dimension equal to the number of pixels in a normalized image patch, after appropriately dealing with edge pixels. A frequency value was selected empirically from {0, 2, 4, 8, 16, 32}, and an orientation value was selected empirically from {0, π/3, π/6, π/2, 3π/4}. As in [5], the tracker was initialized manually in that the first object appearance template was manually selected from the first frame. Four object appearance templates were constructed by perturbing a few pixels in four possible directions at the corner points of the first template at the first frame. In the tracking after the first frame, the tracking results were added to the template set if the size of the template set was less than a predefined value, otherwise the tracking results were used to update the template set.

In the following, the experimental results for single object tracking are shown first, and then the results for multi-object tracking with occlusion handling are shown.

7.1. Single object tracking

We compared our algorithm first with the single feature sparse representation tracker, and then with several state-of-the-art trackers.

7.1.1. Comparison with single feature sparse representation

In order to illustrate the strength of combining multiple cues, we compared our algorithm with the typical L_1-regularized sparse template-based tracker [5] which uses the single gray cue to construct object templates. Five challenging sequences were used for this comparison. For the first three videos, the frequency and orientation of the Gabor filter were set to 2 and $\pi/2$, respectively. For the last two videos, they were set to 4 and $3\pi/4$. The tracking results of our algorithm are shown using red bounding boxes, and the results of the L_1-regularized algorithm are shown using green bounding boxes.

In the first sequence, a woman is occluded by two men walking across her. The appearances of two men are very similar to the woman. This makes it harder to track the woman. The tracking results are shown in Fig. 2. The frame numbers are 1, 12, 59, 79, and 85, respectively. It is seen that the L_1-regularized algorithm fails to track the woman when there is severe occlusion, however the track is recovered when the background disturbance is not strong. Our algorithm obtains accurate tracking results throughout the sequence.

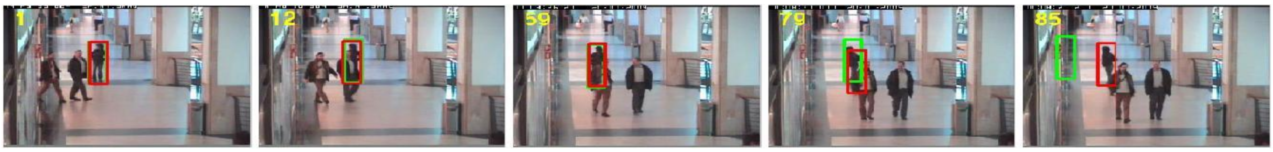


Fig. 2. The results of tracking a woman who is occluded by two men walking across her.

In the second sequence, a woman's face is occluded by a book in various frames. The results of tracking the face are shown in Fig. 3. The frame numbers are 23, 124, 253, 302, and 333, respectively. It is seen that the L_1-regularized algorithm loses track when the occlusion is severe, and the track is recovered when the occlusion is reduced. Our algorithm successfully tracks the object throughout all the frames.

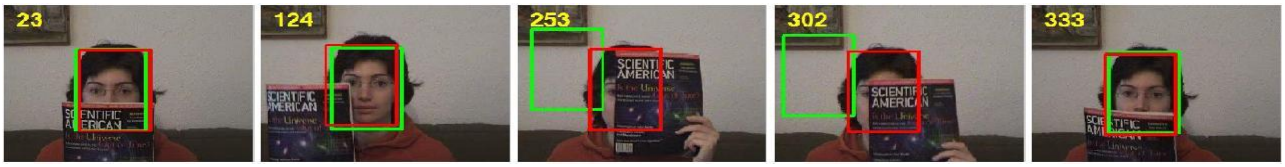


Fig. 3. The results of tracking a face occluded by a book.

In the third sequence, a car moves from the left to the middle of the scene. Its pose and scale change markedly when it moves nearer to the camera and then turns around. The results of tracking the car are shown in Fig. 4. The frame numbers are 20, 80, 173, 196, and 240, respectively. It is seen that the L_1-regularized tracker gradually drifts away and eventually fails to track the car. Our algorithm successfully tracks the car throughout the video.

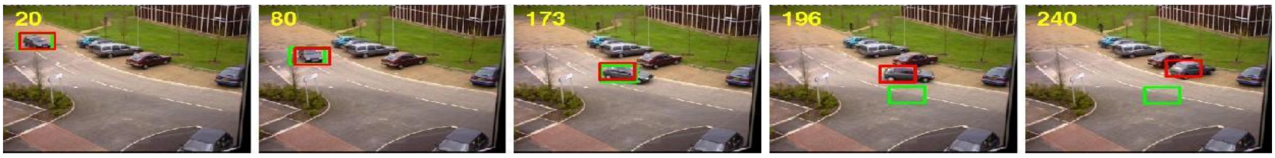


Fig. 4. The results of tracking a car which undergoes large changes in scale and appearance.

In the fourth sequence, a boat moves in a lake. Near to the boat in the image plane, there are tree stumps and houses which have colors similar to the boat. The results of tracking the boat are shown in Fig. 5. The frame numbers are 50, 158, 259, 310, and 384, respectively. It is seen that the L_1-regularized tracker is distracted by the background and drifts off the track in the second half of the video. The reason for this is that the L_1-regularized algorithm only uses the pixel gray levels, and the gray levels of the boat are similar to those in the background. Our algorithm loses track in some frames, but the track is then recovered. Our algorithm, which fuses multiple cues, obtains much more accurate results than the L_1-regularized algorithm.

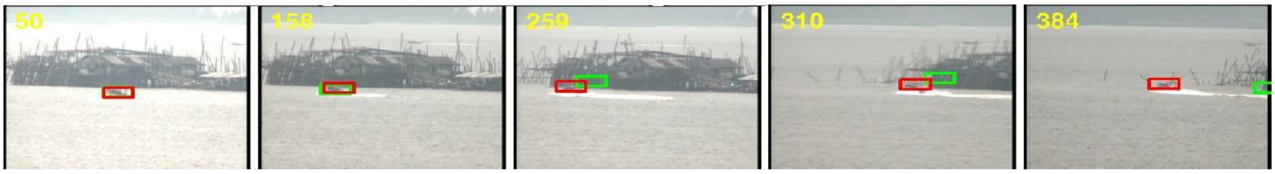


Fig. 5. The results of tracking a boat in a lake.

In the fifth sequence, a car moves against a noisy background. As it is a dark night, the color of the car is similar to the background. The results of tracking the car are shown in Fig. 6. The frame numbers are 20, 80, 171, 211, and 244, respectively. It is seen that both our algorithm and the L_1-regularized algorithm track the car accurately. In this gray sequence, the gray levels alone supply sufficient information for tracking. The shape feature and the texture feature are not discriminative. However, the fusion from the shape and texture cues does not degrade the tracking performance, because they are given low weights.

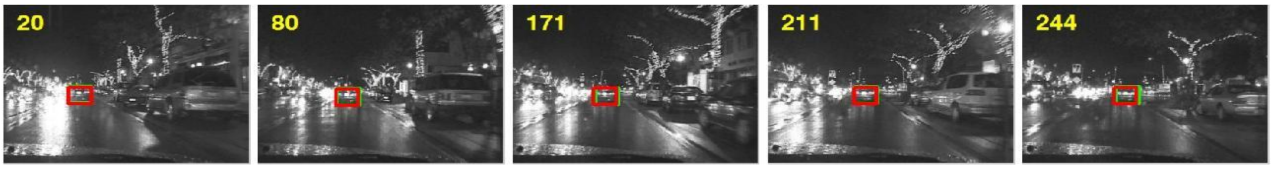


Fig. 6. The results of tracking a car running in a dark night.

From the above tracking results, it is concluded that our algorithm is more robust than the L_1-regularized algorithm. In most cases, the multi-feature joint sparse representation-based tracking algorithm outperforms the sparse representation-based tracking algorithm [5] that only uses the pixel gray levels.

7.1.2. Comparison with state-of-the-art algorithms

Another five challenging sequences were used to compare our algorithm with the following five state-of-the-art algorithms:

- the L_1-regularized sparse template-based tracker (LRST) [5] which is a baseline for our algorithm
- the visual decomposition tracker (VDT) [18] which uses the sparse PCA to fuse multiple cues
- the incremental subspace tracker (IST) [4] which is a typical generative appearance model-based tracker
- the multiple instance learning (MIL) tracker [25] which is a typical discriminative model-based tracker
- the online Adaboost tracker (OBT) [15] which is also a typical discriminative model-based tracker.

The competing algorithms use similar configurations to our algorithm. For example, the same number of particles is used for the particle filter-based competing algorithms (LRST, VDT, IST, and OBT). For the first sequence, the frequency and orientation of the Gabor filter were set to 4 and $3\pi/4$, respectively. For the second sequence, they were set to 2 and $\pi/2$. For the third sequence, they were set to 8 and $\pi/6$. For the fourth sequence, they were set to 2 and $\pi/2$. For the fifth sequence, they were set to 2 and $\pi/6$. In the following, the results of our algorithm are shown using red bounding boxes. The results of the MIL tracker are shown using light blue bounding boxes. The results of the L_1-regularized tracker are shown using green bounding boxes. The results of the visual decomposition tracker are shown using yellow bounding boxes. The results of the online Adaboost tracker are shown using pink bounding boxes. The results of the incremental subspace-based tracker are shown using blue bounding boxes.

In the first sequence, a woman walks from the right to the left, while partially occluded by cars. Some objects in the background have appearances similar to the woman. The results of tracking the woman are shown in Fig. 7. The frame numbers are 4, 41, 141, 206, and 309, respectively. Both our algorithm and the visual decomposition tracker successfully and accurately track the woman through all the frames. The MIL tracker keeps the track of the woman, but the results are not accurate. The other algorithms fail to track the woman.

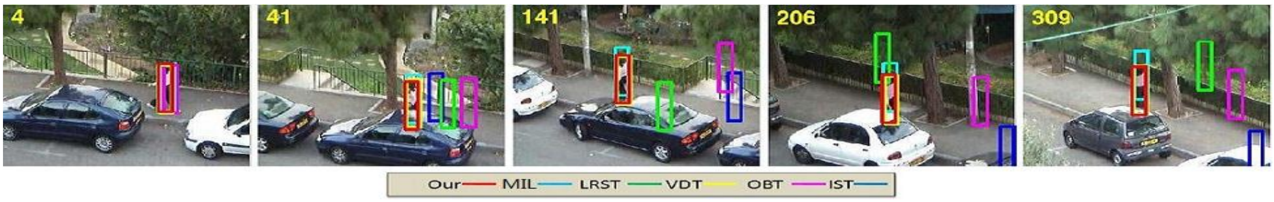


Fig. 7. The results of tracking a woman who is partially occluded by cars.

In the second sequence, a girl changes her head pose by turning her head from full face to look away from the camera, undergoing large changes in appearance. In the middle of the video, the girl's face is severely or even almost completely occluded by a man's head. The results of tracking the girl's head are shown in Fig. 8. The frame numbers are 56, 112, 230, 327, and 438, respectively. Our algorithm, the MIL tracker, and the visual decomposition tracker successfully track the girl's head through large appearance changes and occlusions. The incremental subspace tracker keeps track at the beginning, but gradually loses the track when the changes in appearance become large. The other algorithms cannot accurately track the girl's head.

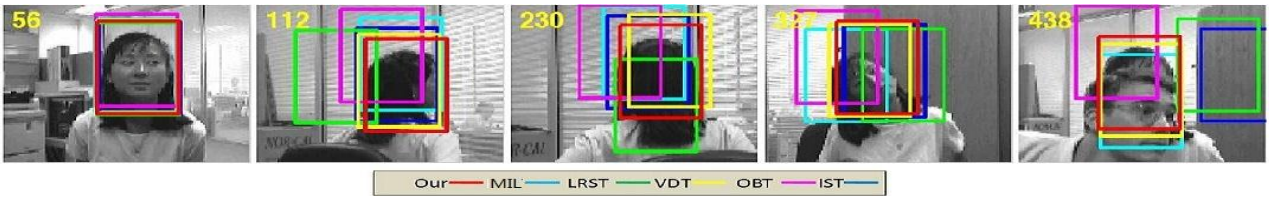


Fig. 8. The results of tracking a girl's head which undergoes large changes in appearance and severe occlusions.

In the third sequence, a pedestrian with a small apparent size walks from the right to the left. He is initially seen from the side and then from the back, and thus undergoes gradual but eventually large changes in appearance. The other pedestrians, who have appearances similar to him, occlude him or are occluded by him from time to time. The results of tracking this pedestrian are shown in Fig. 9. The frame numbers are 17, 88, 305, 373, and 455, respectively. It is seen that only the L_1-regularized tracker and the online Adaboost tracker fail to track the pedestrian. All the other four algorithms keep the track of the pedestrian. However, in some frames the results of the MIL tracker and the incremental subspace tracker are not accurate.

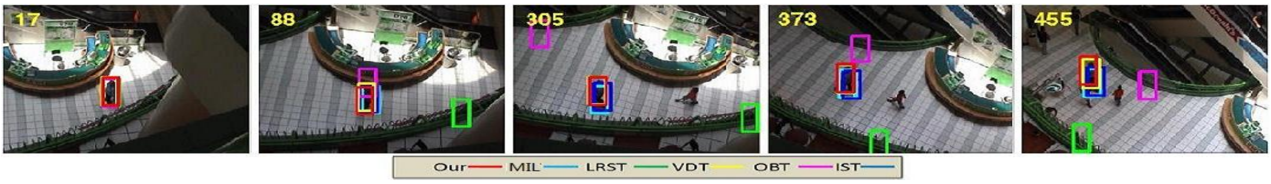


Fig. 9. The results of tracking a pedestrian with a small apparent size and gradual pose changes.

In the fourth sequence, a pedestrian moves in a noisy and cluttered outdoor environment. There are background elements, such as cars and trees, which have colors similar to the pedestrian. The results are shown in Fig. 10. The frame numbers are 6, 28, 52, 78, and 93, respectively. It is seen that the incremental subspace tracker and the visual decomposition tracker fail to track the pedestrian even at the beginning. The MIL tracker, the L_1-regularized tracker, and the online Adaboost tracker keep the track in more than half of the sequence, but finally transfer the tracking to a car in the background. Our algorithm successfully tracks the pedestrian through all the frames.

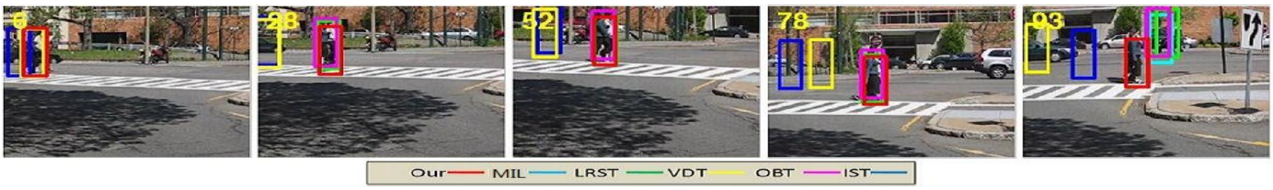


Fig. 10. The results of tracking an outdoor pedestrian in a noisy and cluttered environment.

In the fifth sequence, a deer runs in a noisy background. Other deer, which are very similar in appearance, pass by. The results of tracking the head of the deer are shown in Fig. 11. The frame numbers are 15, 29, 35, 53, and 66, respectively. It is seen that the visual decomposition tracker and the incremental subspace tracker fail to track the head at the beginning of the sequence. The MIL tracker, the L_1-regularized tracker, the online Adaboost tracker, and our algorithm drift off track in some frames due to background disturbances, but the track is recovered when the background disturbances cease.

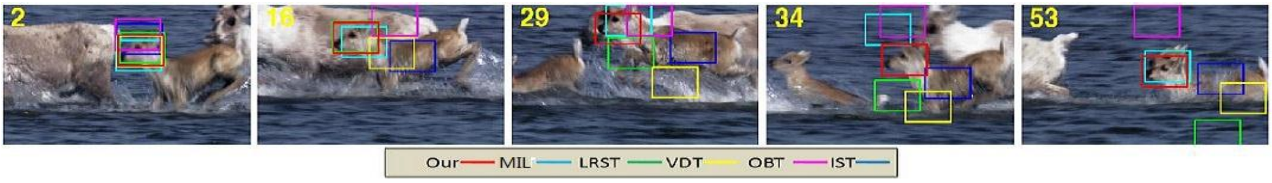


Fig. 11. The results of tracking the head of a deer against a cluttered background.

7.1.3. Quantitative evaluations

Quantitative evaluations were made for the six algorithms using the above five videos. In each frame, four benchmark points were manually labeled corresponding to the four corners of a bounding box which contains the tracked object. We calculated the root mean square error (RMSE) of the four corner points in the bounding box between the tracking results and the ground truths. Figs. 12, 13, 14, 15, and 16 show the RMSE curves of our algorithm and the five competing algorithms for the five sequences respectively, where the x-coordinate is the frame number and the y-coordinate is the RMSE in each frame. Table 3 lists the mean of the RMSEs for all the frames in each of the five sequences. It is seen that the performance of our algorithm is comparable with that of the visual decomposition tracker and that it outperforms the other four algorithms.

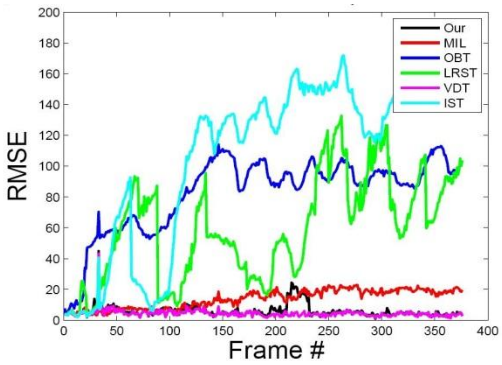


Fig. 12. The RMSE curves of the six algorithms for sequence 1.

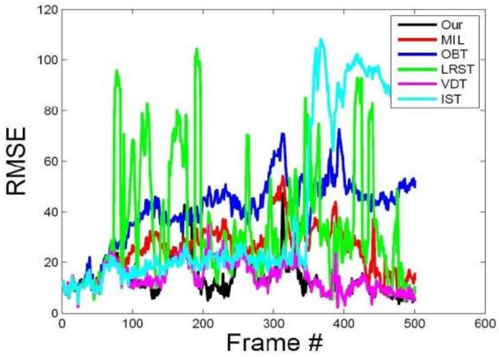


Fig. 13. The RMSE curves of the six algorithms for sequence 2.

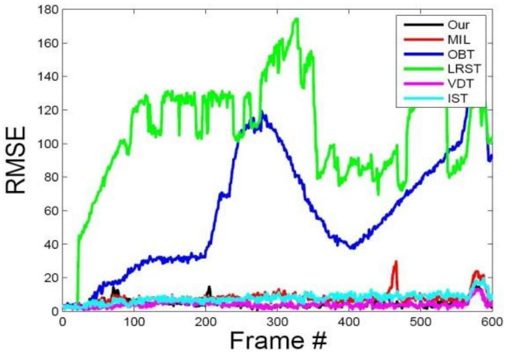


Fig. 14. The RMSE curves of the six algorithms for sequence 3.

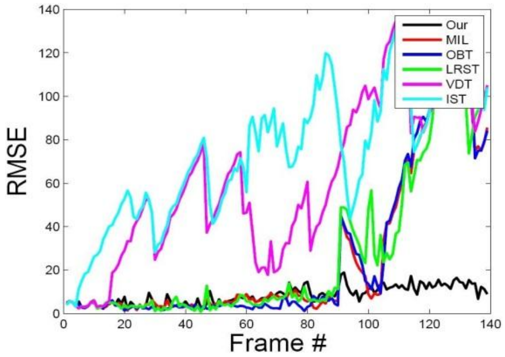


Fig. 15. The RMSE curves of the six algorithms for sequence 4.

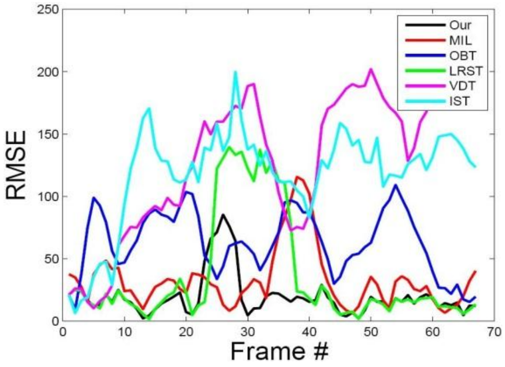


Fig. 16. The RMSE curves of the six algorithms for sequence 5.

Table 3: The mean of the RMSEs for all the frames in each of the five sequences

Trackers / Sequences	MIL-based	Online Adaboost	L_1 regularized	Visual decomposition	Incremental subspace	Our
Video 1	13.7	82.7	59.4	4.3	100.7	5.8
Video 2	24.9	40.3	34.3	14.3	40.8	13.9
Video 3	7.8	59.8	106.1	4.2	7.6	4.6
Video 4	27	26.6	27.1	61.6	73.2	8.3
Video *5*	32.9	62.7	36.1	129.5	117.5	19.5

From the above tracking results, we draw the following conclusion. The algorithms fusing multiple cues, i.e. the visual decomposition tracker and our algorithm, are more accurate than the algorithms which only use a single cue. Our algorithm can effectively fuse multiple cues in a sparse representation for visual tracking. As the MIL tracker deals effectively with the drift problem by using multiple instances, it outperforms the other three algorithms which only use the single cue.

7.2. Multi-object tracking

In order to validate the effectiveness of our algorithm for tracking multiple objects, five different videos, two containing faces, two containing pedestrians, and one containing vehicles, were used to compare our algorithm with three state-of-art algorithms. For the first four videos, the frequency and orientation of the Gabor filter were set to 2 and $\pi/2$, respectively. For the last video, they were set to 16 and $\pi/3$.

The first example corresponds to the second video in Section 7.1.2. In the middle of the sequence, the man's face gradually disappears from the scene and then appears in the scene again. The incremental subspace-based algorithm in [4] is a typical single object tracking algorithm, widely used for comparisons. We extended the incremental subspace-based algorithm to track multi-objects by tracking multi-objects independently, without any occlusion handling. We compared our multi-object tracking algorithm with the incremental subspace algorithm [4] to show the necessity of occlusion handling. The results of tracking the two faces are shown in Fig. 17, where the first row shows the results of our algorithm and the second row shows the results of the incremental subspace algorithm. The frame numbers are 2, 15, 22, 25, and 28, respectively. It is seen that the incremental subspace-based tracker does not effectively handle the occlusions produced by the disappearance of the face of the man. The reason for this is that, for the incremental subspace algorithm, the occluded parts of an object are also used for matching, but the invisible parts cannot be reconstructed by the subspace based-matching. On the contrary, our observation model adopts the visible parts for matching, while ignoring the occluded parts. This is the key of success of our algorithm in tracking the two faces. So, occlusion handling is necessary for tracking multi-objects during occlusions.

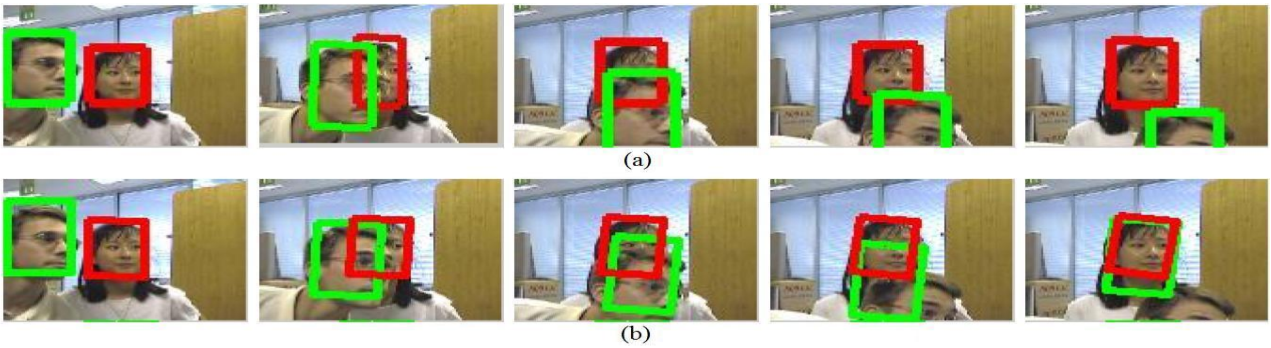


Fig. 17. The results of tracking the faces of a girl and a man: (a) Our algorithm; (b) The incremental subspace algorithm

In the second example, the faces of two men occlude each other and undergo changes in appearance. We compared our multi-object tracking algorithm with the following two state-of-the-art multi-object tracking algorithms with occlusion handling:

- Yang's algorithm [36], which uses a game-theoretic analysis to implicitly handle occlusions
- Zhang's algorithm [44], which use species competition to decompose the tracking of multi-objects with

occlusions into the tracking of a set of un-occluded objects.

The results of tracking the two faces are shown in Fig. 18, where the first row shows the results of our algorithm, the second row shows the results of Zhang's algorithm, and the third row shows the results of Yang's algorithm. The frame numbers are 8, 142, 172, 204, and 237, respectively. It is seen that Zhang's algorithm drifts off the track for the occluded object to some extent. Yang's algorithm fails to track the occluded object during and after the occlusion. Our algorithm effectively tracks both the faces. The changes in the appearances of the faces are effectively handled using our template updating mechanism.

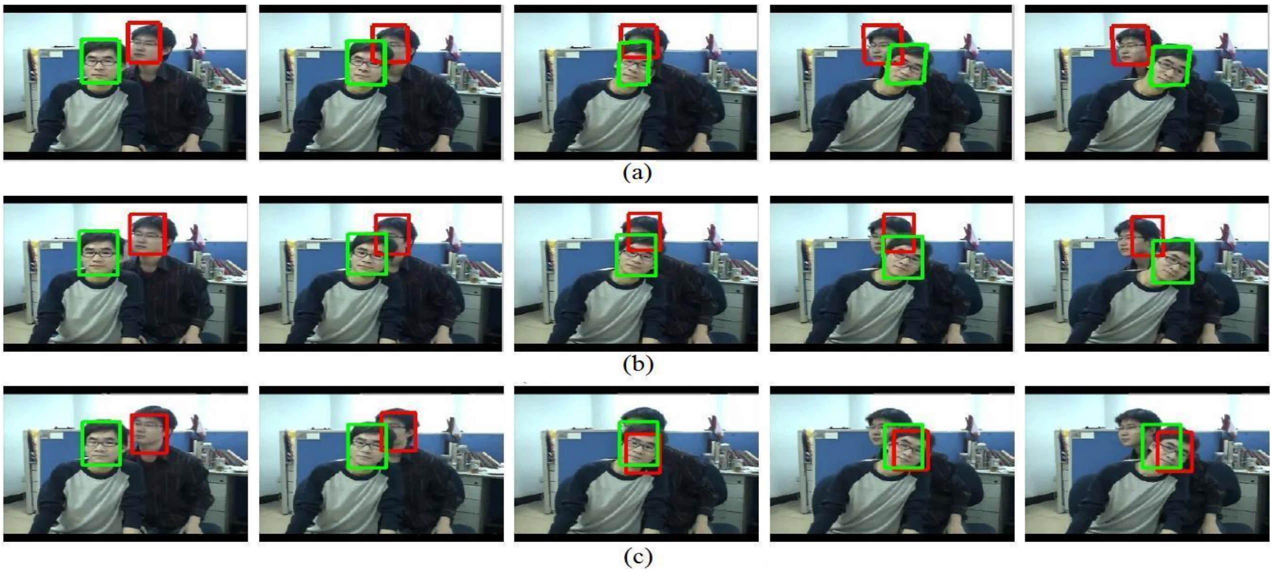
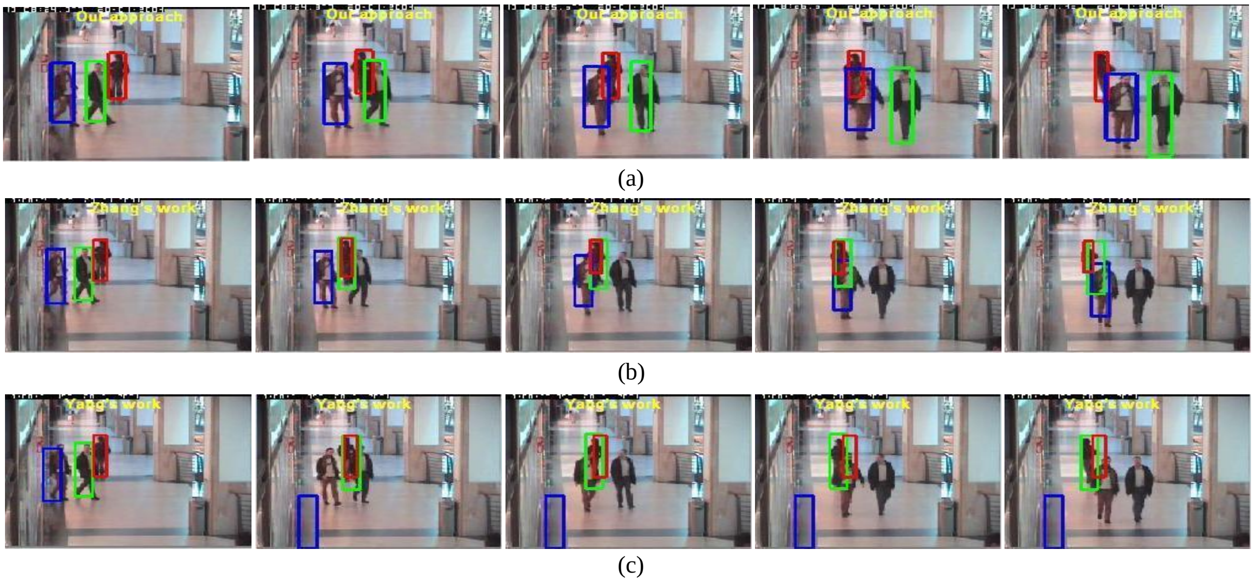

(a)
(b)
(c)

Fig. 18. The results of tracking two men's faces which occlude each other and undergo changes in appearance: (a) Our algorithm; (b) Zhang's algorithm; (c) Yang's algorithm.



(a)
(b)
(c)

Fig. 19. The results for tracking three pedestrians under occlusions: (a) Our algorithm; (b) Zhang's algorithm; (c) Yang's algorithm.

The video in the third example is from the open PETS2004 database. In this video, there are mutual occlusions between three pedestrians. Two men turn around to face the camera and their appearances change. A woman turns around to face away from the camera. Fig. 19 illustrates results of our algorithm, Zhang's algorithm, and Yang's algorithm for tracking all three pedestrians. The frame numbers are 5, 25, 45, 65, and 88, respectively. It is seen that our method successfully tracks all the pedestrians. The changes in the appearances of the pedestrians are successfully tackled through our template updating strategy. Occlusions are effectively handled. However, both the competing algorithms transfer the tracking from the man marked by a green window to the women marked by a red window. Both the algorithms lose the track of the man marked by a blue window.

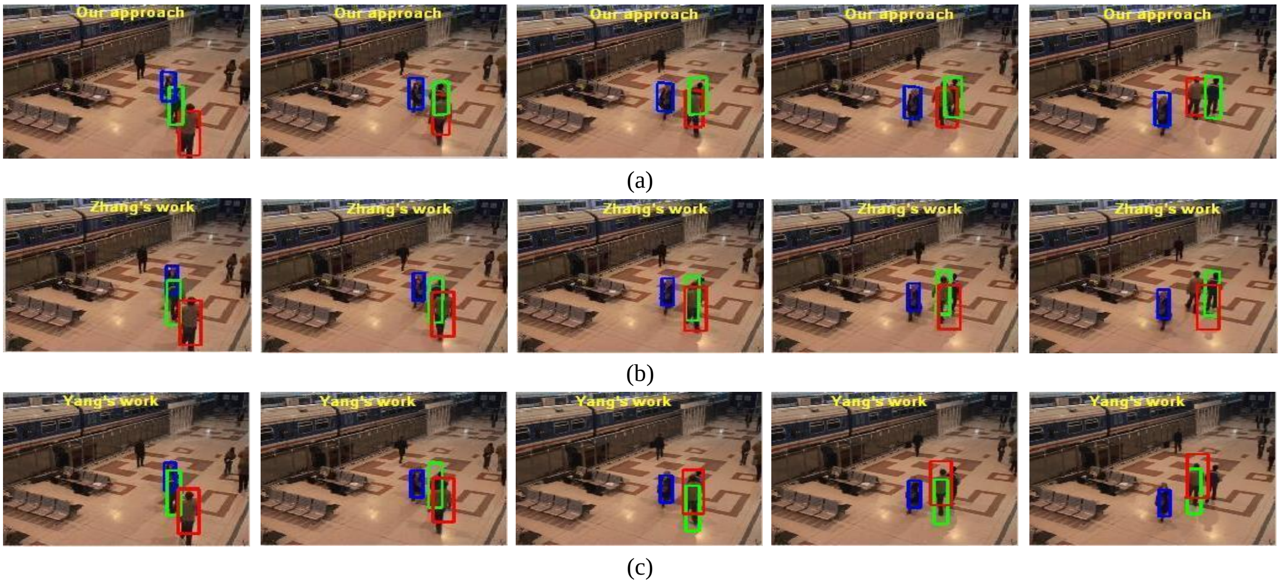
A quantitative evaluation of the tracking accuracy was conducted to further demonstrate better performance of our multi-object tracking algorithm. The evaluation is comprised of the following two aspects:

- the number of successfully tracked frames (tracking is considered to be successful if the estimated bounding box’s center is in the ground truth box)
- the RMSE (root mean square error) between the estimated position and the ground truth.

Table 4 shows the results of quantitative comparisons between our algorithm, Zhang’s algorithm, and Yang’s algorithm for tracking the three pedestrians (persons A, B and C) in the third sequence. It is seen that the mean tracking error of Yang’s algorithm for person A is very large. This is because Yang’s algorithm quickly loses the track of this person. It is apparent that the results of our algorithm are more accurate than the results of Zhang’s algorithm and Yang’s algorithm.

Table 4: Quantitative comparisons between our algorithm, Zhang’s algorithm, and Yang’s algorithm for the third sequence

Algorithms		Our algorithm	Zhang’s algorithm	Yang’s algorithm
Successfully tracked frames	Person A (Blue)	89/90	75/90	5/90
	Person B (Green)	90/90	21/90	18/90
	Person C (Red)	90/90	86/90	40/90
RMSE	Person A (Blue)	6.6566	9.2590	86.2946
	Person B (Green)	5.7107	50.0882	30.0492
	Person C (Red)	2.9384	7.0305	11.5926



(a)

(b)

(c)

Fig. 20. The tracking results for the fourth sequence: (a) Our algorithm; (b) Zhang’s algorithm; (c) Yang’s algorithm.

In the fourth example, three pedestrians (persons A, B, and C) occluding each other are tracked. The video is from the PETS dataset in 2006. Fig. 20 illustrates the tracking results of our algorithm, Zhang’s algorithm, and Yang’s algorithm, where person A is tracked using a blue window, person B is tracked using a green window, and person C is tracked using a red window. The frame numbers are 10, 26, 38, 50, and 63, respectively. It is seen that both Zhang's algorithm and Yang's algorithm fail to accurately track persons B and C during occlusions. The competing algorithms do not deal effectively with the occlusions between persons B and C. Our algorithm successfully tracks all the three pedestrians by effectively handling the occlusions between the three pedestrians. Table 5 shows quantitative results for our algorithm, Zhang's algorithm, and Yang's algorithm for the fourth sequence. It is apparent that the results of our algorithm are the most accurate.

Table 5: Quantitative results of our algorithm, Zhang's algorithm, and Yang's algorithm for the fourth sequence

Algorithms		Our algorithm	Zhang's algorithm	Yang's algorithm
Successfully tracked frames	Person A	89/89	89/89	89/89
	Person B	89/89	76/89	57/89
	Person C	89/89	58/89	89/89
RMSE	Person A	3.1453	3.2985	5.6851
	Person B	4.3988	6.9392	17.8217
	Person C	4.2013	14.4789	8.8054

For the fourth multi-object tracking example, two more pedestrians were tracked, besides the three pedestrians who were tracked in the Fig. 20. The explicit occlusion handling was used. The results are shown in Fig. 21. It is seen that the five pedestrians are correctly tracked.

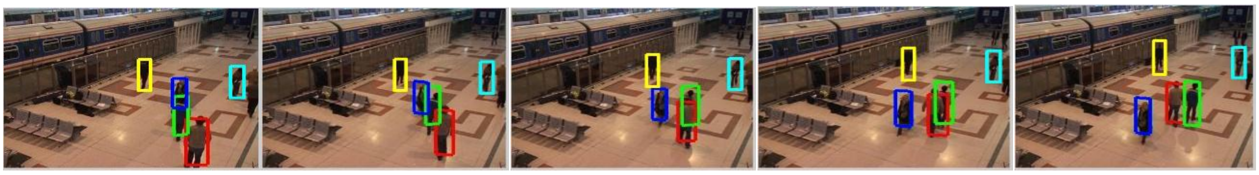


Fig. 21. The results of our algorithm for tracking five pedestrians.

In the fifth example, multi-vehicles were tracked in a crowded traffic intersection scene. Fig. 22 shows the results of our algorithm for tracking eight vehicles in the scene, where two pairs of similar vehicles, which are easily confused with each other, move close to each other. The frame numbers are 3475, 3500, 3525, 3550, and 3600, respectively. It is seen that all the eight vehicles are successfully tracked.

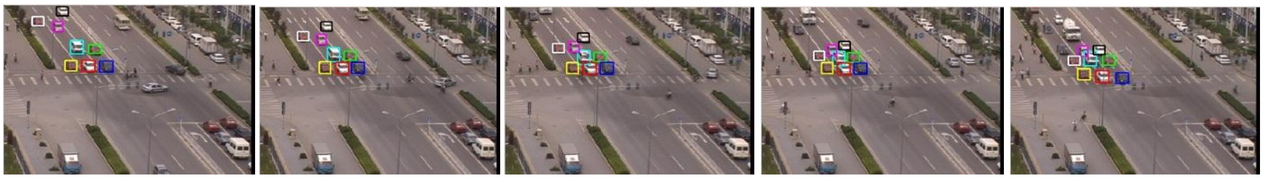


Fig. 22. The results of our algorithm for tracking eight vehicles in a dense traffic intersection scene.

7.3. Computational complexity and runtime

The computational complexity for our single object tracking algorithm is $O(nK\Psi N)$, where Ψ is the number of iterations required to obtain the multi-feature joint sparse representation for an observation. The computational complexity for our multi-object tracking algorithm is $O(nK\Psi N\Upsilon)$.

As measured on an Intel Core i7 3770(3.4GHz/L3) computer, the runtime of our single object tracking algorithm for each frame in all the examples is less than 0.8 seconds and the runtime of our multi-object tracking algorithm is less than 5 seconds per frame in the case of occlusions. As the number of objects increases, the runtime does not quickly increase. It is shown that the computational complexity of the proposed method is not very high. Currently, our algorithm is unable to work in real-time. We expect further research on this problem.

Although multi-feature fusion is applied by our algorithm, our single object tracking algorithm is faster than the L_1-regularized sparse template-based tracker (LRST) because the APG algorithm is used in our algorithm. The speed of our single object tracking algorithm is comparable to the speed of the VDT [18], but slower than the speeds of the IST, the MIL tracker, and the OBT. The speed of our multi-object tracking algorithm is comparable to the speed of Yang's algorithm, but slower than the speed of Zhang's algorithm.

8. Conclusion

In this paper, we have proposed a new tracking algorithm based on a multi-feature joint sparse representation. In our algorithm, multiple cues have been successfully fused together in the sparse representation framework. The variance ratio has been introduced to adapt the weights of different features. The template set has been updated adaptively using the tracking results. By using a sparse weight constraint, a large number of templates can be kept in the template set. We have further proposed an algorithm for tracking multi-objects under occlusions using a multi-feature sparse reconstruction. The matching between the observations and the templates is based on the visible parts of the occluded objects. The experimental results have validated the effectiveness of both our single object tracking algorithm and our multi-object tracking algorithm.

The main limitations of our method are that many parameters are set empirically and the semantic corrections between the different features are not modeled.

References

1. L. Cazzanti, M. Gupta, and A. Koppal, "Generative Models for Similarity-Based Classification," *Pattern Recognition*, vol. 41, no. 7, pp. 2289-2297, July 2008.
2. M. Kim, "Discriminative Semi-supervised Learning of Dynamical Systems for Motion Estimation," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2325-2333, Oct. 2011.
3. B.S. Manjunath, J.R. Ohm, V. Vasudevan, and A. Yamada, "Color and Texture Descriptors," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703-715, June 2001.
4. D.A. Ross, J. Lim, R. Lin, and M. Yang, "Incremental Learning for Robust Visual Tracking," *International Journal of Computer Vision*, vol. 77, no. 2, pp. 125-141, May 2008.
5. X. Mei and H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 234-778, Nov. 2011.
6. J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210-227, Feb. 2009.
7. A. Jepson, D. Fleet, and T. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296-1311, Oct. 2003.
8. S. Zhou, R. Chellappa, and B. Moghaddam, "Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters," *IEEE Trans. on Image Processing*, vol. 13, no. 11, pp. 1491-1506, Nov. 2004.
9. T. Yu and Y. Wu, "Differential Tracking Based on Spatial Appearance Model," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 720-727, 2006.
10. H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive Object Tracking Based on an Effective Appearance Filter," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1661-1667, Sep. 2007.
11. 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, 2009.
12. 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*, pp. 728-735, 2006.
13. R. Collins, Y. Liu, and M. Leordeanu, "Online Selection of Discriminative Tracking Features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631-1643, Oct. 2005.
14. S. Avidan, "Ensemble Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261-271, Feb. 2007.

15. H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-line Boosting," *in Proc. of British Machine Vision Conference*, pp. 47-56, 2006.
16. H. Grabner, C. Leistner, and H. Bischof, "Semi-Supervised On-line Boosting for Robust Tracking," *in Proc. of European Conference on Computer Vision*, pp. 1-8, 2008.
17. A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line Random Forests," *in Proc. of IEEE International Conference on Computer Vision Workshops*, pp. 1393-1400, 2010.
18. J. Kwon and K.M. Lee, "Visual Tracking Decomposition," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1269-1276, 2010.
19. S.J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "A Method for Large-Scale 1-Regularized Least Squares," *IEEE Journal on Selected Topics in Signal Processing*, vol. 1, pp. 606-617, 2007.
20. X.T. Yuan and S. Yan, "Visual Classification with Multi-Task Joint Sparse Representation," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3493-3500, June 2010.
21. J. Zhang, "A Probabilistic Framework for Multi-Task Learning," Technical Report, 2006.
22. K. Yu, T. Zhang, and Y. Gong, "Nonlinear Learning Using Local Coordinate Coding," *in Proc. of Annual Conference on Neural Information Processing Systems*, pp. 1-9, 2009.
23. P. Tseng, "On Accelerated Proximal Gradient Methods for Convex-Concave Optimization," *SIAM Journal of Optimization*, May 2008. http://www.math.washington.edu/~tseng/papers/apgm.pdf
24. M. Isard and A. Blake, "Condensation: Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5-28, 1998.
25. 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.
26. D.L. Donoho and X. Huo, "Uncertainty Principles and Ideal Atomic Decomposition," *IEEE Trans. on Information Theory*, vol. 47, no. 7, pp. 2845-2862, Nov. 2001.
27. B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. Kulikowski, "Robust and Fast Collaborative Tracking with Two Stage Sparse Optimization," *in Proc. of European Conference on Computer Vision*, pp. 624-637, 2010.
28. F. Chen, Q. Wang, S. Wang, W. Zhang, and W. Xu, "Object Tracking via Appearance Modeling and Sparse Representation," *Image and Vision Computing*, vol. 29, no. 11, pp. 787-796, 2011.
29. Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Online Discriminative Object Tracking with Local Sparse Representation," *in Proc. of IEEE Workshop on Applications of Computer Vision*, pp. 425-432, 2012.
30. C. Rasmussen and G. Hager, "Probabilistic Data Association Methods for Tracking Complex Visual Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 560-576, June 2001.
31. A. Elgammal and L. Davis, "Probabilistic Framework for Segmenting People under Occlusion," *in Proc. of International Conference on Computer Vision*, vol. 2, pp. 145-152, 2001.
32. Y. Wu, T. Yu, and G. Hua, "Tracking Appearances with Occlusions," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Madison, vol. 1, pp. 789-795, June 2003.
33. E. Sudderth, M. Mandel, W. Freeman, and A. Willsky, "Distributed Occlusion Reasoning for Tracking with Nonparametric Belief Propagation," *in Proc. of Annual Conference on Neural Information Processing Systems*, pp. 1369-1376, 2004.
34. J. MacCormick and A. Blake, "A Probabilistic Exclusion Principle for Tracking Multiple Objects," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 57-71, 2000.
35. H. Nguyen, Q. Ji, and A. Smeulders, "Spatio-Temporal Context for Robust Multitarget Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 52-64, Jan. 2007.
36. M. Yang, T. Yu, and Y. Wu, "Game-Theoretic Multiple Target Tracking," *in Proc. of IEEE International Conference on Computer Vision*, pp. 1-8, 2007.
37. K. Nummiaro, E. Koller-Meier, and L.V. Gool, "An Adaptive Color-Based Particle Filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99-110, Jan. 2003.
38. P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-Based Probabilistic Tracking," *In Proc. of European Conference on Computer Vision*, vol. 1, pp. 661-675, 2002.
39. Y. Li "On Incremental and Robust Subspace Learning," *Pattern Recognition*, vol. 37, no. 7, pp. 1509-1518, 2004.
40. H. Lim, V.I. Morariu, O.I. Camps, and M. Sznaier, "Dynamic Appearance Modeling for Human Tracking," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 751-757, 2006.
41. J. Ho, K. Lee, M. Yang, and D. Kriegman, "Visual Tracking Using Learned Linear Subspaces," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 782-789, 2004.
42. S. Avidan, "Support Vector Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064-1072, Aug. 2004.
43. O. Tuzel, F. Porikli, and P. Meer, "Human Detection via Classification on Riemannian Manifolds," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, June 2007.
44. X. Zhang, W. Hu, W. Qu, and S. Maybank, "Multiple Object Tracking via Species-Based Particle Swarm Optimization," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1590-1602, Nov. 2010.

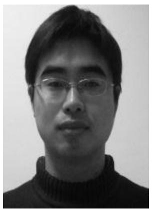
45. X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, “Minimum Error Bounded Efficient ℓ1 Tracker with Occlusion Detection,” *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1257-1264, June 2011.
46. J.B. Tenenbaum, V. de Silva, and J.C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” *Science*, vol. 290, no. 5550, pp. 2319-2323, Dec. 2000.
47. Manjunath and W. Ma, “Texture Features for Browsing and Retrieval of Image Data,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837-842, Aug. 1996.
48. A.A. Butt and R.T. Collins, “Multi-target Tracking by Larangian Relaxation to Min-Cost Network Flow,” *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Oral presentation, June 2013.
49. J. Liu, P. Carr, R.T. Collins, and Y. Liu, “Tracking Sports Players with Context-Conditioned Motion Models,” *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Oral presentation, June 2013.
50. R.T. Collins, “Multitarget Data Association with Higher-Order Motion Models,” *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1744-1751, June 2012.
51. K. Zhang, L. Zhang, and M.-H. Yang, “Real-Time Compressive Tracking,” *in Proc. of European Conference on Computer Vision*, pp. 864-877, 2012.
52. F. Nie, H. Huang, X. Cai, and C. Ding, “Efficient and Robust Feature Selection via Joint $l_{2,1}$-Norms Minimization,” *in Proc. of Advances in Neural Information Processing Systems*, pp. 1813-1821, 2010.
53. J. Liu, S. Ji, and J. Ye, “Multi-Task Feature Learning via Efficient $l_{2,1}$-Norm Minimization,” *in Proc. of Conference on Uncertainty in Artificial Intelligence*, pp. 339-348, 2009.

Acknowledgments

This work is partly supported by the 973 basic research program of China (Grant No. 2014CB349303), the National 863 High-Tech R&D Program of China (Grant No. 2012AA012504), and the Natural Science Foundation of Beijing (Grant No. 4121003).

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, recognition of web objectionable information, and network intrusion detection.

Wei Li received his B.Sc degree in automation from Zhejiang University, China, in 2006. He received the Ph.D. degree at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China, in 2012. Now, he is working in Alibaba Group Company. His research interests include computer vision and pattern recognition.



Xiaoqin Zhang received the B.Sc degree in electronic information science and technology from Central South University, China, in 2005 and PhD degree from the Institute of Automation, Chinese Academy of Sciences, China, in 2010. He is currently a lecture in Wenzhou University, China. His research interests are in visual tracking, motion analysis, and action recognition.

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.