Object tracking using learned feature manifolds

Yanwen Guo a,⇑, Ye Chen a, Feng Tang b, Ang Li c, Weitao Luo a, Mingming Liu a

a National Key Lab for Novel Software Technology, Nanjing University, Nanjing 210023, PR China
b Hewlett-Packard Laboratories, Palo Alto, CA 94304, USA
c University of Maryland, College Park, MD 20740, USA

Abstract

Local feature based object tracking approaches have been promising in solving the tracking problems such as occlusions and illumination variations. However, existing approaches typically model feature variations using prototypes, and this discrete representation cannot capture the gradual changing property of local appearance. In this paper, we propose to model each local feature as a feature manifold to characterize the smooth changing behavior of the feature descriptor. The manifold is constructed from a series of transformed images simulating possible variations of the feature being tracked. We propose to build a collection of linear subspaces which approximate the original manifold as a low dimensional representation. This representation is used for object tracking. Object location is located by a feature-to-manifold matching process. Our tracking method can update the manifold status, add new feature manifolds and remove expiring ones adaptively according to object appearance. We show both qualitatively and quantitatively this representation significantly improves the tracking performance under occlusions and appearance variations using standard tracking dataset.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Object tracking is a central problem in computer vision with many applications, such as activity analysis, automated surveillance, traffic monitoring, and human-computer interaction. It is essentially the problem of finding the most likely estimate of the object state given a sequence of observations. Object tracking is challenging because of:

- Complex object appearance. The object may have complicated appearance which is hard to model. Furthermore, it may undergo significant changes due to the pose and scale variations as well as non-rigid object motions.
- Occlusions. The object may be occluded by the background or other moving objects, making it difficult to be localized.
- Complex object motion. This is caused by either the moving pattern of the object or by camera motion accompanied by object motion.

There are two key components in an object tracking algorithm: object representation and dynamics. Object representation tries to model the object as accurately as possible so that the tracking algorithm can accurately describe the complex object appearance.

Object dynamics model how the object appearance evolves over time to be able to handle appearance variations. The two problems are usually coupled together: the object representation should be designed to be easily updated to model appearance variations, while the object dynamics should be able to take advantage of the characteristics of object representation for model update.

Traditional methods for representing the object, such as global histogram based approach in meanshift tracking [1] and PCA subspace based approach in EigenTracking [2], are global approaches which describe the object to be tracked as a whole. Such methods work well in many practical applications, but have several intrinsic limitations. First, it is usually very difficult for a global representation to capture local details and as a result unable to model complex appearances. Second, global representations are not robust to partial occlusion. Once the object is occluded, the whole feature vector of object representation is affected. Third, global representations are hard to update.

Recently, local representations have opened a promising direction to solve these problems by representing an object as a set of local parts or sparse local features. Part-based trackers generally use sets of connected or global visual properties incorporated local parts or components [3–6]. The parts used for object representation are updated during tracking by removing old parts that exhibit signs of drifting and adding new ones for easy accommodation of appearance changes. Feature-based trackers often represent the target by a set of sparse local features such as SIFT [7] and affine invariant point detectors [8] which are often invariant to changes
in rotation, scale, illumination and viewpoint. These approaches first localize the features at a sparse set of distinctive image points by feature detectors. Then the feature vectors, usually named as descriptors, are computed based on the local image statistics centered at these locations. Two major advantages of sparse local features are the invariance to image changes and robustness to occlusions. Existing local feature based approaches typically model how the local features vary using prototypes. However, this discrete representation cannot capture the gradual changing property of local appearance.

In this paper, we propose a local feature based manifold representation for object tracking. The object is represented by a set of sparse local feature manifolds. Each local feature manifold is computed from a series of SIFT feature descriptors [7] that correspond to different appearances of the same object feature under simulated variations of practical situations. To build it, we first detect a set of interest points on the object by the state-of-the-art feature detectors. For each feature point, we transform the image regions surrounding it for simulating real object changes. A feature manifold is thus obtained by exploring the ensemble of descriptors extracted on the transformed image regions. Such a manifold is an informative yet robust representation in that it captures the local appearance variations of a part of the object over time, making the local representation more robust against object changes. The local feature variation is complicated and nonlinear in practice as an example illustrated in Fig. 1 which shows a feature on a walking man. As can be observed, the feature appearance changes dramatically during the move. As a result, the feature manifold is a highly nonlinear appearance manifold. For computational efficiency, we apply incremental principal component analysis to it and yield a collection of linear subspace approximation.

To model geometric relations among local features, the feature manifolds are organized as a feature manifold graph which is used to represent the target object to be tracked. Each local feature manifold describes object appearance details and relationships among them encode object structure. Such geometric relationships are elastic and have the flexibility to handle objects with coherent motion and a certain amount of variations caused by viewpoint changes and articulated motions. An advantage of the feature manifold graph is that locally the manifold graph reinforces the power of feature description and characterizes variations of object appearance by learning a series of descriptors, while globally it encodes object structure with the geometric relations among those manifolds. Such characteristics make it suitable for many vision tasks.

We apply the feature manifold graph to object tracking as an application. With the feature manifold graph representation, the target object is tracked based on graph-based feature-to-manifold tracking. During tracking, features are extracted in a candidate region of the current frame and then matched with the manifold. Object position is located by integrating all matching in the manifold graph. Since features may appear and disappear due to viewpoint changes and occlusions, our dynamic model is designed to be able to add new feature manifolds and remove expiring ones adaptively and dynamically. To the best of our knowledge, this is the first paper that applies manifold learning to local features for object tracking.

The rest of this paper is organized as follows. Section 2 describes the related work on object tracking. We present our feature manifold model in Section 3. Section 4 shows our main tracking paradigm. Experiments and analysis are given in Section 5, and the whole paper is concluded finally.

2. Related work

Object tracking using local features has been explored by previous researchers. In [9], Shi and Tomasi proposed a method to select corner-based features that are most reliable for tracking. Collins et al. developed an algorithm for unsupervised learning of object models as constellations of features, and proposed a discriminative feature tracker [10]. A simultaneous modeling and tracking method is proposed in [11] to learn the object model during tracking. The object features are selected manually and tracked individually. The posterior distribution of appearance and shape is built up incrementally using an exemplar-based approach. In [12], the object is represented as view-dependent object appearance models corresponding to different viewing angles. This collection of acquired models is indexed with respect to the view sphere. These models are matched to each frame to estimate object motion. In [13], the authors proposed a “feature harvesting” approach that has a training phase to learn the object geometry and appearance using a randomized tree classifier. The online tracking then becomes a detection problem using the learned classifier. Liu et al. proposed to jointly track different types of features by representing the objects of interest with the hybrid templates [14]. A generative model is developed to learn the template and to estimate object location and scale.

It is noted that the state-of-the-art local features such as SIFT [7] and SURF [15] have been used for object tracking recently. In [16], an attributed relational feature graph which represents the object using SIFT features with geometric relations is proposed for object tracking. Zhou et al. presented a SIFT based mean shift tracking algorithm [17]. The similarity between two neighboring frames is measured in terms of color and SIFT correspondence by using an expectation–maximization algorithm. In [18], He et al. proposed to represent the object by a set of SURF features of interest. Object motion is estimated in terms of maximum likelihood feature motion observations. In [19], Sun and Liu proposed an object tracking method which is based on the combination of local SIFT description and global PCA representation. The method is constructed in the framework of particle filter. In fact, the changing of feature appearance is smooth and highly nonlinear in nature, which is hard to be modeled using discrete prototypes. In this

Fig. 1. Appearance variations of a feature during tracking. Feature patches of different frames are shown on the left and a feature manifold is visualized on the right.
paper, we propose to model the feature appearance variations as a feature manifold approximated by several linear subspaces. This significantly enhances the distinctiveness of object representation. Avidan’s ensemble tracker [20] trains an ensemble of weak classifiers and combines them into a strong one using AdaBoost to distinguish between the object and background by pixel labeling. To adapt to object appearance changes and maintain temporal coherence, a new weak classifier is trained per frame and the strong classifier is updated dynamically. Appearance modeling is important for object tracking. In [21], the covariance matrices of image features in five modes are used to represent object appearance. Visual tracking is led by Bayesian inference using the learned Log-Euclidean Riemannian subspace. In [22], Hu et al. presented an incremental learning algorithm which represents object appearance with low dimensional tensor subspace. During tracking, the learning algorithm is used to capture appearance of the dynamic object. In [23], Ross et al. presented a tracking method that incrementally learns a low-dimensional subspace representation, efficiently adapting online to changes in appearance of the target. A sampling algorithm with likelihood estimate is used to estimate object locations during tracking. Originated from [24,25], TLD [26] is a real-time algorithm for tracking of unknown objects in video streams. It simultaneously tracks the object, learns its appearance and detects it whenever it appears in the video. In [27], a solution for particle filtering on general graphs is developed. As the applications, a high-order Markov chains based object tracking method and a multi-object graphical interaction models based distributed multiple object tracking method are developed.

Our work is also inspired by the face tracking methods using probabilistic appearance manifolds [28]. The task of tracking and recognition is formulated as a maximum posteriori estimation problem under the manifold representation which models face appearance. Learning image manifolds from collections of local features has been addressed in [29]. A feature embedding representation that preserves local appearance similarity and spatial structure of the features is learned by solving an eigenvalue problem. A difference between the above representation and our feature manifold is that they generally represent the image with a manifold. However, we use the manifold to characterize variations of each individual feature, which is more flexible and able to describe object details.

3. The feature manifold model

In traditional local feature based representations, each local feature is represented as a single feature vector describing local appearance. During tracking, it is matched to features extracted in the new frame. The assumption is that although object appearance may change due to viewpoint and scale changes, the invariance property of local feature descriptors is able to accommodate appearance variations. However, most existing local feature descriptors are only partially invariant to these transformations. For example, SIFT is only invariant to scale changes, brightness changes, and to some extent robust to viewpoint changes. This is illustrated in the results in the performance evaluation of local feature descriptors [30]. As can be observed, the matching performance of SIFT degrades significantly with more significant viewpoint changes. To solve this problem, we propose to synthetically generate additional training samples simulating possible variations of the features under viewpoint and scale changes. This enriched dataset is used to learn the initial feature manifold. It should be noted that the idea of simulating images under different types of transformations has been used in [31] to learn a patch classifier.

To simulate possible viewpoint situations, transformations are applied to the original image. Each transformation involves three atomic transformations: scaling, rotation and shearing. The mixture of these transformations can accurately simulate all possible viewpoint changes. To produce a simulated image \( I' \), a combination of these three transformations is applied to the original image \( I_0 \).

\[
I' = T_{sh} \ast T_r \ast T_{sc} \ast I_0, \tag{1}
\]

where \( T_{sh}, T_r, \) and \( T_{sc} \) mean shearing, rotation, and scaling transformations separately.

We express them using the homogeneous matrices. Specifically, we represent \( T_{sc} \) as

\[
\begin{pmatrix}
a & 0 & 0 \\
0 & a & 0 \\
0 & 0 & 1
\end{pmatrix}, \text{ and set three levels for } a \text{ to } (0.5, 1, 2). \tag{2}
\]

\( T_r \) is expressed as

\[
\begin{pmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{pmatrix}, \text{ and we rotate the image around its center for a lap and set the interval to } 30^\circ. \tag{3}
\]

We note \( T_{sh} \) by \( \begin{pmatrix} 1 & b & 0 \\ d & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \) and set three levels \([-1, 0, 1] \) for both \( b \) and \( d \) to execute the shearing transformation.

In implementation, we use the OpenCV function cvWarpPerspective() to generate the transformed image. This function applies a perspective transformation to an image, and transforms the source image using the specified matrix.

It is noted that the interest object is assumed to be nearly planar or far away from the camera such that the above combined transformations can be applied to it for simulating the possible variations of practical situations. We do not intend to handle large out-of-plane rotations which may result in self-occlusions of the object. Our experiments show that such scheme works well on a broad range of video sequences.

Without loss of generality, we assume that the image is placed with its center at the world origin. The above combined transformations are applied to every point in the image. This yields a series of transformed images. We then first detect the feature points in the original image and use corresponding transformation to find their corresponding locations in each transformed image. Specifically, we use the SIFT detector, Harris-Affine detector, as well as Hessian-Affine detector to detect feature points, and describe the local appearance of each point using SIFT descriptor. For each point detected by SIFT, Harris-Affine, or Hessian-Affine detectors, the scale used in computing SIFT descriptor is determined by the corresponding detector. For each SIFT feature descriptor \( f_i \) in the original image, its corresponding features are different versions of the original one under viewpoint variations. The manifold \( M_i \) is learned from \( M(f_1, f_2, \ldots) \), in which \( f_1, f_2, \ldots \) are simulated feature descriptors of \( f_i \) under different viewpoints.

In general, the appearance manifold of a local feature is highly nonlinear. Many samples are needed to learn an accurate representation. Although learning such a manifold globally would be an ideal solution, this is mostly infeasible in practice especially for tracking which is hard to capture enough training samples. We assume that local linearity property holds everywhere on a global nonlinear manifold. Therefore, a local feature manifold can be approximated by a collection of linear subspaces [32]. Note that the similar idea of modeling image variations due to changing illuminations by low-dimensional linear subspaces has been exploited by [33]. There have been many methods for manifold learning such as LLE [32] and ISOMAP [34], but most original algorithms are unsuitable for tracking which requires incremental learning for updating. Although the incremental versions of such algorithms have been developed in the literature [35,36], we choose to approximate the nonlinear feature manifold with several PCA subspaces due to computational efficiency.
Given a set of feature descriptors obtained through the above simulation process, we first use K-means clustering to cluster them into $K$ clusters. For each cluster we fit a PCA subspace for dimension reduction. We denote the subspaces as $\mathcal{M}_i = \{ c^{i1}, c^{i2}, \ldots, c^{im} \}$ to approximate the feature manifold. Here $c^{ij}$ is the $j$th linear subspace in feature manifold $\mathcal{M}_i$.

4. Object tracking

Our tracking module includes two stages. The first stage is training, in which we generate the initial feature manifold set using features extracted from the first frame of a video. The other stage is tracking which locates the object in each new frame and updates the object feature model dynamically.

In the training stage, SIFT descriptors are extracted for each detected feature point. We note that for some objects, SIFT detector cannot find enough features for tracking, so we employ several complementary feature detectors including Harris-Affine detector and Hessian-Affine detector. The set of features is represented as $\mathcal{F} = \{ f_1, f_2, \ldots, f_n \}$. Then the approach described in the previous section is used to generate samples for feature manifold learning. We use affine transforms to simulate those transformations. With these transformed images, we extracted features on the corresponding positions for every original feature $f_i$ in $\mathcal{F}$. After all the transformed images are processed, we obtain the feature manifolds for every feature. We further approximate each manifold $\mathcal{M}_i$ with a set of PCA linear subspaces $\{ c^{i1}, c^{i2}, \ldots, c^{im} \}$ by using the aforementioned approach.

In the training stage, we maintain two sets. One is the feature manifold set $\mathcal{MS}$ generated in the training stage and updated according to the current object state. The other is a set that contains the features denoted by $\mathcal{CF} = \{ c_{f_1}, c_{f_2}, \ldots, c_{f_n} \}$ which are the candidates for creating new manifolds. When processing a new frame, we do not extract the SIFT features directly on the original frame. Instead, we choose a region centered at the predicted location of the object with the size twice the object size at the previous frame. Instead of extracting the SIFT features directly on the original frame, we extract the features on the corresponding positions for every original feature $f_i$ in $\mathcal{F}$. After all the transformed images are processed, we obtain the feature manifolds for every feature. We further approximate each manifold $\mathcal{M}_i$ with a set of PCA linear subspaces $\{ c^{i1}, c^{i2}, \ldots, c^{im} \}$ by using the aforementioned approach.

In the tracking stage, we maintain two sets. One is the feature manifold set $\mathcal{M}_i$ generated in the training stage and updated according to the current object state. The other is a set that contains the features denoted by $\mathcal{CF} = \{ c_{f_1}, c_{f_2}, \ldots, c_{f_n} \}$ which are the candidates for creating new manifolds. When processing a new frame, we do not extract the SIFT features directly on the original frame. Instead, we choose a region centered at the predicted location of the object with the size twice the object size at the previous frame. Instead of extracting the SIFT features directly on the original frame, we extract the features on the corresponding positions for every original feature $f_i$ in $\mathcal{F}$. After all the transformed images are processed, we obtain the feature manifolds for every feature. We further approximate each manifold $\mathcal{M}_i$ with a set of PCA linear subspaces $\{ c^{i1}, c^{i2}, \ldots, c^{im} \}$ by using the aforementioned approach.

4.1. Object localization

Once the features have been matched to all feature manifolds representing the object, the correspondences are used to estimate object motion between two successive frames. We compute a homography based on the correspondences between the features with a set of linear PCA subspaces, we calculate the distance between the feature descriptor and each subspace. The minimal distance is taken as $d(\mathcal{M}_i, f_i)$.

$$d(\mathcal{M}_i, f_i) = \min_{j} d(c^{ij}, f_i).$$  (6)

To compute $d(c^{ij}, f_i)$, we denote the eigenvectors of a PCA subspace $c^{ij}$ as $U^{ij}$, and the means as $\mu^j$. Then we project $f_i$ onto $c^{ij}$,

$$f_i = U^{ij} \cdot \text{proj}(f_i - \mu^j).$$  (7)

The reconstruction of the feature $f_i$ on the subspace is therefore,

$$\tilde{f}_i = U^{ij} \cdot \text{proj}(f_i - \mu^j).$$  (8)

The distance $d(c^{ij}, f_i)$ between the feature and the subspace is finally formulated as,

$$d(c^{ij}, f_i) = \sqrt{\sum_{k=1}^{n} (f_{ik} - \tilde{f}_{ik})^2}.$$  (9)

with $K_i$ the feature dimension. $K_i$ is set to 128 for SIFT.

Based on the feature-manifold matching discussed above, we can further enhance the matching performance by leveraging the geometric structure of local feature manifolds. The idea is that matching of individual features can be made more reliable by considering the neighborhood features. We organize the feature manifolds as an attributed manifold graph similar to [16]. Feature matching becomes a graph matching problem which can be solved using relaxation label. Thereinto, the Hungarian algorithm is used to enforce one-to-one matching.

After feature-manifold matching, we run manifold update to incorporate the new data into the model. Manifold update includes two parts: one is manifold self-update which handles feature appearance variations, the other is adding newly appeared manifolds and deleting expiring ones which handles pose changes and occlusions.

- **Manifold self-update.** After obtaining the matched feature for $\mathcal{M}_i$ in the current frame, we calculate the error between this matched feature and its reconstructed representation using the original eigenvectors of the matched subspace of $\mathcal{M}_i$. If the error exceeds a threshold, an incremental PCA (IPCA) procedure is used to update the subspace. There have been many IPCA algorithms, for example the algorithm introduced by Skocaj and Leonardis [37] which requires the computation of covariance matrix, and the covariance-free IPCA (CCIPCA) proposed by Weng et al. [38]. We implemented both algorithms, and the results show that IPCA is more accurate, and CCIPCA requires less computation. As the precision of CCIPCA is just slightly lower than IPCA, we utilize CCIPCA for tracking. For more details about CCIPCA, please refer to [37].

- **Adding newly appeared manifolds and deleting expiring ones.** In a time window such as 5 frames, if the frequency of a manifold $\mathcal{M}_i$ that can obtain matched features is lower than a threshold, it will be deleted from $\mathcal{MS}$. If the frequency of a candidate $c_{f_i}$ that can obtain matched features is higher than a threshold, we will generate its manifold representation, add it into $\mathcal{MS}$ and delete this $c_{f_i}$ from $\mathcal{CF}$ meanwhile.
in the current frame and the manifolds on the object in the previous frame. RANSAC is used to obtain the homography matrix. We then transform the box indicating the object in the previous frame using this matrix and thus get a parallelogram. The bounding box of the parallelogram is taken as the box indicating the target object in the current frame.

The above tracking process is summarized below.

**Algorithm 1.** Feature manifold tracking

**Input:** The video sequence and an initial object box.

**Output:** Tracked object positions.

**Training:** For the first frame.
1. Extract the local features of the initial object to form a feature set 
   \( \mathcal{F} = \{f_1, f_2, \ldots, f_n\} \).
2. Transform the 1st frame or more precisely the object region, and generate a feature manifold \( \mathcal{M}_1 \) for each feature \( f_i \).
3. Represent \( \mathcal{M}_1 \) as the PCA linear subspaces \( \{c^{1}, c^{2}, \ldots, c^{m}\} \).
4. Build the feature manifold graph.

**Tracking:** For each of the successive frames.

1. Extract local features \( \mathcal{F} = \{f'_1, f'_2, \ldots, f'_n\} \) in a candidate region centered at the object.
2. Perform feature-to-manifold matching. Find the matched feature \( f'_i \) for each manifold \( \mathcal{M}_i \).
3. Estimate object motion and transform the object box.
4. Update manifold subspaces using IPCA, add new manifolds and remove expiring ones.

5. Experiments and discussion

5.1. Matching performance

We first compare our feature-to-manifold matching with feature-to-feature matching. A standard dataset from the VGG datasets is used for comparison.

**VGG datasets** have six groups of images. Every group includes six images for simulating variations under different imaging conditions, including viewpoint changes, scale changes, illumination variations, and so on. For viewpoint changes, the ground truth homography is provided. We choose this group as the dataset for comparison. We take the Graf dataset as an example. In this dataset, we select the reference image as our original image, and carry out the transformations to simulate possible viewpoints of practical situations. The number of transformation matrices is 324. Then we choose an image from other five images as the query. The original and query images are shown in Fig. 2. The query one has 2806 features. We use the ground truth homography to evaluate our matching performance. The comparison result is shown in Fig. 2 right. From the precision and recall curves, it is obvious that the performance of our feature-to-manifold matching is better than traditional feature-to-feature matching.

5.2. Tracking results

5.2.1. Parameter setting

Several parameters are used in our tracker. For each feature manifold in the current frame, we use a fixed window around its position in the next frame as the search window for matching. In the experiments, we set this window to \( 30 \times 30 \) for most videos. The size of search window for a sequence named Pets09_2 is \( 60 \times 60 \) for accommodating fast object movement. The distance threshold for determining whether or not a feature is matched with a manifold is 0.8 for admission of more feature-to-manifold correspondences. In a time window in the following frames, if the number of matched features for a manifold in the current frame is less than a threshold \( \tau \), this manifold will be removed. For a newly detected feature in the current frame, if the number of matched features in the time window is great than or equal to \( \tau \), we will generate its manifold. We set the time window to 5 frames, and set \( \tau \) to 4 in our implementation. We would like to emphasize that all the parameters, except for the window size set to \( 60 \times 60 \) in the Pets09_2 sequence, were kept constant for our experiments.

5.2.2. Qualitative results

We apply our tracker to a set of very diverse and challenging video sequences that contain background clutters, camera motions, rapid changes of object appearance, illumination changes, in-plane/out-of-plane pose changes, scale changes of the targets and so on. Some of these videos are widely employed by previous tracking approaches for evaluating the performance of their trackers. The resolution and number of frames of each video are listed in Table 1.

In the tracking results, the tracked objects are marked as red boxes. All the results are shown in our project webpage.\(^1\) We visually inspect the tracking results, and classify them as “keep track” or “lose track” as shown in the last column of Table 1. In the following,

\(^1\) [Link to project webpage](http://cs.nju.edu.cn/ywguo/tracking2013/result.html).
Table 1
Video sequences used for experiments. The last column shows whether c not our tracker successfully tracks each sequence by visually inspecting the results without resorting to an accurate measure of the success rate.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Resolution</th>
<th>Number of frames</th>
<th>Keep track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagwoman</td>
<td>720 × 576</td>
<td>96</td>
<td>YES</td>
</tr>
<tr>
<td>Courtyard</td>
<td>720 × 480</td>
<td>143</td>
<td>YES</td>
</tr>
<tr>
<td>Crosswalk_shadow</td>
<td>360 × 240</td>
<td>117</td>
<td>YES</td>
</tr>
<tr>
<td>David</td>
<td>320 × 240</td>
<td>537</td>
<td>YES</td>
</tr>
<tr>
<td>David_indoor</td>
<td>320 × 240</td>
<td>761</td>
<td>YES</td>
</tr>
<tr>
<td>David_outdoor</td>
<td>640 × 480</td>
<td>252</td>
<td>NO</td>
</tr>
<tr>
<td>Face</td>
<td>640 × 480</td>
<td>383</td>
<td>YES</td>
</tr>
<tr>
<td>Faceocc2</td>
<td>320 × 240</td>
<td>819</td>
<td>YES</td>
</tr>
<tr>
<td>Gym</td>
<td>426 × 234</td>
<td>767</td>
<td>YES</td>
</tr>
<tr>
<td>Jumping</td>
<td>352 × 288</td>
<td>313</td>
<td>NO</td>
</tr>
<tr>
<td>Oldman</td>
<td>720 × 576</td>
<td>112</td>
<td>YES</td>
</tr>
<tr>
<td>PET509_2</td>
<td>768 × 576</td>
<td>66</td>
<td>YES</td>
</tr>
<tr>
<td>Security officer</td>
<td>720 × 576</td>
<td>200</td>
<td>YES</td>
</tr>
<tr>
<td>Sylvester</td>
<td>320 × 240</td>
<td>1344</td>
<td>YES</td>
</tr>
<tr>
<td>Uniformwoman</td>
<td>720 × 576</td>
<td>286</td>
<td>YES</td>
</tr>
<tr>
<td>Wallcott</td>
<td>640 × 352</td>
<td>253</td>
<td>YES</td>
</tr>
</tbody>
</table>

Fig. 5 shows the facial tracking result for the David_indoor video sequence by our tracker. David’s face undergoes significant illumination changes and partial occlusion. Furthermore, David changed his posture by moving forward, shaking head, and turning round. Since SIFT is robust to illumination changes, our tracker keeps track when the face area shifts from dark to bright from around frames 200 to 300. From Fig. 5 bottom, although the curve of manifold numbers shows small fluctuations frequently, the number of stable manifolds ensures that our tracker keeps track for this sequence, as is the case for the Sylvester sequence. The result on this sequence also demonstrates that our tracker works well for facial tracking.

Fig. 6 shows another facial tracking result by our tracker. This video is from the Honda/UCSD Video Database, and the resolution is 640 × 480. In this video, the man changes his posture by moving forward, backward and shaking head, and varies facial expression simultaneously. Our algorithm successfully keeps track throughout the whole sequence. Other examples of tracking faces are demonstrated in the results of David and Faceocc2 sequences, among which the latter one undergoes partial facial occlusions occasionally. The David_indoor video sequence is of size 320 × 240 and the number of frames is 761. Besides this video, our tracker successfully tracks the target objects for the long video sequences such as the Sylvester sequence with 1344 frames, the Faceocc2 sequence with 819 frames, the David sequence with 537 frames, and the Gym sequence with 767 frames.

Our feature manifold model assumes that the object undergoes in-plane pose changes in that we use the concatenations of tree atomic transformations applied to each feature to simulate object transformations. However, our tracker can accommodate out-of-plane pose changes of the target objects, for instance the Sylvester, David_indoor, Gym, and Wallcott video sequences. Our tracker successfully keeps track for each of the above videos throughout the whole sequence. Fig. 7 shows the tracking results of ten intermediate frames of the Sylvester sequence. The target object undergoes fast and large out-of-plane pose changes throughout the sequence. We carefully examine each of these videos to see the frames of out-of-plane pose changes. The reason why our tracker can handle out-of-plane pose changes is that the number of feature manifolds on the target object remains relatively stable over time in these frames, although small fluctuations occur frequently. Our online updating scheme can adapt to the appearance changes caused by pose variations.

5.2.2.1. Failure cases. Our tracker is sensitive to motion blur and strong noises, since local features like SIFT cannot be found in the heavily blurred frames, and local feature descriptor in the noisy frames is often unreliable. As shown in Fig. 8, Our tracker fails to locate the target from the 16th frame in the Jumping sequence. Significant blur in the face area of most frames due to fast motion significantly degrades the performance of our tracker. Obviously, in this sequence the number of stable manifolds on the object is fewer than other video sequences our tracker keeps track. Furthermore, the curve visualizing the variation of manifold numbers has an obvious downside at frame 16 where losing track occurs, but on the contrary, the downside of manifold number does not necessarily imply losing track.

Our tracker keeps track of David from the 1st frame to the 124th frame, while loses track at frame 129 in the David_outdoor sequence. The reason is that the object box shrinks heavily in frame 125 when David turns round to face the camera. We have carefully checked the manifold graph representing the target in that frame. The repeated grids on the plaid shirt David wears make matching between the manifolds and features error-prone.

---

**Table 1** Video sequences used for experiments. The last column shows whether our tracker successfully tracks each sequence by visually inspecting the results without resorting to an accurate measure of the success rate.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Resolution</th>
<th>Number of frames</th>
<th>Keep track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagwoman</td>
<td>720 × 576</td>
<td>96</td>
<td>YES</td>
</tr>
<tr>
<td>Courtyard</td>
<td>720 × 480</td>
<td>143</td>
<td>YES</td>
</tr>
<tr>
<td>Crosswalk_shadow</td>
<td>360 × 240</td>
<td>117</td>
<td>YES</td>
</tr>
<tr>
<td>David</td>
<td>320 × 240</td>
<td>537</td>
<td>YES</td>
</tr>
<tr>
<td>David_indoor</td>
<td>320 × 240</td>
<td>761</td>
<td>YES</td>
</tr>
<tr>
<td>David_outdoor</td>
<td>640 × 480</td>
<td>252</td>
<td>NO</td>
</tr>
<tr>
<td>Face</td>
<td>640 × 480</td>
<td>383</td>
<td>YES</td>
</tr>
<tr>
<td>Faceocc2</td>
<td>320 × 240</td>
<td>819</td>
<td>YES</td>
</tr>
<tr>
<td>Gym</td>
<td>426 × 234</td>
<td>767</td>
<td>YES</td>
</tr>
<tr>
<td>Jumping</td>
<td>352 × 288</td>
<td>313</td>
<td>NO</td>
</tr>
<tr>
<td>Oldman</td>
<td>720 × 576</td>
<td>112</td>
<td>YES</td>
</tr>
<tr>
<td>PET509_2</td>
<td>768 × 576</td>
<td>66</td>
<td>YES</td>
</tr>
<tr>
<td>Security officer</td>
<td>720 × 576</td>
<td>200</td>
<td>YES</td>
</tr>
<tr>
<td>Sylvester</td>
<td>320 × 240</td>
<td>1344</td>
<td>YES</td>
</tr>
<tr>
<td>Uniformwoman</td>
<td>720 × 576</td>
<td>286</td>
<td>YES</td>
</tr>
<tr>
<td>Wallcott</td>
<td>640 × 352</td>
<td>253</td>
<td>YES</td>
</tr>
</tbody>
</table>
The errors are accumulated in the successive frames and lead to lose track. Although the tracker rejuvenates at around frame 189, it finally loses track again at frame 237. As shown in the red rectangles of Fig. 9 bottom, we also see obvious downside of manifold numbers near the frames where losing track occurs. We infer that the fluctuations of manifold numbers degrade the accuracy the manifold-based object representation, leading to tracking failure.

Besides, our tracker loses track in the last a few frames of the Security officer sequence, when the officer went upstairs in the

Fig. 3. The Security officer sequence. Up: The frames 1, 50, 74, 147, and 192 are shown. The red boxes indicate object locations tracked by our approach. Bottom: Each point of the curve represents the number of manifolds on the object of the current frame that match with the features in the predicted location on its successive frame. We mark the frames of the top row where they are taken in the bottom plot with red spots. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 4. The Courtyard sequence. The frames 1, 38, 74, 110, and 142 are shown.

Fig. 5. The David_indoor sequence. Up: The frames 1, 181, 336, 457, and 695 are shown. The red boxes are the tracked facial regions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
194th frame. In this frame, his body is completely concluded by the advertising board. Our tracker cannot recover from this situation since he disappeared from the view soon.

5.2.3. Quantitative comparisons

To quantitatively evaluate our feature manifold model-based tracker, we compare our results to those produced by six different
tracking algorithms. They are the meanshift tracker [1], incremental visual tracker (IVT) [23], track-learning-detection method (TLD) [26], Hough-based tracker [5], L1-APG tracker [39], and superpixel tracker [40] which represent state-of-the-art tracking algorithms. For fair evaluation, we use the source codes provided by the websites of authors of these trackers. The parameters of each tracker are carefully selected for best performance. For instance, for IVT ten groups of parameters suggested by the authors are tried. For all the trackers, the objects are initialized at the same position at the first frames. Most above approaches use a bounding box on each of the successive frames to represent the target location, while the Hough-based tracker [5] delivers a non-rigid contour representation by roughly segmenting the object from the background.

We use two criteria, tracking success rate and location error with respect to object center for quantitative evaluations. To compute the success rate, we evaluate whether each tracking result is a success or not by employing the following criterion. Given the tracked bounding box or non-rigid object representation \( R_T \) and the ground truth box \( R_G \), the score is defined as \( \frac{R_T \cap R_G}{R_T} \). We consider the tracking result of one frame as a success when this score is above 0.25, and we report the percentage of successfully tracked frames for every tracker on each video sequence. We define the location error as the distance between object center and ground truth.

The statistical data for each tracker on the videos are given in Tables 2 and 3. Fig. 10 shows the curves of center errors for all the testing video sequences. The horizontal axis represents frame number and vertical axis denotes center error in pixel. The inset image in each sub-figure is the first frame of each sequence with the red box indicating initial object position. Note that, for each sequence, the tracker that fails from the beginning is excluded here. From Table 2, we see that our feature manifold tracker achieves the highest success rate for ten sequences while being close to the top for five sequences of the rest videos, except for the Jumping sequence. From Table 3, our tracker achieves the lowest average error for eight of the sixteen sequences. It should be noted that for each tracker, the average error on each sequence is computed by only considering the tracking errors on those frames that are successfully tracked. Tracking errors obtained after tracking failure are excluded. Looking at the success rate and average center error, we also see that the David_outdoor sequence is the most difficult video to track, not just for our tracker but also for many other trackers. The tracking result on the Wallcott sequence shows that our tracker can successfully track the target with highly articulated human motions and rapid appearance changes. The football player in this sequence changes his poses dramatically and rapidly with the ball. Moreover, the defenders in red shirts occlude him occasionally. These result in the failure of many trackers including IVT and TLD. We have carefully checked the number of manifolds on each video frame during tracking, and found that the number of manifolds that match features on the football player remains stable throughout the sequence. As shown in Tables 2 and 3, our tracker

<table>
<thead>
<tr>
<th>Success rates (%)</th>
<th>Meanshift tracker</th>
<th>IVT</th>
<th>TLD</th>
<th>Hough tracker</th>
<th>L1-APG tracker</th>
<th>Superpixel tracker</th>
<th>Feature manifold tracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagwoman</td>
<td>6</td>
<td>99</td>
<td>9</td>
<td>10</td>
<td>5</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>Courtyard</td>
<td>40</td>
<td>99</td>
<td>100</td>
<td>–</td>
<td>100</td>
<td>29</td>
<td>100</td>
</tr>
<tr>
<td>Crosswalk_shadow</td>
<td>100</td>
<td>25</td>
<td>56</td>
<td>100</td>
<td>51</td>
<td>48</td>
<td>100</td>
</tr>
<tr>
<td>David</td>
<td>100</td>
<td>62</td>
<td>100</td>
<td>96</td>
<td>60</td>
<td>9</td>
<td>98</td>
</tr>
<tr>
<td>David_indoor</td>
<td>91</td>
<td>69</td>
<td>28</td>
<td>76</td>
<td>16</td>
<td>98</td>
<td>81</td>
</tr>
<tr>
<td>David_outdoor</td>
<td>99</td>
<td>100</td>
<td>100</td>
<td>95</td>
<td>99</td>
<td>9</td>
<td>100</td>
</tr>
<tr>
<td>Face</td>
<td>80</td>
<td>100</td>
<td>99</td>
<td>100</td>
<td>50</td>
<td>9</td>
<td>100</td>
</tr>
<tr>
<td>Faceocc2</td>
<td>31</td>
<td>54</td>
<td>95</td>
<td>26</td>
<td>33</td>
<td>44</td>
<td>87</td>
</tr>
<tr>
<td>Gym</td>
<td>42</td>
<td>38</td>
<td>98</td>
<td>64</td>
<td>22</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Oldman</td>
<td>34</td>
<td>90</td>
<td>71</td>
<td>49</td>
<td>59</td>
<td>98</td>
<td>100</td>
</tr>
<tr>
<td>PET509_2</td>
<td>8</td>
<td>14</td>
<td>61</td>
<td>8</td>
<td>14</td>
<td>15</td>
<td>97</td>
</tr>
<tr>
<td>Security Officer</td>
<td>12</td>
<td>26</td>
<td>25</td>
<td>87</td>
<td>22</td>
<td>45</td>
<td>96</td>
</tr>
<tr>
<td>Sylvester</td>
<td>91</td>
<td>46</td>
<td>94</td>
<td>100</td>
<td>45</td>
<td>41</td>
<td>100</td>
</tr>
<tr>
<td>Uniformwoman</td>
<td>92</td>
<td>67</td>
<td>30</td>
<td>15</td>
<td>67</td>
<td>99</td>
<td>98</td>
</tr>
<tr>
<td>Wallcott</td>
<td>100</td>
<td>17</td>
<td>30</td>
<td>83</td>
<td>99</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Table 3
Average center location errors (in pixel). “-” Means that the tracker fails from the beginning. For each tracker, the average error on each sequence is computed by only considering the tracking errors on those frames that are successfully tracked. For each sequence, the lowest average center location error is shown in bold-font.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>Meanshift tracker</th>
<th>IVT</th>
<th>TLD</th>
<th>Hough tracker</th>
<th>L1-APG tracker</th>
<th>Superpixel tracker</th>
<th>Feature manifold tracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagwoman</td>
<td>56</td>
<td>6</td>
<td>50</td>
<td>45</td>
<td>47</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>Courtyard</td>
<td>33</td>
<td>7</td>
<td>7</td>
<td>-</td>
<td>4</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>Crosswalk_shadow</td>
<td>9</td>
<td>45</td>
<td>27</td>
<td>7</td>
<td>31</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>David</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>39</td>
<td>5</td>
</tr>
<tr>
<td>David_indoor</td>
<td>50</td>
<td>11</td>
<td>11</td>
<td>16</td>
<td>23</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>David_outdoor</td>
<td>24</td>
<td>11</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Face</td>
<td>35</td>
<td>14</td>
<td>16</td>
<td>25</td>
<td>32</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>Faceocc2</td>
<td>30</td>
<td>10</td>
<td>18</td>
<td>16</td>
<td>22</td>
<td>47</td>
<td>7</td>
</tr>
<tr>
<td>Gym</td>
<td>36</td>
<td>22</td>
<td>21</td>
<td>33</td>
<td>34</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Jumping</td>
<td>30</td>
<td>34</td>
<td>15</td>
<td>15</td>
<td>39</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>Oldman</td>
<td>27</td>
<td>9</td>
<td>25</td>
<td>36</td>
<td>20</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>PET509_2</td>
<td>41</td>
<td>31</td>
<td>5</td>
<td>38</td>
<td>42</td>
<td>37</td>
<td>6</td>
</tr>
<tr>
<td>Security officer</td>
<td>35</td>
<td>33</td>
<td>29</td>
<td>12</td>
<td>29</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Sylvester</td>
<td>13</td>
<td>18</td>
<td>12</td>
<td>4</td>
<td>29</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>Uniformwoman</td>
<td>17</td>
<td>13</td>
<td>28</td>
<td>60</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Wallcott</td>
<td>6</td>
<td>56</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Fig. 10. Center error plots for all the videos. For each sub-figure, the horizontal axis represents frame number and vertical axis denotes center error in pixel. The inset image in each sub-figure is the first frame of each sequence with the red box indicating initial object position. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
achieves the success rate of 100%, and the average center error across the whole sequence is 4. Our tracker tracks the player more precisely than the meanshift tracking approach. This is attributed to the learned feature manifold representation which can accommodate rapid object changes as well as detail variations. The dancer in Gym sequence undergoes severe deformations and fast appearance changes. Visually inspecting the results, our tracker almost keeps track throughout the whole sequence. From Tables 2 and 3, our tracker achieves the success rate of 87%, and the average center error hits 11 pixels. The dancer undergoes highly articulated motions, we only use a bounding box to approximate it. The tracking errors in some frames such as frames 157–169, 178–185, and 196–233 are thus very high. The tracking results on such frames are classified as “failure” as a result.

Our tracker outperforms other trackers on tracking the pedestrians in the Crosswalk_Sequ and Pets09_2 sequences. In the sequence Pets09_2, many students wander in the campus and the pedestrian we track moves from left to right. The background is cluttered with many distracters some of which look similar to the target. Our tracker keeps track for most frames, and only fails in the 11th frame when the pedestrian passing by the white roadblock. In this frame, the roadblock interferes with the tracker since many manifolds on the object match with the features on the roadblock. As a result, the box indicating the target in this frame deviates from the ground truth trajectory severely. We believe that incorporating the Kalman filter into our tracker will handle this problem.

5.2.4. Computation cost
Our tracker was implemented in C++ on a PC with Intel Core i7-2700 CPU @ 3.5 GHz. Computational complexity of our approach mainly depends on the number of features detected. The computation time runs roughly linearly with the number of feature manifolds. Since processing different feature manifolds are completely independent, we parallelize this process for acceleration. Initializing the tracker at the first frame is time consuming since we need to extract the features from each transformed image, use K-means to cluster the feature descriptors of each feature, and fit PCA subspaces for dimension reduction. Tracking the object in the successive frames however only involves manifold-to-feature matching and manifold update in an incremental manner, and thus runs fast. On average, our tracker needs around 3 s to initialize the tracker, and runs at approximately 2 frames per second afterwards.

6. Conclusions
We have proposed the feature manifold model, a new object representation, which represents the object as local feature manifolds. The feature manifold as the core of this representation is more informative and robust compared with an individual feature. The experiments on challenging video sequences show that the feature manifold model-based object tracker can stably track the dynamic objects. We intend to further explore applications of the feature manifold object representation to other vision tasks such as object recognition in future.

Acknowledgments
The authors would like to thank the reviewers for their constructive comments which helped improve this paper greatly. This work was supported in part by the National Natural Science Foundation of China under Grants 61073098, 61021062, and 61373059, and the National Basic Research Program of China (2010CB327903).

References


