Int J Comput Vis (2011) 91: 303–327
DOI 10.1007/s11263-010-0399-6

# Incremental Tensor Subspace Learning and Its Applications to Foreground Segmentation and Tracking

**Weiming Hu · Xi Li · Xiaoqin Zhang · Xinchu Shi · Stephen Maybank · Zhongfei Zhang**

Received: 28 September 2009 / Accepted: 1 October 2010 / Published online: 15 October 2010

**Abstract** Appearance modeling is very important for background modeling and object tracking. Subspace learning-based algorithms have been used to model the appearances of objects or scenes. Current vector subspace-based algorithms cannot effectively represent spatial correlations between pixel values. Current tensor subspace-based algorithms construct an offline representation of image ensembles, and current online tensor subspace learning algorithms cannot be applied to background modeling and object tracking. In this paper, we propose an online tensor subspace learning algorithm which models appearance changes by incrementally learning a tensor subspace representation through adaptively updating the sample mean and an eigenbasis for each unfolding matrix of the tensor. The proposed incremental tensor subspace learning algorithm is applied to foreground segmentation and object tracking for grayscale and color image sequences. The new background models capture the intrinsic spatiotemporal characteristics of scenes. The new tracking algorithm captures the appearance characteristics of an object during tracking and uses a particle filter to estimate the optimal object state. Experimental evaluations against state-of-the-art algorithms demonstrate the promise and effectiveness of the proposed incremental tensor subspace learning algorithm, and its applications to foreground segmentation and object tracking.

**Keywords** Incremental learning · Tensor subspace · Foreground segmentation · Tracking

## 1 Introduction

Modeling the appearances of objects or scenes plays an important role in computer vision applications such as background modeling, tracking, and behavior analysis. Color histograms (Nummiaroa et al. 2003; Perez et al. 2002) of regions are widely used for appearance modeling due to their robustness to region scaling, rotation, and shape variations. Their limitation is that they ignore the spatial distribution of pixel values. Kernel density estimation-based appearance models (Elgammal et al. 2002; Yang et al. 2005) use spatial weighted kernels to represent the spatial distribution of pixel values. Their limitation is their high computational and memory complexities. GMM (Gaussian mixture model)-based appearance models (Zhou et al. 2004; Wu and Huang 2004) use a mixture of weighted Gaussian

**Electronic supplementary material** The online version of this article (doi:10.1007/s11263-010-0399-6) contains supplementary material, which is available to authorized users.

W. Hu (✉) · X. Li · X. Shi
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
e-mail: wmhu@nlpr.ia.ac.cn

X. Li
e-mail: lixichinanlpr@gmail.com

X. Shi
e-mail: xcshi@nlpr.ia.ac.cn

X. Zhang
College of Mathematics & Information Science, Wenzhou University, Wenzhou 325000, Zhejiang, China
e-mail: xqzhang@wzu.edu.cn

S. Maybank
Department of Computer Science and Information Systems, Birkbeck College, Malet Street, London WC1E 7HX, UK
e-mail: sjmaybank@dcs.bbk.ac.uk

Z. Zhang
State University of New York, Binghamton, NY 13902, USA
e-mail: zhongfei@cs.binghamton.edu

distributions to learn a statistical model for colors. Their limitation is that they deal with each pixel independently and thus the relations between the values of nearby pixels are not effectively characterized. Conditional random field-based appearance models (Wang et al. 2006) use Markov random fields to model the relations between the values of neighboring pixels. Their limitations are that their training is very expensive and the global distributions of pixels are not considered. Online subspace learning-based appearance models (Skocaj and Leonardis 2003; Ross et al. 2008) flatten appearance regions to vectors in order to describe global statistical information about pixel values. Their limitation is that spatial information, which is invariant under certain global appearance variations e.g. lighting changes and robust to image noise, is missing due to the flattening.

Recently, multi-linear subspace analysis has attracted much attention and has been applied to image representation and appearance modeling, etc. Yang et al. (2004) develop a 2-dimensional PCA (principal component analysis) for image representation. An image covariance matrix is constructed. The eigenvectors of this matrix are derived for image feature extraction. Ye et al. (2004a) present a learning method called 2-dimensional linear discriminant analysis in which classification is based on operations on image matrices. Ye (2005) propose an algorithm for low rank approximations of a collection of matrices using an iterative algorithm which reduces the reconstruction error sequentially, and improves the resulting approximation during successive iterations. Ye et al. (2004b) present a new dimension reduction algorithm which constructs the low-order matrix representation of images directly by projecting the images to a vector space that is the product of two lower-dimensional vector spaces. Some pioneering methods use tensors to construct object models. Wang and Ahuja (2005) propose a rank-R tensor approximation which can effectively capture spatiotemporal redundancies in the tensor entries. Yan et al. (2005) propose an algorithm for discriminant analysis with tensor representation. This algorithm is derived from the popular vector-based linear discriminant analysis algorithm. Vasilescu and Terzopoulos (2002, 2003) apply the $N$-mode SVD (singular value decomposition), i.e. multi-linear subspace analysis, to construct a compact representation of facial image ensembles factorized by different faces, expressions, viewpoints, and illuminations. He et al. (2005) present a tensor subspace analysis algorithm, which learns a lower dimensional tensor subspace, to characterize the intrinsic local geometric structure within the tensor space. Wang et al. (2007) give a convergent solution for general tensor-based subspace learning. Sun et al. (2006a, 2006b, 2008) propose three tensor subspace learning methods: DTA (dynamic tensor analysis), STA (streaming tensor analysis), and WTA (window-based tensor analysis), for representing data streams over time.

The above tensor analysis algorithms cannot be applied to background modeling and object tracking directly. We point out the following aspects:

1) Except for the DTA, STA, and WTA algorithms, the above tensor analysis algorithms learn tensor subspaces offline, i.e. when new data arrives, the subspace model is retrained using the new data and the previous data. This results in high memory and time costs, while spatiotemporal redundancies are substantially reduced. In the context of background modeling and object tracking, it is necessary to use the new data to online update the previously learned model. This is because appearance updating for background modeling and object tracking is more effective if the recent frames in a video are weighted more heavily than previous frames.

2) The DTA, STA, and WTA algorithms include incremental tensor subspace learning which adaptively updates the subspaces. However, they cannot be applied to background modeling and object tracking. These three algorithms use column spaces of the three unfolding matrices obtained from the corresponding three modes of the tensor. In fact, the column space of the unfolding matrix on the third mode is of no use for background modeling and object tracking, but the row space of this matrix is useful. Furthermore, DTA and WTA update the covariance matrix formed from the columns of each of the unfolding matrices and then obtain an eigenvector decomposition of the updated covariance matrix, assuming that the mean of the previous unfolding matrix is equal to the mean of the unfolding matrix of new data. As a result, the updating is not accurate if the mean changes. DTA and WTA have the small size problem: the number of new samples is much less than the rank of the covariance matrix. STA applies the SPIRIT (streaming pattern discovery in multiple timeseries) iterative algorithm (Papadimitriou et al. 2005) to the new coming data to approximate DTA without diagonalization. The tensor subspaces learned using STA are less accurate than the tensor subspaces learned using DTA. (More descriptions to DTA, WTA, and STA are given in Sect. 3.3.6.)

In this paper, we develop a new incremental tensor subspace learning algorithm, and apply it to foreground segmentation and object tracking. The main contributions of our work are as follows:

- The proposed algorithm learns online a low dimensional tensor subspace representation of the appearance of an object or a scene by adaptively updating the sample mean and an eigenbasis for each unfolding matrix using tensor decomposition and incremental SVD. Compared with existing vector subspace algorithms for appearance modeling, our algorithm more efficiently captures the intrinsic spatiotemporal characteristics of the appearance of an object and a scene. Furthermore, our method works online, resulting in much lower computational and memory complexities.

- Based on the proposed incremental subspace learning algorithm, two background models, one for grayscale image sequences and the other for color image sequences, are developed to capture the spatiotemporal characteristics of scenes based on a likelihood function which is constructed from the learned tensor subspace model. The background models are used to segment the foreground from the background. The experimental results show that our algorithm obtains more accurate foreground segmentation results than the vector subspace-based algorithm (Li 2004) and the GMM-based algorithm (Stauffer and Grimson 1999).
- We propose a visual object tracking algorithm in which the proposed incremental tensor subspace learning algorithm is used to capture the appearance of an object during tracking (Li et al. 2007). Particle filtering is used to propagate the sample distributions over time. The experimental results show that our algorithm tracks objects more robustly than the vector subspace-based algorithm (Ross et al. 2008) and the Riemannian metric-based algorithm (Porikli et al. 2006).

The remainder of the paper is organized as follows: Sect. 2 discusses the related work. Section 3 describes our incremental tensor subspace learning algorithm. Sections 4 and 5 present our background segmentation algorithm and our object tracking algorithm respectively. Section 6 demonstrates experimental results. The last section summarizes the paper.

## 2 Related Work

In Sect. 1, we reviewed the work closely related to tensor-based appearance modeling in order to strengthen the motivation of this paper. For completeness, in the following, we briefly discuss the developments in foreground segmentation and visual object tracking.

### 2.1 Foreground Segmentation

Segmentation of foreground regions in an image sequence is accomplished by comparing each new frame with the learned background model. Effective modeling of the background is crucial for foreground segmentation. However, changes in dynamic scenes, such as illumination variations, shadow movements, and tree swaying, make background modeling quite difficult.

Much work has been done in background modeling and foreground segmentation. Stauffer and Grimson (1999) propose an online adaptive background model in which a GMM is used to model the sequence of values associated with each pixel. Each pixel in a new frame is classified by matching the value of the pixel with one of the distributions in the GMM associated with this pixel. Sheikh and Shah (2005) use nonparametric density estimation over a joint domain-range representation of image pixels to directly model multimodal spatial uncertainties and complex dependencies between pixel location and color. Jacques et al. (2006) present an adaptive background model for grayscale video sequences. The model utilizes local spatiotemporal statistics to detect shadows and highlights. It can adapt to illumination changes. Haritaoglu et al. (2000) build a statistical background model which represents each pixel by three values which are its minimum intensity value, its maximum intensity value, and the maximum intensity difference between consecutive frames. Wang et al. (2005) present a probabilistic method for background subtraction and shadow removal. Their method detects shadows by a combined intensity and edge measure. Tian et al. (2005) propose an adaptive Gaussian mixture model based on a local normalized cross-correlation metric and a texture similarity metric. These two metrics are used for detecting shadows and illumination changes, respectively. Patwardhan et al. (2008) propose a framework for coarse scene modeling and foreground detection using pixel layers. The framework allows for integrated analysis and detection in a video scene. Wang et al. (2006) use a dynamic probabilistic framework based on a conditional random field to capture spatial and temporal statistics of pixels for foreground and shadow segmentation. Li (2004) constructs a subspace-based background model. An online PCA is used to incrementally learn the background's subspace representation.

The aforementioned methods for background modeling are unable to fully exploit the spatiotemporal redundancies within image ensembles. In particular, the vector subspace techniques (Li 2004) lose local spatial information, perhaps leading to incorrect foreground segmentation results. Consequently, it is interesting to develop the tensor-based learning algorithms to effectively capture the spatiotemporal characteristics of the background pixels.

### 2.2 Visual Object Tracking

The effective modeling of object appearance variations plays a critical role in visual object tracking. There are two types of appearance variations: intrinsic appearance variations resulting from objects themselves such as object pose variation or object shape deformation, and extrinsic appearance variations associated with the environment of the objects, such as changes in illumination, camera motion, or occlusions. Much work has been done on modeling object appearance for visual tracking. Hager and Belhumeur (1996) propose a tracking algorithm which uses an extended gradient-based optical flow method to track objects under varying illumination. Black and Jepson (1998) present a nice subspace learning-based tracking algorithm. A pre-trained, view-based eigenbasis representation is used to model appearance variations. Isard and Blake (1996) use curves or splines to represent the boundary of an object and develop the Condensation algorithm for contour tracking. Black et al. (1998) employ a mixture model to represent and recover object appearance changes in consecutive frames, providing more reliable estimates of image motion than traditional optical flow-based approaches. Jepson et al. (2003) develop a robust tracking algorithm using wavelet features which can be used to model the spatial correlations in images directly and are suited to multiple scales. They use a wavelet basis to decompose each image at two different scales. Each scale has four orientations. In total, 8 filter masks are required. The extraction of wavelet features is time consuming and the number of extracted features is large. Zhou et al. (2004) embed adaptive appearance models into a particle filter to achieve robust visual object tracking. Yu and Wu (2006) propose a non-rigid object tracking algorithm based on a spatial-appearance model which captures non-rigid appearance variations and recovers all the motion parameters effectively. Li et al. (2005) use a generalized geometric transform to handle object deformation, articulated objects, and occlusions. Wong et al. (2006) present a robust appearance-based tracking algorithm using an online-updating Bayesian classifier. Lee and Kriegman (2005) present a tracking method based on an online learning algorithm which incrementally learns a generic appearance model from a video. Lim et al. (2006) present a human tracking framework using robust identification of system dynamics and a nonlinear dimension reduction technique. Ho et al. (2004) present a visual tracking algorithm based on linear subspace learning. Ross et al. (2008) propose a generalized tracking framework based on the incremental vector subspace learning method with sample mean updating. Han et al. (2009) apply continuous density propagation in sequential Bayesian filtering to real-time object tracking. The techniques of density interpolation and density approximation are used to represent the likelihood and the posterior densities with Gaussian mixtures. Gall et al. (2008) combine patch-based matching and region-based matching to track both structured and homogeneous object body parts. Nickel and Stiefelhagen (2008) propose a person tracker based on dynamic integration of generalized cues. A layered sampling strategy is adopted when particle filtering is applied to these cues. Chen and Yang (2007) present a spatial bias appearance model which exploits local region confidences for tracking objects under complex backgrounds and partial occlusions. Mahadevan and Vasconcelos (2009) propose an object tracking algorithm which combines a top-down saliency mode and a bottom-up saliency mode to localize the object. Yang et al. (2009) propose an algorithm for tracking objects with nonstationary appearances. In this algorithm, negative data constraints and bottom-up pair-wise data constraints are used to dynamically adapt to the changes in the object appearance. Kwon and Lee (2009) track a non-rigid object using a local patch-based appearance model which maintains relations between local patches by online updating. Ramanan et al. (2007) propose a human body tracking algorithm which models the appearance of each body part individually and represents the deformable assembly of parts by spring-like connections between pairs of parts. Matthews et al. (2004) propose an appearance template updating algorithm that does not suffer from tracker drift. The template can be updated in every frame and yet still stay attached to the original object. The template is first updated at the current template location. To eliminate drift, this updated template is then aligned with the benchmark template extracted from the first frame. Grabner et al. (2008) formulate the tracker updating process in a semi-supervised fashion by combining the decisions obtained from a given prior classifier and an online classifier. The prior classifier, which is trained using samples extracted from the first frame, is used to deal with tracking drift. Both the above two algorithms (Matthews et al. 2004; Grabner et al. 2008) use a fixed template, which is constructed in the first frame, to significantly alleviate tracking drift. The limitation of these two algorithms is that the template constructed in the first frame becomes unreliable if the object appearance undergoes large changes.

It is noted that the above tracking algorithms do not fully exploit the spatiotemporal information in the image ensembles obtained while tracking an object. This is particularly true for the vector subspace-based tracking algorithms (Ross et al. 2008) in which information about correlations between neighboring pixels is for the most part lost when the images are flattened into vectors. In order to achieve robust tracking, it is necessary to develop the tensor-based learning algorithms for effective subspace analysis to more effectively utilize the spatiotemporal information about an object's appearance.

## 3 Incremental Tensor Subspace Learning

First, basic concepts of tensor algebra, as well as its notations and symbols, are briefly introduced. Then, our online tensor subspace learning algorithm is described.

### 3.1 Tensor Algebra

Tensor algebra (Lathauwer et al. 2000) is the mathematical foundation of multi-linear analysis. A tensor can be regarded as a multi-order "array" lying in multiple vector spaces. We denote an $N$-order tensor as $\hat{A} \in R^{I_1 \times I_2 \times \cdots I_n \cdots \times I_N}$, where $I_n$ $(n = 1, 2, \ldots, N)$ is a positive integer. Each element in this tensor is represented as $a_{i_1 \ldots i_n \ldots i_N}$, where $1 \leq i_n \leq I_n$. Each order of a tensor is associated with a "mode". By unfolding a tensor along a mode, a tensor's unfolding matrix

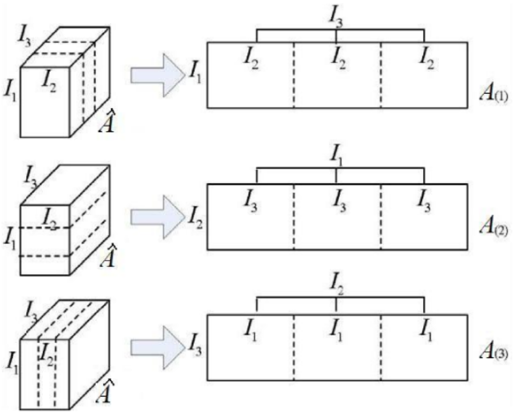

**Fig. 1** Illustration of unfolding a 3-order tensor

corresponding to this mode is obtained. For example, the mode-$n$ unfolding matrix $A_{(n)} \in R^{I_n \times (\prod_{i \neq n} I_i)}$ of $\hat{A}$ consists of $I_n$-dimensional mode-$n$ column vectors which are obtained by varying the $n$th-mode index $i_n$ and keeping indices of the other modes fixed, i.e. the column vectors of $A_{(n)}$ are just the mode-$n$ vectors. Figure 1 shows the process of unfolding a 3-order tensor $\hat{A}$ into three matrices: the mode-1 matrix $A_{(1)}$ consisting of $I_1$-dimensional column vectors, the mode-2 matrix $A_{(2)}$ consisting of $I_2$-dimensional column vectors, and the mode-3 matrix $A_{(3)}$ consisting of $I_3$-dimensional column vectors. The inverse operation of the mode-$n$ unfolding is the mode-$n$ folding which restores the original tensor $\hat{A}$ from the mode-$n$ unfolding matrix $A_{(n)}$, represented as $\hat{A} = fold(A_{(n)}, n)$. The mode-$n$ rank $R_n$ of $\hat{A}$ is defined as the dimension of the space generated by the mode-$n$ vectors: $R_n = rank(A_{(n)})$.

The operation of mode-$n$ product of a tensor and a matrix forms a new tensor. The mode-$n$ product of tensor $\hat{A}$ and matrix $U$ is denoted as $\hat{A} \times_n U$. Let matrix $U \in R^{J_n \times I_n}$. Then, $\hat{A} \times_n U \in R^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$ and its elements are calculated by:

$$(\hat{A} \times_n U)_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 \ldots i_N} u_{j_n i_n}. \tag{1}$$

Of course, $\hat{A} \times_n U$ can be obtained by calculating $U \cdot A_{(n)}$ first where the operation "." represents matrix multiplication, and then operating mode-$n$ folding on $U \cdot A_{(n)}$. Given a tensor $\hat{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and three matrices $C \in R^{J_n \times I_n}$, $D \in R^{K_n \times J_n}$, and $E \in R^{J_m \times I_m}$ $(n \neq m)$, tensor's mode-$n$ product has the following properties:

1. $(\hat{A} \times_n C) \times_m E = (\hat{A} \times_m E) \times_n C = \hat{A} \times_n C \times_m E$
2. $(\hat{A} \times_n C) \times_n D = \hat{A} \times_n (D \cdot C)$

The scalar product of two tensors $\hat{A}$ and $\hat{B}$ with the same set of indices is defined as:

$$\langle \hat{A}, \hat{B} \rangle = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} a_{i_1 i_2 \cdots i_N} b_{i_1 i_2 \cdots i_N}. \tag{2}$$

The Frobenius norm of $\hat{A}$ is defined as: $\|\hat{A}\|_F = \sqrt{\langle \hat{A}, \hat{A} \rangle}$.

### 3.2 Tensor Decomposition

Tensor decomposition is higher-order SVD (Vasilescu and Terzopoulos 2003) which is a generalization of the conventional matrix SVD. The SVD of a matrix $X \in R^{m \times n}$ can be represented as $X = U \Sigma V^T$, where matrix $U \in R^{m \times m}$, matrix $\Sigma \in R^{m \times n}$ and matrix $V \in R^{n \times n}$. The column vectors in $U$ are the eigenvectors of $XX^T$ and $\Sigma$ is a diagonal matrix containing the singular values of $X$. The tensor decomposition of a $N$-order tensor $\hat{A}$ which lies in $N$ vector spaces involves $N$ orthonormal matrices $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$ to generate these $N$ spaces respectively: the orthonormal column vectors of $U^{(n)}$ span the column space of the mode-$n$ unfolding matrix $A_{(n)}$ $(1 \leq n \leq N)$. Then, the tensor $\hat{A}$ is decomposed in the following way:

$$\hat{A} = \hat{B} \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_N U^{(N)} \tag{3}$$

where $\hat{B}$ is the core tensor controlling the interaction between the $N$ mode matrices $U^{(1)}, \ldots, U^{(N)}$. In this way, each mode matrix $U^{(n)}$ $(1 \leq n \leq N)$ (Vasilescu and Terzopoulos 2003) is computed by finding the SVD for the mode-$n$ unfolding matrix: $A_{(n)} = \tilde{U}_n \tilde{\Sigma}_n \tilde{V}_n^T$ and setting the mode matrix $U^{(n)}$ as the orthonormal matrix $\tilde{U}_n$ $(U^{(n)} = \tilde{U}_n)$. The core tensor is computed by Vasilescu and Terzopoulos (2003):

$$\hat{B} = \hat{A} \times_1 U^{(1)^T} \cdots \times_n U^{(n)^T} \cdots \times_N U^{(N)^T}. \tag{4}$$

Such a decomposition can only be achieved offline, i.e. it cannot be used for incremental tensor subspace learning.

In real applications, dimension reduction is necessary for a compact representation of a tensor. Lathauwer et al. (2000) propose a rank-$(R_1, R_2, \ldots, R_N)$ approximation algorithm for the dimension reduction. The algorithm applies the technique of alternate least squares to find the dominant projection subspaces of a tensor. Given an $N$-order tensor $\hat{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, a rank-$(R_1, R_2, \ldots, R_N)$ tensor $\hat{D}$ is found to minimize the square of the Frobenius norm of the error tensor:

$$\hat{D} = \arg\min_{\hat{C}} (\|\hat{A} - \hat{C}\|_F^2). \tag{5}$$

The computational and memory costs are high.

### 3.3 Incremental Rank-$(R_1, R_2, R_3)$ Tensor Subspace Learning

In this section, we describe the proposed incremental rank-$(R_1, R_2, R_3)$ tensor subspace learning algorithm for 3-order tensors. The algorithm applies an incremental SVD algorithm (Ross et al. 2008) to identify the dominant projection subspaces of a 3-order tensor and incrementally update these subspaces when new data arrive.

### *3.3.1 Incremental SVD*

Gu and Eisenstat (1995) propose an efficient and stable algorithm for finding the SVD of a matrix obtained by deleting a row from an original matrix. The algorithm updates the SVD of the original matrix. They (Gu and Eisenstat 1993) also propose a stable and fast algorithm for finding the SVD of a matrix obtained by appending a row to an original matrix with a known SVD. The techniques for downdating and updating SVD are quite similar to each other. In incremental SVD, the aim is to update a given SVD when new data arrives. The SVD contains only a relatively small number of non-zero singular values. Incremental SVD is suitable for background modeling and object tracking, as it emphasizes the current observations and retains information about the previous observations in the previous SVD. The algorithm in Ross et al. (2008) extends the classic incremental SVD (Levy and Lindenbaum 2000) by computing the subspace of a dynamic matrix with the mean updating which removes the assumption that the mean of the previous data is equal to the mean of the new data. More accurate incremental SVD is obtained.

In this paper, we apply the incremental SVD algorithm in Ross et al. (2008) to our incremental tensor subspace learning algorithm. In the following, this incremental SVD algorithm is briefly described. Let $A'$ be the previous data matrix where the data are represented by column vectors. Let $F'$ be a new data matrix. Let $\mu_A$, $\mu_F$, and $\mu_{A^*}$ be the column mean vectors of $A'$, $F'$, and $(A'|F')$ respectively, where the operation "|" merges the left and the right matrices. Let $\{U_A, \Sigma_A, V_A\}$ be the SVD of $A'$, where only the principal components with larger singular values are retained. The SVD $\{U_{A^*}, \Sigma_{A^*}, V_{A^*}\}$ of $(A'|F')$ is estimated from $\mu_A$, $\{U_A, \Sigma_A, V_A\}$, and $F'$. This incremental updating process is outlined as follows:

**Step 1:** Compute

$$\mu_{A^*} = \frac{I_A}{I_A + I_F}\mu_A + \frac{I_F}{I_A + I_F}\mu_F \tag{6}$$

where $I_A$ is the number of the columns in $A'$ and $I_F$ is the number of columns in $F'$.

**Step 2:** Construct a new matrix $E$:

$$E = \left((F' - \mu_F l_{1\times I_F})|\sqrt{\frac{I_A I_F}{I_A + I_F}}(\mu_A - \mu_F)\right) \tag{7}$$

where $l_{1\times I_F}$ is a $I_F$-dimensional row vector whose elements are all "1", i.e. $\overbrace{1, 1, \ldots, 1}^{I_F}$.

**Step 3:** Compute the QR decomposition of $E$ to obtain the eigenbasis $\tilde{E}$ of $E$. Let matrix $U'$ be $U' = (U_A|\tilde{E})$.

**Step 4:** Let matrix $V'$ be

$$V' = \begin{pmatrix} V_A & 0 \\ 0 & \Omega_{I_F} \end{pmatrix} \tag{8}$$

where $\Omega_{I_F}$ is the $I_F \times I_F$ identity matrix. Then, matrix $\Sigma'$ is defined as:

$$\Sigma' = \begin{pmatrix} \Sigma_A & (U_A)^T \quad E \\ 0 & \tilde{E}^T \quad E \end{pmatrix}. \tag{9}$$

**Step 5:** Compute the SVD of $\Sigma'$ : $\Sigma' = \tilde{U}\tilde{\Sigma}\tilde{V}^T$. Then, the of SVD of $(A'|F')$ is obtained: $U_{A^*} = U'\tilde{U}$, $\Sigma_{A^*} = \tilde{\Sigma}$, and $(V_{A^*})^T = (\tilde{V})^T(V')^T$.

The forgetting factor $\lambda$ in Ross et al. (2008) is used to weight the data streams, in order that recent observations are given higher weights than historical ones. This is achieved by replacing the matrix $A'$ with $\lambda A'$ in the incremental updating process described above.

As shown in Levy and Lindenbaum (2000), Ross et al. (2008), Golub and Van Loan (1996), the result of incremental SVD for a matrix is the same as the result of the SVD for the matrix in the batch mode if all the non-zero singular values at the previous step are retained and used for incremental SVD at the current step. The subspace obtained using the incremental SVD in this way is very accurate. Except for matrix SVD, there is no iterative process included in the incremental SVD algorithm. There are reliable, stable algorithms for the matrix SVD that converge rapidly to the correct result. So, the incremental SVD avoids the convergence problem generally associated with iterative algorithms.

Let $m$ be the number of rows in the data matrix and let $n$ be the number of singular values retained in the SVD of the data matrix. In general, updates to the SVD require $O(mn^2)$ operations in practice, although they can be done in $O(mn\log(n))$ in theory. According to Ross et al. (2008), the above incremental SVD only requires $O(mnI_F)$ operations. As $I_F$ is a small integer, this incremental SVD is very fast.

### *3.3.2 Unfolding an Extended 3-Order Tensor*

In line with the requirements for foreground segmentation and object tracking, we only consider the extension of 3-order tensors along one order. Given a 3-order tensor $\hat{A} \in R^{I_1\times I_2\times I_3}$, when a new 3-order tensor $\hat{F} \in R^{I_1\times I_2\times I'_3}$ arrives, $\hat{A}$ is extended along the third order to form a tensor $\hat{A}^* = (\hat{A}|\hat{F}) \in R^{I_1\times I_2\times I^*_3}$ where the operation "|" merges the left and the right tensors along the third order, and $I^*_3 = I_3 + I'_3$. Figure 2 illustrates the process of unfolding $\hat{A}^*$ and the relations between the previous unfolding matrices $A_{(1)}, A_{(2)}, A_{(3)}$, the newly added unfolding matrices $F_{(1)}, F_{(2)}, F_{(3)}$ and the current unfolding matrices $A^*_{(1)}, A^*_{(2)}, A^*_{(3)}$. The three different modes of unfolding a extended 3-order tensor are shown in the left of Fig. 2. The three unfolding matrices $A^*_{(1)}$, $A^*_{(2)}$, and $A^*_{(3)}$ corresponding to the three different modes are shown in the right of Fig. 2. In the figure, the white regions represent the previous tensor

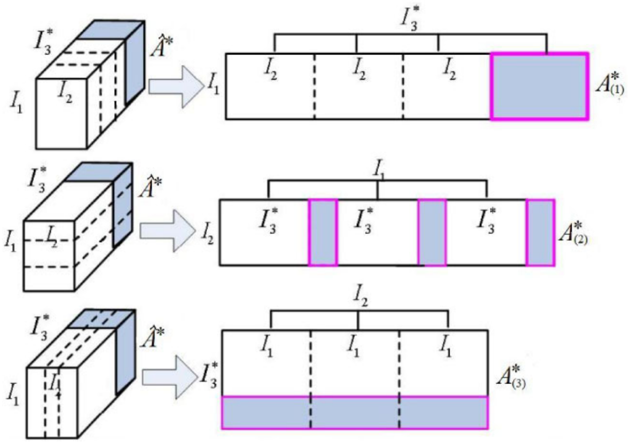

**Fig. 2** Unfolding an extended 3-order tensor

$\hat{A}$ and the previous unfolding matrices, and the gray regions denote the newly added tensor $\hat{F}$ and its unfolding matrices $F_{(1)}, F_{(2)}, F_{(3)}$.

### 3.3.3 Incremental Learning for Unfolding Matrices

After addition of the new tensor, the column spaces of the two unfolding matrices on modes 1 and 2 are extended, and the row space of the mode-3 matrix is extended. Consequently, our incremental tensor subspace learning algorithm needs to online track the changes in these three extended spaces, and online identify the three corresponding dominant projection subspaces for a compact representation of the tensor. These three spaces are handled in the following ways:

1) With respect to $A^*_{(1)}$, as $A^*_{(1)} = (A_{(1)}|F_{(1)})$, the SVD of $A^*_{(1)}$ can be obtained from the SVD of $A_{(1)}$ and the data in $F_{(1)}$ using the incremental SVD technique described in Sect. 3.3.1.

2) With respect to $A^*_{(2)}$, it is noted that $A^*_{(2)}$ can be decomposed as: $A^*_{(2)} = (A_{(2)}|F_{(2)}) \cdot P$, where $P$ is an elementary counterchange matrix obtained by column exchange and transpose operations on an identity matrix $Z$ with rank $I_1 I_3^*$. Let

$$Z = (\overbrace{E_1}^{I_3} \mid \overbrace{Q_1}^{I_3'} \mid \overbrace{E_2}^{I_3} \mid \overbrace{Q_2}^{I_3'} \ldots \mid \overbrace{E_{I_1}}^{I_3} \mid \overbrace{Q_{I_1}}^{I_3'}) \qquad (10)$$

which is generated by partitioning $Z$ into $2I_1$ blocks along the column dimension. The partition of $Z$ corresponds to $A^*_{(2)}$'s block partition shown in Fig. 2, i.e. $E_1, E_2, \ldots E_{I_1}$ correspond to the white regions, and $Q_1, Q_2, \ldots Q_{I_1}$ correspond to the gray regions. Then, the elementary counterchange matrix $P$ is formulated as:

$$P = (E_1|E_2 \ldots |E_{I_1}|Q_1|Q_2 \ldots |Q_{I_1})^T. \qquad (11)$$

In this way, the column subspace (column projection matrix) of $A^*_{(2)}$ can be efficiently obtained by the SVD of $(A_{(2)}|F_{(2)})$ from the SVD of $A_{(2)}$ and $F_{(2)}$ using the incremental SVD technique.

3) With respect to $A^*_{(3)}$, we estimate the row subspace, instead of the column subspace. This is because we only consider the extension of a 3-order tensor along the third order with the increase of the dimension of the third order, and we then only need to track the changes in the row space of $A^*_{(3)}$, and identify the new row projection subspace. For the applications to background modeling and object tracking, this row space contains information about the changes in the appearance of an object or background over time. But the subspace of the column space is of no use for background modeling and object tracking, because values of each pixel in an appearance over time form a vector, and the subspace of this column space only uses a small number of vectors to describe the vectors of all the pixels in the appearance. The dimension of this column space becomes larger and larger over time until finally the column space is too large for practical applications. Consequently, for $A^*_{(3)}$, we should calculate the SVD of the matrix

$$\left(\frac{A_{(3)}}{F_{(3)}}\right)^T \qquad (12)$$

where the operation "—" merges the upper and lower matrices. Formula (12) is equal to $(A_{(3)})^T|(F_{(3)})^T$. We obtain the SVD of $(A_{(3)})^T|(F_{(3)})^T$ from the SVD of $(A_{(3)})^T$ and $(F_{(3)})^T$ using the incremental SVD technique.

The first $R_1$ dominant singular vectors with the larger singular values are selected from the SVD of $(A_{(1)}|F_{(1)})$ to form the subspace of $(A_{(1)}|F_{(1)})$. The first $R_2$ dominant singular vectors with the larger singular values are selected from the SVD of $(A_{(2)}|F_{(2)})$ to form the subspace of $(A_{(2)}|F_{(2)})$. The first $R_3$ dominant singular vectors with the larger singular values are selected from the SVD of $((A_{(3)})^T|(F_{(3)})^T)$ to form the subspace of $((A_{(3)})^T|(F_{(3)})^T)$. The obtained three subspaces form the result of the incremental rank-$(R_1, R_2, R_3)$ tensor subspace learning.

### 3.3.4 Incremental SVD for 3-Order Tensors

We can use the results of incremental SVD of $(A_{(1)}|F_{(1)})$, $(A_{(2)}|F_{(2)})$, and $((A_{(3)})^T|(F_{(3)})^T)$ to formulate online 3-order tensor decomposition which is based on the SVD of the unfolding matrices $A^*_{(1)}$, $A^*_{(2)}$, and $A^*_{(3)}$.

1) Let $\{\tilde{U}^{(1)}, \tilde{\Sigma}^{(1)}, \tilde{V}^{(1)}\}$ be the SVD of $(A_{(1)}|F_{(1)})$. As $A^*_{(1)} = (A_{(1)}|F_{(1)})$, the SVD $U^{(1)}, D^{(1)}, V^{(1)}$ of $A^*_{(1)}$ is obtained by: $U^{(1)} = \tilde{U}^{(1)}$, $\Sigma^{(1)} = \tilde{\Sigma}^{(1)}$, and $V^{(1)} = \tilde{V}^{(1)}$.

2) Let $\{\tilde{U}^{(2)}, \tilde{\Sigma}^{(2)}, \tilde{V}^{(2)}\}$ be the SVD of $(A_{(2)}|F_{(2)})$. As $A^*_{(2)} = (A_{(2)}|F_{(2)}) \cdot P$, where $P$ is defined in (11), the SVD

$U^{(2)}, D^{(2)}, V^{(2)}$ of $A^*_{(2)}$ is obtained by: $U^{(2)} = \tilde{U}^{(2)}$, $\Sigma^{(2)} = \tilde{\Sigma}^{(2)}$, and $V^{(2)} = P^T \cdot \tilde{V}^{(2)}$.

3) Let $\{\tilde{U}^{(3)}, \tilde{\Sigma}^{(3)}, \tilde{V}^{(3)}\}$ be the SVD of $((A_{(3)})^T | (F_{(3)})^T)$. As $A^*_{(3)} = (\frac{A_{(3)}}{F_{(3)}})$, the SVD $U^{(3)}, \Sigma^{(3)}, V^{(3)}$ of $A^*_{(3)}$ is obtained by: $U^{(3)} = \tilde{V}^{(3)}$, $\Sigma^{(3)} = (\tilde{\Sigma}^{(3)})^T$, $V^{(3)} = \tilde{U}^{(3)}$.

It is noted that our online 3-order tensor decomposition is different from the offline tensor decomposition formulated in (3) and (4) in that we use the row vectors in the mode-3 unfolding matrix $A^*_{(3)}$ rather than its column vectors. This can be regarded as an extension of the offline tensor decomposition, because if column vectors are used for $A^*_{(3)}$ the incremental subspace learning for $A^*_{(3)}$ cannot be achieved. The mode-3 unfolding matrix and its incremental subspace learning just correspond to the vector subspace learning algorithm in Ross et al. (2008). In this way, the subspace learned using the algorithm in Ross et al. (2008) is kept in our tensor subspace learning algorithm. This ensures, in theory, that our incremental tensor subspace learning algorithm can obtain more accurate results than the incremental vector subspace learning algorithm.

#### 3.3.5 Likelihood Evaluation

It is necessary for a subspace learning-based algorithm to evaluate the likelihood of the test sample given the learned subspace or subspaces. The likelihood evaluation method with respect to the proposed incremental 3-order tensor subspace learning algorithm is described as follows: As we only consider the increase of the dimension of the third order of a 3-order tensor $A \in R^{I_1 \times I_2 \times I_3}$ with the dimensions of the first and second orders held constant, the learned subspaces of the tensor should be composed of the mode-1 column projection matrix $U^{(1)} \in R^{I_1 \times R_1}$, the mode-2 column projection matrix $U^{(2)} \in R^{I_2 \times R_2}$, and the mode-3 row projection matrix $V^{(3)} \in R^{(I_1 \cdot I_2) \times R_3}$. Let $\mu^{(1)}$ and $\mu^{(2)}$ be the column mean vectors of $A_{(1)}$ and $A_{(2)}$ respectively. Let $\mu^{(3)}$ be the row mean vector of $A_{(3)}$. Tensors $\hat{M}_1$ and $\hat{M}_2$ are defined by:

$$\begin{aligned} \hat{M}_1 &= (\overbrace{\mu^{(1)}, \ldots, \mu^{(1)}}^{I_2}) \in R^{I_1 \times I_2 \times 1} \\ \hat{M}_2 &= (\underbrace{\mu^{(2)}, \ldots, \mu^{(2)}}_{I_1})^T \in R^{I_1 \times I_2 \times 1} \end{aligned}. \tag{13}$$

The sum of the reconstruction error norms of a test sample $\hat{J} \in R^{I_1 \times I_2 \times 1}$ corresponding to the three modes is computed by:

$$\begin{aligned} RE = \Bigg( &\sum_{i=1}^{2} \left\| (\hat{J} - \hat{M}_i) - \left( (\hat{J} - \hat{M}_i) \times_i (U^{(i)} \cdot U^{(i)^T}) \right) \right\|_F^2 \\ &+ \left\| (J_{(3)} - \mu^{(3)}) \right. \\ &\left. - \left( (J_{(3)} - \mu^{(3)}) \cdot (V^{(3)} \cdot V^{(3)^T}) \right) \right\|_F^2 \Bigg)^{\frac{1}{2}} \end{aligned} \tag{14}$$

where $J_{(3)}$ is the mode-3 unfolding matrix of$\hat{J}$, and the reconstruction error for a mode represents the distance from the sample to the subspace corresponding to this mode. The likelihood of the test sample $\hat{J}$ given the learned tensor subspaces is computed by:

$$p(J | U^{(1)}, U^{(2)}, V^{(3)}) \propto \exp\left( -\frac{RE^2}{2\sigma^2} \right) \tag{15}$$

where $\sigma$ is a scale factor. The smaller the *RE*, the larger the likelihood.

#### 3.3.6 Theoretical Comparison

In the following, we compare our algorithm with DTA, WTA, and STA in Sun et al. (2006a, 2006b, 2008).

DTA: For DTA in Sun et al. (2006a, 2008), the incremental updating of the subspace of the unfolding matrix of a tensor on each mode is performed by updating the covariance matrix formed from the columns of the unfolding matrix and then obtaining an eigenvalue decomposition of the updated covariance matrix. The updating is achieved by:

$$C_d \leftarrow \lambda C_d + X_{(d)}^T X_{(d)} \tag{16}$$

where $C_d$ on the right hand side of (16) is the covariance matrix of the unfolding matrix on mode $d$ for the data observed at the previous steps, and $X_d$ is the unfolding matrix on mode $d$ for the incoming data at the current step. According to Ross et al. (2008), this updating of the covariance matrix is based on the assumption that $\mu_d = \mu'_d$ where $\mu_d$ is the mean of the previous unfolding matrix, and $\mu'_d$ is the mean of $X_d$, i.e. any changes in the mean are not considered. So, this updating is not accurate if there are significant changes in the mean. If the updating is applied to background modeling and object tracking, the corresponding model cannot adapt to large appearance changes.

According to (Ross et al. 2008) the accurate updating of the covariance matrix with mean updating should be

$$C_d \leftarrow \lambda C_d + X_{(d)}^T X_{(d)} + \frac{pq}{p+q} \left( \mu_d - \mu'_d \right) \left( \mu_d - \mu'_d \right)^T \tag{17}$$

where $p$ is the number of columns in the previous matrix and $q$ is the number of columns in the new incoming data matrix. Even if we introduce (17) into DTA, the obtained subspace is still not accurate for the applications to background modeling and object tracking, because DTA has the small size problem. For a 3-order tensor ($R^{I_1 \times I_2 \times I_3}$), the dimension of the row vectors in the unfolding matrix on mode 3 is $I_1 I_2$. The corresponding covariance matrix is a $I_1 I_2 \times I_1 I_2$

matrix. It is assumed that there are $s$ new samples. In the applications to background modeling and object tracking, $s \ll I_1 I_2 \times I_1 I_2$. The rank of the mode-3 unfolding matrix of the new incoming samples is equal to or less than $s$. The eigenvector decomposition of a $I_1 I_2 \times I_1 I_2$ matrix obtained from $s$ samples is degenerate. This is the small sample problem. One of the motivations in Ross et al. (2008) is to solve the small sample problem using the incremental SVD. So, our incremental subspace tensor learning method is more accurate than DTA for the applications to background modeling and object tracking.

2) WTA: WTA in Sun et al. (2006b, 2008) updates the covariance matrix of the unfolding matrix in a similar way to the DTA, but it only focus on a window of $w$ recent samples, i.e. the covariance matrix is more dependent on the most recent $w$ samples. Only one sample is used in each step of the updating process:

$$C_d \leftarrow \lambda C_d - X_{n-w,(d)} X_{n-w,(d)}^T + X_{n,(d)} X_{n,(d)}^T \tag{18}$$

where $X_{n-w,(d)}$ is the $d$th mode unfolding matrix of the $(n-w)$th sample, and $X_{n,(d)}$ is the mode-$d$ unfolding matrix of the $n$-th sample, i.e. the new sample. WTA has the same limitations as the DTA: it is assumed that the mean of the previous data is equal to the mean of the new data and it has the same small sample size problem. Furthermore, the fact that only the $w$ most recent samples are focused on leads to model drift in applications to tracking. According to Sun et al. (2008), the tensor subspaces learned using WTA are less accurate than the tensor subspaces learned using DTA. So, WTA is less accurate than our tensor subspace learning method for the applications to background modeling and object tracking.

3) STA: STA in Sun et al. (2006a, 2008) applies the SPIRIT (streaming pattern discovery in multiple timeseries) iterative algorithm (Papadimitriou et al. 2005) to the incoming data to update the column subspace of the data matrix for approximating DTA without diagonalization. The tensor subspaces learned using STA are less accurate than the tensor subspaces learned using DTA while STA is faster than DTA. So, STA is less accurate than our tensor subspace learning method.

#### 3.3.7 *Remarks*

We discuss the following aspects:

1) Compared with the offline SVD for tensors, the proposed incremental SVD for 3-order tensors adapts to appearance variations with a much lower complexity. The incremental tensor subspace learning algorithm requires $O[I_1 I_2 (I_3 + I_3')(R_1 + R_2 + R_3)]$ operations and $O[I_1 R_1 + I_2 R_2 + I_1 I_2 (R_3 + I_3')]$ memory units. In comparison, the offline SVD for 3-order tensors requires $O[I_1 I_2 (I_1 + I_2 + I_3 + I_3')(I_3 + I_3')]$ operations and $O[I_1 (I_3 + I_3') I_2]$ memory units (Wang and Ahuja 2008). It is obvious that $I_3$ is much larger than $R_1$, $R_2$, $R_3$, and $I_3'$. So, the complexity of the offline SVD for 3-order tensors is much higher than that of our incremental tensor subspace learning algorithm.

2) The batch tensor subspace learning is not suitable for background modeling and object tracking, due to the following points: A) The time taken for batch computation increases indefinitely as the length of the video increases. B) The batch processing algorithm considers all the changes in object appearance in all the observed frames. The information in the sequence of frames is not used. This reduces the accuracy of appearance updating especially when the number of frames is very large, as recent frames provide more accurate information about the appearance model.

3) When our incremental tensor subspace learning method is applied to appearance modeling for background modeling and object tracking, the appearance should be treated as a 2-D matrix, like the vector subspace-based algorithms in Li (2004), Ross et al. (2008). In Li (2004), Ross et al. (2008), the 2-matrix is unfolded into a vector, and the SVD technique is used to obtain the subspace representing changes in object appearance over time. In our method, the tensor technique is introduced to object appearance representation over time by treating appearances using 2-D matrices. The spatial information of the object appearance is included in the subspaces of the unfolding matrices on the first and the second modes of the tensor. The subspace of the unfolding matrix on the third mode of the tensor describes appearance changes over time. In contrast to the vector subspace-based algorithm in Li (2004), Ross et al. (2008), our tensor subspace-based algorithm captures more spatial information in images.

## 4 Foreground Segmentation

We apply the proposed incremental tensor subspace learning algorithm to foreground segmentation from image sequences. Figure 3 shows the architecture of our foreground segmentation algorithm. A tensor subspace-based background appearance model is obtained using the incremental tensor subspace learning algorithm. When new frames arrive, they are matched with the background appearance model to detect foreground pixels in these new frames. The background images corresponding to these new frames are then constructed and used to update the tensor subspace-based background model using the incremental tensor subspace learning.

Let $\hat{G}$ be a sequence $B_1, B_2, \ldots, B_q, \ldots, B_t$ of background appearance images of a scene, i.e. a 3-order background appearance tensor, where $B_q$ represents the $q$th background image. We define a rectangular pixel region centered at pixel $(u, v)$, where the region has the height of $I_1$ pixels and the width of $I_2$ pixels. Then, we define a back-

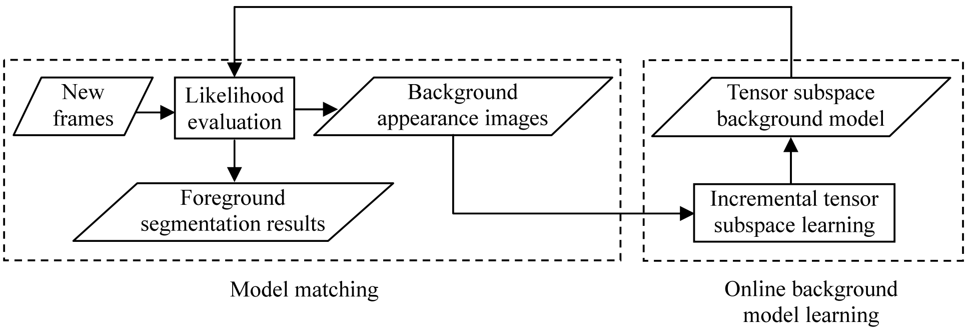
New frames
Likelihood evaluation
Background appearance images
Foreground segmentation results
Tensor subspace background model
Incremental tensor subspace learning
Model matching
Online background model learning

**Fig. 3** The architecture of our foreground segmentation algorithm

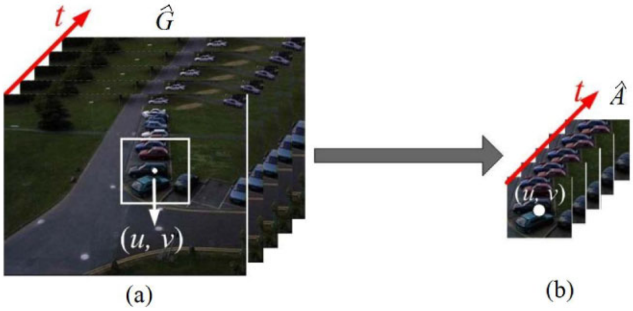

**Fig. 4** The relation between tensors $\hat{G}$ and $\hat{A}$

ground appearance tensor $\hat{A} = B_1^{uv}, B_2^{uv}, \dots, B_q^{uv}, \dots, B_t^{uv}$ which is smaller than $\hat{G}$, where $B_q^{uv}$ is composed of pixels in the rectangular pixel region. This tensor captures the spatiotemporal interactions between pixel $(u, v)$ and its $I_1 I_2 - 1$ neighboring pixels in the rectangular region. The relation between tensors $\hat{G}$ and $\hat{A}$ is illustrated in Fig. 4. We apply the proposed incremental tensor subspace learning to tensor $\hat{A}$ to effectively mine statistical properties of $\hat{A}$. In this paper, we develop two background appearance models for grayscale image sequences and color image sequences respectively.

### 4.1 Grayscale Background Model

For grayscale image sequences, the tensor subspace model for the tensor $\hat{A} \in R^{I_1 \times I_2 \times t}$ associated with pixel $(u, v)$ consists of the mode-1 column projection matrix $U^{(1)} \in R^{I_1 \times R_1}$, the mode-2 column projection matrix $U^{(2)} \in R^{I_2 \times R_2}$, and the mode-3 row projection matrix $V^{(3)} \in R^{(I_1 \cdot I_2) \times R_3}$. Given the rectangular pixel region $J_{t+1}^{uv} \in R^{I_1 \times I_2}$ centered at $(u, v)$ in a new frame $t+1$, the likelihood $P(J_{t+1}^{uv} | U^{(1)}, U^{(2)}, V^{(3)})$ for $J_{t+1}^{uv}$ given the learned tensor subspace model $\{U^{(1)}, U^{(2)}, V^{(3)}\}$ for pixel $(u, v)$ is estimated using (14) and (15). In this way, the criterion for foreground segmentation is defined as:

$$(u, v) \in \begin{cases} \text{background} & \text{if } P(J_{t+1}^{uv} | U^{(1)}, U^{(2)}, V^{(3)}) > T_{gray} \\ \text{foreground} & \text{otherwise} \end{cases} \tag{19}$$

where $T_{gray}$ denotes a threshold.

After foreground segmentation is implemented for all the pixels at frame $t+1$, a background appearance image $B_{t+1}$ at time $t+1$ is constructed according to the result of foreground segmentation at time $t+1$ and the previous background appearance images $B_1, B_2, \dots, B_t$. Let $M_t$ be the mean background appearance image at time $t$:

$$M_t = \frac{1}{t} \sum_{k=1}^{t} B_k. \tag{20}$$

Typically, $M_t$ is computed recursively:

$$M_t = \frac{t-1}{t} M_{t-1} + \frac{1}{t} B_t. \tag{21}$$

Then, the value $B_{t+1}(u, v)$ of each pixel $(u, v)$ in $B_{t+1}$ is computed by:

$$B_{t+1}(u, v) = \begin{cases} (1-\omega) M_t(u, v) + \omega J_{t+1}(u, v) \\ \quad \text{if } (u, v) \in \text{foreground} \\ J_{t+1}(u, v) \quad \text{otherwise} \end{cases} \tag{22}$$

where $\omega$ is a learning rate factor which is set to a very small value. Formula (22) means that if a pixel is determined to belong to the background, then its value in the current background appearance image is of course set to its value in the current frame; otherwise its value in the current background appearance image is interpolated between its value in the current frame and its value in the mean background appearance image. This ensures that the background model can adapt to the changes in the environment such as lighting variations. Subsequently, the estimated background appearance images are used to update the tensor subspace model of the background appearance tensor $\hat{A}$ of each pixel $(u, v)$ by applying the proposed incremental tensor subspace learning algorithm.

### 4.2 Color Background Model

There are two typical background modeling algorithms which can deal with color: the GMM-based algorithm

(Stauffer and Grimson 1999) and the kernel-based algorithm (Elgammal et al. 2002). In the GMM-based algorithm, each mixture component of the background model is a single Gaussian distribution in the RGB color space. The covariance matrix of each Gaussian is a diagonal matrix $\sum = \sigma^2 I$ where $I$ is an identity matrix and $\sigma$ is a standard deviation. The GMM-based algorithm deals with the RGB channels separately, and assumes the same standard deviation for each channel. Although dealing with color in this way is not consistent with real data distributions, it can greatly increase the speed with only a slight loss of accuracy. The kernel density estimation-based background modeling algorithm (Elgammal et al. 2002) also deals with the color channels separately, i.e. the joint probability density of color pixels is decomposed into the product of independent kernel density functions of each color channel. In contrast to the GMM-based algorithm, the kernel-based algorithm uses different Gaussian kernel variances for each channel.

Referring to the GMM-based algorithm (Stauffer and Grimson 1999) and the Kernel-based algorithm (Elgammal et al. 2002), we extend the proposed background model for grayscale sequences to the background model for color image sequences. In the color background model, we use the $(r, g, s)$ color space which is defined in terms of the RGB color space by $r = R/(R+G+B)$, $g = G/(R+G+B)$, and $s = (R+G+B)/3$. The effects of shadows are reduced in the $(r, g, s)$ color space (Elgammal et al. 2002).

Let $\hat{A}^r \in R^{I_1 \times I_2 \times t}$ be a 3-order tensor $B_1^r B_2^r \ldots B_q^r \ldots B_t^r$, where $B_q^r$ is a matrix which is composed of the $r$-components of pixels in the rectangular appearance region centered at pixel $(u, v)$ at time $q$. Similarly, we define tensors $\hat{A}^g \in R^{I_1 \times I_2 \times t}$ $(B_1^g B_2^g \ldots B_t^g)$ and $\hat{A}^s \in R^{I_1 \times I_2 \times t}$ $(B_1^s B_2^s \ldots B_t^s)$ for the $g$-components and the $s$-components of the pixels in the region centered at $(u, v)$. For each component's tensor $\hat{A}^{\mathbb{C}}$ $(\mathbb{C} \in \{r, g, s\})$, a tensor subspace model is incrementally learned using the proposed incremental tensor subspace learning algorithm. In this way, three tensor subspace models for each pixel are obtained corresponding to the three color components. The learning process for the tensor of each component is similar to that for the grayscale sequence. The subspace model for tensor $\hat{A}^{\mathbb{C}}$ $(\mathbb{C} \in \{r, g, s\})$ consists of the following terms: 1) the maintained subspace dimensions $(R_1^{\mathbb{C}}, R_2^{\mathbb{C}}, R_3^{\mathbb{C}})$ corresponding to the three tensor unfolding modes; 2) the column projection matrices $U_{\mathbb{C}}^{(1)} \in R^{I_1 \times R_1^{\mathbb{C}}}$ and $U_{\mathbb{C}}^{(2)} \in R^{I_2 \times R_2^{\mathbb{C}}}$ of modes 1 and 2, and the mode-3 row projection matrix $V_{\mathbb{C}}^{(3)} \in R^{(I_1 \cdot I_2) \times R_3^{\mathbb{C}}}$; 3) the column mean vectors $\mu_{\mathbb{C}}^{(1)}$ and $\mu_{\mathbb{C}}^{(2)}$ of the unfolding matrices $A_{(1)}^{\mathbb{C}}$ and $A_{(2)}^{\mathbb{C}}$ of modes 1 and 2, and the row mean vector $\mu_{\mathbb{C}}^{(3)}$ of the mode-3 unfolding matrix $A_{(3)}^{\mathbb{C}}$. Let $J_{t+1}^r \in R^{I_1 \times I_2}$, $J_{t+1}^g \in R^{I_1 \times I_2}$ and $J_{t+1}^s \in R^{I_1 \times I_2}$ be, respectively, the matrices of $r, g, s$-components of pixels in the appearance region centered at $(u, v)$ at time $t+1$. The distances $RM_{uv}^r$, $RM_{uv}^g$ and $RM_{uv}^s$ between $\{r, g, s\}$-components of pixels in the rectangular image region $J_{t+1}^{uv}$ centered at $(u, v)$ at frame $t+1$ and the learned $\{r, g, s\}$-component tensor-based subspace models are calculated, respectively, using (14). Then, the likelihood $P_{uv}$ for $\hat{J}_{t+1}^{uv}$ given the learned tensor subspace models $\{U_r^{(1)}, U_r^{(2)}, V_r^{(3)}\}$, $\{U_g^{(1)}, U_g^{(2)}, V_g^{(3)}\}$, and $\{U_s^{(1)}, U_s^{(2)}, V_s^{(3)}\}$ for pixel $(u, v)$ is estimated by:

$$P_{uv} = \exp\left(-\frac{1}{2}\left(\frac{RM_{uv}^r}{\sigma_r}\right)^2 - \frac{1}{2}\left(\frac{RM_{uv}^g}{\sigma_g}\right)^2 - \frac{1}{2}\left(\frac{RM_{uv}^s}{\sigma_s}\right)^2\right) \tag{23}$$

where $\sigma_r$, $\sigma_g$, and $\sigma_s$ are three scale factors. The criterion for foreground segmentation is defined as:

$$(u, v) \in \begin{cases} \text{background} & \text{if } P_{uv} > T_{color} \\ \text{foreground} & \text{otherwise} \end{cases} \tag{24}$$

where $T_{color}$ is a threshold.

Using the result of foreground segmentation at the current time $t+1$, we can estimate the current background appearance image which consists of the $(r, g, s)$-component background appearance matrices: $B_{t+1}^r \in R^{I_1 \times I_2}$, $B_{t+1}^g \in R^{I_1 \times I_2}$, and $B_{t+1}^s \in R^{I_1 \times I_2}$. The elements in these three matrices are estimated by:

$$B_{t+1}^{\mathbb{C}}(u, v) = \begin{cases} (1 - \omega_{\mathbb{C}}) M_{t+1}^{\mathbb{C}}(u, v) + \omega_{\mathbb{C}} J_{t+1}^{\mathbb{C}}(u, v) & \\ \quad \text{if } (u, v) \in \text{foreground} & \\ J_{t+1}^{\mathbb{C}}(u, v) \quad \text{otherwise} & \end{cases} \tag{25}$$

where $\mathbb{C} \in \{r, g, s\}$, $\omega_{\mathbb{C}}$ is a learning rate factor, and $M_t^{\mathbb{C}}$ is the mean matrix of $B_1^{\mathbb{C}}, B_2^{\mathbb{C}}, \ldots,$ and $B_t^{\mathbb{C}}$ at time $t$. As formulated in (20) and (21), $M_t^{\mathbb{C}}$ is computed recursively.

Subsequently, the newly estimated background appearance component matrices are used to incrementally update the subspace models of the component tensors of the color background appearance region centered at each pixel by applying the incremental tensor subspace learning algorithm. The tensor subspace model for each component is learned and updated in the same way as the tensor subspace model for the grayscale sequences. Figure 5 is used to further illustrate the foreground segmentation process for color image sequences.

## 5 Visual Tracking

We apply the proposed incremental tensor subspace learning algorithm to appearance-based object tracking. Figure 6 shows the architecture of our object tracking algorithm. In the algorithm, a low dimensional subspace model for the appearance tensor of an object is learned. The model uses the incremental rank-$(R_1, R_2, R_3)$ tensor subspace learning

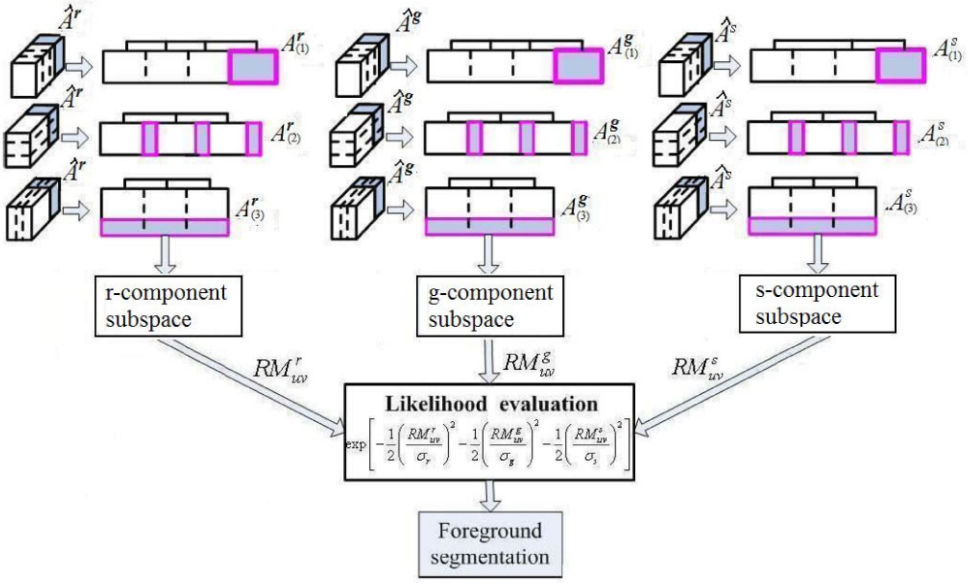

**Fig. 5** Foreground segmentation for color image sequences

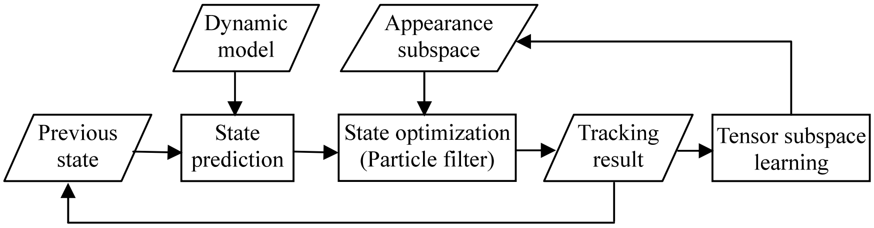

**Fig. 6** The architecture of our tracking algorithm

algorithm to find the dominant projection subspaces of the 3-order tensor of the object appearance. The current object state, which is initialized according to the previous state and the dynamic model, is optimized using a particle filter. The appearance region specified by the optimal state is the tracking result which is used to further update the object appearance tensor subspace model.

### 5.1 Object Appearance Tensor Subspace Model

For tracking in grayscale images, the appearance regions of an object at different frames are normalized to the same size using linear interpolation and the intensities of the pixels in these normalized appearance regions form a tensor $\hat{A}$. The length of the tensor increases with time. For this tensor $\hat{A}$, unfolding matrices $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$ of modes 1, 2, and 3 are obtained. Column subspaces $U^{(1)}$ and $U^{(2)}$ of modes 1 and 2 and row subspace $V^{(3)}$ of mode 3 are learned using the proposed incremental tensor subspace learning algorithm. These three subspaces are combined via tensor reconstruction as formulated in (14) to form a tensor subspace-based object appearance model. We compute the likelihood for a candidate object appearance region given the tensor subspaces of the object appearance. The value of the likelihood is a measure of the similarity between the candidate region and the tensor subspaces of the object appearance.

When tracking an object in a color image sequence, the $r$, $g$ and $s$ values of the pixels in the normalized appearance regions of the object at different frames form, respectively, three tensors $\hat{A}^r$, $\hat{A}^g$, and $\hat{A}^s$ for the $r, g$, and $s$ components. For each tensor $\hat{A}^{\mathbb{C}}$ ($\mathbb{C} \in \{r, g, s\}$), unfolding matrices $A^{(1)}_{\mathbb{C}}$, $A^{(2)}_{\mathbb{C}}$, and $A^{(3)}_{\mathbb{C}}$ of modes 1, 2, and 3 are obtained, and the corresponding column subspaces $U^{(1)}_{\mathbb{C}}$ and $U^{(2)}_{\mathbb{C}}$ of modes 1 and 2 and the corresponding row subspace $V^{(3)}_{\mathbb{C}}$ of mode 3 are then learned. We compute the reconstruction error $RM^{\mathbb{C}}$ ($\mathbb{C} \in \{r, g, s\}$) of the component $\mathbb{C}$ matrix of a candidate object appearance region given the component$\mathbb{C}'$s tensor subspaces of the object using (14). Then, the likelihood for the candidate object region given the tensor subspaces of the object is estimated by:

$$\exp\left(-\frac{1}{2}\left(\frac{RM^r}{\sigma_r}\right)^2-\frac{1}{2}\left(\frac{RM^g}{\sigma_g}\right)^2-\frac{1}{2}\left(\frac{RM^s}{\sigma_s}\right)^2\right) \quad (26)$$

where $\sigma_r$, $\sigma_g$, and $\sigma_s$ are three scale factors. Formula (26) is similar to (23).

### 5.2 Bayesian Inference for Tracking

A Markov model with hidden state variables is used for motion estimation. An affine image warping is applied to model the object motion between two consecutive frames. The object state variable vector $X_t$ at time $t$ is described using the six parameters of the affine motion transform: $x_t, y_t, \eta_t, s_t, \beta_t$, and $\phi_t$, which are respectively the translation parameters of the $x, y$ coordinates, the rotation angle, the scale, the aspect ratio, and the skew direction. The location, size and pose of the object are indicated in the affine motion parameters. Given a set of observed image regions $O_1, O_2, \dots, O_t$, the posterior probability of the object state is formulated by Bayes' theorem:

$$\begin{aligned} &p(X_t|O_1, O_2, \dots, O_t) \\ &\quad \propto p(O_t|X_t) \int p(X_t|X_{t-1}) \\ &\qquad \times p(X_{t-1}|O_1, O_2, \dots, O_{t-1})dX_{t-1} \end{aligned} \tag{27}$$

where $p(O_t|X_t)$ is the likelihood for the observation $O_t$ given the object state $X_t$, and $p(X_t|X_{t-1})$ is the probability model for the object state transition. The terms $p(O_t|X_t)$ and $p(X_t|X_{t-1})$ determine the tracking process. A Gaussian distribution is employed to model the state transition distribution $p(X_t|X_{t-1})$:

$$p(X_t|X_{t-1}) = N(X_t : X_{t-1}, \Sigma) \tag{28}$$

where $\Sigma$ denotes a diagonal covariance matrix with six diagonal elements $\sigma_x^2, \sigma_y^2, \sigma_\eta^2, \sigma_s^2, \sigma_\beta^2$, and $\sigma_\phi^2$. The observation model $p(O_t|X_t)$ is evaluated using the likelihood for a sample image region given the learned appearance tensor subspaces. For grayscale image sequences, this probability is estimated using (15). For color image sequences, this probability is estimated using (26).

A standard particle filter (Isard and Blake 1996) is used to estimate the object motion state. The components of each particle correspond to the six affine motion parameters. For the maximum a posteriori estimate, the particle which maximizes the observation model is selected as the optimal state of the object in the current frame. The affinely warped image region associated with the optimal state of the object is used to incrementally update the tensor subspace-based object appearance model.

## 6 Experiments

The experimental results for the foreground segmentation are shown first, and then the experimental results for the visual object tracking are shown. The runtimes are measured on a computer with 3.25 GB RAM and Intel Core2 Quad CPU at 2.83 GHz.

### 6.1 Foreground Segmentation

In order to evaluate the performances of the proposed incremental tensor subspace learning-based algorithms for foreground segmentation for grayscale image sequences and for color image sequences, four examples corresponding to four videos are shown to demonstrate the claimed contributions of our algorithms. The first two videos consist of 8-bit grayscale images while the two remaining videos consist of 24-bit color images. The first video is selected from the PETS2001 database, available on http://www.cvg.cs.rdg.ac.uk/slides/pets.html. It consists of 650 frames. In the video, a person walks along a road in a well lit scene, and vehicles enter or leave the scene every now and then. The second video consists of 1012 frames. In the video, three persons walk in a scene containing a side of a building, two lightly swaying trees, and two cars. In the middle of the video, these three people occlude each other. The third video consists of 89 frames. In the video, two cars are moving in a dark and blurry traffic scene. The last video is selected from CAVIAR2, available on http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/. It consists of 1000 frames. In the video, several people walk along a corridor, or enter or leave the corridor from time to time. The first two videos are used to evaluate the foreground segmentation performance of the proposed algorithm for grayscale image sequences, while the third and fourth videos are used to evaluate the foreground segmentation for color image sequences. The tensor subspace-based background models for either grayscale image sequences or color image sequences are updated every three frames. The rectangular regions centered at each pixel are of size 5 × 5 pixels, i.e. the values of $I_1$ and $I_2$ in Sect. 4 are both set to 5.

We compare our background modeling algorithm with the representative incremental vector subspace learning-based algorithm in (Li 2004) and the standard GMM-based algorithm in Stauffer and Grimson (1999), with respect to accuracy and speed. The competing algorithms are briefly described below:

- The vector subspace-based algorithm in (Li 2004) incrementally constructs, using online PCA, a scene's background model represented by a low dimensional vector subspace. Although it uses a PCA model defined over the whole image, information on each pixel is included in the PCA model and each pixel should be handled one by one to detect foreground pixels using the learned PCA model. In the algorithm, each image is flattened into a vector whose dimension is equal to the product of the width and the height of the image. In our algorithm the maximum of the lengths of the flattened vectors is only $I_1 I_2$ which is much less than the number of pixels in the full image. Our algorithm can be applied to images which are in practice too large for the vector subspace-based algorithm in Li (2004).

- For more fair comparison, we modify the original version of the vector subspace-based algorithm in Li (2004) by defining a separate PCA model for each $I_1 \times I_2$ rectangle sampled from the image, and compare our algorithm with the modified version of the vector subspace-based algorithm, besides the original version of the vector subspace-based algorithm.
- The GMM-based algorithm provides a very good tradeoff between representational power and algorithmic complexity, allowing for good results in real time.

As the vector subspace-based algorithm in Li (2004) is only available for grayscale image sequences, the grayscale videos in Examples 1 and 2 are used to achieve the comparisons between our algorithm and the vector subspace-based algorithms. The GMM-based algorithm is available for both grayscale and color image sequences, so results of the GMM-based algorithm for all the four videos are reported. In the following, the results of the examples are illustrated, and then the analysis of the results is given.

### *6.1.1 Example 1*

In the first example, the subspace dimensions $R_1$, $R_2$, and $R_3$ for the incremental tensor subspace learning algorithm are empirically set to 3, 3, and 10 respectively. The scale factor $\sigma$ in (15) is set to 15. The threshold $T_{gray}$ in (19) is set to 0.8. The learning rate factor $\omega$ in (22) is set to 0.08. The parameters for the vector subspace-based algorithm are chosen to ensure that it performs as accurately as possible. The PCA subspace dimension $p$ is 12; the updating rate $\alpha$ in Li (2004) is 0.96; and the coefficient $\beta$ in Li (2004) is 11. The parameters for the GMM-based algorithm are set as defaults in the OpenCV tool.

Six representative frames 345, 486, 511, 529, 563, and 624 of the foreground segmentation results are shown in Fig. 7, where the first and the sixth rows are from the original sequence, the second and the seventh rows show the results of our algorithm, the third and the eighth rows show the results of the original version of the vector subspace-based algorithm, the fourth and the ninth rows show the results of the modified version of the vector subspace-based algorithm, and the fifth and the tenth rows show the results of the GMM-based algorithm. It can be seen that the segmentation results of our algorithm are clean, connected for each object, and almost noiseless, and furthermore almost all of the associated shadows are omitted. Our algorithm obtains more accurate foreground segmentations than the vector subspace-based algorithms and the GMM-based algorithm.

The frame rate of our algorithm for this example is 1.2 frames per second. The frame rate of the original version of the vector subspace-based algorithm in Li (2004) is 8.3 frames. The frame rate of the modified version of the vector subspace-based algorithm is 2.3 frames. The frame rate of the GMM-based algorithm is 7.6 frames per second. Although our algorithm is slower than the vector subspace-based algorithms and the GMM-based algorithm, the speed of our algorithm is still acceptable.

### *6.1.2 Example 2*

In the second example, the subspace dimensions $R_1$, $R_2$, and $R_3$ for the incremental tensor subspace learning algorithm are empirically assigned as 3, 3, and 12 respectively. The scale factor $\alpha$ in (15) is set to 20. The threshold $T_{gray}$ in (19) is chosen as 0.81. The learning rate factor $\omega$ in (22) is assigned as 0.09. The parameters for the vector subspace-based algorithm are set as follows: the PCA subspace dimension $p$ is 13; the updating rate $\alpha$ is 0.95; and the coefficient $\beta$ is 9. The parameters for the GMM-based algorithm are set as defaults in the OpenCV tool.

Five representative frames 178, 197, 203, 215, and 243 of the foreground segmentation results are displayed in Fig. 8, where the first row is from the original image sequence and the second, the third, the fourth, and the fifth rows correspond to our algorithm, the original version of the vector subspace-based algorithm in Li (2004), the modified version of the vector subspace-based algorithm, and the GMM-based algorithm, respectively. It is shown that our algorithm obtains a much cleaner segmented background than the vector subspace-based algorithms and the GMM-based algorithm, and the foreground segmented by our algorithm is cleaner, better connected for each object, less noisy, and less affected by shadows than the foreground regions segmented by the vector subspace-based algorithms and the GMM-based algorithm.

For this example, the frame rates of our algorithm, the original version of the vector subspace-based algorithm, the modified version of the vector subspace-based algorithm, and the GMM-based algorithm are 5.7, 39.6, 11.2, and 30.9 frames per second, respectively.

### *6.1.3 Example 3*

In the third example, the tensor subspace dimensions ($R_1^r$, $R_2^r$, $R_3^r$), ($R_1^g$, $R_2^g$, $R_3^g$), and ($R_1^s$, $R_2^s$, $R_3^s$) corresponding to the three components $(r, g, s)$ in the color space are empirically assigned as (3, 3, 11), (3, 3, 11), and (3, 3, 10), respectively. The learning rate factors $\omega_r$, $\omega_g$, and $\omega_s$ in (25) are all assigned as 0.08. The scale factors $\sigma_r$, $\sigma_g$, and $\sigma_s$ in (23) are set to 0.12, 0.13, and 16, respectively. The threshold $T_{color}$ in (24) is chosen as 0.79. The parameters for the GMM-based algorithm are set as defaults in the OpenCV tool.

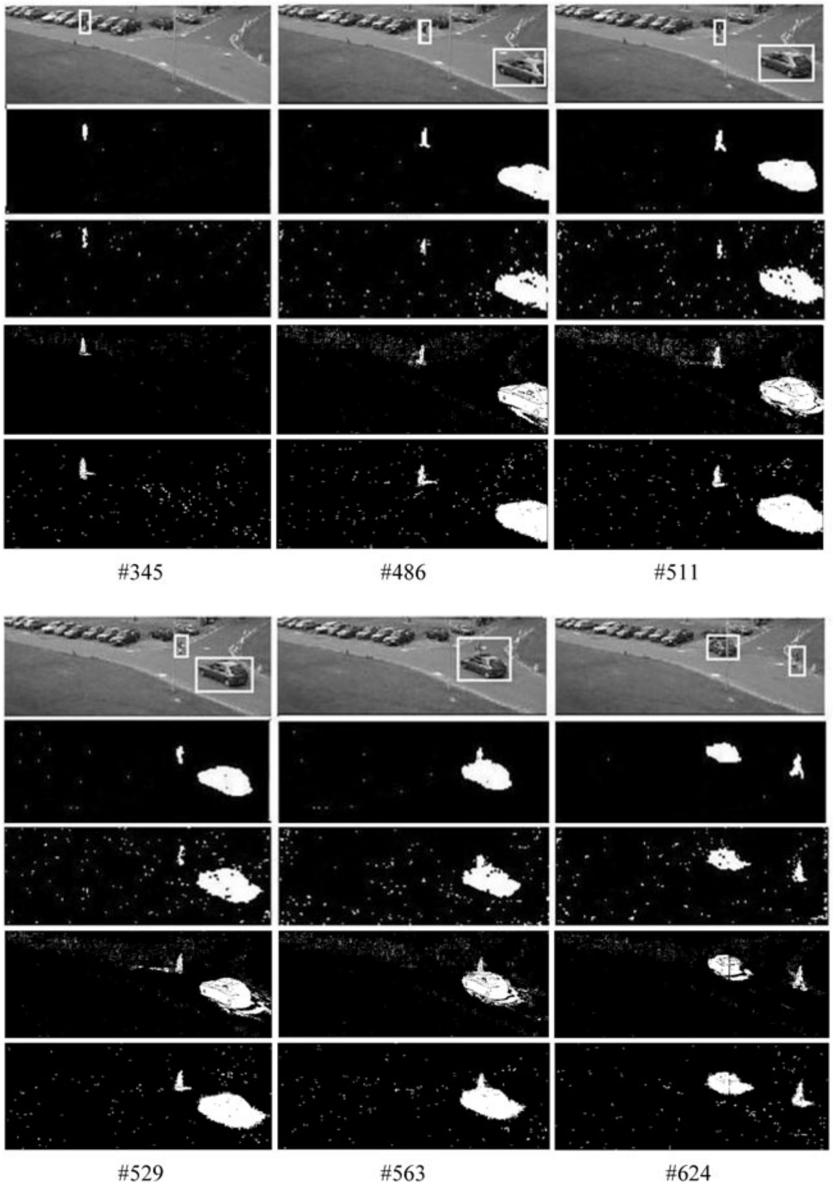

**Fig. 7** Foreground segmentation results for the first video: *Rows* 1 and 6 are from the original sequence where the moving regions are highlighted by white boxes; *Rows* 2 and 7 show the results of our algorithm; *Rows* 3 and 8 show the results of the original version of the vector subspace-based algorithm; *Rows* 4 and 9 show the results of the modified version of the vector subspace-based algorithm; and *Rows* 5 and 10 show the results of the GMM-based algorithm

Five representative frames 52, 69, 79, 83, and 87 of the foreground segmentation results of our algorithm and the GMM-based algorithm are shown in Fig. 9, where Rows 1, 2 and 3 display the original images, the results of our algorithm and the results of the GMM-based algorithm, respectively. The foreground regions are accurately segmented using our algorithm even though their sizes are small; almost all the street lamps are segmented as background, even those which are particularly bright; and our algorithm successively identifies regions corresponding to traffic lights which change over time as background. However, some of these regions are mistakenly identified as foreground by the GMM-based algorithm. In this example, our algorithm more accurately models the color background than the GMM-based algorithm. The frame rates of our algorithm and the GMM-based algorithm are 5.3 and 32.0 frames per second, respectively.

### 6.1.4 Example 4

In the fourth example, the tensor subspace dimensions $(R_1^r, R_2^r, R_3^r)$, $(R_1^g, R_2^g, R_3^g)$, and $(R_1^s, R_2^s, R_3^s)$ corresponding to the three components in the $(r, g, s)$ color space are empirically set to $(3, 3, 9)$, $(3, 3, 9)$, and $(3, 3, 11)$ respectively. The learning rate factors $\omega_r, \omega_g$, and $\omega_s$ in (25) are all set to 0.08. The scale factors $\sigma_r, \sigma_g$, and $\sigma_s$ in (23) are set to 0.11, 0.13, and 20 respectively. The threshold $T_{color}$ in (24) is chosen as 0.78. The parameters for the GMM-based algorithm are set as defaults in the OpenCV tool.

The five representative frames 296, 312, 472, 790, and 814 of the foreground segmentation results are displayed in

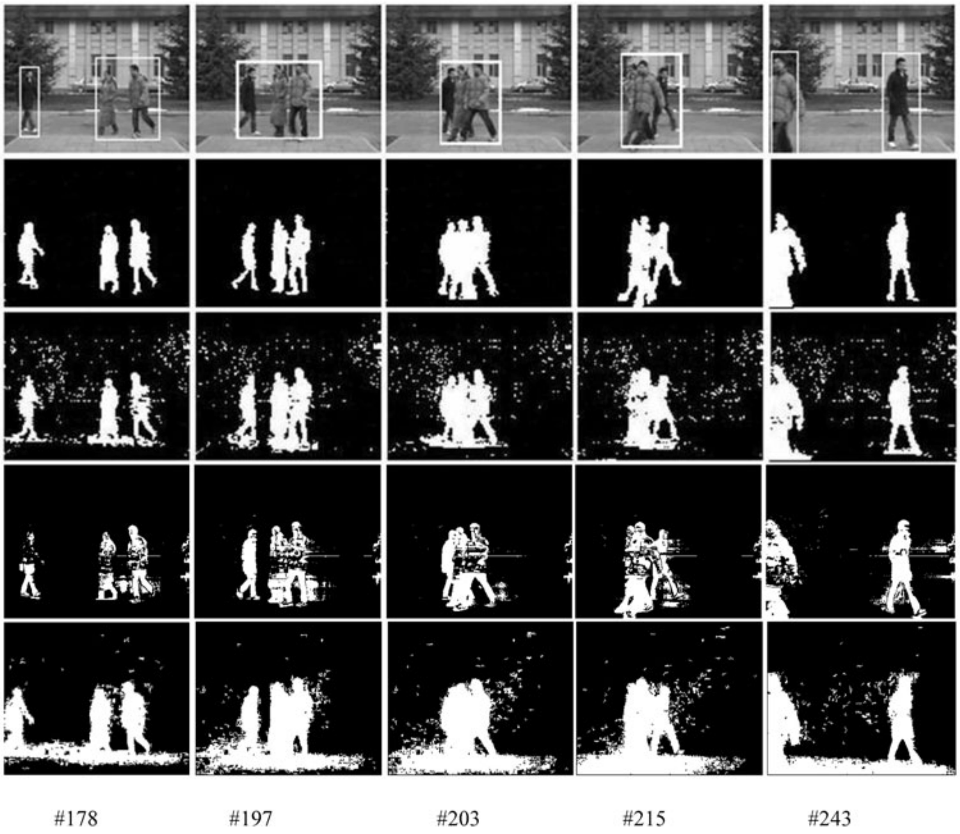

**Fig. 8** Foreground segmentation for the second video: In *row* 1, the moving regions are highlighted by *white boxes*; *Rows* 2, 3, 4, and 5 show the results of our algorithm, the original version of the vector subspace-based algorithm, the modified version of the vector subspace-based algorithm, and the GMM-based algorithm, respectively

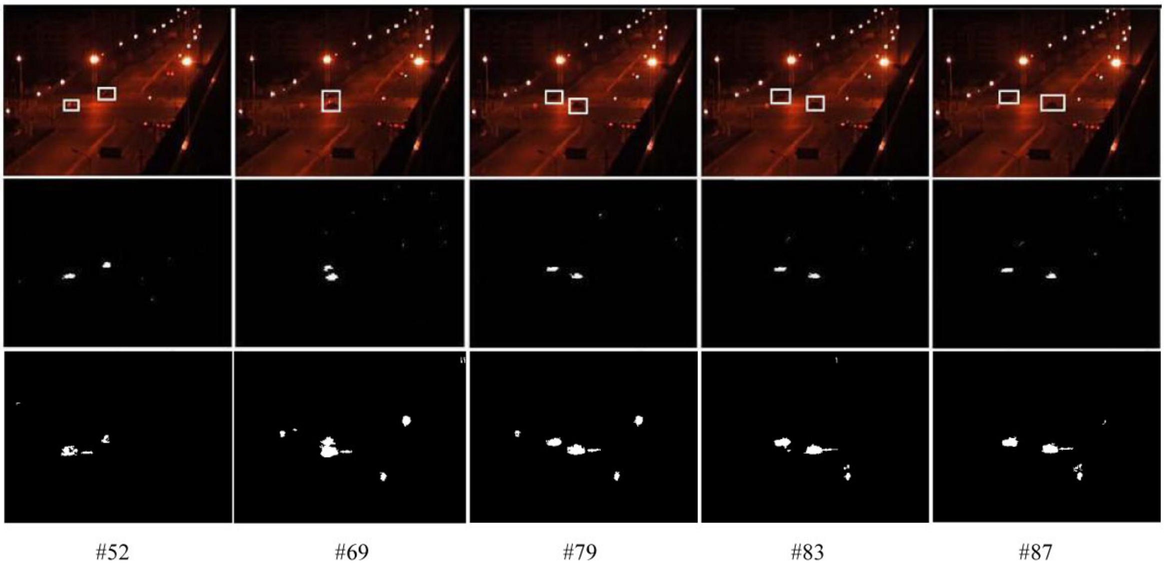

**Fig. 9** The foreground segmentation results for Example 3: In *row* 1, the moving regions are highlighted by *white boxes*; *Row* 2 displays the results of our algorithm; and *Row* 3 corresponds to the results of the GMM-based algorithm

Fig. 10, where Row 1 shows the original images, and Rows 2 and 3 show the results of our algorithm and the GMM-based algorithm respectively. It can be seen that good foreground segmentation results are obtained using our algorithm in that the background is clean and the foreground is connected. In contrast, the GMM-based algorithm does not effectively handle shadows, in that many shadows are mistakenly classified as foreground. The frame rates of our algorithm and the GMM-based algorithm for this example are 5.2 and 28.7 frames per second, respectively.

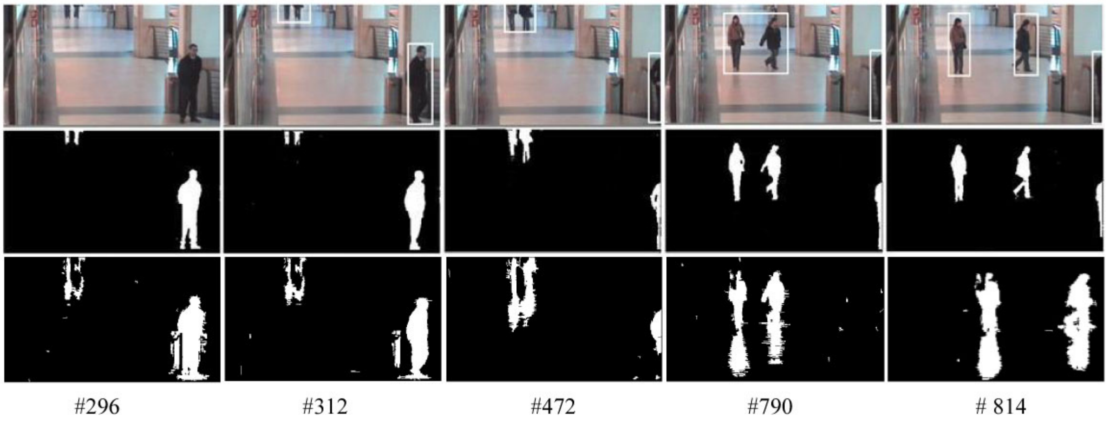

**Fig. 10** The foreground segmentation results for Example 4: In *row* 1, the moving regions are highlighted by white boxes; *Rows* 2 and 3 show the results of our algorithm and the GMM-based algorithm respectively

### *6.1.5 Analysis of Results*

The reason why our algorithm obtains more accurate foreground segmentation results in complicated scenes, as compared with the incremental vector subspace-based algorithms and the GMM-based algorithm, is that our algorithm is able to exploit the spatiotemporal correlations of values of pixels within the image ensembles by the incremental tensor subspace learning. The vector subspace-based algorithm flattens the images into vectors, and as a result loses most of the spatial correlation information. The global or local variations in a scene would substantially change the vector subspace, resulting in foreground segmentation errors. The GMM-based algorithm models all the pixels in an image independently: spatial correlations of the image are not considered.

The reason why our foreground segmentation algorithm is slower than the GMM-based algorithm and the vector subspace-based algorithms is that the GMM-based algorithm directly models individual pixels; the vector-based algorithm uses a PCA model defined over the whole image; the modified vector subspace-based algorithm uses a flattened matrix to describe appearance changes in the rectangular region centered at each pixel; but our algorithm uses a tensor corresponding to three unfolding matrices to describe appearance changes in the rectangular pixel region centered at each pixel. The speed of our algorithm is still acceptable because the size of the rectangular region is small, and the incremental SVD in Ross et al. (2008) is fast.

The frame rate of foreground segmentation is dependent on the image size. For example, the frame rate in Example 1 is much lower than the frame rates in Examples 2, 3, and 4, because the image size in Example 1 is much larger than the image sizes in other examples.

## 6.2 Tracking

To evaluate the performance of the proposed tracking algorithm, five videos are used, one video for each example. The different scenarios include noisy images, scene blurring, objects with small apparent sizes, object pose variations, and occlusions. In the first, fourth, and fifth examples, moving faces are tracked. In the second and third examples, pedestrians are tracked. The face tracking is initialized using the face detection algorithm (Yan et al. 2007). For tracking a pedestrian in the video captured by a stationary camera, the initialization is based on background subtraction. For tracking a pedestrian in the video taken by a mobile camera, the initialization is based on optical flow region analysis (Zhou et al. 2007).

For the tensor subspace representation of object appearance during tracking, object regions obtained at different times are normalized to a size of 20 × 20 pixels using linear interpolation. The forgetting factor $\lambda$ in the incremental SVD is set to 0.99. For the particle filtering in the visual tracking, the number of particles is set to 300. The six diagonal elements $\sigma_x^2, \sigma_y^2, \sigma_\eta^2, \sigma_s^2, \sigma_\beta^2$, and $\sigma_\phi^2$ in the covariance matrix $\Sigma$ in (28) are assigned the values $5^2, 5^2, 0.03^2, 0.03^2, 0.005^2$, and $0.001^2$, respectively. The tensor subspaces are updated every three frames.

In the experiments, we compare our tracking algorithm with three state-of-the-art representative and typical tracking algorithms: the vector subspace learning-based tracking algorithm (Ross et al. 2008), the Riemannian metric-based tracking algorithm (Porikli et al. 2006), and a DTA-based tracking algorithm which is designed for this paper. The competing algorithms are briefly described below:

- The vector subspace learning-based tracking algorithm flattens object appearances in frames in which the object appears into vectors on which online subspace learning is

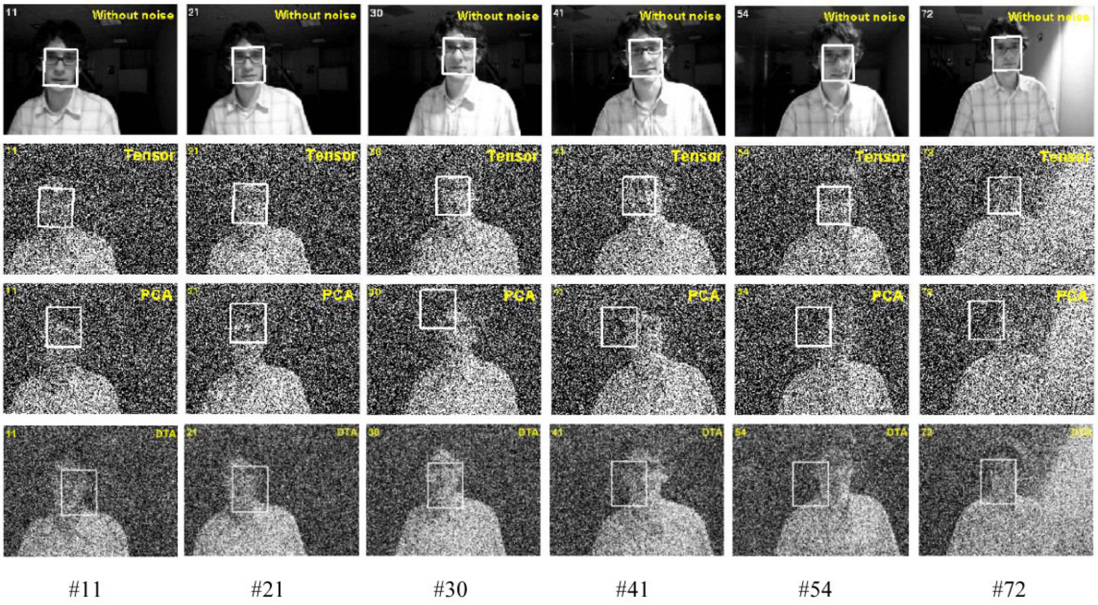

**Fig. 11** Tracking results for Example 1: The *first row* shows the results of our algorithm without added noise; the second, the *third* and the *fourth rows* show, respectively, the results of our algorithm and the vector subspace-based algorithm and the DTA-based algorithm in the presence of high amplitude noise

implemented to obtain a vector subspace-based representation of the object appearance.

- The Riemannian metric-based tracking algorithm (Porikli et al. 2006) uses the covariance matrix of image features for object representation. An affine-invariant Riemannian metric is used in updating of the appearance model.
- According to Sun et al. (2008), DTA is more accurate than STA and WTA. So, we choose to apply DTA to object tracking. DTA itself cannot be applied to tracking directly. We modify DTA by using the row space of the mode-3 unfolding matrix instead of the column space, as required by the tracking application. The values of all parameters such as the forgetting factor, the particle number, and the covariance matrix in (28) are the same as those used in our tensor subspace-based tracking algorithm.

In the following, the results of these five examples are first demonstrated, a quantitative comparison of the results is then made, and analysis of results is finally given.

### *6.2.1 Example 1*

The video for this example is captured in an indoor scene from a mobile camera, available on http://www.cs.toronto.edu/~dross/ivt/. It consists of 100 frames. Each frame is an 8-bit grayscale image. In the video, a man walks in a room, changing his pose and facial expression over time under varying lighting conditions. In order to investigate tracking performance in the presence of high amplitude image noise, we add Gaussian random noise to the video. The process of adding the noise is formulated as: $I'(x, y) = g(I(x, y) + s \cdot Z)$, where $I(x, y)$ is the original pixel value at $(x, y)$, $I'(x, y)$ is the pixel value after the noise is added, $Z$ is a sample from the standard normal distribution $N(0, 1)$, $s$ is a scale factor controlling the noise amplitude, and the function $g(\varphi)$ is defined as:

$$g(\varphi) = \begin{cases} 0 & \varphi < 0 \\ 255 & \varphi > 255 \\ [\varphi] & 0 \le \varphi \le 255 \end{cases} \tag{29}$$

where $[\varphi]$ stands for the floor of the element $\varphi$.

In the experiment, $s$ is set to 200. The subspace dimensions $R_1$, $R_2$, and $R_3$ are empirically assigned as 3, 3, and 5 respectively. For the vector subspace learning-based tracking algorithm, 5 singular vectors are maintained during the tracking, and the remaining singular vectors are discarded at each subspace updating. Six representative frames 11, 21, 30, 41, 54 and 72 of the tracking results are shown in Fig. 11, where the first row shows the tracking results of our algorithm without added noise, and the remaining three rows show, respectively, the tracking results of our algorithm, the vector subspace-based tracking algorithm, and the DTA-based algorithm in the presence of the added noise. The results shown in the first row are used as references for the accuracies of the tracking results shown in the remaining three rows. From Fig. 11, we see that the proposed tracking algorithm is more resistant to strong noise. Two points are made out:

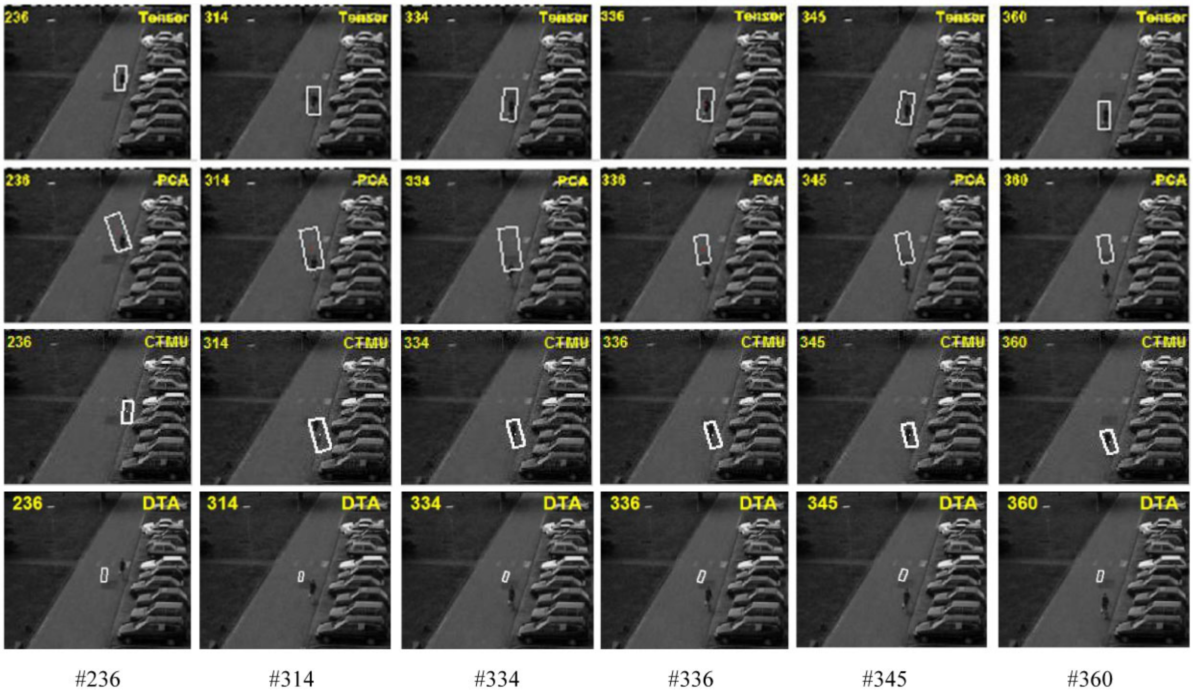

**Fig. 12** Results of tracking a pedestrian with a very small apparent size in a dark scene: *Rows* 1, 2, 3, and 4 correspond to our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm, respectively

- Our algorithm exhibits much more robust results than the vector subspace-based tracking algorithm. The reason for this is that the added noise substantially changes the vector subspace representation of the object appearance in the vector subspace-based tracking algorithm, resulting in tracking errors. In comparison, our tracking algorithm relies on a tensor subspace model which makes use of more spatial information from the three modes, resulting in more accurate results.
- Our algorithm tracks the face of the man more accurately than the DTA-based algorithm. The reason for this is that the tensor subspaces learned using our algorithm are more accurate than the tensor subspaces learned using the DTA-based algorithm.

The frame rates of our algorithm, the vector subspace-based algorithm, and the DTA-based algorithm are 11.2, 38.0, and 8.3 frames per second, respectively. So our algorithm is faster than the DTA-based algorithm, but slower than the vector subspace-based algorithm.

### *6.2.2 Example 2*

The video used in this example is chosen from the open PETS2001 database. It is recorded in an outdoor scene by a stationary camera and it consists of 500 8-bit grayscale images. In this video, a pedestrian with a very small apparent size moves down a road in a dark scene. The motivation of this example is to check the tracking performance in handling image blurring and objects with small apparent sizes.

In the experiment, the tensor subspace dimensions $R_1$, $R_2$, and $R_3$ in our algorithm and the DTA-based algorithm are empirically set to 5, 5, and 8 respectively. For the vector subspace-based tracking algorithm, 16 singular vectors are maintained during the tracking. Six representative frames 236, 314, 334, 336, 345, and 360 of the tracking results are shown in Fig. 12, where the first, the second, the third, and the fourth rows correspond to our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm, respectively. It can be seen that the results of our algorithm are the most accurate. The Riemannian metric-based algorithm tracks more accurately than the vector subspace-based algorithm. The DTA-based algorithm loses the track after Frame 173. This example reflects that our algorithm makes a more compact object appearance representation and thus more efficiently reduces spatiotemporal redundancy of object appearance information particularly for tracking an object with a small apparent size in blurred images.

The frame rates of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm are 10.9, 33.6, 25.6, and 8.6 frames per second, respectively. Our algorithm is faster than the DTA-based algorithm, but slower than the vector subspace-based algorithm and the Riemannian metric-based algorithm.

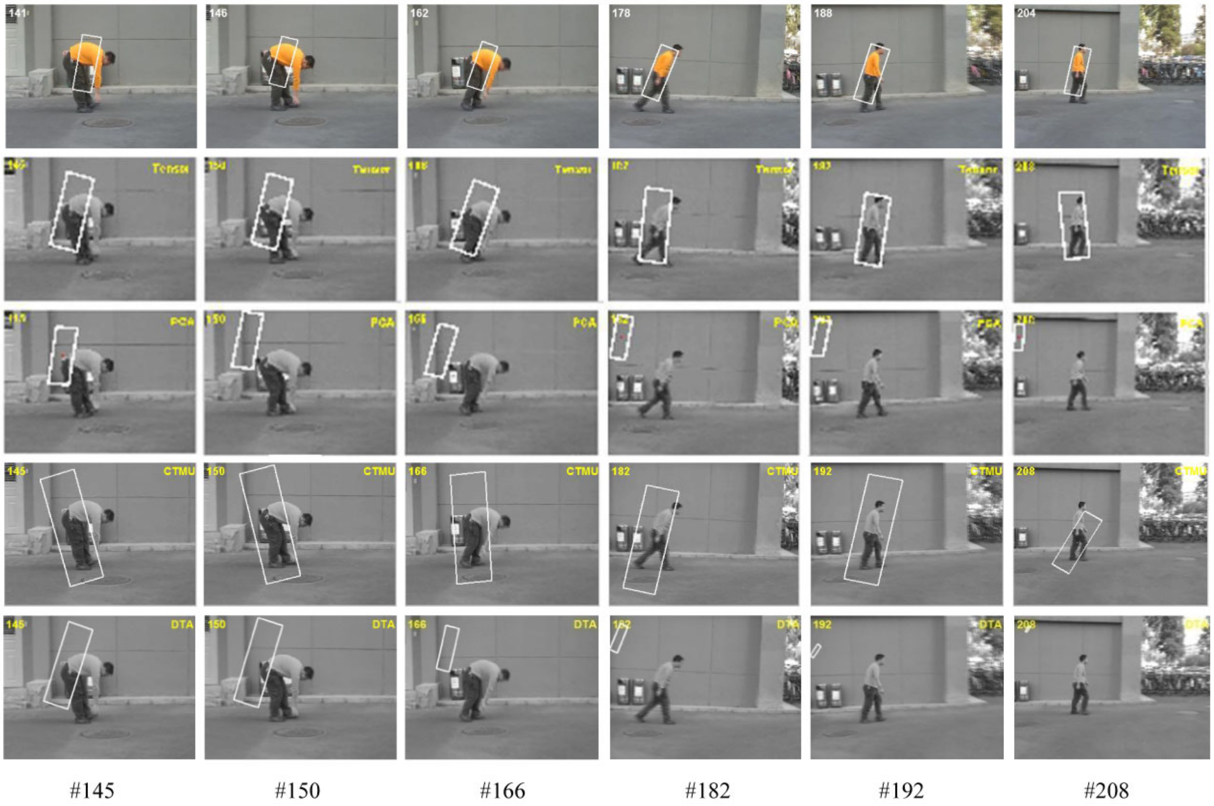

**Fig. 13** Tracking during drastic pose changes: *Rows* 1 and 2 show the results of our algorithm for the color sequence and the grayscale sequence respectively; *Rows* 3, 4, and 5 show, respectively, the results of the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm for the grayscale sequence

### *6.2.3 Example 3*

The video for this example is recorded in an outdoor scene using a mobile camera. It consists of 235 frames. In the video, a man walks from left to right on a well lit road and his body pose varies over time. In the middle of the video, there are drastic motions and pose changes: the man bows down to reach the ground and then stands back up again. The motivation of this example is to make a comparison between our algorithm and the competing algorithms in handling pose variations.

The 8-bit grayscale image sequence and the 24-bit RGB color image sequence are both considered. The tensor subspace dimensions $R_1$, $R_2$ and $R_3$ are empirically assigned as 8, 8, and 10 respectively. For the vector subspace-based algorithm, 16 singular vectors are maintained during the tracking. Six representative frames 145, 150, 166, 182, 192, and 208 of the tracking results are shown in Fig. 13, where Rows 1 and 2 show the results of our algorithm for the color sequence and the grayscale sequence respectively; and Rows 3, 4, and 5 show, respectively, the results of the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm for the grayscale image sequence. It can be seen that in the color sequence and in the grayscale sequence, our algorithm tracks the object successfully even when there are drastic pose and motion changes. The results for the color sequence are slightly more accurate than the results for the grayscale image sequence as more information is available in the color sequence. However, both the vector subspace-based algorithm and the DTA-based algorithm lose the track during and after the drastic pose and motion changes. The results of the Riemannian metric-based algorithm are very inaccurate in many frames. For this example, the frame rates of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm are 10.8, 34.2, 27.6, and 8.3 frames per second, respectively.

### *6.2.4 Example 4*

The video for this example is recorded from an indoor scene by a stationary camera. It consists of dark and motion-blurring grayscale images, and its length is 535 frames. In this video, a man shakes his head, first takes off and then wears his glasses, and sometimes uses his hands to occlude his face. The motivation of this example is to compare the performance of our algorithm with those of the competing algorithms in handling partial occlusions.

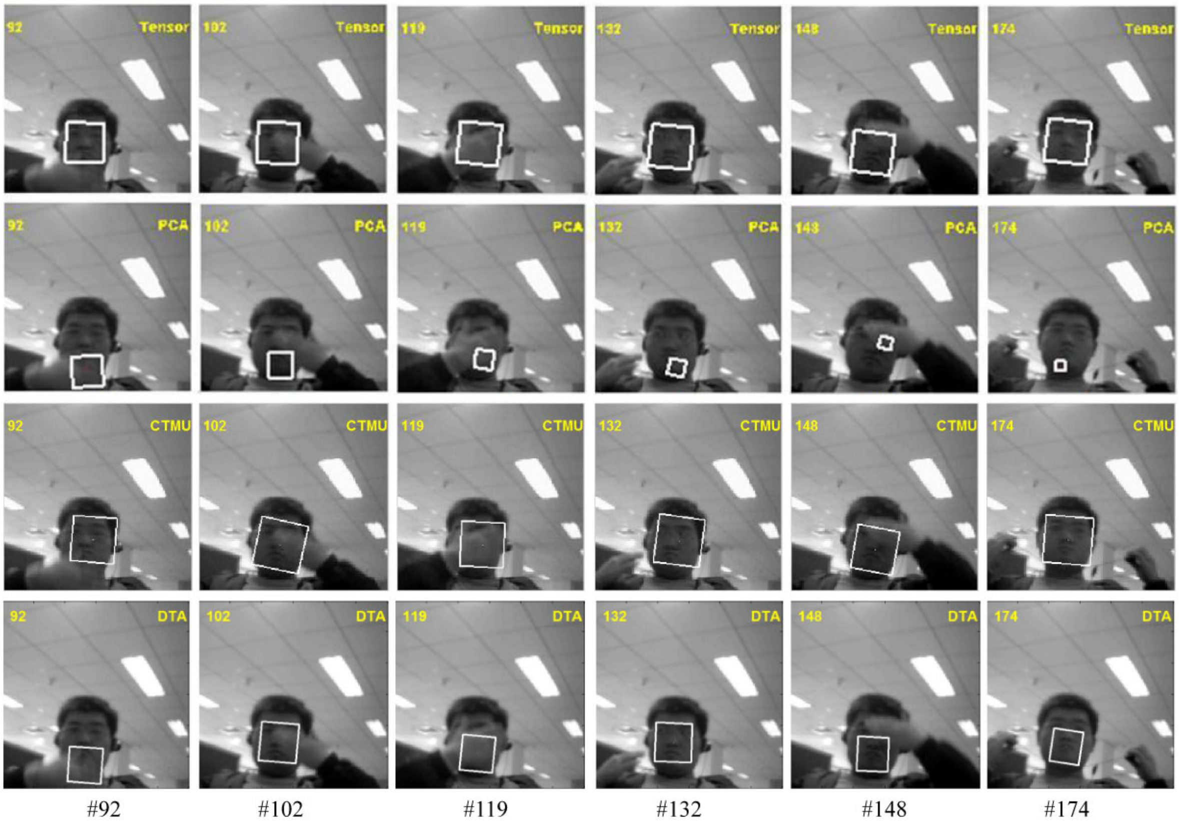

**Fig. 14** Tracking a face during partial occlusions in blurred images: *Rows* 1, 2, 3, and 4 show the results of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm, respectively

In this example, the tensor subspace dimensions $R_1$, $R_2$ and $R_3$ are empirically set to 3, 3, and 5, respectively. For the vector subspace-based algorithm, 10 singular vectors are maintained during the tracking. Six representative frames 92, 102, 119, 132, 148, and 174 of the tracking results are shown in Fig. 14, where rows 1, 2, 3, and 4 correspond to our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm, respectively. From the figure, we see that our algorithm tracks the face accurately under poor lighting conditions even when the face of the man is occluded seriously by the hands from time to time. However, the vector subspace-based algorithm loses track of the face in several frames. The results of the Riemannian metric-based algorithm are acceptable, although they are less accurate than the results of our algorithm. The DTA-based algorithm can complete the tracking all the time but its results are inaccurate, in particular when the face is occluded (see Frames 92, 119, and 148). For this example, the frame rates of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm are 11.2, 40.6, 27.7, and 9.2 frames per second, respectively.

### 6.2.5 Example 5

This example is widely used for testing face tracking algorithms. The video is available on http://www.cs.toronto.edu/vis/projects/dudekfaceSequence.html. It is recorded with a mobile camera, and its length is 573 frames. In this video, a man who sits in a chair changes his pose and facial expression over time and from time to time his hand occludes his face. (There are benchmark points in this example, and the corresponding quantitative results are shown in Sect. 6.2.6.)

In the experiment, 8-bit grayscale images are used. All the tensor subspace dimensions are set to 8. The subspace dimension of the vector subspace-based algorithm is set to 13. Six representative frames 50, 76, 106, 107, 118, and 124 of the tracking results are shown in Fig. 15, where the first, the second, the third, and the fourth rows correspond to our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm, respectively. It can be seen that our algorithm tracks the face accurately even during occlusions and pose changes, while the vector subspace-based algorithm, the Riemannian metric-based algorithm and the DTA-based algorithm loses the track in many frames. For this example, the frame rates of our algorithm, the vector subspace-based

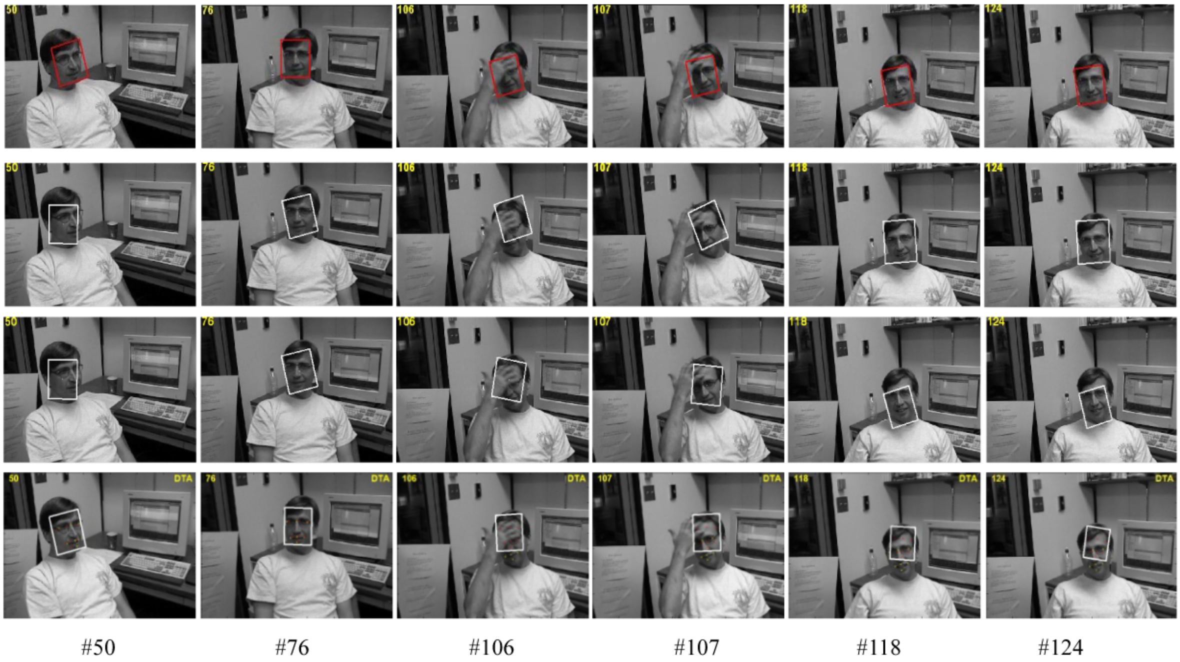

**Fig. 15** Tracking a face under partial occlusions and pose variations: *Rows* 1, 2, 3, and 4 show the results of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm respectively

algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm are 10.7, 26.7, 24.2, and 9.4 frames per second, respectively.

### 6.2.6 *Quantitative Comparisons*

For Examples 1, 2, 3, and 4, there are no existing benchmarks, so we provide a quantitative comparison between our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm by labeling a number of representative frames in each of these examples. The centers of object locations in the representative frames are labeled manually as the ground truth. Then, we quantitatively evaluate the tracking performances by computing the mean location deviation which is the average of the pixel distances between the center of the rectangle which representing the tracking result in each frame and the corresponding center of the ground truth. The less the deviation, the higher the localization accuracy. Table 1 lists the mean localization deviations of the tracking results of our algorithm, the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm in the representative frames in the four examples. From the table, it can be seen that the object localization accuracy of our algorithm is higher than those of the vector subspace-based algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm.

**Table 1** Mean localization deviations of the tracking results

| Algorithms | Examples | | | |
|---|---|---|---|---|
| | Example 1 | Example 2 | Example 3 | Example 4 |
| Our algorithm | 5.1 | 2.5 | 3.3 | 2.5 |
| Vector subspace-based | 31.7 | 28.7 | 77.2 | 28.6 |
| Riemannian metric-based | – | 12.7 | 19.6 | 6.5 |
| DTA-based | 27.3 | 68.3 | 79.2 | 11.3 |

For Example 5, each frame contains seven manually labeled benchmark points, which characterize the location and the shape of the face. These benchmark points are used to evaluate the accuracy of tracking results. During the tracking, the values of the object's affine motion parameters for each frame are used to obtain seven validation points corresponding to the seven benchmark points. In each frame, the deviation which is defined as the average of the pixel distances between each validation point and its corresponding benchmark point is used to quantitatively evaluate the tracking accuracy in this frame. The quantitative comparison results are displayed in Fig. 16, where the x-coordinate is the frame number and the y-coordinate is the tracking deviation. From the figure, we see that the average pixel distances for our tracking algorithm are always lower than those for the vector subspace-based tracking algorithm, the Riemannian metric-based algorithm, and the DTA-based algorithm: our algorithm obtains more accurate results than the competing algorithms.

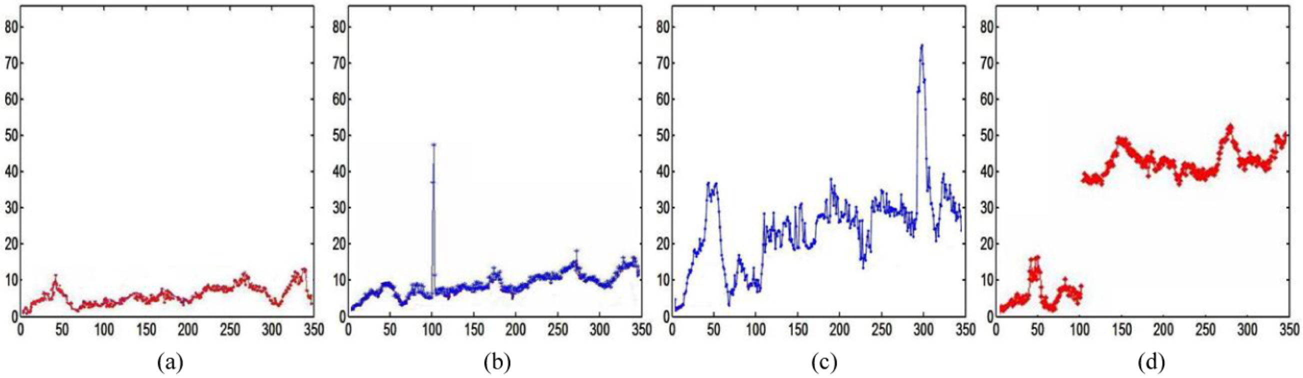

**Fig. 16** The quantitative comparison for Example 5: (**a**) Our algorithm; (**b**) The vector subspace-based algorithm; (**c**) The Riemannian metric-based algorithm; (**d**) The DTA-based algorithm

### *6.2.7 Analysis of Results*

From the aforesaid experimental results, it is shown that our tracking algorithm is able to robustly track objects through changes in appearance, such as noise disturbance, image blurring, objects with small apparent sizes, drastic pose changes, and occlusions. In the experiments, tracker drift (Matthews et al. 2004; Grabner et al. 2008), that is small inaccuracies in localization lead to gradual corruption of the appearance model and loss of track, did not occur for our algorithm. The reasons for robustness of our algorithm to appearance change and tracker drift are as follows: 1) Our algorithm can provide accurate tensor subspaces which record the object or background appearances over time. 2) The tensor representation encodes spatial information as well as the object appearance variations, and thus makes the appearance model discriminative. Even if the subspace information on one mode of the tensor is partially lost or drastically varies, the cues obtained from the subspace information in the other modes of the tensor can recover the subspace information on this mode. As a result, small inaccuracies in the location of the object do not accumulate.

In contrast to our tracking algorithm, the vector subspace-based tracking algorithm only considers information in one mode. If there is a change in the appearance of the object, the vector subspace-based tracking algorithm is more likely to lose the track. The tensor subspaces obtained by the DTA-based algorithm are less accurate than those obtained by our algorithm. The Riemannian metric-based tracking algorithm does not directly model the changes of each pixel in object appearance over time. These problems make these algorithms less robust to appearance changes and tracker drift than our algorithm.

## 7 Conclusion

In this paper, we have developed an incremental tensor subspace learning algorithm based on subspace analysis within a multi-linear framework. The appearance of an object or a scene and the changes in appearance over time are modeled by incrementally learning a low dimensional tensor subspace representation which is updated incrementally as new images arrive. We have applied the proposed incremental tensor subspace learning algorithm to foreground segmentation and object tracking. Our foreground segmentation algorithms for grayscale image sequences or color image sequences capture the intrinsic spatiotemporal characteristics of scenes based on a likelihood function which is constructed on the basis of the learned tensor subspace model. Our tracking algorithm captures the appearance characteristics of an object during tracking, and uses particle filtering to propagate the sample distributions over time. Experimental results show that our proposed algorithms for foreground segmentation and object tracking are robust to noise or low quality images, occlusions, lighting changes, scene blurring, objects with small apparent sizes, and object pose variations. Consequently, our incremental tensor subspace learning algorithm performs effectively in modeling appearance changes of objects or scenes in complex scenarios.

Our future work will focus on the following aspects:

- We will use something like a wavelet basis, which represents the image using coefficients that capture the spatial correlations, and then take wavelet-like features into the incremental subspace learning algorithm which is then applied to background modeling and object tracking.
- We will consider the image a 3D tensor (with color created the third axis) and do the fourth order tensor decomposition to deal with changes in color appearance. Correspondingly, incremental subspace learning for 4-order tensors should be achieved, especially for the applications to background modeling and object tracking.

**Acknowledgements** This work is partly supported by NSFC (Grant No. 60825204, 60935002) and the National 863 High-Tech R&D Program of China (Grant No. 2009AA01Z318).

## References

Black, M. J., & Jepson, A. D. (1998). EigenTracking: robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, *26*(1), 63–84.

Black, M. J., Fleet, D. J., & Yacoob, Y. (1998). A framework for modeling appearance change in image sequence. In *Proc. of IEEE international conference on computer vision* (pp. 660–667), Jan. 1998.

Chen, D., & Yang, J. (2007). Robust object tracking via online dynamic spatial bias appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(12), 2157–2169.

Elgammal, A., Duraiswami, R., Harwood, M., & Davis, L. S. (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, *99*(7), 1151–1163.

Gall, J., Rosenhahn, B., & Seidel, H.-P. (2008). Drift-free tracking of rigid and articulated objects. In *Proc. of IEEE conference on computer vision and pattern recognition* (pp. 1–8), June 2008.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations*. Baltimore: Johns Hopkins University Press.

Grabner, H., Leistner, C., & Bischof, H. (2008). Semi-supervised online boosting for robust tracking. In *Proc. of European conference on computer vision* (pp. 234–247).

Gu, M., & Eisenstat, S. C. (1993). A stable and fast algorithm for updating the singular value decomposition. *Research report YALEU/DCS/RR-966*, Department of Computer Science, Yale University, New Haven, June 1993.

Gu, M., & Eisenstat, S. C. (1995). Downdating the singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, *16*(3), 793–810.

Hager, G. D., & Belhumeur, P. N. (1996). Real-time tracking of image regions with changes in geometry and illumination. In *Proc. of IEEE conference on computer vision and pattern recognition* (pp. 403–410), June 1996.

Han, B., Zhu, Y., Comaniciu, D., & Davis, L. S. (2009). Visual tracking by continuous density propagation in sequential bayesian filtering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 919–930.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 809–830.

He, X., Cai, D., & Niyogi, P. (2005). Tensor subspace analysis. In *Proc. of annual conference on neural information processing systems* Dec. 2005 Cambridge: MIT Press.

Ho, J., Lee, K., Yang, M., & Kriegman, D. (2004). Visual tracking using learned linear subspaces. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 782–789).

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *Proc. of European conference on computer vision* (vol. 2, pp. 343–356).

Jacques, J. C. S. Jr., Jung, C. R., & Musse, S. R. (2006). A background subtraction model adapted to illumination changes. In *Proc. of IEEE international conference on image processing* (pp. 1817–1820), Oct. 2006.

Jepson, A. D., Fleet, D. J., & El-Maraghi, T. F. (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(10), 1296–1311.

Kwon, J., & Lee, K. M. (2009). Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping Monte Carlo sampling. In *Proc. of IEEE conference on computer vision and pattern recognition workshops* (pp. 1208–1215), June 2009.

Lathauwer, L. D., Moor, B. D., & Vandewalle, J. (2000). On the best Rank-1 and Rank-$(R1, R2, \dots, Rn)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications*, *21*(4), 1324–1342.

Lee, K., & Kriegman, D. (2005). Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 852–859).

Levy, A., & Lindenbaum, M. (2000). Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, *9*, 1371–1374.

Li, Y. (2004). On incremental and robust subspace learning. *Pattern Recognition*, *37*(7), 1509–1518.

Li, J., Zhou, S. K., & Chellappa, R. (2005). Appearance modeling under geometric context. In *Proc. of IEEE international conference on computer vision* (vol. 2, pp. 1252–1259).

Li, X., Hu, W. M., Zhang, Z. F., Zhang, X. Q., & Luo, G. (2007). Robust visual tracking based on incremental tensor subspace learning. In *Proc. of IEEE international conference on computer vision* (pp. 1–8), Oct. 2007.

Lim, H., Morariu, V. I., Camps, O. I., & Sznaier, M. (2006). Dynamic appearance modeling for human tracking. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 751–757).

Mahadevan, V., & Vasconcelos, N. (2009). Saliency-based discriminant tracking. In *Proc. of IEEE conference on computer vision and pattern recognition workshops* (pp. 1007–1013), June 2009.

Matthews, I., Ishikawa, T., & Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(4), 810–815.

Nickel, K., & Stiefelhagen, R. (2008). Dynamic integration of generalized cues for person tracking. In *Proc. of European conference on computer vision*, Part IV. Lecture notes in computer science (vol. 5305, pp. 514–526), Oct. 2008.

Nummiaroa, K., Koller-Meierb, E., & Gool, I. V. (2003). An adaptive color-based particle filter. *Image and Vision Computing*, *21*(1), 99–110.

Papadimitriou, S., Sun, J., & Faloutsos, C. (2005). Streaming pattern discovery in multiple timeseries. In *Proc. of international conference on very large data bases* (pp. 697–708).

Patwardhan, K., Morellas, V., & Sapiro, G. (2008). Robust foreground detection in video using pixel layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(4), 746–751.

Perez, P., Hue, C., Vermaak, J., & Gangnet, M. (2002). Color-based probabilistic tracking. In *Proc. of European conference on computer vision*, Part I. Lecture notes in computer science (vol. 2350, pp. 661–675).

Porikli, F., Tuzel, O., & Meer, P. (2006). Covariance tracking using model update based on Lie algebra. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 728–735).

Ramanan, D., Forsyth, D. A., & Zisserman, A. (2007). Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 65–81.

Ross, D. A., Lim, J., Lin, R.-S., & Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, *77*(2), 125–141.

Sheikh, Y., & Shah, M. (2005). Bayesian object detection in dynamic scenes. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 74–79).

Skocaj, D., & Leonardis, A. (2003). Weighted and robust incremental method for subspace learning. In *Proc. of IEEE international conference on computer vision* (vol. 2, pp. 1494–1501), Oct. 2003.

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 2, pp. 246–252).

Sun, J., Tao, D., & Faloutsos, C. (2006a). Beyond streams and graphs: dynamic tensor analysis. In *Proc. of ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 374–383), Aug. 2006.

Sun, J., Papadimitriou, S., & Yu, P. S. (2006b). Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Proc. of international conference on data mining* (pp. 1076–1080), Dec. 2006.

Sun, J., Tao, D., Papadimitriou, S., Yu, P. S., & Faloutsos, C. (2008). Incremental tensor analysis: theory and applications. *ACM Transactions on Knowledge Discovery from Data*, *2*(3), 1–37.

Tian, Y., Lu, M., & Hampapur, A. (2005). Robust and efficient foreground analysis for real-time video surveillance. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 1182–1187).

Vasilescu, M. A. O., & Terzopoulos, D. (2002). Multilinear subspace analysis of image ensembles: TensorFaces. In *Proc. of European conference on computer vision* (pp. 447–460), May 2002.

Vasilescu, M. A. O., & Terzopoulos, D. (2003). Multilinear subspace analysis of image ensembles. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 2, pp. 93–99), June 2003.

Wang, H., & Ahuja, N. (2005). Rank-R approximation of tensors using image-as-matrix representation. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 2, pp. 346–353).

Wang, H., & Ahuja, N. (2008). A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision*, *76*(3), 217–229.

Wang, Y., Tan, T., Loe, K. F., & Wu, J. K. (2005). A probabilistic approach for foreground and shadow segmentation in monocular image sequences. *Pattern Recognition*, *38*(11), 1937–1946.

Wang, Y., Loe, K., & Wu, J. (2006). A dynamic conditional random field model for foreground and shadow segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(2), 279–289.

Wang, H., Yan, S., Huang, T., & Tang, X. (2007). A convergent solution to tensor subspace learning. In *Proc. of international joint conference on artificial intelligence* (pp. 629–634).

Wong, S., Wong, K. K., & Cipolla, R. (2006). Robust appearance-based tracking using a sparse bayesian classifier. In *Proc. of international conference on pattern recognition* (vol. 3, pp. 47–50).

Wu, Y., & Huang, T. S. (2004). Robust visual tracking by integrating multiple cues based on co-inference learning. *International Journal of Computer Vision*, *58*(1), 55–71.

Yan, S., Xu, D., Yang, Q., Zhang, L., Tang, X., & Zhang, H. (2005). Discriminant analysis with tensor representation. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 526–532), June 2005.

Yan, S., Shan, S., Chen, X., Gao, W., & Chen, J. (2007). Matrix-structural learning (MSL) of cascaded classifier from enormous training set. In *Proc. of IEEE conference on computer vision and pattern recognition* (pp. 1–7), June 2007.

Yang, J., Zhang, D., Frangi, A. F., & Yang, J. (2004). Two-dimensional PCA a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(1), 131–137.

Yang, C., Duraiswami, R., & Davis, L. S. (2005). Efficient mean-shift tracking via a new similarity measure. In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 176–183), June 2005.

Yang, M., Fan, Z., Fan, J., & Wu, Y. (2009). Tracking nonstationary visual appearances by data-driven adaptation. *IEEE Transactions on Image Processing*, *18*(7), 1633–1644.

Ye, J. (2005). Generalized low rank approximations of matrices. *Machine Learning*, *61*(1–3), 167–191.

Ye, J., Janardan, R., & Li, Q. (2004a). Two-dimensional linear discriminant nalysis. In *Proc. of neural information processing systems conference* (pp. 1569–1576). Cambridge: MIT Press.

Ye, J., Janardan, R., & Li, Q. (2004b). GPCA: an efficient dimension reduction scheme for image compression and retrieval. In *Proc. of ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 354–363), Aug. 2004.

Yu, T., & Wu, Y. (2006). Differential tracking based on spatial-appearance model (SAM). In *Proc. of IEEE conference on computer vision and pattern recognition* (vol. 1, pp. 720–727), June 2006.

Zhou, S. K., Chellappa, R., & Moghaddam, B. (2004). Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, *13*(11), 1491–1506.

Zhou, X., Hu, W. M., Chen, Y., & Hu, W. (2007). Markov random field modeled level sets method for object tracking with moving cameras. In *Proc. of Asian conference on computer vision*, Part I (pp. 832–842).