Abstract

We present a particle filtering algorithm for visual tracking, in which the state equations for the object motion evolve on the two-dimensional affine group. We first formulate, in a coordinate-invariant and geometrically meaningful way, particle filtering on the affine group that allows for combined state-covariance estimation. Measurement likelihoods are also calculated from the image covariance descriptors using incremental principal geodesic analysis, a generalization of principal component analysis to curved spaces. Comparative visual tracking studies demonstrate the increased robustness of our tracking algorithm.

KEY WORDS—visual tracking, particle filtering, affine group, Lie group

1. Introduction

Since Isard and Blake (1998) first showed that particle filtering (see, e.g., Gordon et al. (1993), Doucet et al. (2001), Arulampalam et al. (2002) and the references therein for an introduction and survey of recent developments in particle filtering) can be effectively applied to visual tracking, numerous particle filtering-based visual trackers have been proposed. What makes the visual tracking problem challenging, particularly from the point of view of robustness, are sudden and frequent changes in, e.g., pose and facial expressions, lighting, and the background environment (resulting from occlusions, for example).

Not surprisingly much of the recent effort in particle filtering-based tracking has been devoted to attempting to confront these challenges directly. Lee et al. (2005) apply an offline learning algorithm using training data for different viewpoints, whereas Jepson et al. (2003), Zhou et al. (2004), and Wang et al. (2007) estimate the appearance model parameters while adapting to appearance changes in an online manner. Incremental subspace learning methods have also been widely applied: Ross et al. (2008) develop a principal component-based tracker with an online incremental principal component update feature, Li et al. (2007) extend the incremental principal component method to higher-dimensional tensor data, and Mei et al. (2007) propose an incremental kernel as a non-linear counterpart to incremental principal components. The image covariance descriptor introduced by Tuzel et al. (2006) has also been applied with some success to object detection (Tuzel et al. 2007) and tracking (Porikli et al. 2006). Other related notable contributions in particle filtering-based visual tracking include Perez et al. (2004), Rui and Chen (2001), Khan et al. (2004), and Chang and Ansari (2005).

While tracked objects are typically represented by an image region surrounding the object (e.g., an ellipse or rectangle), it has been pointed out recently that information about any shape deformations of the object image region are also highly useful; Zhou et al. (2004) and Lee et al. (2005), for example, argue that such information can lead to better performance in,
e.g., human–robot interaction, video surveillance, and other applications that require simultaneous recognition and tracking. Under mild assumptions, such an image region transformation can be effectively realized as a two-dimensional affine transformation. A number of recent approaches to particle filtering-based visual tracking explicitly make use of the two-dimensional affine transformation as the underlying state space e.g. Zhou et al. (2004), Li et al. (2007), and Ross et al. (2008).

As is well known, the set of two-dimensional affine transformations is not a vector space, but rather a curved space that also has the structure of a Lie group (the two-dimensional affine group \( \text{Aff}(2) \)). Existing tracking algorithms for the most part employ a set of local coordinates to parameterize \( \text{Aff}(2) \) in vector form, and formulate the state and measurement equations as non-linear equations evolving on a vector space. The advantage of doing so is that existing vector space particle filtering algorithms can now be readily applied to the problem at hand.

Under usual circumstances it is not unreasonable to expect that such local coordinate-based tracking methods will work reliably. There are of course many possible choices for local coordinates on \( \text{Aff}(2) \), and the performance of any local coordinate-based tracker will inevitably depend on the particular coordinates used. However, as long as the tracking scenario is reasonably well behaved, and the (typically) additive noise models formulated in the chosen coordinates is locally meaningful, local coordinate-based algorithms should work for the most part.

Realistic visual tracking scenarios, however, often involve extreme changes in the background environment, lighting conditions, and other external disturbances that venture to the limits of the local coordinate’s domain of validity. As local coordinates typically cannot globally parameterize a curved space in a homogeneous fashion (consider, for example, the severe distortion that occurs at the poles in the standard two-dimensional latitudinal–longitudinal maps of the Earth), local coordinate-based trackers that fail to take into account the intrinsic geometry of \( \text{Aff}(2) \) will have at best uneven, and at worst highly unreliable, tracking performance. Partly as a consequence of this failure to take the underlying geometry into account, human intervention in the form of, e.g., adjusting the covariance of noise models and other various filtering parameters, often becomes necessary for reliable performance over a realistic range of operating regimes.

Performing filtering on \( \text{Aff}(2) \) correctly, therefore, requires a geometrical formulation of the state and measurement equations. Chiuso and Soatto (2000), Srivastava (2000), and Srivastava et al. (2002) have investigated particle filtering on Lie groups for the case of linear state dynamics and measurements, with particular focus given to the orthogonal and Euclidean groups. In previous work (Kwon et al. 2007) we have developed a particle filtering algorithm for general non-linear stochastic systems evolving on the Euclidean group, and more general matrix Lie groups, that also allows for simultaneous state and covariance parameter estimation; the latter is essentially a geometric extension of the kernel smoothing with shrinkage method of Liu and West (2001), that correctly accounts for the geometric structure of the Lie group state space and \( P(n) \), the space of symmetric positive-definite matrices.

In this paper we take the general matrix Lie group filtering framework of Kwon et al. (2007) and the affine matrix formulation of visual tracking as our common point of departure, and develop a particle-filtering-based visual tracking algorithm that correctly accounts for the underlying geometry. In addition to working out the details of the general Lie group particle filtering algorithm for the case of the affine group (e.g., formulas for the sample mean on \( \text{Aff}(2) \) and \( P(n) \)), tracking robustness is also enhanced via the following additional features:

- We construct and apply an auto-regressive (AR) state dynamics on the affine group. While past work has employed random walk models for their simplicity (whose disadvantages can be overcome to some extent by using a larger number of particles), we show that an AR state dynamics offers a reasonable compromise between model simplicity and flexibility, and is a highly useful means of improving tracking performance.

- We apply the combined state–covariance estimation framework for Lie group filtering (Kwon et al. 2007) to the visual tracking problem. Experience suggests that setting appropriate covariance values in the state particle propagation is crucial to good tracking, and the combined state–covariance estimation eliminates the need for the user to determine appropriate covariance values \textit{a priori}. The computational efficiency of the state–covariance estimation procedure is further enhanced by use of the log-Euclidean metric (Arsigny et al. 2007), thereby eliminating the optimization procedure required to calculate the intrinsic sample mean of \( P(n) \).

- Measurement likelihoods are calculated by applying principal geodesic analysis (PGA; essentially a generalization of principal component analysis (PCA) to Riemannian manifolds) to the image covariance descriptor \( P(n) \) (Fletcher et al. 2004; Pennec et al. 2006; Fletcher and Joshi 2007).

Our experimental results convincingly show that by drawing together a diverse collection of geometric tools—some of it developed in previous work by the authors, others newly developed as a geometric generalization of existing vector space ideas—tracking robustness can be enhanced considerably. Apart from providing the algorithmic details of our visual tracker, which as we show below involves numerous geometrical subtleties of practical relevance, our main contribution is a convincing demonstration that, when combined in the
right way, geometrical methods, despite their increased computational complexity, can indeed have a significant impact on improving tracking performance.

This paper is organized as follows. We begin in Section 2 with some geometric preliminaries on the affine group, including a discussion of general particle filtering on the affine group, and a demonstration of how purely local coordinate-based methods can break down. Section 3 provides a detailed account of our Aff(2) particle-filtering framework for visual tracking: modeling the state dynamics as an AR process on Aff(2), combined state–covariance estimation, calculating measurement likelihoods with incremental PGA and the log-Euclidean metric on $P(n)$, and formulas for the sample mean closed under the Lie bracket operation.

2. Particle Filtering on the Affine Group

2.1. State Dynamics

The object image region is initially assumed as shown in the left panel of Figure 1. The affine transformation of a point $p = (p_x, p_y)^T \in \mathbb{R}^2$ in the object image region is realized by an invertible linear transformation $G \in \mathbb{R}^{2 \times 2}$, followed by a translation $t = (t_x, t_y)^T \in \mathbb{R}^2$, i.e. $p' = Gp + t$. In homogeneous coordinates, such a transformation can be equivalently represented as

$$[p'] = \begin{bmatrix} G & t \\ 0 & 1 \end{bmatrix} [p],$$

where $G$, $t$, and $p$ are 2D vectors.

The homogeneous matrix representation of the two-dimensional affine transformation $[\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}]$ can be identified as the matrix Lie group of two-dimensional affine transformations, denoted by Aff(2).

Figure 2 illustrates the geometric transformations induced by each basis element $E_i$. A general affine transformation is the result of a combination of the transformation modes shown in Figure 2.

We now consider left-invariant stochastic state equations on the affine group of the form

$$dX = X \cdot A(X) \, dt + X \sum_{i=1}^{m} b_i E_i \, d\omega_i, \quad (3)$$

where $X \in \text{Aff}(2)$ is the state, the map $A : \text{Aff}(2) \to \text{aff}(2)$ is possibly non-linear, the $b_i$ are scalar constants, and the $d\omega_i \in \mathfrak{R}$ denotes independent Wiener process noise. The exponential Euler discretization of (3) leads to

$$X_k = X_{k-1} \cdot \exp \left( A(X_k) \Delta t + \sum_{i=1}^{6} E_i \sqrt{\Delta t} \epsilon_{k,i} \right), \quad (4)$$

where $\epsilon_k = (\epsilon_{k,1}, \ldots, \epsilon_{k,6})^T$ is six-dimensional zero-mean Gaussian noise with a specified covariance matrix $S \in \mathfrak{R}^{6 \times 6}$. The discrete version of the measurement equation is expressed as

$$y_k = g(X_k) + n_k, \quad (5)$$

where $g : X_k \to \mathbb{R}^{N_y}$ is a non-linear function and $n_k$ is zero-mean Gaussian noise with a covariance $R \in \mathbb{R}^{N_y \times N_y}$. 

---

1. Preliminary versions of this work have also been described by Kwon and Park (2008). Also, upon publication of our results, we have become aware that Li et al. (2008) also reports using the log-Euclidean metric on the image covariance descriptor for incremental subspace learning.
minimal geodesics are given by the left and right translations of the one-parameter subgroups of the form $e^{tM}, M \in gl(2), t \in \mathbb{R}$. For our case where the particles are resampled according to their associated weights, all of the affine transformations represented by the particles can be expected to be close to each other.

We therefore approximate the sample mean of $\{G_k^{(1)}, \ldots, G_k^{(N)}\}$, $GL(2)$ components of the resampled particles $X_k^{(*)}$, as

$$\tilde{G}_k = G_{k, \text{max}} \cdot \exp(\tilde{M}_k),$$

where $G_{k, \text{max}}$ denotes the $GL(2)$ part of the particle possessing the greatest weight before resampling. The sample mean of $\{X_k^{(1)}, \ldots, X_k^{(N)}\}$ can then be readily obtained as $\left[\hat{g}_k, \hat{t}_k\right]$, where $\hat{t}_k \in \mathbb{R}^2$ is the arithmetic mean of the translation parts.

### 2.3. Pitfalls of Purely Local Coordinate Formulations

This section illustrates in some detail the pitfalls of purely local coordinate-based particle filtering algorithms that do not properly take into account the geometry of the affine group. Zhou et al. (2004) defined the state $x \in \mathbb{R}^6$ as the direct Euclidean embedding of $Aff(2)$, i.e. $x = (a_1, \ldots, a_6)^T$, where each $a_i$ constitutes an affine transformation matrix of the form $\left[\begin{array}{cc} a_{i1} & a_{i2} \\ a_{i3} & 0 \end{array}\right]$. In Li et al. (2007) and Ross et al. (2008), $G \in GL(2)$ is decomposed via singular value decomposition (SVD) as $G = UDV^T$, where $U$ and $V$ are unitary matrices and $D = \left[\begin{array}{cc} t_1 & 0 \\ 0 & s_2 \end{array}\right]$. Then, as shown by Hartley and Zisserman (2000), $G$ can be represented as $G = (UV^T)VDU^T = \text{Rot}(\phi_1)\text{Rot}(\phi_2)\text{Rot}(\phi_3)\text{Rot}(\phi_4)$, where $\phi_1$ and $\phi_2$ are rotation angles. Ross et al. (2008) and Li et al. (2007) then define the state $x \in \mathbb{R}^6$ as $x = (\phi_1, \phi_2, s_1, t_1, t_2)^T$, where $\alpha = s_1/s_2$.

In the above cases, the state equation is obtained by discretizing a stochastic differential equation of the form

$$dx = f(x,t)dt + F(x,t)dw,$$

where $f: \mathbb{R}^6 \rightarrow \mathbb{R}^6$ and $F: \mathbb{R}^6 \rightarrow \mathbb{R}^{6 \times m}$ are possibly time-varying non-linear functions and $dw$ is $m$-dimensional Wiener process noise. To illustrate the potential pitfalls of applying existing particle filtering algorithms to (discretized versions of) Equation (9) without a proper geometric accounting of the affine group, consider two initial object image regions as shown in Figure 3(a). Observe that each side of the object image region corresponding to $X_0$ is twice as long as the side corresponding to $X_0'$. At time $k-1$, both transformed image regions appear identical as shown in Figure 3(b), and the states
Fig. 3. There are two different states \( X \) and \( X' \) representing the affine transformations of two different initial object image regions as shown in (a). At time \( k - 1 \), both transformed image regions represented by \( X_{k-1} \) and \( X'_{k-1} \) appear the same as shown in (b). Using the Euclidean embedding to represent \( X \) and \( X' \) in vector form as \( x \) and \( x' \), the same perturbation to \( x_{k-1} \) and \( x'_{k-1} \) results in the different degrees of shape deformation as shown in (c). In contrast, the degrees of shape deformation induced by the same perturbation to the different states \( X_{k-1} \) and \( X'_{k-1} \) are as shown in (d).

\[
X_{k-1} = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}, \quad X'_{k-1} = \begin{bmatrix} 2 & 0 & 10 \\ 0 & 0 & 1 \end{bmatrix}.
\]

Using the Euclidean embedding, the states \( X_{k-1} \) and \( X'_{k-1} \) can be represented in vector form as \( x_{k-1} = (1, 0, 0, 1, 10, 10)^T \) and \( x'_{k-1} = (2, 0, 0, 2, 10, 10)^T \), respectively. Assuming the state transition is induced via a random walk for simplicity, \( x_{k} = x_{k-1} + w_{k} \), where \( w_{k} \) is six-dimensional Gaussian noise, the states at time \( k \) perturbed by the same random noise \( w_{k} = (0.2, 0.2, 0.2, 0.2, 2, 2)^T \) become

\[
x_{k} = x_{k-1} + w_{k} = (1.2, 0.2, 0.2, 1.2, 12, 12)^T,
\]

\[
x'_{k} = x'_{k-1} + w_{k} = (2.2, 0.2, 2.2, 2, 12, 12)^T.
\]

Figure 3(c) shows the transformed image regions by each perturbed state \( x_{k} \) by the dashed line and \( x'_{k} \) by the dotted line. It can be verified that the degrees of shape deformation induced by \( x_{k} \) and \( x'_{k} \) are dependent on \( x_{k-1} \) and \( x'_{k-1} \), in spite of the identical level of perturbation \( w_{k} \) applied to both. This phenomenon is clearly undesirable from the perspective of visual tracking.

In contrast, expressing random walk state transitions in terms of our geometric framework, the state equation becomes

\[
X_{k} = X_{k-1} \cdot \exp \left( \sum_{i=1}^{6} E_{i} \sqrt{\Delta t} \epsilon_{k,i} \right).
\]
Figure 3(d) shows the transformed image regions for each state perturbed by the same random noise $\epsilon_{k,i}$, each corresponding to the same transformation induced by $x_k$ ($X_k$ by the dashed line and $X_k'$ by the dotted line). We can see that for the same state perturbation, the shapes of the transformed image regions are exactly the same for both $X_{k-1}$ and $X_{k-1}'$. The degree of shape deformation induced by a specific perturbation is always the same regardless of the state.

In Figure 3(d), it is worth noting that the locations of the transformed image regions by $X_k$ and $X_k'$ are different, in contrast to their identically transformed shapes. This can be understood by simply examining the result of multiplying two affine transformations $Q_1$ and $Q_2$:

$$Q_1 Q_2 = \begin{bmatrix} G_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} G_2 & t_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} G_1 G_2 & G_1 t_2 + t_1 \\ 0 & 1 \end{bmatrix}.$$  \hspace{1cm} (14)

Considering $Q_1$ and $Q_2$ to be respectively $X_{k-1}$ and $\exp\left(\sum_{i=1}^{\Delta T} E_i \sqrt{\Delta t} \epsilon_{k,i}\right)$ in (13), it can be recognized that the translation of the transformed image region by $X_k$ depends on $X_{k-1}$, even for the same state perturbation.

We now argue that this phenomenon is in fact an advantage of our geometric framework. Figure 4 physically illustrates the situation discussed so far. Initially, two cubes of different sizes are placed at the same three-dimensional position. Here $X_{k-1}'$ represents the affine transformation of the image region of the small cube, which is moved half the distance toward the camera from its initial three-dimensional position. Similarly, $X_{k-1}$ represents the affine transformation of the image region of the large cube, which is twice the size of the small cube. The large cube is moved from its initial three-dimensional position while maintaining the same distance to the camera, and its image region (represented by $X_{k-1}$) is the same as that for the small cube (represented by $X_{k-1}'$).

In this scenario, the perturbation to $X_{k-1}$ and $X_{k-1}'$ can be physically understood as three-dimensional object rotation and translation over some fixed time interval. Since the small cube is twice as close to the camera as the large cube, the image translation induced by the three-dimensional translation of the small cube is twice as great as that induced by the same three-dimensional translation of the large cube. In contrast, the degrees of shape deformation induced by the same three-dimensional rotations of the two cubes are necessarily same. The example shown in Figure 3(d) exactly matches this physical interpretation. From this perspective, our geometric framework can be considered as a natural representation of image transformations induced by object movement in the three-dimensional space.

For the SVD-based local coordinate representation, the same problem occurring in the Euclidean embedding case is also inevitable (with the possible exception of the Rot($\phi_1$) being factored out to represent the object rotation). Another drawback of the SVD-based approach is that the decomposition is not unique. For example, suppose

$$x_k = \left(\begin{array}{c} \frac{\pi}{8} \\ \frac{\pi}{12} \end{array} 1.2, 0.8, 10, 10\right)^\top.$$  

The affine transformation matrix represented by $x_k$ becomes

$$\begin{bmatrix} 1.1168 & -0.3106 & 10 \\ 0.2485 & 0.9273 & 10 \\ 0 & 0 & 1 \end{bmatrix}.$$  \hspace{1cm} (15)

Conversely, if we represent the above matrix as $x_k$ via SVD, the result is

$$x_k = \left(\begin{array}{c} \frac{\pi}{8} \\ \frac{-5\pi}{12} \\ 0.96, 1.25, 10, 10\end{array}\right)^\top.$$  

We can see that $\phi_2$ are changed to $\phi_2 - \pi/2$ (with appropriate changes made to $s_1$ and $s_2$). This is triggered by the coordinate change to transform the unitary matrices obtained via SVD into valid rotation matrices. As a result tracking performance may be diminished, since particles perturbed by different amounts of random noise may represent the same affine transformation.

3. Visual Tracking Framework

We now apply the affine particle filtering framework described in the previous section to the visual tracking problem. Particular focus will be given to features that are specific to visual tracking, including the choice of an AR process on Aff(2) for the state dynamics, combined state–covariance estimation on Aff(2) and $P(\alpha)$, and calculating measurement likelihoods using PGA on the image covariance descriptors.
3.1. AR State Dynamics

Tracking performance depends to a large extent on the accuracy of the state dynamics model. The simplest state dynamics model is a random walk. Within our framework, a random walk model implies that the drift \( A(X, t) \) in (4) is set to zero as (13). While the random walk model has been shown to be effective for a surprisingly large class of simple and well-behaved application scenarios, tracking performance can be further enhanced if the particle propagation is guided by the appropriate state dynamics beyond the random walk.

Balancing model flexibility with simplicity, in our framework we choose to model the state dynamics via a first-order AR process. The standard vector space formulation of an AR process of order \( p \) can be expressed as

\[
x_k = \sum_{i=1}^{p} a_i x_{k-i} + w_k,
\]

where \( a_i \) are the AR process parameters. For our case where the state space is \( \text{Aff}(2) \), the AR process must be defined in a different way that respects the geometry of \( \text{Aff}(2) \). The definition of AR processes on general Riemannian manifolds has been addressed by Xavier and Manton (2006); extending this formulation to the affine group (and to general matrix Lie groups in a straightforward way), the state transition equation is expressed as a first-order AR process on \( \text{Aff}(2) \) of the form

\[
X_k = X_{k-1} \cdot \exp \left( A_{k-1} + \sum_{i=1}^{6} E_i \sqrt{\Delta t} \epsilon_{k,i} \right),
\]

\[
A_{k-1} = a \log(X_{k-2}^{-1} X_{k-1}),
\]

where \( a \) is the AR process parameter. For typical tracking applications (where video frame rates will typically range between 30 and 60 frames per second, and object speeds are not excessively fast), it can be assumed that \( X_{k-2}^{-1} X_{k-1} \) is close to the identity. Thus, \( \log(X_{k-2}^{-1} X_{k-1}) \) can be defined uniquely. Note that it is not possible to compute \( A_{k-1} \) immediately from each particle at \( k - 1 \) because the particles are resampled at every time step. Therefore, in practice, the state is augmented as \( \{X_k, A_k\} \).

3.2. Tracking with Unknown Covariance

Visual tracking performance also depends to a great extent on the value of the covariance \( S \) related with the Wiener process in (17). However, in general, it is difficult to select appropriate values of \( S \) a priori. Instead of trying to find the appropriate value of \( S \), we adopt the approach described by Kwon et al. (2007), further extending the combined state-parameter estimation method of Liu and West (2001) to \( P(n) \), the space of \( n \times n \) symmetric positive-definite matrices (recall that covariance matrices are elements of \( P(n) \)).

If there are unknown parameters \( \theta \) in the state equation, the state \( X_k \) can be augmented with \( \theta_k \) as \( \{X_k, \theta_k\} \) to estimate \( \theta_k \) simultaneously with \( X_k \). To solve the degeneracy problem owing to the non-existence of the dynamics for the parameters, it is commonplace to add an artificial dynamics to the parameter particles \( \theta_k^{(i)} \) as \( \theta_k^{(i)} \sim N(\theta_k^{(i-1)}, \Sigma_{\theta_k}^{(i-1)}) \) with the covariance \( \Sigma_{\theta_k}^{(i-1)} \) decreasing with time. In this case, the “over-dispersion” or “loss of information” (Liu and West 2001) occurs since the covariance of the parameter particles are increased owing to the artificial noise. To make the covariance of the parameter particles unchanged even with the artificial noise, Liu and West (2001) have proposed kernel smoothing with shrinkage, where each parameter particle \( \theta_k^{(i)} \) is randomly sampled from a normal kernel of the form \( N(\theta_k^{(i-1)} + (1-a)\delta_k^{(i-1)}, h^2 \Sigma_{\theta_k}^{(i-1)}) \) (here \( \delta_k^{(i-1)} \) and \( \Sigma_{\theta_k}^{(i-1)} \) are respectively the mean and covariance of \( \delta_k^{(i-1)} \); by setting \( a = \sqrt{1-h^2} \), the “over-dispersion” is trivially corrected. A discount factor \( \delta \) with typical values between 0.95 and 0.99 is used as the control factor, with \( a = (3\delta-1)/2\delta \) (Liu and West 2001).

Extending the combined state-parameter estimation approach of Liu and West (2001) to arbitrary covariances \( S \) requires methods for finding, on \( P(n) \), minimal geodesics between two arbitrary elements of \( P(n) \), the sample mean and sample covariance, and how to perform Gaussian random sampling on \( P(n) \). In this regard the log-Euclidean metric recently proposed by Arsigny et al. (2007) offers a computationally efficient way to implement the combined state–covariance estimation.

3.2.1. Log-Euclidean Metric on \( P(n) \)

Since \( P(n) \) is convex, the arithmetic mean of \( P(n) \) samples always lies within \( P(n) \). However, the arithmetic mean of \( P(n) \) has a number of undesirable properties, e.g., singular values are not preserved (Pennec et al. 2006; Fletcher and Joshi 2007). Thus, we focus on the intrinsic mean for \( P(n) \) samples \( \{P_1, \ldots, P_N\} \), defined as

\[
\arg \min_{\bar{P} \in P(n)} \sum_{i=1}^{N} d(\bar{P}, P_i)^2,
\]

where \( d(\cdot, \cdot) \) represents a geodesic distance between two \( P(n) \) elements.

To compute the geodesic distance, an appropriate metric for \( P(n) \) should be first identified. The affine-invariant metric invariant under the \( GL(n) \) group action proposed by Pennec et al. (2006) and Fletcher and Joshi (2007) can be regarded as a natural choice, but requires an optimization procedure to compute the sample mean on \( P(n) \). The log-Euclidean metric of Arsigny et al. (2007) preserves much of the natural properties
of the affine-invariant metric while being computationally trivial. Its basic formula is straightforward: the distance between $P_1$ and $P_2$ is given by

$$d(P_1, P_2) = \| \log(P_2) - \log(P_1) \|,$$  

(20)

where $\| \cdot \|$ represents the standard Euclidean vector norm. The geodesic between $P_1$ and $P_2$ is simply expressed as

$$\gamma(t) = \exp((1 - t) \log(P_1) + t \log(P_2)).$$  

(21)

With the log-Euclidean distance equation (20), the log-Euclidean mean of $P(n)$ elements $\{P_1, \ldots, P_N\}$ is obtained in closed form as

$$\bar{P} = \exp \left( \frac{1}{N} \sum_{i=1}^{N} \log(P_i) \right).$$  

(22)

Then the sample covariance $\Sigma_{\bar{P}}$ can be calculated as

$$\Sigma_{\bar{P}} = \frac{1}{N-1} \sum_{i=1}^{N} (Z_i - \bar{Z})(Z_i - \bar{Z})^T,$$  

(23)

where $Z_i = \log(P_i) - \log(\bar{P})$ in column vector form.

3.2.2. Kernel Smoothing with Shrinkage for Covariance Estimation

Kernel smoothing with shrinkage as proposed by Liu and West (2001) can now be performed for covariance parameter estimation. First, the sample mean $\bar{\theta}_{k-1}$ and covariance $\Sigma_{\bar{\theta}_{k-1}}$ of the covariance particles $\theta^{(i)}_{k-1}$ can be easily calculated from (22) and (23). The kernel mean $a \theta^{(i)}_{k-1} + (1 - a) \bar{\theta}_{k-1}$ can be understood as the corresponding point on the geodesic connecting $\theta_{k-1}$ and $\bar{\theta}_{k-1}$ with $\gamma(0) = \theta_{k-1}$ and $\gamma(1) = \bar{\theta}_{k-1}$. With the log-Euclidean geodesic equation (21), $a \theta^{(i)}_{k-1} + (1 - a) \bar{\theta}_{k-1}$ corresponds to $\exp(a \log(\theta^{(i)}_{k-1}) + (1 - a) \log(\bar{\theta}_{k-1}))$. Finally, Gaussian random sampling with the mean $\mu$ and covariance $\Sigma$ can be realized as follows: (i) perform Cholesky decomposition on the covariance $\Sigma$ as $\Sigma = CC^T$, (ii) generate a zero-mean, unit-variance Gaussian random vector $Z \in \mathbb{R}^{n(n+1)/2}$ and form $Z = CZ$, and (iii) compute $\exp(\log(\mu) + [Z])$ where $[Z]$ denotes $Z$ reshaped as an element of Sym($n$), the space of symmetric matrices.

3.3. Robust Measurement Likelihood Calculation

3.3.1. Image Covariance Descriptor for Object Image Representation

Our visual tracking framework assumes that the tracking is initiated when the object is detected automatically, and its rectangular image template from the initial frame is given. To represent the object image template, we use the image covariance descriptor recently proposed by Tuzel et al. (2006), which has been applied with some success to both object tracking (Porikli et al. 2006) and detection (Tuzel et al. 2007).

The object image template is first transformed to an $W \times H$ image patch, and then normalized to have unit variance and zero mean (to minimize the effect of illumination changes); to every pixel in the normalized $W \times H$ image patch $T_{X_0}$, a feature vector $f_i \in \mathbb{R}^{10}$ is assigned as

$$f_i = \left( p_x, p_y, pr, pH, I, |I_{px}|, |I_{py}|, \tan^{-1} \left( \frac{I_{px}}{I_{py}} \right), |I_{px, py}|, |I_{px, py}| \right)^T,$$  

(24)

where $(p_x, p_y)^T$ and $(pr, pH)^T$ respectively represent the Cartesian and polar coordinates of each pixel, $I$ denotes the pixel intensity, and $I_{px}, I_{py, px}, I_{py, py}$ are the first- and second-order image derivatives with respect to the Cartesian coordinates. The image covariance descriptor $C_{X_0} \in \mathbb{R}^{10 \times 10}$ for the object template is then given by

$$C_{X_0} = \frac{1}{WH - 1} \sum_{i=1}^{WH} (f_i - \bar{f})(f_i - \bar{f})^T,$$  

(25)

where $\bar{f}$ is the mean value of $f_i$.

By representing the object template in the form of the image covariance descriptor rather than the raw image has the following advantages: (i) not only the intensity information but the spatial information can be considered simultaneously; (ii) multiple features beyond the image intensity, e.g., color information, edge-like information embedded in the image derivatives, images from different modalities such as an infrared image, can be easily fused; (iii) the dimension of the image covariance descriptor is quite low compared with the raw image; (iv) the effect of outlier pixels can be smoothed out during the covariance calculation.

During tracking, as depicted in Figure 5 the image covariance descriptor $C_{X_k}$ is also calculated from $T_{X_k}$, the normal-
3.3.2. PGA-based Measurement Likelihood

As tracking proceeds, the object image covariance descriptors \(C_{\hat{x}_k}\) corresponding to the estimated state \(\hat{x}_k\) can be collected. The object appearance generally undergoes a gradual change, induced by various internal and external causes such as changes in the object pose, facial expressions, and lighting conditions. For purposes of robust measurement likelihood calculation, it would be helpful to use in some appropriate fashion the image covariance descriptors \(\{C_{\hat{x}_1}, C_{\hat{x}_2}, \ldots\}\) collected up to the present time. One possible way is to use subspace methods such as PCA, which has been used with wide success in various recognition and tracking applications. In our case, where the object image covariance is not a vector but a symmetric and positive-definite matrix, PCA for vector-valued data cannot be applied directly.

Fletcher et al. (2004) generalizes PCA to manifold-valued data in a coordinate-invariant way, and refers to it as PGA. The idea of PGA is to apply standard PCA to the tangent space at the intrinsic mean as follows: (i) compute the intrinsic mean of the manifold-valued samples; (ii) compute the sample covariance of the manifold-valued samples on the tangent space at the computed intrinsic mean; (iii) find the eigenvectors and eigenvalues of the sample covariance.

Recently, PGA of \(P(n)\) has been presented with the affine-invariant metric by Fletcher and Joshi (2007). However, applying PGA as described by Fletcher and Joshi (2007) to our case has a number of disadvantages, the most notable being that the affine-invariant mean is obtained via a gradient descent optimization procedure. In contrast, the log-Euclidean mean of \(P(n)\) samples can be calculated in a fixed time without iteration; PGA can thus be applied to the image covariance descriptors efficiently. In short, PGA of \(P(n)\) elements \(\{P_1, \ldots, P_N\}\) is equivalent to obtaining the eigenvalues and eigenvectors of the sample covariance obtained using (23), and with the sample mean obtained using (22). Furthermore, the incremental PCA of Ross et al. (2008) can also be applied directly to the image covariance descriptors without any algorithmic modification, as long as we retain the matrix logarithms of the image covariance descriptors in column vector form. (For space reasons the incremental PCA algorithm of Ross et al. (2008) is not recounted here.)

For the measurement likelihood calculation with the incrementally updated mean and principal eigenvectors of the object image covariance descriptors, the probabilistic approach of Moghaddam and Pentland (1997) can be adopted. Given the mean \(\tilde{C}\) of the collected image covariance descriptors \(\{C_{\tilde{x}_0}, C_{\tilde{x}_1}, C_{\tilde{x}_2}, \ldots\}\) and the first \(M\) principal eigenvectors \(u_i, i = 1, \ldots, M\), the residual error \(e^2_r\) for the image covariance \(C_{\tilde{x}_k}\) is defined as

\[
e^2_r = \|\log(C_{\tilde{x}_k}) - \log(\tilde{C})\|^2 - \sum_{i=1}^{M} c_i^2,
\]

where the \(c_i\) are the projection coefficients of \((\log(C_{\tilde{x}_k}) - \log(\tilde{C}))\) for each principal eigenvector, i.e., \(c_i = u_i^T (\log(C_{\tilde{x}_k}) - \log(\tilde{C}))\). The residual error \(e^2_r\) is understood to be the “distance-from-feature-space” (DFFS) while the “distance-in-feature-space” (DIFS) is defined as the Mahalanobis distance, i.e., \(\sum_{i=1}^{M} (c_i^2 / \lambda_i)\) where the \(\lambda_i\) are the eigenvalues corresponding to \(u_i\) (Moghaddam and Pentland 1997). Therefore, the measurement equation (5) can be explicitly expressed as

\[
y_k = \left[ e_r \sqrt{\sum_{i=1}^{M} c_i^2 / \lambda_i} \right] + n_k, \tag{27}
\]

where \(n_k\) is zero-mean Gaussian noise with covariance \(R = \text{Diag}(\sigma^2_{PGA}, 1) \in \mathbb{R}^{2 \times 2}\). The measurement likelihood \(p(y_k | X_k)\) is then calculated from

\[
p(y_k | X_k) \propto \exp\left( -\frac{1}{2} y_k^T R^{-1} y_k \right). \tag{28}
\]

The collected object images \(\{T_{\tilde{x}_1}, T_{\tilde{x}_2}, \ldots\}\) can also be used for similarity comparison purposes; we simply add the square root of the sum of squared differences (SSD) between \(T_{\tilde{x}_1}\) and the mean intensity image of \(\{T_{\tilde{x}_0}, T_{\tilde{x}_1}, T_{\tilde{x}_2}, \ldots\}\) to \(y_k\) to further increase the robustness of the measurement likelihood calculation. The measurement equation is finally given by

\[
y_k = \left[ e_r \sqrt{\sum_{i=1}^{M} c_i^2 / \lambda_i} \right] + n_k, \tag{29}
\]

where \(\tilde{T}\) represents the object mean intensity image incrementally updated, and \(n_k\) is Gaussian noise with \(R = \text{Diag}(\sigma^2_{\tilde{PGA}}, 1, \sigma^2_{\text{SSD}}) \in \mathbb{R}^{3 \times 3}\). Until the minimal number of the object image covariance descriptors sufficient to perform PGA is collected, at the very initial phase of tracking where the object appearance change is not intense, the measurement becomes simply the square root of SSD between \(T_{\tilde{x}_0}\) and \(T_{\tilde{x}_1}\), i.e.,

\[
y_k = \|T_{\tilde{x}_0} - T_{\tilde{x}_1}\| + n_k, \tag{30}
\]

where \(n_k\) is sampled from \(N(0, R)\), \(R = \sigma_{\text{SSD}}^2\).

We now present the visual tracking algorithm described thus far.
Algorithm

1. Initialization

(a) Set \( k = 0 \).

(b) Set the initial phase period as \( k_{\text{init}} \) and update period as \( k_{\text{update}} \) for the incremental PGA.

(c) Set \( \delta \) between 0.95 \( \sim \) 0.99 for the covariance estimation.

(d) Set number of particles as \( N \).

(e) For \( i = 1, \ldots, N \),
   - Set \( X_{0}^{(i)} = I, A_{0}^{(i)} = 0 \).
   - Draw the covariance parameters \( \theta_{0}^{(i)} \) for \( S \) from \( p(\theta_{0}) \).

2. Importance sampling step

(a) Set \( k = k + 1 \).

(b) Compute the mean \( \bar{\theta}_{k-1} \) and covariance \( \Sigma_{\theta_{k-1}} \) of \( \{\theta_{k-1}^{(1)}, \ldots, \theta_{k-1}^{(N)}\} \) from (22) and (23).

(c) For \( i = 1, \ldots, N \), draw \( x_{k}^{(ei)} \sim p(x_{k} | x_{k-1}^{(ei)}, \theta_{k}^{(ei)}) \), i.e.
   - Determine the kernel mean \( a\theta_{k-1}^{(i)} + (1 - a)\bar{\theta}_{k-1} \) from (21) where \( a = (3\delta - 1)/2\delta \).
   - Draw \( \theta_{k}^{(ei)} \) from \( N(a\theta_{k-1}^{(i)} + (1 - a)\bar{\theta}_{k-1}, h^{2}\Sigma_{\theta_{k-1}}) \) where \( h^{2} = 1 - a^{2} \).
   - Generate the Gaussian \( \epsilon_{k} \) from \( N(0, \theta_{k}^{(ei)}) \), and propagate \( x_{k}^{(i)} \) to \( x_{k}^{(ei)} \) with \( A_{k}^{(i)} \) via (17).
   - Compute \( A_{k}^{(ei)} \) with (18).

(d) For \( i = 1, \ldots, N \), calculate the importance weights by \( w_{k}^{(i)} \propto p(y_{k} | x_{k}^{(ei)}) \) with (30) if \( k \leq k_{\text{init}} \), or with (29).

(e) For \( i = 1, \ldots, N \), normalize the importance weights by \( \tilde{w}_{k}^{(i)} = w_{k}^{(i)} \left[ \sum_{j=1}^{N} w_{k}^{(j)} \right]^{-1} \).

3. Selection step (resampling)

(a) Resample from \( \{X_{k}^{(1)}, \ldots, X_{k}^{(N)}\}, \{A_{k}^{(1)}, \ldots, A_{k}^{(N)}\} \), and \( \{\theta_{k}^{(1)}, \ldots, \theta_{k}^{(N)}\} \) with probability proportional to \( \tilde{w}_{k}^{(i)} \) to produce independent and identically distributed (i.i.d.) random samples \( \{X_{k}^{(1)}, \ldots, X_{k}^{(N)}\}, \{A_{k}^{(1)}, \ldots, A_{k}^{(N)}\} \), and \( \{\theta_{k}^{(1)}, \ldots, \theta_{k}^{(N)}\} \).

(b) For \( i = 1, \ldots, N \), set \( w_{k}^{(i)} = \tilde{w}_{k}^{(i)} = 1/N \).

(c) If \( k \geq k_{\text{init}} \) and mod\( (k, k_{\text{update}}) = 0 \),

4. Go to importance sampling step

4. Experiments

In this section, we demonstrate the feasibility of our proposed visual tracking algorithm via various experiments, comparing the performance of our geometric tracker with existing local coordinate-based trackers. In the first set of experiments, we track a cube moving under a large-scale change. The second set of experiments are performed with four widely used test video clips (Jepson et al. 2003; Ross et al. 2008); these test clips are particularly challenging because of the sudden and extreme appearance changes resulting from, e.g., temporal occlusions, changes in the object pose, and abruptly varying lighting conditions.

4.1. Experiment 1: A Moving Cube

The video clip used in this experiment contains images of a cube moving under rather mild tracking conditions, i.e. a fixed camera and indoor environment. As shown in Figure 6, the scale of the cube image undergoes relatively large changes. To focus on verifying to what extent tracking performance depends on the state representation, we run our tracker without many of the features proposed in Section 3, i.e. the AR state dynamics, covariance parameter estimation, and incremental PGA of the image covariance descriptors. The state transition is instead generated by a random walk with pre-specified covariance, with Equation (30) taken as the measurements.
4.1.1. Implementation Details

The covariance for the Wiener process noise for the cube video clip is set to

\[ S = \text{diag} \left( 0.2^2, 0.02^2, 0.4^2, 0.02^2, 10^2, 10^2 \right)^\top \]  

(31)

with the discrete time interval \( \Delta t = \frac{1}{30} \), with this choice the cube video clip is sampled at 30 frames per second (fps). The Wiener process noise \( \epsilon_k \in \mathbb{R}^6 \) is sampled from \( N(0, S) \) and fed into (17) with \( \Delta_{k-1} = 0 \).

For comparison, we also apply the local coordinate representation-based trackers using the Euclidean embedding and SVD. For the Euclidean embedding-based tracker, the state transition is set to be \( x_k = x_{k-1} + w_k \), where

\[
\begin{bmatrix}
\eta_{k,1} + \eta_{k,2} \\
-\eta_{k,3} + \eta_{k,4} \\
\eta_{k,3} + \eta_{k,4} \\
\eta_{k,1} - \eta_{k,2} \\
\eta_{k,5} \\
\eta_{k,6}
\end{bmatrix}
\]

(32)

with \( \eta_k = (\eta_{k,1}, \ldots, \eta_{k,6})^\top \in \mathbb{R}^6 \) sampled from \( \eta_k \sim N(0, S' \Delta t) \); the above can be justified by noting that the following approximation is valid for sufficiently small \( \eta_k \):

\[
\exp \left( \sum_{i=1}^{6} \eta_{k,i} E_i \right) \\
\approx \begin{bmatrix} 1 + \eta_{k,1} + \eta_{k,2} & -\eta_{k,3} + \eta_{k,4} & \eta_{k,5} \\
\eta_{k,3} + \eta_{k,4} & 1 + \eta_{k,1} - \eta_{k,2} & \eta_{k,6} \\
0 & 0 & 1 \end{bmatrix}.
\]  

(33)

Setting the state covariance for the SVD-based tracker to be equivalent to our geometric tracker proves to be somewhat problematic, because the transformations induced by \( E_1 \), \( E_2 \), and \( E_4 \) are involved together in \( s_1 \), \( a \), and \( \phi_2 \). Instead we have experimentally found that the state transition via the Wiener process with

\[ S' = \text{diag} \left( 0.4^2, 0.02^2, 0.2^2, 0.02^2, 10^2, 10^2 \right)^\top \]  

(34)

is almost equivalent to our case around the identity. The state equation for the SVD-based tracker thus becomes \( x_k = x_{k-1} + w_k \), with \( w_k \sim N(0, S' \Delta t) \).

As mentioned earlier in Section 2.3, for local coordinate representation-based trackers, larger covariance values are needed to ensure successful tracking when the GL(2) component of the state is farther from the identity. Therefore, we run the two local coordinate representation-based trackers with various covariance values and various numbers of particles.

To quantitatively compare tracking performance, the two-dimensional pixel coordinates of the four corner points of the cube image are extracted manually and used as ground truth data. Tracking performance is judged based on the difference between the ground truth data and corner point coordinates estimated by each tracker. The experiments are performed using Matlab R2008a running on an Intel Core-2 Quad 2.4 GHz processor with 3 GB memory.

4.1.2. Results

Tracking results for the cube video clip are summarized in Table 1. In the table, “L1” and “L2” respectively represent the Euclidean embedding-based and SVD-based trackers, while “Cov.”, “N”, “Err.” and “N. Err.” respectively denote the covariance for the Wiener process noise, number of particles, and average computation time per frame in seconds. “Err.” is the average of \( e_i \), calculated as

\[
e_i = \sqrt{\sum_{j=1}^{4} (p_{i,j} - \tilde{p}_{i,j})^2}, \quad i = 1, \ldots, N_f, \]  

(35)

where \( N_f \) is the total number of video frames, and \( p_{i,j} \) and \( \tilde{p}_{i,j} \) respectively represent the coordinates of the four corner points corresponding to the estimated state and ground truth data. Since \( e_i \) is clearly affected by the cube image scale, we

<table>
<thead>
<tr>
<th>Case</th>
<th>Tracker</th>
<th>Cov.</th>
<th>N</th>
<th>Err.</th>
<th>N. Err.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ours</td>
<td>S</td>
<td>200</td>
<td>5.4058</td>
<td>2.8038</td>
<td>0.2035</td>
</tr>
<tr>
<td>2</td>
<td>Ours</td>
<td>S</td>
<td>500</td>
<td>5.3078</td>
<td>2.7919</td>
<td>0.5027</td>
</tr>
<tr>
<td>3</td>
<td>Ours</td>
<td>S</td>
<td>1,000</td>
<td>5.0648</td>
<td>2.6634</td>
<td>1.047</td>
</tr>
<tr>
<td>4</td>
<td>L1</td>
<td>S</td>
<td>500</td>
<td>32.2654</td>
<td>14.5886</td>
<td>0.266</td>
</tr>
<tr>
<td>5</td>
<td>L1</td>
<td>S</td>
<td>1,000</td>
<td>15.3742</td>
<td>6.9193</td>
<td>0.531</td>
</tr>
<tr>
<td>6</td>
<td>L1</td>
<td>S</td>
<td>1,000</td>
<td>6.6966</td>
<td>3.4259</td>
<td>0.542</td>
</tr>
<tr>
<td>7</td>
<td>L1</td>
<td>S</td>
<td>2,000</td>
<td>6.7121</td>
<td>3.4618</td>
<td>1.086</td>
</tr>
<tr>
<td>8</td>
<td>L1</td>
<td>S</td>
<td>1,000</td>
<td>6.0824</td>
<td>3.1889</td>
<td>0.539</td>
</tr>
<tr>
<td>9</td>
<td>L1</td>
<td>S</td>
<td>2,000</td>
<td>5.6238</td>
<td>2.9250</td>
<td>1.084</td>
</tr>
<tr>
<td>10</td>
<td>L2</td>
<td>S'</td>
<td>500</td>
<td>6.3431</td>
<td>3.4203</td>
<td>0.281</td>
</tr>
<tr>
<td>11</td>
<td>L2</td>
<td>S'</td>
<td>1,000</td>
<td>5.9576</td>
<td>3.1621</td>
<td>0.563</td>
</tr>
<tr>
<td>12</td>
<td>L2</td>
<td>S'</td>
<td>1,000</td>
<td>6.0190</td>
<td>3.2449</td>
<td>0.561</td>
</tr>
<tr>
<td>13</td>
<td>L2</td>
<td>S'</td>
<td>2,000</td>
<td>5.8082</td>
<td>3.1127</td>
<td>1.142</td>
</tr>
</tbody>
</table>
calculate another error $\tilde{e}_i$ normalized by the scales of the cube image calculated from the ground truth data as shown in Figure 7. “N. Err.” in Table 1 is the average of $\tilde{e}_i$. Figure 8 and 9 show respectively $e_i$ and $\tilde{e}_i$ produced by each tracker.

As shown in Table 1, the cases corresponding to our tracker, i.e. “1”, “2”, and “3”, yield the lowest tracking errors among all of the cases (based on comparing “Err.” and “N. Err.” for cases “1”, “2”, and “3” with the others). Of particular note is that even with a small number of particles as in case “1”, satisfactory tracking performance is achieved. The plots of $\tilde{e}_i$ in Figure 9(a) imply tracking errors that are more or less invariant to scale changes (with the exception of some frames around the 127th frame, where an abrupt scale change occurs accompanied by relatively large rotation and position changes).

As suggested by the large error values for the cases “4” and “5” in Table 1, tracking failures occur for the Euclidean embedding-based tracker when the covariance $S$ is set identical to ours. From Figure 8(b) and Figure 9(b), we can see that the tracking errors after the 100th frame are greater than those for the earlier phase, owing to the large-scale change. One implication is that this choice of $S$ is appropriate only when the $GL(2)$ component of the state is close to the identity.

As shown in Table 1 for the cases “6”, “7”, “8”, and “9”, tracking accuracy of the Euclidean embedding-based tracker is improved with an increase in number of particles (1,000 and 2,000) and covariance values ($2S$ and $3S$). However, it should be noted that the tracking errors for the large number of particles is still greater than the cases “1”, where only 200 particles are used for our tracker.

For the SVD-based tracker, we can see from Table 1 that the tracking errors for the cases “10” and “11” with the covariance $S'$ equivalent to our case is much smaller than that of the Euclidean embedding-based tracker with the covariance $S$ (cases “4” and “5”). This superiority arises from the fact that the rotation component is factored out via the SVD-based representation, which is not the case for the Euclidean embedding. Although tracking accuracy of the SVD-based tracker is slightly improved with an increase in the number of particles (2,000) and covariance values ($2S'$) for the case “13”, it is still worse than that of our tracker. The tracking accuracy with $3S'$ was not noticeably improved, and the results for such cases are not included here.

As given in Table 1, the computational complexity of our tracker is about twice that of the local coordinate representation-based trackers. The tracking results for the cases “1”, “4”, and “10” can also be seen in Extension 1.

### 4.2. Experiment 2: Video Clips from Previous Literature

We now test the tracking performance of our algorithm using some well-known video clips from the literature: the “Dudek” sequence from Jepson et al. (2003) and the “David”, “Sylvester”, and “Trellis” sequences from Ross et al. (2008). For comparison purposes all of the video clips are resampled to be 15 fps. Further aspects of the test video clips are summarized in Table 2.

#### 4.2.1. Implementation Details

For the prior distributions of the unknown parameters, i.e. $p(\theta_0)$, a uniform distribution between possible intervals is suggested by Higuchi (2001). Since in our case the parameter space is $P(n)$, we first sample $\text{Sym}(n)$ using a uniform distribution between $-0.25$ and $0.25$. Then the initialized covariance particles are obtained as the matrix exponential of the sampled $\text{Sym}(n)$ particles; this initialization procedure is consistent with

| Table 2. Characteristics of the Video Clips used in Experiment 2 |
|------------------|------------------|------------------|------------------|------------------|
| Environment      | Dudek            | David            | Sylvester        | Trellis          |
| Indoor           | Indoor           | Indoor           | Indoor           | Outdoor          |
| Facial Change    | Large            | Large            | No               | Small            |
| Pose Change      | Moderate         | Large            | Very large       | Large            |
| Lighting Change  | Moderate         | Large            | Large            | Very large       |
| Temporal Occlusion | Yes             | No              | No              | No               |
| Overall Difficulty | 4               | 3               | 2               | 1               |

Fig. 7. A plot of the scale change of the cube image.
the log-Euclidean metric used for $P(n)$. The control factor $\delta$ is set to 0.99.

Since the magnitude of the noise for the translational part is quite different from that for the $GL(2)$ part, the random noise generated with the covariance particles should be appropriately scaled. Furthermore, the geometric transformations induced by $E_2$ and $E_4$ in (2) are not significant in the test video clips. Therefore, we scale the random noise generated with the covariance particles with scale parameters $(0.05, 0.001, 0.05, 0.001, 15, 15)^T$ in the order of the Lie algebra basis elements of $\text{Aff}(2)$ in (2).

We set the initial phase period to be the first 15 frames, and the update period for the incremental PGA to be 5 frames. We used 500 particles for the experiments, with an AR parameter value of 0.5. These experiments are also performed in Matlab R2008a running on an Intel Core-2 Quad 2.4 GHz processor with 3 GB memory.

4.2.2. Results

Tracking results for the “Dudek” sequence shown in the first column of Figure 10 appear to be quite satisfactory. Note that changes in the object appearance (in this case due to changes in the object pose, facial expressions, lighting conditions, and the wearing and removal of glasses) and object scale occur
Gradually. Owing to the inherent stochastic nature of particle filtering, the tracker can recover from the temporary drift induced by the temporal object occlusion as shown in the 103rd and 110th frames.

The “David” sequence appears more challenging than the “Dudek” sequence because of the abrupt change in lighting conditions, together with a long-term object pose change. However, as shown in the second column of Figure 10, our tracker tracks the object quite well over the entire frame sequence. The tracker maintains its focus around the object while the object changes its pose around the 160th frame, and continues to successfully track the object when the frontal face of the object reappears after the 177th frame. Tracking results for the remainder of the “David” sequence after the 177th frame are also successful despite the facial expression change, removing and wearing the glasses, and changes in lighting as shown in the 226th, 299th, and 429th frames.

For the “Sylvester” sequence, our proposed tracker also yields quite satisfactory tracking results as shown in the third column of Figure 10. Around the 309th frame, the tracker temporarily loses the object because of the abrupt pose change. However, as shown in the 322nd and 437th frames, our tracker...
does not lose the object permanently, and quickly recovers from any temporal drift whenever the frontal face of the object reappears.

The “Trellis” sequence, possibly the most challenging among the four test video clips, involves an extreme change in lighting that occurs at the same time as a long-term object pose change. Again, tracking results shown in the fourth column of Figure 10 are highly encouraging: as shown in the 63rd, 150th, and 268th frames, our tracker is able to track the object in spite of the extreme change in lighting. After the long-term object pose change around the 332nd frame, the tracker continues to track the object when the frontal object face reappears, despite the abrupt change in lighting that occurs at the 371st and 442nd frames. Tracking results of our tracker can also be seen in Extension 1.
4.2.3. Comparison with an Intensity-based Tracker

To measure the tracking benefits of measurement likelihood calculation using incremental PGA of the image covariance descriptors, we run our tracker using only the incrementally updated object mean image $\bar{T}$, i.e. the measurement equation is now given by

$$y_k = \|\bar{T}x_k - \bar{T}\| + n_k,$$

(36)

where $n_k \sim N(0, R) , R = \sigma^{2}_{\text{SSD}}$. A comparison between our tracker (solid rectangle) and the simple intensity-based tracker (dashed rectangle) is shown in Figure 11. For the “Dudek” sequence, the intensity-based tracker yields surprisingly good tracking results as shown in the first column of Figure 11. We presume that these can be attributed to the fact that changes in lighting and object pose are relatively moderate, and that the object template image has been normalized to have unit variance and zero mean so as to minimize the effects of illumination changes. However, at some frames (such as around the 374th and 472nd frames), tracking accuracy is noticeably worse than that of our proposed tracker. Tracking results for the simple intensity-based tracker are not as good for the remaining three sequences. As evident from the second column of Figure 11, the intensity-based tracker eventually loses the object for the “David” sequence after the long-term object pose change around the 160th frame, although tracking results are satisfactory up to this point. Tracking failures also occur for the “Sylvester” and “Trellis” sequences. As shown in the third column of Figure 11, the tracker permanently loses the object after the 231st frame, when extreme changes in the object pose and lighting occur simultaneously. Around the 182nd frame when an extreme change in lighting take place, tracking accuracy is worse than that of our tracker. For the “Trellis” sequence, as shown in the fourth column of Figure 11, the tracker loses the object at the very early stages because of the extreme change in lighting. The tracking result comparison shown in Figure 11 is also given in Extension 1.

4.2.4. Tracking Without the AR State Dynamics

To verify the effect of the state dynamics using the AR process on $\text{Aff}(2)$, we run our proposed tracker again without the AR state dynamics, i.e. we set $a = 0$ in (18). A comparison of tracking results with (solid rectangle) and without (dashed rectangle) the AR state dynamics is shown in Figure 12. For the “Dudek” sequence shown in the first column of Figure 12, the tracker fails to recover quickly from the drift induced by the temporal occlusion as shown in the 117th and 187th frames, and slowly adjusts to the fast object scale changes as shown in the 504th frame. As shown in the second column of Figure 12, the tracker also adjusts to the object pose change around the 160th frame, which is much slower than for
Fig. 12. A comparison of the tracking results for our proposed tracker with (solid rectangle) and without (dashed rectangle) the AR state dynamics. The first, second, third, and fourth columns represent the tracking results respectively for the “Dudek”, “David”, “Sylvester”, and “Trellis” sequences.

Without the AR state dynamics, tracking results for the “Sylvester” (the third column of Figure 12) and “Trellis” (the fourth column of Figure 12) sequences are considerably worse. For the “Sylvester” sequence, tracking accuracy is comparable to that of our tracker with the AR state dynamics before the tracker loses the object around the 304th frame. Unlike our tracker with the AR state dynamics, the tracker cannot recover quickly from the temporal drift, even when the frontal object reappears in the 335th frame. For the “Trellis” sequence, without the AR state dynamics, the tracker eventually loses the object at an early stage of the sequence. The tracking result comparison shown in Figure 12 is also available in Extension 1.

4.2.5. Comparison with IVT Particle Filtering Tracker

Finally, we compare our tracker’s performance with that of the IVT tracker of Ross et al. (2008). Tracking results of the IVT tracker are obtained by directly running the Matlab code available from Ross (2008) without any parameter modification. For the “Dudek” and “David” sequences, the initial object regions used for our tracker are different from those for the IVT tracker. During experiments, we have found that the IVT tracker’s performance for the “Dudek” sequence is sensitive to the initial object region. Thus, for a fair comparison, we run both trackers with the same initial object image region suggested by Ross (2008). The comparison between our tracker (solid rectangle) and the IVT tracker (dashed rectangle) is shown in Figure 13.

For the “Dudek” sequence, both trackers accurately track the object over the entire sequence as shown in the first column of Figure 13. On the whole, our tracker’s performance is comparable to that of the IVT tracker. Around the 373rd frame, the IVT tracker outperforms our tracker while the opposite is the case around the 476th frame. Our tracker outperforms the IVT tracker for the “David” sequence as shown in the second column of Figure 13. Although both trackers never lose the object, our tracker displays superior tracking performance around the 78th and 243rd frames when compared with the initial object image region.

The better performance of our tracker is more clearly demonstrated for the “Sylvester” (the third column of Figure 13) and “Trellis” (the fourth column of Figure 13) sequences. For the “Sylvester” sequence, both trackers lose the object around the 305th frame as a result of the abrupt object pose change. As tracking proceeds, the IVT tracker cannot recover from the drift and eventually loses the object, unlike our tracker. As shown in the 180th frame, the tracking accuracy of the IVT tracker is worse than that of our tracker.

For the “Trellis” sequence, both trackers accurately track the object over the initial lighting condition change. After the long-term object pose change around the 320th frame, the IVT tracker eventually loses the object while our tracker continues to track the object until the end of the sequence. The tracking result comparison of Figure 13 can also be seen in Extension 1.
4.3. Discussion

The improved performance of our geometric tracker over purely local coordinate-based trackers has been quantitatively shown via experiments with the cube video clip. As argued earlier in Section 2.3, our geometric tracker more naturally represents the image transformations induced by the three-dimensional object movement; we suspect that the difference in tracking accuracy can be traced to the fact that each Lie algebra basis element of our tracker naturally represents each possible geometric transformation mode as shown in Figure 2, which is not the case for the local coordinate-based trackers.

We have also qualitatively demonstrated our geometric tracker’s improved performance through a series of experiments involving benchmark test video clips. A simple intensity-based measurement likelihood was sufficient when changes in lighting are not abrupt as in the “Dudek” sequence. However, for abrupt changes in lighting such as those found in the “David”, “Sylvester”, and “Trellis” sequences, intensity-based measurement likelihoods were not robust. In contrast, such changes in lighting could be dealt with appropriately with the incremental PGA of the image covariance descriptors. Another advantage of our tracker is the ability to quickly recover from large and abrupt object pose changes whenever the frontal object image reappears.

The advantages of using an $\text{Aff}(2)$ AR process for the state dynamics has also been demonstrated. In principle, a random walk model is appropriate for most visual tracking applications, particularly if a large number of particles, and large covariance values for particle propagation, are used. However, in practice the number of particles one can effectively use is limited by the requirement that visual tracking should inherently be performed online. Including an appropriate state dynamics is one means of compensating for this limitation, and our studies suggest that the $\text{Aff}(2)$ AR process is quite adequate for our purposes.

We also have shown that our tracker outperforms the IVT tracker, especially for the “Sylvester” and “Trellis” sequences. The aforementioned tracking failures of the IVT tracker for the “Sylvester” and “Trellis” sequences have already been reported by Ross et al. (2008). For the “Sylvester” sequence, note that our tracker’s performance using the video clip resampled in 15 fps is better than that of the IVT tracker using the original video clip at 30 fps. From Extension 1, we can see that the IVT tracker’s tracking results are prone to chatter, whereas ours tends to be smoother. This comes from the fact that the state estimation of the IVT tracker is given by the particle whose weight is greatest. We believe that it is better to estimate the state using the sample mean of the particles rather than the maximum weight particle. We have also found that the tracking performance of our tracker is not sensitive to the initial object image region; it is instructive to compare our tracking results for the “Dudek” and “David” sequences using our own initial object image regions (Figure 10 and Extension 1) with using the initial object image region suggested by Ross et al. (2008) (Figure 13 Extension 1).
Fig. 14. The estimated covariance elements for the “Sylvester” sequence. The solid and dashed lines respectively denote the diagonal and off-diagonal elements of the estimated covariance.

For the covariance estimation, note that the settings for the four test video clips are identical, i.e. the covariance particles are initialized from the matrix exponential of $\text{Sym}(n)$ sampled from the uniform distribution between $-0.25$ and $0.25$, and the sampled noise is scaled with the same scale parameters. We cannot assert that the estimated covariance via our combined state–covariance estimation approach is highly accurate for a given specific situation. However, an important benefit is that difficult and time-consuming noise parameter tuning for specific scenarios can be avoided. One case of covariance estimation is shown in Figure 14. With a sufficiently large number of particles, more general and larger affine transformations can be tracked with larger values of the noise scale parameters.

The average computational time per frame of our proposed visual tracker is about 1.8 seconds with 500 particles. Although the computation times indicate that the current speed of our tracker is far from being online, we expect that further speedup, up to rates of 15 fps, is possible (the current implementation is in Matlab, with little consideration given to improving speed at this stage).

5. Conclusions

We have presented a visual tracking framework based on a coordinate-invariant, geometrically well-defined particle filtering algorithm developed for $\text{Aff}(2)$. To increase robustness, the tracker adopts the following additional features: (i) the state dynamics is given by an AR process defined on $\text{Aff}(2)$; (ii) combined state–covariance estimation; and (iii) robust measurement likelihood calculation based on incremental PGA of the image covariance descriptors. The log-Euclidean metric on $\mathcal{P}(n)$ plays a fundamental role in covariance estimation and incremental PGA of the image covariance descriptors. The proposed visual tracking framework has been tested via various comparative experiments, with considerably improved tracking performance relative to current state-of-the-art visual trackers.

Possibilities for future work include extending our visual tracking framework to multiple object tracking in more realistic situations, and how to deal with long-term object occlusions. Both problems can be approached in the context of the data association problem.

Acknowledgements

The authors are grateful to Professor Kyoung Mu Lee for his comments and suggestions that helped to improve the manuscript. This research was supported in part with funding from CBMS-KRF-2007-412-J03001, KIST-CIR, the Intelligent Autonomous Manipulation Research Center, ROSAEC, and IAMD-SNU.

Appendix: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org

Table of Multimedia Extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>Moving pictures corresponding to all of the tracking results shown in Section 4</td>
</tr>
</tbody>
</table>

References


