A Geometric Particle Filter for Template-Based Visual Tracking

Junghyun Kwon, Hee Seok Lee, Frank C. Park, and Kyoung Mu Lee

Abstract—Existing approaches to template-based visual tracking, in which the objective is to continuously estimate the spatial transformation parameters of an object template over video frames, have primarily been based on deterministic optimization, which as is well-known can result in convergence to local optima. To overcome this limitation of the deterministic optimization approach, in this paper we present a novel particle filtering approach to template-based visual tracking. We formulate the problem as a particle filtering problem on matrix Lie groups, specifically the Special Linear group $SL(3)$ and the two-dimensional affine group $Aff(2)$. Computational performance and robustness are enhanced through a number of features: (i) Gaussian importance functions on the groups are iteratively constructed via local linearization; (ii) the inverse formulation of the Jacobian calculation is used; (iii) template resizing is performed; and (iv) parent-child particles are developed and used. Extensive experimental results using challenging video sequences demonstrate the enhanced performance and robustness of our particle filtering-based approach to template-based visual tracking. We also show that our approach outperforms several state-of-the-art template-based visual tracking methods via experiments using the publicly available benchmark data set.

Index Terms—Visual tracking, object template, particle filtering, Lie group, special linear group, affine group, Gaussian importance function

1 INTRODUCTION

TEMPLATE-BASED object tracking is a particular form of visual tracking, in which the objective is to continuously fit an object template to a sequence of video frames [62]. Beginning with the seminal work of Lucas and Kanade [37], template-based visual tracking has primarily been formulated as a nonlinear optimization problem, e.g., [3], [8], [9], [11], [18]. As is well known, however, nonlinear optimization-based tracking methods have the disadvantage of potentially becoming trapped in local minima, particularly when inter-frame object motions are large. Global optimization strategies such as multi-scale pyramids and simulated annealing, despite their considerably larger computational requirements, also cannot consistently ensure better tracking performance.

In this paper we propose an alternative approach to template-based visual tracking, based on geometric particle filtering. Particle filtering can be regarded as a Monte Carlo method for solving the (unnormalized) conditional density equations for state estimation [14], in which the state and observation equations are modelled as nonlinear stochastic differential equations. Because of its stochastic nature, particle filtering can, when done properly, be made more robust to the problem of local minima than existing deterministic optimization-based methods, without sacrificing much in the way of computationally efficiency. Moreover, particle filtering yields better estimation performance for nonlinear problems like template-based visual tracking than, e.g., extensions of the linear Kalman filter, because of its non-parametric density representation.

In applying particle filtering to template-based visual tracking, the first issue that needs to be addressed is how to represent the state. The estimated transformation parameters are typically in the form of matrices satisfying certain group properties—in our case these will be homographies and affine transformations—and the subsequent state space is no longer a vector space, but a curved space. To be able to directly apply existing particle filtering algorithms, which are primarily developed in the context of systems whose state space is a vector space, one must therefore choose a local coordinate representation for the curved state space, but there are several well-known problems associated with this approach: (i) several local coordinate patches may be required to cover the entire state space, and (ii) the choice of local coordinates will strongly affect tracking performance. See [2] for a detailed discussion on coordinate parametrizations for homographies.

A second issue involves the choice of importance function, a critical step in particle filtering. The optimal importance function minimizes the particle weight variance, and eventually maintains the number of effective particles to be as large as possible [15], so that accurate tracking is ensured even with a limited number of particles. Since for general nonlinear systems the optimal importance function cannot be determined in closed-form, a computationally efficient and sufficiently accurate approximation is required.
The main contributions of this paper directly address these two fundamental issues:

- Rather than applying conventional vector space particle filtering to some local coordinate formulation of the template-based visual tracking problem, we instead develop a geometric particle filtering algorithm that directly takes into account the curved nature of the state space. Specifically, homographies can be identified with the three-dimensional Special Linear group $SL(3)$, while the two-dimensional affine transformation matrices can be identified with the two-dimensional affine group $Aff(2)$. As is well-known, both $SL(3)$ and $Aff(2)$ possess the structure of an algebraic group and a differentiable manifold, or Lie group. Our resulting algorithms are geometric (in the sense of being local coordinate-invariant), so that their performance does not depend on the choice of local coordinates.

- We derive an exponential coordinate-based approximation of Gaussian distributions for Lie groups, and use these as optimal importance functions for particle sampling. This approach can in some sense be regarded as a generalization of the Gaussian importance function of [15] to our Lie groups in question.

- We develop several techniques that, collectively, enhance the accuracy and computational efficiency of our template-based visual tracking in a significant way: the iterative Gaussian importance function generation, inverse formulation of Jacobian calculation, template resizing, and use of parent-child particles.

The overall performance of our tracking framework is demonstrated in Section 7 via experiments with various challenging test video sequences. The quantitative performance comparison between the previously employed importance functions and ours as well as that between different parameter settings is also performed in Section 7. We finally compare the performance of our tracking framework with several state-of-the-art template-based visual tracking methods using the Metaio benchmark data set [35] in Section 7. The compared methods include the ESM tracker [8] and L1 tracker [4], which are widely regarded as the state-of-the-art template-based visual tracking algorithms based on deterministic optimization and particle filtering, respectively. Key-point matching-based motion estimation methods using descriptors such as SIFT [36], SURF [6], and FERNS [40] are also compared with our framework.

## 2 RELATED WORK

While regarded as more straightforward relative to other visual tracking problems [24], [41], [63], template-based visual tracking continues to arise in a wide range of mainstream applications, e.g., visual servoing [8], augmented reality [21], [45], and visual localization and mapping [29], [51]. What is common between such applications is that it is necessary to estimate the 3D camera motion from tracking results. In order to do so, we should accurately estimate the object template motion up to the homography that is the most general spatial transformation of planar object images, i.e., we should perform projective motion tracking for those applications.

However, previous template-based visual tracking algorithms employing particle filtering [4], [31], [32], [33], [46], [64] are all confined to affine motion tracking, i.e., template-based visual tracking with affine transformations. The exception is [60], but the accuracy of the tracking results as presented leave something to be desired. To the best of our knowledge, ours is the first work to solve projective motion tracking via particle filtering, with sufficient accuracy and efficiency by employing Gaussian importance functions to approximate the optimal importance function.

Following the seminal work of Isard and Blake to successfully apply particle filtering to contour tracking [24], the efforts to approximate the optimal importance function for visual tracking have mainly concentrated on contour tracking [34], [44], [47], [49] by means of the unscented transformation (UT) [25]. One notable work is [1], in which the optimal importance function is analytically obtained with linear Gaussian measurement equations for point set tracking. When approximating a nonlinear function, UT is generally recognized as more accurate than first-order Taylor expansions [25]. However, in our approach we rely on Taylor expansions instead of UT, because UT requires repeated trials of the measurement process involving image warping with interpolation, which may prohibit real-time tracking. Moreover, it is not clear how to choose in a systematic way the many parameters required in UT.

Regression-based learning also has been successfully applied to template-based visual tracking [22], [26], [33], [57], [65]. Regression methods estimate a regression relation between the spatial transformation parameters and image similarity values using training samples, which are then directly used in tracking. It is notable that the regression is performed on $Aff(2)$ in [57] and [33]. Since our framework does not require learning with training samples, the comparison with regression-based methods is not considered here.

Previous work on particle filtering on Lie groups was focused on the special orthogonal group $SO(3)$ representing 3D rotation matrices [12], [53] and the special Euclidean group $SE(3)$ representing 3D rigid body transformation matrices [28], [52]. Recently, Gaussian importance functions on $SE(3)$ obtained by UT were utilized for visual localization and mapping with a monocular camera [29]. In [33], particle filtering on $Aff(2)$ is employed as a basic inference tool for affine motion tracking. It is worth to note that the unscented Kalman filtering (or equivalently UT) on general Riemannian manifolds has been formulated in [20], which can be used to approximate the optimal importance function for particle filtering on Lie groups.

Two previous papers by the authors, [31] and [30], are the most closely related with this paper. Compared with [31], we extend the framework from affine motion tracking to projective motion tracking, and utilize Gaussian importance functions instead of the state transition density-based importance functions. [30] is a preliminary conference version of this paper. In addition to the extension to projective motion tracking, we have made several
important improvements over [30] to increase the accuracy and efficiency that include the iterative Gaussian importance function generation, inverse Jacobian formulation, and use of parent-child particles.

3 BACKGROUND

3.1 Lie Group
A Lie group $G$ is a group which is a differentiable manifold with smooth product and inverse group operations. The Lie algebra $\mathfrak{g}$ associated with $G$ is identified as a tangent vector space at the identity element of $G$. A Lie group $G$ and its Lie algebra $\mathfrak{g}$ are related via the exponential map, $\exp : \mathfrak{g} \to G$. The logarithm map $\log$ is defined as the inverse (near the identity) of $\exp$, i.e., $\log : G \to \mathfrak{g}$. For the matrix Lie groups we address, $\exp$ and $\log$ are given by the ordinary matrix exponential and logarithm, defined as

$$\begin{align*}
\exp(x) &= \sum_{m=0}^{\infty} \frac{x^m}{m!}, \\
\log(X) &= \sum_{m=1}^{\infty} \frac{(-1)^{m+1}(X-I)^m}{m}.
\end{align*}$$

where $x \in \mathfrak{g}$, $X \in G$, and $I$ is the identity matrix. For sufficiently small $x, y \in \mathfrak{g}$, the Baker-Campbell-Hausdorff (BCH) formula states that $z$ satisfying $\exp(z) = \exp(x) \exp(y)$ is given by

$$z = x + y + \frac{1}{2}[x, y] + \frac{1}{12}[x, [x, y]] - \frac{1}{12}[y, [x, y]] + \cdots,$$

where $[\cdot, \cdot]$ is the matrix commutator (Lie bracket) given by $[A, B] = AB - BA$. A more detailed description of Lie groups and Lie algebras can be found in, e.g., [19].

As is well known, the exponential map $\exp$ locally defines a diffeomorphism between a neighborhood of $G$ containing the identity and an open set of the Lie algebra $\mathfrak{g}$ centered at the origin, i.e., given $X \in G$ sufficiently near the identity, the exponential map $X = \exp(\sum u_i E_i)$, where $E_i$ are the basis elements of $\mathfrak{g}$, is a local diffeomorphism. These local exponential coordinates can be extended to cover the entire group by left or right multiplication, e.g., the neighborhood of any $Y \in G$ can be well defined as $Y(u) = Y \cdot \exp(\sum u_i E_i)$. Note also that in a neighborhood of the identity, the minimal geodesics (i.e., paths of shortest distance) are given precisely by these exponential trajectories (and their left- and right-translated versions).

3.2 Transformation Matrices as Lie Groups
A projective transformation is represented by a linear equation in the homogeneous coordinates with a nonsingular matrix $H \in \mathbb{R}^{3 \times 3}$ called a homography. To remove scale ambiguity of homogeneous coordinates, homographies can be normalized to have the unit determinant. In this case, homographies can be identified with the 3D special linear group $SL(3)$ defined as $SL(3) = \{ X \in GL(3) : \det(X) = +1 \}$ where $GL(3)$ is the 3D general linear group corresponding to the set of $3 \times 3$ invertible real matrices. $sl(3)$, the Lie algebra associated with $SL(3)$, corresponds to the set of $3 \times 3$ real matrices with the zero trace. We choose the basis elements of $sl(3)$ as

$$E_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, E_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$E_4 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, E_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, E_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The spatial transformations induced by exponentials of $E_i$ are shown in Fig. 1. We can define the map $v_{sl(3)} : sl(3) \to \mathbb{R}^3$ to represent $sl(3)$ elements in column vectors with respect to the basis elements given in (4). For

$$x = \begin{bmatrix} x_1 & x_4 & x_7 \\ x_2 & x_5 & x_8 \\ x_3 & x_6 & x_9 \end{bmatrix} \in \mathbb{R}^3,$$

the map is given by $v_{sl(3)}(x) = (x_1, x_9, \frac{x_2-x_4}{2}, \frac{x_2+x_4}{2}, x_7, x_8, x_5, x_6)^T$.

Note that our choice of $sl(3)$ basis elements in (4) is different from the one in [7] and [8]. The difference is that we separately represent the rotation and skew by $E_3$ and $E_4$, respectively. This helps to set the state covariance for particle filtering properly in practice. For example, if we can expect the object rotational motion is not significant, we can set the covariance value corresponding to $E_3$ smaller to maximize the tracking performance with the limited computational power.

An affine transformation is also frequently used in template-based visual tracking; such a transformation is represented by a matrix of the form $\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$ where $A$ belongs to $GL(2)$, the set of $2 \times 2$ invertible real matrices, and $b \in \mathbb{R}^2$. We can regard the set of affine transformation matrices as the affine group $Aff(2)$, which is a semi-direct product of $GL(2)$ and $\mathbb{R}^2$. The Lie algebra associated with $Aff(2)$, denoted $\mathfrak{aff}(2)$, is represented as $\mathfrak{aff}(2)$, where $U$ belongs to $gl(2)$ and $v \in \mathbb{R}^2$. $gl(2)$ is the Lie algebra associated with $GL(2)$ and corresponds to the set of $2 \times 2$ real matrices. The
six basis elements of \( \text{aff}(2) \) can be chosen as the same as the first six basis elements of \( s(l)(2) \) in (4) with the exception of \( E_2 \). \( E_2 \) of \( \text{Aff}(2) \) is given by

\[
E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

representing isotropic scaling. The map \( v_{\text{aff}(2)} : x \rightarrow \mathbb{R}^6 \) is given by \( v_{\text{aff}(2)}(x) = (\frac{x_1 + x_4}{2}, \frac{x_2 + x_5}{2}, \frac{x_3 + x_6}{2}, x_5, x_6, x_0) \) where

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \in \text{aff}(2).
\]

### 4 Probabilistic Tracking Framework

In this section, we describe how projective motion tracking can be realized by particle filtering on \( \text{SL}(3) \) with appropriate definitions of state and measurement equations. As shown in Fig. 2, the initial state \( X_0 \) becomes a \( 3 \times 3 \) identity matrix. Representing the image pixel coordinates by \( p = (p_x, p_y)^T \), the objective is to find \( X_k \in \text{SL}(3) \) that transforms the object template \( T(p) \) to the most similar image region of the \( k \)-th video frame \( I_k \).

#### 4.1 State Equation on \( \text{SL}(3) \)

In order to formulate a state equation on \( \text{SL}(3) \), we first consider a left-invariant stochastic differential equation on \( \text{SL}(3) \) of the form

\[
dX = X \cdot A(X)dt + X \sum_{i=1}^{n} b_i E_i dw_i,
\]

where \( X \in \text{SL}(3) \) is the state, the map \( A : \text{SL}(3) \rightarrow \text{sl}(3) \) is possibly nonlinear, the \( b_i \) are scalar constants, the \( dw_i \in \mathbb{R} \) denote independent Wiener process noise, and the \( E_i \) are the basis elements of \( \text{sl}(3) \) given in (4). The continuous state equation (5) can be represented in a more tractable form by the simple first-order Euler discretization:

\[
X_k = X_{k-1} \cdot \exp(A(X, t)\Delta t + v_{\text{sl}(3)}^{-1}(dW_k)\sqrt{\Delta t}),
\]

where \( dW_k \in \mathbb{R}^3 \) is the Wiener process noise with a covariance \( P \in \mathbb{R}^{3 \times 3} \) rather than the identity matrix, in which the \( b_i \) terms in (5) are reflected.

\( A(X, t) \) in (6) can be regarded as the state dynamics. Here we use a simple state dynamics based on the first order auto-regressive (AR) process on \( \text{SL}(3) \) approximating the generalized AR process on Riemannian manifolds of [61] using the logarithm map. The state equation on \( \text{SL}(3) \) that we actually use for projective motion tracking is then represented as

\[
X_k = X_{k-1} \cdot \exp(A_{k-1} + v_{\text{sl}(3)}^{-1}(dW_k)\sqrt{\Delta t}),
\]

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

where \( a \) is the AR parameter, usually set to be smaller than 1. We set \( a = 0.5 \) for all experiments in Section 7. We can assume in general tracking applications that \( X_{k-2}X_{k-1} \) lies within the injectivity radius of \( \exp \), because video frame rates typically range at least between 15 and 60 frames per second (fps), and object speeds are not excessively fast; thus \( A_{k-1} \) can be considered to be uniquely determined.

#### 4.2 Measurement Equation

To represent the measurement equation in a form such that analytic Taylor expansion is possible, we utilize the warping function \( W(p; X_k) \) employed in [3]. For a projective transformation of the pixel coordinates \( p = (p_x, p_y)^T \) of the template \( T(p) \) with

\[
X_k = \begin{bmatrix} a_1 & a_4 & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{bmatrix},
\]

\( W(p; X_k) \) is defined as

\[
W(p; X_k) = \begin{bmatrix} p'_x \\ p'_y \end{bmatrix} = \begin{bmatrix} a_3 p_x + a_4 p_y + a_7 \\ a_2 p_x + a_5 p_y + a_8 \\ a_1 p_x + a_6 p_y + a_9 \end{bmatrix}.
\]

The region corresponding to \( X_k \) can be expressed by \( I_k(W(p; X_k)) \), which is warped back to the template coordinates involving image interpolation. In our implementation, we simply use the nearest neighbor interpolation for computational efficiency. The measurement is then a measure of the image similarity between \( T(p) \) and \( I_k(W(p; X_k)) \), and can be represented by \( g(T(p), I_k(W(p; X_k))) \).

We regard the object template \( T(p) \) as a set composed of the initial template \( T_{\text{init}}(p) \) and the incremental PCA model of [46] with the mean \( T(p) \) and \( M \) principal eigenvectors \( b_i(p) \). Then our measurement equation is given by

\[
y_k = g(T(p), I_k(W(p; X_k))) + n_k = \frac{g_{\text{NCC}}}{g_{\text{PCA}}} + n_k,
\]

where \( n_k \sim N(0, R) \), \( R = \text{diag}(\sigma_{\text{NCC}}^2, \sigma_{\text{PCA}}^2) \). Here, \( g_{\text{NCC}} \) is a normalized cross correlation (NCC) score between \( T_{\text{init}}(p) \) and \( I_k(W(p; X_k)) \), while \( g_{\text{PCA}} \) is a reconstruction error determined by

\[
\sum_{i=1}^{M} [I_k(W(p; X_k)) - T(p)]^2 - \sum_{i=1}^{M} c_i^2,
\]

where \( c_i \) are the projection coefficients of the mean-normalized image to \( b_i(p) \).

The roles of NCC and PCA are mutually complementary: incremental PCA deals with appearance changes such as illumination change and pose change of non-planar
objects [46], which cannot be properly dealt with by NCC, while NCC minimizes any possible drift problems associated with incremental PCA. The possible drawback of conjunctive use of NCC and PCA is the tracking failure due to too large error in NCC for severe local appearance changes such as specular reflection. As a simple remedy, we detect outliers whose PCA reconstruction error is above the threshold (0.15 in our implementation) and exclude them in calculating 
g_{NCC}.

### 4.3 State Estimation via Particle Filtering

Tracking is now equivalent to estimating the filtering density \( \rho(X_k | y_{1:k}) \), where \( y_{1:k} \equiv \{ y_1, \ldots, y_k \} \), via particle filtering on SL(3). Assume that \( \{X_{1:k-1}^{(i)}, w_k^{(i)} \}, i = 1, \ldots, N \) , a set of SL(3) samples called particles and their weights, approximate \( \rho(X_{1:k-1} | y_{1:k-1}) \) well. The new particles \( X_k^{(i)} \) are then sampled from the importance function \( \pi(X_k | X_{1:k-1}^{(i)}; y_{1:k}) \), and the weights \( w_k^{(i)} \) for \( X_k^{(i)} \) are calculated as

\[
w_k^{(i)} = w_{k-1}^{(i)} \frac{\rho(y_k | X_k^{(i)}) \rho(X_k^{(i)} | X_{1:k-1}^{(i)})}{\pi(X_k^{(i)} | X_{1:k-1}^{(i)}; y_{1:k})}.
\]

Then, \( \{X_k^{(i)}, w_k^{(i)}, i = 1, \ldots, N\} \) is considered to be distributed to approximate \( \rho(X_k | y_{1:k}) \).

\( \rho(X_k^{(i)} | X_{1:k-1}^{(i)}) \) in (11) is calculated with (7) and (8) as

\[
\rho(X_k^{(i)} | X_{1:k-1}^{(i)}) = \frac{1}{\sqrt{(2\pi)^3 \det(Q)}} \exp\left(-\frac{1}{2} d_1^T Q^{-1} d_1\right),
\]

where \( d_1 = v_{3|3}(\log((X_k^{(i)})^{-1} X_{1:k-1}^{(i)}) - A_k^{-1} y_{1:k} ) \) and \( Q = P\Delta t \).

The measurement likelihood \( \rho(y_k | X_k^{(i)}) \) in (11) is similarly determined from (10) as

\[
\rho(y_k | X_k^{(i)}) = \frac{1}{\sqrt{(2\pi)^2 \det(R)}} \exp\left(-\frac{1}{2} d_2^T R^{-1} d_2\right),
\]

where \( d_2 = y_k - \hat{y}_k^{(i)} \) and \( \hat{y}_k^{(i)} = g(T(p), I_k(W(p, x_k^{(i)}))). \)

The real measurement \( y_k \) is given by the value of \( g \) when \( I_k(W(p, x_k)) = T(p) \). That is, \( g_{NCC} \) and \( g_{PCA} \) of \( y_k \) are always given by 1 and 0, respectively.

Usually, the particles \( X_k^{(i)} \) are resampled in proportion to their weights \( w_k^{(i)} \) in order to minimize the degeneracy problem [14]. Note that a particle can be copied multiple times as a result of resampling. Therefore, the same indices at time \( k - 2 \) and \( k - 1 \) can represent different particle histories actually. Since it is not possible to compute \( A_{k-1} \) in (8) immediately from \( X_{1:k-2} \) and \( X_{1:k-1} \) due to this reason, the state is augmented as \( \{\hat{X}_k, A_k\} \) in practice.

Assuming that particle resampling is conducted at every time step, the state estimation \( \hat{X}_k \) can be given by the sample mean of the resampled particles. An efficient gradient descent algorithm to obtain the mean of SL(3) samples using the Riemannian exponential map given by the form of \( \exp(-x^T) \exp(x + x^T) \) and its inverse log map is presented in [7]. For fast computation, we instead use the ordinary matrix exponential and log as an approximation of Riemannian exponential and log maps using the BCH formula (3). The corresponding algorithm is given in Algorithm 1. Although the convergence of Algorithm 1 cannot be guaranteed in the most general case, in our extensive experiments the sample mean of the resampled particles typically converges within just three or four iterations, as long as we choose the particle whose weight before resampling is the greatest among all particles as the initial mean.

Algorithm 1 An algorithm to obtain the mean of SL(3) samples \( \{X_1, \ldots, X_N\} \)

1. \( \mu = X_1 \)
2. repeat \( \Delta X_i = \mu^{-1} X_i \)
3. \( \Delta \mu = \exp\left(\frac{1}{N} \sum_{i=1}^{N} \log(\Delta X_i)\right) \)
4. \( \mu = \mu \Delta \mu \)
5. until \( ||\log(\Delta \mu)|| < \epsilon \)

### 5 GAUSSIAN IMPORTANCE FUNCTION FOR PROJECTIVE MOTION TRACKING

The performance of particle filtering depends heavily on the choice of importance function. The state transition density \( \rho(X_k | X_{k-1}) \) is popularly used as the importance function because of its simplicity in implementation, as the particle weights are calculated simply in proportion to the measurement likelihood \( \rho(y_k | X_k^{(i)}) \). However, since the object can move unpredictably (i.e., differently from our assumption of smooth motion implied by the AR process-based state equation in (7) and (8)) and a peaked measurement likelihood (i.e., a small covariance \( R \) for \( n_k \) in (10)) is inevitable for accurate tracking, most particles sampled from \( \rho(X_k | X_{k-1}) \) will have negligible weights and be wasted; this will result in tracking inaccuracy.

In the ideal case, the variance of particle weights will be minimum, i.e., evenly distributed particle weights. The optimal importance function minimizing the particle weight variance is given by the form of \( \rho(X_k | X_{k-1}) \) [15], which clearly states that the current measurement \( y_k \) should be considered in particle sampling. Unfortunately, this optimal importance function cannot be obtained in closed form for general nonlinear state space models (except for those with linear Gaussian measurement equations). Our measurement equation (10) is clearly nonlinear because the warping function \( W(p; X_k) \) itself is nonlinear and arbitrary video frames \( I_k \) are involved. Therefore, the optimal importance function must be properly approximated for accurate template-based visual tracking with a limited number of particles.

In this section, we present in detail how we can approximate the optimal importance function as Gaussian distributions on SL(3) by following the Gaussian importance function approach of [15].

#### 5.1 Gaussian Importance Function Approach

We consider the following vector state space model:

\[
x_k = f(x_{k-1}) + w_k,
\]

\[
y_k = g(x_k) + n_k.
\]
where \( x_k \in \mathbb{R}^{N_x}, y_k \in \mathbb{R}^{N_y}, \) and \( \omega_k \) and \( n_k \) are zero-mean Gaussian noise with covariances \( P \in \mathbb{R}^{N_x \times N_x} \) and \( R \in \mathbb{R}^{N_y \times N_y}, \) respectively. For this model, we briefly review the principal idea of the Gaussian importance function in [15].

In [15], \( \rho(x_k, y_k|x_{k-1}^{(i)}) \) is first assumed to be jointly Gaussian with the following approximated mean \( \mu \) and covariance \( \Sigma \):

\[
\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} E[x_k|x_{k-1}^{(i)}] \\ E[y_k|x_{k-1}^{(i)}] \end{bmatrix} = \begin{bmatrix} f(x_{k-1}^{(i)}) \\ g(f(x_{k-1}^{(i)})) \end{bmatrix},
\]

(16)

\[
\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} \text{Var}(x_k|x_{k-1}^{(i)}) & \text{Cov}(x_k, y_k|x_{k-1}^{(i)}) \\ \text{Cov}(y_k, y|x_{k-1}^{(i)}) & \text{Var}(y_k|x_{k-1}^{(i)}) \end{bmatrix} \begin{bmatrix} P \\ \text{JP} \end{bmatrix} + \begin{bmatrix} P \text{JT} \\ \text{JP} \text{JT} \end{bmatrix} R,
\]

(17)

where \( \tilde{y}_k \) is the approximation of \( y_k \) by the first-order Taylor expansion at \( f(x_{k-1}^{(i)}) \) with the Jacobian \( J \in \mathbb{R}^{N_y \times N_x}. \)

Then by conditioning \( \rho(x_k, \tilde{y}_k|x_{k-1}^{(i)}) \) on the recent measurement \( y_k \) via the conditional distribution formula for multivariate Gaussian, \( \rho(x_k|x_{k-1}^{(i)}, y_k) \) is approximated as \( N(m_k, \Sigma_k) \) where

\[
m_k = \mu_1 + \Sigma_{12}(\Sigma_{22})^{-1}(y_k - \mu_2),
\]

(18)

\[
\Sigma_k = \Sigma_{11} - \Sigma_{12}(\Sigma_{22})^{-1}\Sigma_{12}^{T}.
\]

(19)

If \( g \) in (15) is linear, then \( \rho(x_k|x_{k-1}^{(i)}, y_k) \) is exactly determined without approximation as the case of the Kalman filter. In the subsequent subsections, we show how we can apply this Gaussian importance function approach to our \( \text{SL}(3) \) case.

5.2 Gaussian Distributions on \( \text{SL}(3) \)

We first consider proper notions of Gaussian distributions on \( \text{SL}(3) \). Definitions of Gaussian distributions on general Riemannian manifolds are given in, e.g., [17] and [42]. These coordinate-free characterizations, while geometrically appealing, are much too computationally involved to be used for template-based visual tracking, where fast computation is essential. Instead, we make use of the exponential coordinates discussed in Section 3.1. Note that Gaussian distributions on the Lie algebra can be well-defined because the Lie algebra is a vector space. We therefore construct Gaussian distributions on \( \text{SL}(3) \) as the exponential of Gaussian distributions on \( \text{sl}(3) \) provided that the covariance values for Gaussian distributions on \( \text{sl}(3) \) are sufficiently small enough to guarantee the local diffeomorphism property of the exponential map. The exponential coordinates are preferable to other coordinates, since (locally in a neighborhood of the identity) the minimal geodesics are given by exponentials of the form \( \exp(xt) \), where \( x \in \text{sl}(3) \) and \( t \in \mathbb{R} \); noting that minimal geodesics are the equivalent of straight lines on curved spaces, the exponential map has the desirable property of mapping “straight lines” to “straight lines”.

With this background, we approximately construct a Gaussian distribution on \( \text{SL}(3) \) as

\[
N_{\text{SL}(3)}(X; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^3 \det(\Sigma)}} \exp\left( -\frac{1}{2} \epsilon^T \Sigma^{-1} \epsilon \right),
\]

(20)

where \( \epsilon \in \mathbb{R}^3 \) is sampled from the zero-mean Gaussian distribution on \( \text{sl}(3) \) with covariance \( \Sigma \).

This definition can be regarded as an approximation of the one in [7], where we use the exponential map instead of the Riemannian exponential map as done in the case of the \( \text{SL}(3) \) sample mean in Section 4.3. On the rotation group, where a bi-invariant metric exists, the definition of Gaussian distribution as given in [7] is identical to ours. We remark that the underlying notion behind our construction of Gaussian distributions on Lie groups is quite similar to the nonlinear mean shift concept on Lie groups as given in [54], in which mean shift on a Lie group is realized via the exponential mapping of ordinary vector space mean shift on the Lie algebra.

5.3 \( \text{SL}(3) \) Case

To make our problem more tractable, we apply the BCH formula to (7) using only the first two terms of (3). The resulting approximated state equation is given as

\[
X_k = f(X_{k-1}) \cdot \exp(v_{\text{sl}(3)}^{-1}(dW_k)) \sqrt{\Delta t},
\]

(22)

where \( f(X_{k-1}) = X_{k-1} \cdot \exp(A_{k-1}) \). Comparing (22) with the definition of random sampling from Gaussian on \( \text{SL}(3) \) in (21), we can easily identify that \( \rho(X_k|X_{k-1}^{(i)}) \) corresponds to \( N_{\text{SL}(3)}(X_k; A_{k-1}Q) \), where \( X_k^* = f(X_{k-1}^{(i)}) \) and \( Q = P\Delta t \). Then we seek to obtain Taylor expansion of the measurement equation (10) at \( X_k^* \).

As discussed in Section 3.1, the neighborhood of \( X_k^* \) is parametrized by the exponential coordinates as \( X_k^*(u) = X_k \cdot \exp(v_{\text{sl}(3)}(u)) \). Thus the first-order Taylor expansion of (10) at \( X_k^* \) can be realized by differentiating \( g(X_k^*(u)) \) with respect to \( u \). The exponential coordinate-based approach to obtain the Taylor expansion can also be found in the literature on optimization methods on Lie groups [38], [56], [58].

With the Jacobian \( J \) of \( g(X_k^*(u)) \) evaluated at \( u = 0, y_k \) at \( X_k^* \) can be approximated as

\[
\tilde{y}_k = g(X_k^*) + Ju + n_k.
\]

(23)

1. For notational convenience we shall also use \( g(X_k) \) instead of \( g(T(p); \mathcal{I}_k(W(p; X_k))) \).
Note that $\tilde{y}_k$ in (23) is essentially an equation of $u$ representing the local neighborhood of $X_k^*$. With a slight abuse of notation, we let the same $u$ also denote the random variable of Gaussian distributions on $s(3)$ for $\rho(X_k^*|X_{k-1}^*) = N_{SL(3)}(X_k^*, Q)$, i.e., $\rho(u) = N(0, Q)$. Then analysis of $\rho(X_k^*, \tilde{y}_k|X_{k-1}^*)$ becomes equivalent to that of $\rho(u, \tilde{y}_k)$ with $X_k^*$.

We now regard $\rho(u, \tilde{y}_k)$ to be jointly Gaussian with the following mean $\mu$ and covariance $\Sigma$:

$$
\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} E[u] \\ E[\tilde{y}_k] \end{bmatrix},
$$

$$
\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} \text{Var}(u) & \text{Cov}(u, \tilde{y}_k) \\ \text{Cov}(u, \tilde{y}_k)^T & \text{Var}(\tilde{y}_k) \end{bmatrix}.
$$

Then the conditional distribution $\rho(u|\tilde{y}_k = y_k)$ is determined as $N(\pi, \Sigma_k)$, with

$$
\pi = \mu_1 + \Sigma_{12}(\Sigma_{22})^{-1}(y_k - \mu_2),
$$

$$
\Sigma_k = \Sigma_{11} - \Sigma_{12}(\Sigma_{22})^{-1}\Sigma_{12}^T.
$$

The above implies that $\rho(u)$ on $s(3)$ used for sampling $X_k$ from $N_{SL(3)}(X_k^*, Q)$ is transformed from $N(0, Q)$ to $N(\pi, \Sigma_k)$ by incorporating $y_k$.

The sampling of $X_k$ with $X_k^*$ and the updated $N(\pi, \Sigma_k)$ can be realized as

$$
X_k^* \sim \text{exp}(v_{\text{a}(3)}(\pi + \epsilon)),
$$

where $\epsilon \sim N(0, \Sigma_k)$. Again by using the BCH formula approximation, (28) can be rewritten as

$$
X_k^* \sim \text{exp}(v_{\text{a}(3)}(\pi)) \cdot \text{exp}(v_{\text{a}(3)}^{-1}(\epsilon)).
$$

This implies that $\rho(X_k^*|X_{k-1}^*, \tilde{y}_k = y_k)$ is eventually given by $N_{SL(3)}(m_k, \Sigma_k)$ where $m_k = X_k^* \cdot \text{exp}(v_{\text{a}(3)}(\pi))$. That is, the optimal importance function $\rho(X_k^*|X_{k-1}^*, \tilde{y}_k = y_k)$ is finally approximated as $N_{SL(3)}(m_k, \Sigma_k)$ with

$$
\pi = \mu_1 + \Sigma_{12}(\Sigma_{22})^{-1}(y_k - \mu_2),
$$

$$
m_k = X_k^* \cdot \text{exp}(v_{\text{a}(3)}(\pi)),
$$

$$
\Sigma_k = \Sigma_{11} - \Sigma_{12}(\Sigma_{22})^{-1}\Sigma_{12}^T.
$$

Here, $\mu_1$ and $\Sigma_{11}$ are immediately given by $0$ and $Q$, respectively, because $\rho(u) = N(0, Q)$. The remaining $\mu_2$, $\Sigma_{12}$, and $\Sigma_{22}$ are also straightforwardly determined as $g(X_k^*)$, $QJ^T$, and $JQJ^T + R$, respectively. Particle sampling from the obtained $N_{SL(3)}(m_k, \Sigma_k)$ is realized by (21), and the denominator of (11) can be straightforwardly calculated by (20).

5.4 Jacobian Calculation

To obtain the Jacobian $J$ of $g(X_k^*(u))$ in (23), we employ the chain rule

$$
J_i = \frac{\partial g_i(X_k^*(u))}{\partial u} \bigg|_{u=0} = \frac{\partial g_i(X_k^*)}{\partial X_k^*} \cdot \nabla_u I_k(p_u'),
$$

where $p' = W(p; X_k^*)$, $p_u' = W(p; X_k^*(u))$, and $\nabla_u I_k(p_u') = \frac{\partial x(p_u')}{\partial u}$. Here, $J_i$ and $g_i$ represent the $i$th row of $J$ and $i$th element of $g$, respectively. $\frac{\partial g_i(X_k^*)}{\partial X_k^*}$ and $\frac{\partial x(p_u')}{\partial u}$ are given in [11] and [30], respectively.

$$
\nabla_u I_k(p_u') \text{ in (33)} \text{ is related with the image gradient of } I_k, \text{ and can be further decomposed via the chain rule as}
$$

$$
\nabla_u I_k(p_u') = \nabla I_k(p') \cdot \frac{\partial p'}{\partial X_k^*} \cdot \frac{\partial X_k^*}{\partial u} \bigg|_{u=0},
$$

The first term $\nabla I_k(p')$ of (34) is the image gradient of $I_k$ calculated at the coordinates of $I_k$ and warped back to the coordinates of $I(p)$. The second term $\frac{\partial x(p_u')}{\partial u}$ of (34) can be obtained by differentiating (9) with respect to $a = (a_1, \ldots, a_9)\top$, where $X_k^*$ is given by

$$
X_k^* = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{bmatrix},
$$

the result is

$$
\begin{bmatrix}
\frac{p_3 - p_4}{p_3} \\ \frac{p_4 - p_5}{p_4} \\ 0 \\ \frac{p_5 - p_6}{p_5} \\ \frac{p_6 - p_7}{p_6} \\ 0 \\ \frac{p_7 - p_8}{p_7} \\ \frac{p_8 - p_9}{p_8} \\ 0 \\ \frac{p_9 - p_1}{p_9} \\
\end{bmatrix},
$$

with $p_1 = a_1p_x + a_4p_z + a_7$, $p_2 = a_3p_x + a_5p_y + a_8$, and $p_3 = a_2p_x + a_6p_y + a_9$. The last term $\frac{\partial x(p_u')}{\partial u}$ of (34) is a $9 \times 8$ matrix each of whose columns is given by $X_k^\top \cdot E_i$ with $E_i$ in (4) represented as a vector in $\mathbb{R}^8$ by $a$.

6 ENHANCEMENT OF ACCURACY AND EFFICIENCY

6.1 Iterative Gaussian Importance Function Generation

In [30], it was shown that Gaussian importance functions by local linearization yields better results for affine motion tracking than the state transition density. However, such Gaussian importance functions would poorly approximate the optimal importance function for projective motion tracking, whose warping function is nonlinear. Although the second-order approximation using the Hessian is possible [48], the exact Hessian computation for our case is time-consuming and its approximation as in [11] would not much enhance the accuracy.

To increase the accuracy of Gaussian importance functions by local linearization, we can apply iteratively a local linearization-based approximation such as the iterated extended Kalman filter (EKF) [5]. However, when $X_k^*$ ventures out of the basin of convergence, it can result in importance functions even worse than the state transition density because of possible divergence.
To overcome this divergence problem, we select the best among all Gaussian importance functions obtained at each iteration by the following two measures:

\[ s_1 = y_k - g(m_{k,j}), \quad (37) \]
\[ s_2 = v(\log(X_k^{-1}m_{k,j})), \quad (37) \]

where \( m_{k,j} \) denotes the mean of the Gaussian importance function obtained at the \( j \)-th iteration. \( s_1 \) represents how dissimilar the predicted measurement of \( m_{k,j} \) is to \( y_k \), while \( s_2 \) represents how far \( m_{k,j} \) is from \( X_k \). To balance the influences of \( s_1 \) and \( s_2 \), we define a criterion \( C \) in the form of a Gaussian using \( R \) in (10) and \( Q = P\Delta t \) in (22) as

\[ C(j) = \exp\left(-\frac{1}{2} s_1^T R^{-1} s_1\right) \exp\left(-\frac{1}{2} s_2^T Q^{-1} s_2\right). \quad (38) \]

Among all Gaussian importance functions obtained at each iteration, we select the best maximizing \( C \). The selected Gaussian importance function can be intuitively understood as the one most similar to \( y_k \) while minimizing divergence from \( X_k \). The algorithm is given in Algorithm 2.

**Algorithm 2** An algorithm for selective iterative Gaussian importance function generation

1. Set \( m_{k,0} = X_k^T \) and \( \Sigma_{k,0} = Q = P\Delta t \).
2. Calculate \( C(0) \) with \( m_{k,0} \) via (38).
3. For \( j = 1, \ldots, N_{\text{iter}} \),
   a. Calculate the Jacobian \( J_j \) of \( y_k \) at \( m_{k,j-1} \).
   b. Set \( \mu_1 = m_{k,j-1}, \mu_2 = g(m_{k,j-1}), \Sigma_{11} = \Sigma_{k,j-1}, \Sigma_{12} = \Sigma_{k,j-1}J_j^T + R, \Sigma_{22} = J_j\Sigma_{k,j-1}J_j^T + R, \Sigma_{22} \).
   c. Obtain \( m_{k,j} \) and \( \Sigma_{k,j} \) via (30), (31), and (32).
   d. Calculate \( C(j) \) with \( m_{k,j} \) via (38).
4. Determine the importance function as \( N_{\text{iter}}(m_{k,j}, \Sigma_{k,j}) \) with \( j = \arg \max_{j \in \{0, 1, \ldots, N_{\text{iter}}\}} C(j) \).

In this case, \( w_k^{(i)} \) are calculated via (11) with

\[ \rho(X_k^{(i)} | X_{k-1}) = \frac{1}{\sqrt{(2\pi)^d\det(Q)}} \exp\left(-\frac{1}{2} \overline{d}_1 Q^{-1} \overline{d}_1\right), \quad (39) \]

where \( \overline{Q} = \Sigma_{k-1}^{-1} \) and \( \overline{d}_1 = v(\log(m_{k-1,j}^{-1}X_k^{(i)})) \), in light of the fact that \( \rho(X_k^{(i)}) \) has been deterministically transformed to \( N_{\text{iter}}(m_{k,1}, \Sigma_{k,1}) \) before calculation of \( J_1 \).

Note that in Algorithm 2, the uncertainty of \( \mu_1 \) decreases as the iteration proceeds, because \( \Sigma_{11} \) at the \( j \)-th iteration is given by \( \Sigma_{k,j-1} \). In the general iterated EKF, \( \Sigma_{11} \) is fixed to the initial value \( Q \) in our case) because the decrease in the determinant of \( \Sigma_{11} \) during iteration implies that there are multiple \( y_k \)'s at one time step, and it does not apply to general cases. However, since \( y_k \) in our framework is always given by the value when \( I_k(W(p; X_k)) = T(p) \), we can freely assume multiple \( y_k \)'s at one time step and gradually decrease the uncertainty of \( \mu_1 \). This can contribute to accurately track objects with a limited number of particles.

Though the proposed iterative Gaussian importance function generation method can approximate the optimal importance function more accurately than by local linearization, the required iteration likely lowers overall computational efficiency. In the subsequent subsections, we describe a collective set of methods that together increase computational efficiency.

### 6.2 Inverse Formulation of Jacobian Calculation

In analogy with the inverse approach of [3], we utilize the inverse formulation of the Jacobian calculation. The idea is that the derivative of \( g(T(p), I_k(W(p; X_k)(u))) \) with respect to \( u \) is equivalent to that of \( g(T(W(p; X_0(-u)), I_k(p')) \) where \( X_0 \) is the initial identity matrix.

Denoting \( W(p; X_0(-u)) \) by \( p''_u \), the Jacobian \( J \) of \( g(T(p''_u), I_k(p')) \) at \( u = 0 \) is given by

\[ J_u = \frac{\partial g(T(p), I_k(p'))}{\partial T(p)} \nabla_u T(p''_u), \quad (40) \]

where \( \nabla_u T(p''_u) = \frac{\partial T(p''_u)}{\partial u} \bigg|_{u=0} \). Note that \( \nabla_u T(p''_u) \) is constant and can be pre-computed because it only depends on \( T(p) \) and \( X_0 \), which are always constant. To obtain \( J \) of \( g(T(p''_u), I_k(p')) \), we only need to evaluate \( \frac{\partial g(T(p), I_k(p'))}{\partial T(p)} \) with \( T(p) \) and \( I_k(p') \).

### 6.3 Template Resizing

The computational complexity of our framework increases with the size of \( T(p) \) because the measurement process involves image interpolation. To avoid the increased computational complexity with large templates, we resize the template to the specified size as described in Fig. 3a.

The affine transformation matrix \( A \) for coordinate transformation only affects the actual image interpolation, all calculations including the Jacobian can be done on the transformed template coordinates \( \hat{p} \) without additional care related to template resizing.

The template resizing in our framework has further significance beyond just reducing computational complexity. Consider two initial templates with different sizes shown in Fig. 3b in solid lines. The dotted quadrilaterals in Fig. 3b represent the transformed templates with

\[ X_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.005 & 0 & 1 \end{bmatrix}. \]
The shapes of transformed templates are different from each other despite the same state. This means we should set the covariance $P$ in (7) differently according to the template size even with the same expected object movement. By resizing the template, we can set $P$ reasonably according to the expected object movement regardless of the original template size. To reduce the computational complexity, one can alternatively use a subset of template pixels as in [13]. However, it still suffers from the problem described in Fig. 3b and thus is not suitable for our framework.

### 6.4 Use of Parent-Child Particles

To reduce computational complexity further, we propose to use parent-child particles as described in Fig. 4. In Fig. 4, blue and green circles denote $N$ parent particles $X(k)$ to which the iterative Gaussian importance function generation is applied. Green circles are $N_c$ child particles $X(i,j)$ sampled from each Gaussian importance function $N_{SL(3)}(X(k))$ for each parent particle $X(k)$. Thus, $N \cdot N_c$ particles are sampled while the iterative Gaussian importance function generation is applied only $N$ times. After weighting all $N \cdot N_c$ particles, we resample only $N$ parent particles using residual systematic resampling [10], which is most suitable when the number of particles varies during resampling.

Theoretically, it is possible to reduce the number of particles during resampling as long as the true filtering density is well represented by a reduced set of particles. Our underlying assumption about the use of parent-child particles is that the true filtering density will have a single mode for projective motion tracking of a single object. If a density is single-mode, it can be represented well by smaller number of particles than the case of multi-mode distributions.

One of empirical measures for how well a probability density is represented by particles and their weights is the number of effective particles defined as $N_{eff} = \sum_i \left( \frac{\bar{w}_i(i)}{\bar{w}_i(i)} \right)^2$ [15], where $\bar{w}_i(i)$ are the normalized weights. The number of effective particles varies between one (in the worst case) and the total number of particles (in the ideal case). If the number of effective particles does not change much by the use of the proposed parent-child particles, it can be said that the parent-child particles does not alter the statistical property of particle filtering. In Section 7.2, we experimentally verify that the use of parent-child particles is effective in reducing computational complexity without altering the statistical property of particle filtering by showing the effective numbers of particles for different pairs of $N$ and $N_c$ of the same total number of particles are consistent.

Now we present the overall algorithm of projective motion tracking via particle filtering on $SL(3)$ with Gaussian importance functions in Algorithm 3.

### Algorithm 3 An algorithm for projective motion tracking via particle filtering on $SL(3)$ with Gaussian importance functions

1. **Initialization**
   a. Set $k = 0$.
   b. Set number of parent and child particles as $N$ and $N_c$.
   c. For $i = 1, \ldots, N$,
      - Set $X(0) = I$, $A(i) = 0$.
      - For $j = 1, \ldots, N_c$, set $w(j) = \frac{1}{N_c}$.
   d. Compute $\nabla_u T(p_k)$.

2. **Importance sampling step**
   a. Set $k = k + 1$.
   b. For $i = 1, \ldots, N$,
      - Compute $N_{SL(3)}(m_k, \Sigma_k)$ via Algorithm 2 with $\nabla_u T(p_k)$.
      - For $j = 1, \ldots, N_c$,
         - Draw $X(k+i, j) \sim N_{SL(3)}(m_k, \Sigma_k)$ via (21) and compute $A(k+i, j)$ with (8).
         - Calculate the weights $w(k+j) = (11)$ and (39).
   c. Normalize the weights as $\bar{w}(k+j) = \frac{w(k+j)}{\sum_i w(k+i)}$.

3. **Selection step (resampling)**
   a. Resample using residual systematic resampling from 
      \{X(k+i, j), A(k+i, j), i = 1, \ldots, N, j = 1, \ldots, N_c\} according to 
      \{w(k+i), \bar{w}(k+i), i = 1, \ldots, N\} to obtain i.i.d.
      \{X(k), A(k), i = 1, \ldots, N\}.
   b. Calculate $X(k)$ from \{X(k), X(k+1), \ldots, X(k+N)\} via Algorithm 1.
   c. For $i = 1, \ldots, N$,
      - For $j = 1, \ldots, N_c$, set $w(k+j) = \frac{1}{N_c}$.

4. **Go to the importance sampling step**

### 6.5 Adaptation to Affine Motion Tracking

We now show how to adapt our projective motion tracking framework to the group of affine transformations. The major difference in performing affine motion tracking via particle filtering on $Aff(2)$ is in the warping function $W(p, X_k)$. All other details, such as the state equation and Gaussian importance functions on $Aff(2)$, can be easily adapted from the $SL(3)$ case.

The warping function $W(p; X_k)$ for an affine transformation with

$$X_k = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$
is defined as

$$W(p; X_k) = \begin{bmatrix} a_1 p_x + a_2 p_y + a_5 \\ a_2 p_x + a_4 p_y + a_6 \end{bmatrix}.$$  

Accordingly, the second term \( \frac{\partial W}{\partial X} \) of (34) for \( \text{Aff}(2) \) is given by differentiating \( W(p; X_k^*) \) with respect to \( a = (a_1, \ldots, a_6) \), with \( X_k^* \) defined as

$$X_k^* = \begin{bmatrix} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \\ 0 & 0 & 1 \end{bmatrix} ;$$

the result is

$$\begin{bmatrix} p_x & 0 & p_y & 0 & 1 & 0 \\ 0 & p_x & 0 & p_y & 0 & 1 \end{bmatrix}.$$  

The last term of (34) for \( \text{Aff}(2) \) becomes a \( 6 \times 6 \) matrix.

We can also perform similarity motion tracking via particle filtering on \( \text{Aff}(2) \) by setting the state covariance as

$$P = \text{diag}(\sigma_1^2, \ldots, \sigma_3^2),$$

with nearly zero \( \sigma_1 \) and \( \sigma_3 \). Note that this is not possible with particle filtering on \( \text{SL}(3) \) because isotropic scaling is not present in (4).

7 EXPERIMENTS

In this section, we demonstrate the feasibility of our particle filtering-based probabilistic visual tracking framework through extensive experiments using various test video sequences. For all test sequences, we perform projective motion tracking by particle filtering on \( \text{SL}(3) \) with C implementation on a desktop PC equipped with Intel Core i7 2600K CPU and 16 GB RAM. The experimental results can be also seen in the supplementary video.

7.1 Overall Tracking Performance

In order to demonstrate the overall performance of the proposed tracking framework, we use the following video sequences of planar objects: “Towel” and “MousePad” from [65], “Bear” and “Book” from [50], and “Tea” and “Acronis” of our own. For non-rigid (originally) planar objects, we use “Paper” from [50] and “Magazine” from [39] while we use “Doll” and “David” from [46] for rigid non-planar objects.

We crop the initial object template \( T_{\text{init}}(p) \) from the first frame manually and resize it to a size of \( 40 \times 40 \). We set \( (N, N_c) = (40, 10) \) for parent-child particles and \( N_{\text{iter}} = 5 \) for the iterative Gaussian importance function generation. The PCA bases are incrementally updated at every fifth frame after the initial PCA basis extraction from the first 15 frames. The state covariance \( P \) and measurement covariance \( R \) are tuned specifically for each sequence to yield satisfactory tracking results. The parameters tuned for each sequence are summarized in the separate supplementary document, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.170. Our C implementation with this setting runs in about 15 fps. By the inverse formulation of the Jacobian calculation, the tracking speed is increased from 8 to 15 fps. There is no noticeable performance degradation by template resizing.

Fig. 5 shows the tracking results by our framework for the test sequences of planar objects. Please also watch the supplementary video for better understanding of the tracking performance of our framework. Figs. 5a and 5b show that our framework accurately tracks the planar objects in “MousePad” and “Towel” despite the motion blur caused by fast and large inter-frame object motion. Our framework is also able to track the planar object in “Tea” consistently despite temporal occlusion as Fig. 5c shows. The robustness to motion blur and temporal occlusion can be considered as originating from the stochastic nature of particle filtering.

The effect of conjunctive use of NCC and PCA in our measurements can be verified from the results for “Bear” in Fig. 5d. When using only \( g_{\text{NCC}} \) for the measurement (red lines in Fig. 5d), tracking failure occurs because of local illumination changes. On the contrary, our framework using both \( g_{\text{NCC}} \) and \( g_{\text{PCA}} \) (yellow lines in Fig. 5d) accurately tracks the object by accounting for the local illumination change through an incrementally updated PCA model.

The effect of outlier detection can be identified from the tracking result for “Acronis” in Fig. 5e. Without outlier detection (red lines in Fig. 5e), our framework fails to track the object because of large NCC error induced by severe local illumination changes. On the contrary, by detecting outliers whose PCA reconstruction error is above the threshold and excluding them in calculating \( g_{\text{NCC}} \) (yellow lines in Fig. 5e), our framework is able to successfully track the object.

2. The video is also available at http://cv.snu.ac.kr/jhkwon/tracking_video.avi.
Fig. 6. Tracking results by our framework for (a) “Paper” and (b) “Magazine”, which contain non-rigid object images, and (c) “Doll” and (d) “David”, which contain rigid non-planar object images.

Fig. 5f shows the tracking failure for “Book” even with the conjunctive use of NCC and PCA with outlier detection. In “Book”, severe specular reflection changes the object appearance so rapidly that even the incremental PCA model cannot capture such appearance changes adequately, which leads to the tracking failure.

Although our framework is inherently best suited for tracking rigid planar objects, it can also yield satisfactory tracking results up to a certain extent for non-rigid and non-planar object cases. Figs. 6a and 6b show the tracking results for “Paper” and “Magazine” in which originally planar objects are being deformed non-rigidly. For both sequences, our framework estimates planar homographies locally optimal in minimizing the difference to the initial object template. We can confirm that such homography estimation is reasonably acceptable via a visual check of the tracking results in the supplementary video, available online.

Figs. 6c and 6d show the tracking results for “Doll” and “David” in which rigid non-planar objects are changing their poses frequently under the varying illumination condition. In spite of appearance changes owing to object pose and illumination changes, our framework estimates the object image region quite accurately for both sequences. These successful tracking results can be mainly attributed to \( g_{PCA} \) in the measurement as such appearance changes are captured by the incrementally updated PCA bases. The tracking results by our framework for “Doll” and “David” are in good agreement with the results in [46] where the incremental PCA model for visual tracking is originally proposed.

7.2 Tracking Performance Analysis with Parameter Variations

In this subsection, we analyze the tracking performance of our framework with parameter variations. Fixing the template size and \( N_{iter} \) to 40 \( \times \) 40 and five, respectively, we vary \( (N, N_c) \) for parent-child particles, state covariance \( P \), and measurement covariance \( R \). For this experiment, we use our own video sequences, “Mascot”, “Bass”, and “Cube” shown in Fig. 7, with the ground truth corner points extracted manually. Since there is little local illumination change in the test sequences, we use only \( g_{NCC} \) for the measurement in this experiment to simplify the performance analysis.

In order to analyze the tracking performance quantitatively, we consider the followings: “Time”, “Error”, “Std”, and \( N_{eff} \). “Time” is the average computation time per frame in milliseconds. With the ground truth corner points \( \hat{c}_k \) and estimated corner points \( \hat{c}_k' \) at time \( k \), we calculate the RMS error as
\[
\epsilon_k = \sqrt{\frac{1}{K} \sum_{j=1}^{K} (\hat{c}_{k,j} - \hat{c}_{k,j}')^2} / s_k,
\]
where \( s_k \) is the ground truth object area at time \( k \). The purpose of error normalization by the object area is to prevent the error from being scaled according to the object scale change. “Error” and “Std” are the mean and standard deviation of \( \epsilon_k \). As aforementioned, \( N_{eff} \) represents how well a probability density is represented by particles and their weights. The larger \( N_{eff} \) is, the stabler the tracking is.

We first vary \( (N, N_c) \) for parent-child particles, which are the most important factors for tracking performance in practice. We fix \( P \) and \( R \) to the values to yield satisfactory tracking results for each sequence. \( P \) and \( R \) tuned for each sequence can be found in the separate supplementary document, available online. There are 12 combinations of \( (N, N_c) \) corresponding to 200, 400, and 600 particles in total.

Table 1 shows combinations of \( (N, N_c) \) and corresponding tracking performance. The numbers in Table 1 are obtained by averaging over 10 independent runs for each combination of \( (N, N_c) \). It is clear from Table 1 that as the total number of particles increases, “Error” and “Std” decrease along with the increase of “Time”. On the other hand, within the same total number of particles, there is a trend that as \( N_c \) increases, “Error” and “Std” increase along with the decrease of “Time”.

We can verify the effectiveness of the use of parent-child particles by specifically comparing the cases of \( (N, N_c) = (400,1) \) and \( (N, N_c) = (60,10) \). For all test sequences, “Time” for \( (N, N_c) = (60,10) \) is much less than that for \( (N, N_c) = (400,1) \) while both cases yield similar “Error” and “Std”. This implies that we can enhance the computational efficiency considerably without sacrificing the tracking accuracy by employing the parent-child particles. It is also worth to note that in Table 1 the values of \( N_{eff} \) for different values of \( N_c \) are quite similar within the same total number of particles, i.e., the same \( N \cdot N_c \). This means that the use

3. Note that the tracking speed gain is not strictly inversely proportional to the number of parent particles, because particles copied from the same particle during resampling can share the same importance function without re-computation.
of parent-child particles does not alter the statistical property of particle filtering and tracking stability.

It is necessary in practice to find the appropriate combination of \( (N, N_c) \) that balances the tracking accuracy and computation time. Fig. 8 shows the scatter plots of “Error” and “Time” for each combination of \( (N, N_c) \). From Fig. 8, we can identify that our choice of \( (N, N_c) = (40, 10) \) for the experiments in Section 7.1 is reasonable because green circles in Fig. 8 representing the cases of \( (N, N_c) = (40, 10) \) can be regarded as points of good compromise between “Error” and “Time”.

We now check the tracking performance changes according to the changes of the state and measurement covariances, \( P \) and \( R \). For each sequence, we vary \( P \) tuned previously as \( P_1 = 0.8^2 P, P_2 = 0.9^2 P, P_3 = P, P_4 = 1.1^2 P \), and \( P_5 = 1.2^2 P \) and similarly for \( R \). We set \( (N, N_c) = (40, 10) \) for this experiment. Tables 2 and 3 summarize the tracking performance comparison between different state and measurement covariances. The numbers in Tables 2 and 3 are obtained by averaging over 10 independent runs for each case.

It is hard to find out certain trends from Table 2 except one that \( N_{\text{eff}} \) decreases as \( P \) increases. This can be intuitively understood: a larger \( P \) will result in larger variations of sampled SL(3) particles and accordingly smaller number of sampled particles will be similar to the ground truth. In practice, it is important to set \( P \) to cover a possible expected range of the object motion. As we show in Fig. 3, our template resizing approach is important in setting \( P \) correctly regardless of the initial size of the object image. Note that the values of \( P \) for various test video sequences given in the separate supplementary document, available online, are in very similar ranges.

There is a clear trend in Table 3 that “Error”, “Std”, and \( N_{\text{eff}} \) all increase as \( R \) increases. This can be intuitively understood: if \( R \) is set to the very large value, all

### TABLE 1

<table>
<thead>
<tr>
<th>Mascot</th>
<th>((N, N_c))</th>
<th>((200,1))</th>
<th>((40,5))</th>
<th>((20,10))</th>
<th>((10,20))</th>
<th>((400,1))</th>
<th>((80,5))</th>
<th>((40,10))</th>
<th>((20,20))</th>
<th>((600,1))</th>
<th>((120,5))</th>
<th>((60,10))</th>
<th>((30,20))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>25.61</td>
<td>14.99</td>
<td>11.78</td>
<td>9.76</td>
<td>50.92</td>
<td>29.74</td>
<td>25.47</td>
<td>19.31</td>
<td>76.16</td>
<td>44.42</td>
<td>35.10</td>
<td>28.96</td>
<td>35.10</td>
</tr>
<tr>
<td>Error</td>
<td>0.0260</td>
<td>0.0262</td>
<td>0.0266</td>
<td>0.0280</td>
<td>0.0251</td>
<td>0.0253</td>
<td>0.0283</td>
<td>0.0249</td>
<td>0.0299</td>
<td>0.0251</td>
<td>0.0253</td>
<td>0.0255</td>
<td>0.0255</td>
</tr>
<tr>
<td>Std</td>
<td>0.0155</td>
<td>0.0153</td>
<td>0.0059</td>
<td>0.0071</td>
<td>0.0050</td>
<td>0.0051</td>
<td>0.0053</td>
<td>0.0049</td>
<td>0.0049</td>
<td>0.0051</td>
<td>0.0053</td>
<td>0.0054</td>
<td>0.0054</td>
</tr>
<tr>
<td>(N_{\text{eff}})</td>
<td>25.61</td>
<td>14.99</td>
<td>11.78</td>
<td>9.76</td>
<td>50.92</td>
<td>29.74</td>
<td>25.47</td>
<td>19.31</td>
<td>76.16</td>
<td>44.42</td>
<td>35.10</td>
<td>28.96</td>
<td>35.10</td>
</tr>
</tbody>
</table>

### TABLE 2

<table>
<thead>
<tr>
<th>Mascot</th>
<th>(P)</th>
<th>(P_1)</th>
<th>(P_2)</th>
<th>(P_3)</th>
<th>(P_4)</th>
<th>(P_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0.0271</td>
<td>0.0261</td>
<td>0.0254</td>
<td>0.0252</td>
<td>0.0249</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>0.0061</td>
<td>0.0056</td>
<td>0.0055</td>
<td>0.0052</td>
<td>0.0051</td>
<td></td>
</tr>
<tr>
<td>(N_{\text{eff}})</td>
<td>57.26</td>
<td>56.94</td>
<td>54.55</td>
<td>51.24</td>
<td>47.35</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 3

<table>
<thead>
<tr>
<th>Mascot</th>
<th>(R)</th>
<th>(R_1)</th>
<th>(R_2)</th>
<th>(R_3)</th>
<th>(R_4)</th>
<th>(R_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0.0231</td>
<td>0.0241</td>
<td>0.0255</td>
<td>0.0269</td>
<td>0.0286</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>0.0047</td>
<td>0.0050</td>
<td>0.0054</td>
<td>0.0056</td>
<td>0.0063</td>
<td></td>
</tr>
<tr>
<td>(N_{\text{eff}})</td>
<td>42.06</td>
<td>49.05</td>
<td>54.01</td>
<td>59.91</td>
<td>64.08</td>
<td></td>
</tr>
</tbody>
</table>

---

Fig. 8. Scatter plots of “Error” and “Time” for each combination of \((N, N_c)\) in Table 1. The legend for (a) applies to all sub figures.
particles will have essentially the same weights regardless of their similarity to the ground truth. In this case, a larger $N_{\text{eff}}$ does not necessarily result in more stability in tracking. It should be understood that the comparison of different values of $N_{\text{eff}}$ is only meaningful with the same $P$ and $R$. In practice, we should set $R$ to balance the tracking accuracy and stability. The difficulty is that the proper value of $R$ depends on the object image characteristics as identified from the separate supplementary document, available online. Therefore, it would be an interesting topic to find a way to set a proper $R$ automatically based on object image properties such as gradient and texture.

7.3 Comparison with Different Importance Functions

We now compare the performance of our Gaussian importance functions with that of the state transition density (“STD”) and the Gaussian importance functions by local linearization (“LL”), which is employed in [30]. We implement “LL” also with the inverse formulation of the Jacobian calculation to make its computation time comparable to ours. We use the same test sequences as in Section 7.2. For “STD” and “LL”, we set $N$ as 400, 800, and 1,200 without the use of parent-child particles while we set $(N, N_c) = (40, 10)$ for ours. All the remaining parameters are the same as in Section 7.2.

Table 4 summarizes the performance comparison between different importance functions. The numbers in Table 4 are obtained by averaging over 10 independent runs for each case. For “STD” and “LL”, it is clear that as $N$ increases, “Error” and “Std” decrease along with the increase of “Time” and $N_{\text{eff}}$. Comparing the cases of “STD” and “LL” with the same $N$, we can confirm from the decreased “Error” and “Std” and increased $N_{\text{eff}}$ that the tracking performance is enhanced by the local linearization-based Gaussian importance function.

However, “LL” fails to outperform our framework for all test sequences in terms of “Error” and “Std”. Note that our framework yields the least “Error” and “Std” for all test sequences with the greatest $N_{\text{eff}}$. It is remarkable that $N_{\text{eff}}$ for our framework with 400 particles in total is even greater than that for “LL” with 1200 particles.

Fig. 9 shows the scatter plots of “Error” and “Time” for each case. We can infer from Fig. 9 that “STD” and “LL” with the same “Time” as ours would yield greater “Error” than ours, and they would require much greater “Time” to yield the same “Error” as ours. This can be better illustrated by the supplementary video, available online, which shows tracking results for “Mascot”, “Bass”, and “Cube” by each importance function with the similar computational complexity, i.e., $N = 1200$ for “STD”, $N = 600$ for “LL”, and $(N, N_c) = (40, 10)$ for ours. The result videos clearly indicate that our framework yields more accurate and stabler tracking results than “STD” and “LL” when run with the similar computational complexity.

7.4 Benchmark Test Using Metaio Data Set

We finally evaluate the tracking performance of our framework using the Metaio benchmark data set [35], which has been developed to evaluate template-based tracking algorithms. The benchmark data set is composed of the following four groups classified according to the object texture property: “Low”, “High”, “Repetitive”, and “Normal”. Each group possesses two different objects as shown in Fig. 10. For each object, there are five different video sequences focusing on different types of dynamic behaviors, namely, “Angle” for the camera viewing angle change, “Range” for the object image size change, “FastFar” for the fast moving object far from the camera, “FastClose” for the fast moving object close to the camera, and “Illumination” for the illumination change. Therefore, there are 40 test video sequences in total in the benchmark data set.

For each sequence, the initial coordinates of four corner fiducial points outside the object image region are given.
The goal is to estimate those fiducial point coordinates for the remaining frames. We manually extract the object template from the initial frame and perform projective motion tracking for the remaining frames. Then we estimate the fiducial point coordinates by transforming their initial coordinates using the estimated homographies. In [35], the tracking at a specific frame is considered to be successful if the root-mean-square (RMS) error of the estimated fiducial point coordinates is less than 10 pixels. There is no consideration of the object scale change in calculating the tracking error.

For experiments using the Metaio benchmark data set, we set \( (N, N_c) = (80, 10) \) to increase the tracking accuracy with the reasonable computational complexity. The template size is \( 40 \times 40 \) and \( N_{err} \) is set as five as before. The state covariance \( P \) and measurement covariance \( R \) are tuned specifically for each sequence to yield satisfactory tracking results. The parameter settings for each sequence are summarized in the separate supplementary document, available online. Although the video sequences in the Metaio benchmark data set are in color, we use their gray-scale versions.

We compare our tracking performance with five different template tracking methods: the ESM tracker based on nonlinear optimization [8], the L1 tracker based on particle filtering [4], and motion estimation by key-point matching using SIFT [36], SURF [6], and FERNS [40]. The ESM tracker and L1 tracker are generally considered as the state-of-the-art tracking methods in their own categories, i.e., nonlinear optimization and particle filtering. The main strength of the ESM tracker is a large basin of convergence achieved by using both forward and inverse gradients of the object template while that of the L1 tracker is the robustness to object appearance changes achieved by the advanced object appearance model based on sparse representation. SIFT, SURF, and FERNS are also widely used for key-point matching because of their ability to extract distinctive local features stably.

We reuse the tracking results of the ESM tracker and key-point matching-based motion estimation methods reported in [35] without re-running them. For the L1 tracker, we use the source code provided by the authors of [4]. In order to achieve higher tracking accuracy of the L1 tracker, we use 2,000 particles, which is rather many, with the state covariance values somewhat increased from the default and tuned differently for each sequence. The other parameters remain unchanged from the default.

It is rather meaningful to compare our tracking performance with the ESM tracker because it also treats \( SL(3) \) as an optimization variable. That is, it is a comparison between deterministic and probabilistic template-based visual tracking algorithms employing the same Lie group-based representation. The comparison with the ESM tracker can verify our claim that the well-designed particle filtering can be more robust to the problem of local optima than the deterministic optimization.

It is also meaningful to compare our tracking performance with the L1 tracker because it also employs particle filtering as the state estimation tool as ours but only with the state transition density-based importance function. Furthermore, the L1 tracker can estimate the object motion up to affine transformations. Therefore, the comparison with the L1 tracker can verify whether our two contributions, namely the use of the Gaussian importance functions and the extension of our affine motion tracking framework of [30] to projective motion tracking, are really important for highly accurate template tracking.

### 7.4.1 Overall Performance

Bar graphs in Fig. 11 summarize the evaluation of tracking results by each tracking method in terms of the tracking success rate. Table 5 shows the average tracking success rates for each specific sub-groups. Our framework yields higher tracking success rates for “Repetitive” and “Normal” than for “Low”.

The reason of the lower success rate for “High” is due to the quite low success rate for the grass image in Fig. 10d in which no distinct structure exists.

On the other hand, our framework yields higher tracking success rates for “Angle”, “Range”, and “Illumination” than for “FastFar” and “FastClose”. The reason of the lower success rate for “FastFar” is severe motion blur caused by fast camera motion while that for “FastClose” is partially visible objects due to the closeness to the camera. The higher success tracking rates for “Illumination” implies that our measurement using NCC and PCA can properly deal with illumination changes.

In comparison with other template tracking methods, our framework yields the highest tracking success rate averaged over all 40 test sequences as Table 5 shows. Specifically, our framework yields the highest tracking success rates for “Low”, “Repetitive”, “Normal”, “Angle”, and “FastFar” while our framework is the second best for “High”, “Range”, and “Illumination”, following SIFT. The reason of the best tracking success rates of SIFT for “Range” and “FastClose” is it estimates homographies only using visible key-points unlike our framework. Overall, the performance ranking is as follows: ours (74.53 percent), SIFT (64.35 percent), ESM (47.28 percent), FERNS (44.92 percent), SURF (31.95 percent), and L1 (21.10 percent).

### 7.4.2 Comparison with ESM Tracker

Our framework consistently outperforms the ESM tracker for all sub-groups. The performance difference for “Repetitive” and “FastFar” well characterizes the
different inherent properties of deterministic optimization and stochastic particle filtering. The lower tracking success rate of the ESM tracker for “Repetitive” is due to being trapped in the local optima with similar appearances while our framework suffers less from such local optima problems because of the stochastic nature of particle filtering. Although both methods do not employ a specific model of motion blur, our framework yields far better tracking success rates than the ESM tracker for “FastFar” where severe motion blur exists. This can be also attributed to the stochastic nature of particle filtering.

### 7.4.3 Comparison with L1 Tracker

Our framework also consistently outperforms the L1 tracker for all sub-groups with considerable margins. The average success rate of our framework is 74.53 percent while that of the L1 tracker is 21.10 percent. When compared with our framework, the main reason of the poor performance of the L1 tracker is it can estimate the object motion only up to affine transformations. Among “Angle”, “Range”, “FastFar”, “FastClose”, and “Illumination”, the L1 tracker performs worst for “Angle”, where the perspective effect is severe due to large camera viewing angle changes. Conversely, the L1 tracker performs best for “Range”, where the camera viewing angle change is more limited than other cases.

#### TABLE 5

<table>
<thead>
<tr>
<th>Method</th>
<th>Low</th>
<th>Repetitive</th>
<th>Normal</th>
<th>High</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FERNS</td>
<td>29.49</td>
<td>48.26</td>
<td>58.61</td>
<td>43.32</td>
<td>44.92</td>
</tr>
<tr>
<td>SIFT</td>
<td>49.77</td>
<td>71.49</td>
<td>67.86</td>
<td>68.26</td>
<td>64.35</td>
</tr>
<tr>
<td>SURF</td>
<td>19.56</td>
<td>33.38</td>
<td>50.18</td>
<td>24.66</td>
<td>31.95</td>
</tr>
<tr>
<td>ESM</td>
<td>57.78</td>
<td>28.78</td>
<td>71.62</td>
<td>30.94</td>
<td>47.28</td>
</tr>
<tr>
<td>L1</td>
<td>21.19</td>
<td>22.69</td>
<td>20.08</td>
<td>20.43</td>
<td>21.10</td>
</tr>
<tr>
<td>Ours</td>
<td>67.81</td>
<td>84.58</td>
<td>81.95</td>
<td>63.76</td>
<td>74.53</td>
</tr>
</tbody>
</table>

#### Method | Angle | Range | FastFar | FastClose | Illum. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FERNS</td>
<td>44.78</td>
<td>52.84</td>
<td>12.45</td>
<td>44.30</td>
<td>70.24</td>
</tr>
<tr>
<td>SIFT</td>
<td>59.33</td>
<td>87.46</td>
<td>24.69</td>
<td>56.62</td>
<td>93.66</td>
</tr>
<tr>
<td>SURF</td>
<td>27.11</td>
<td>41.11</td>
<td>6.27</td>
<td>39.32</td>
<td>46.92</td>
</tr>
<tr>
<td>ESM</td>
<td>69.98</td>
<td>57.05</td>
<td>15.11</td>
<td>41.72</td>
<td>52.53</td>
</tr>
<tr>
<td>L1</td>
<td>8.35</td>
<td>38.80</td>
<td>21.11</td>
<td>5.54</td>
<td>31.99</td>
</tr>
<tr>
<td>Ours</td>
<td>88.36</td>
<td>81.22</td>
<td>66.56</td>
<td>43.48</td>
<td>93.00</td>
</tr>
</tbody>
</table>

The numbers are average tracking success rates for each specific sub-groups.
Another reason for the poor performance of the L1 tracker is that it employs the state transition density-based importance function unlike our framework. Only considering the frames where the perspective effect is not severe, the qualitative tracking performance of the L1 tracker is far better than the numbers in Table 5 suggest, i.e., it tracks the object with seemingly good accuracy, but the RMS tracking errors are often greater than the threshold (10 pixels). We can enhance the tracking accuracy of the L1 tracker with the state transition density-based importance function by using more particles, e.g., 5,000, but it is not practical considering the computational complexity. The average tracking error of the L1 tracker for the successfully tracked frames is 7.02 while that of our framework is 4.88. These suggest that the state transition density-based importance function employed by the L1 tracker is not suitable for applications where high tracking accuracy is highly required such as augmented reality and visual servoing.

8 Discussion

From experiments using various test video sequences, we have demonstrated that our probabilistic template-based visual tracking framework can provide quite accurate tracking results with sufficient computational efficiency. Compared with [30], such satisfactory performance is largely derived from a collection of techniques, including the iterative Gaussian importance function generation, inverse formulation of the Jacobian calculation, template resizing, and use of parent-child particles.

The benchmark test using the Metaio data set clearly demonstrates the practical value of our framework for applications such as visual servoing, augmented reality, and visual localization and mapping, where accurate projective motion tracking is necessary for 3D camera motion estimation. Note that the Metaio benchmark data set has been developed specifically in consideration of augmented reality applications. We have shown that our framework yields far better tracking performance for the Metaio benchmark data set than the tracking methods currently popularly used for augmented reality applications, namely, the deterministic optimization-based ESM tracker and motion estimation by key-point matching using SIFT, SURF, and FERNS.

The extension of our previous affine motion tracking algorithm of [30] to projective motion tracking is another important factor for the satisfactory tracking performance for the Metaio benchmark data set. When the perspective effect is not perceivable (e.g., when the object is far enough from the camera or its size is small), an affine motion tracker can estimate the object template motion rather accurately. However, in most augmented reality applications usually targeted for indoor environments, the perspective effect nearly always exists as shown by our test video images in Fig. 5 and the Metaio benchmark data set images. This also holds true for the cases of visual servoing and indoor visual localization and mapping. Affine motion tracking should not be used for such applications where the 3D camera motion is estimated from tracking results because of its limited ability to estimate the object template motion accurately. Therefore, it should be regarded that what we have done to our previous work [30] in this paper, i.e., the extension of the affine motion tracking framework to projective motion tracking with increased computational efficiency and tracking accuracy, has not only theoretical merits but also considerable practical importance.

Especially for visual tracking, it is difficult to model the object dynamics precisely for freely moving objects. Due to this reason, it has been the focus to improve the object appearance model in improving the performance of particle filtering-based visual tracking, e.g., [4], [23], [46], [59]. However, we argue that the improvement of particle sampling other than the object appearance model is also important especially for projective motion tracking. The degree of freedom of projective motion is eight rather higher than those of other forms of motion such as similarity and affine. When we sample particles naively using the state transition density-based importance function, the number of particles necessary to get satisfactory results increases with the state dimension exponentially. Therefore, the improvement of particle sampling is essential for the practicality of projective motion tracking via particle filtering. The poor performance of the L1 tracker of [4] for the Metaio benchmark data set also suggests that even with the advanced object appearance model, the naive particle sampling using the state transition density-based importance function can lead to tracking inaccuracy. In this paper, we have considerably improved particle sampling by the proposed iterative Gaussian importance function generation.

The way to represent the state also affects the tracking performance considerably. In this paper, we have not presented a performance comparison between our Lie group-based state representation and a conventional vector space one. The superiority of our Lie group-based state representation to a conventional vector space one has been well demonstrated both conceptually and experimentally in [31] and [30], and the reader is referred to these sources for further details.

The question of how to design an image similarity measure robust to various challenging causes such as motion blur, temporal occlusion, and illumination change is obviously relevant and important to template-based visual tracking performance. However, the design of image similarity measures is not the focus of this paper. Instead, we have shown how well-established image similarity measures suitable to template-based visual tracking, such as NCC and PCA reconstruction error, can easily fit into our framework and yield satisfactory tracking performance. The only restriction of our framework on image similarity measures is that they should be differentiable, and that the derivatives can be obtained at least approximately. The practical performance shall be dependent on the linearity of the employed similarity measure. In this sense, the entropy-based similarity measures such as mutual information [43] and disjoint information [55] can be also used for measurements in our framework because their derivatives can be approximately obtained [11], [16].

We have not addressed the occlusion issue explicitly in this paper, since the stochastic nature of particle filtering can deal (at least partly) with short-term and partial occlusion, as experimentally shown in Section 7.1 and previously in [31], [46]. One can apply more specific
strategies such as the robust error norm [9] and the sub-templates [27] to our framework in order to effectively deal with long-term occlusion.

In [50], as an efficient alternative to global optimization strategies, it is suggested to run the Kalman filter as a pre-processing step to give a reasonable initial point for optimization. Similarly, to refine our tracking results we can run an optimization procedure as a post-processing step using the estimated state as the initial point. We can also consider fusion with the detection-based approach such as [21] to be able to escape from temporal tracking failure. Although these conjunctive uses of different kinds of methods for template-based visual tracking may not be so appealing from a theoretical point of view, in practice they can potentially result in considerable gains both in accuracy and efficiency.

9 Conclusion
In this paper, we have proposed a novel particle filtering framework to solve template-based visual tracking probabilistically. We have formulated particle filtering on SL(3) with appropriate state and measurement equations for projective motion tracking that does not require vector parametrization of homographies. In order to utilize the optimal importance function, we have clarified the notions of Gaussian distributions and Taylor expansion on SL(3) using the exponential coordinates, and approximated the optimal importance function as a Gaussian distribution on SL(3) by local linearization. We have also proposed an iterative Gaussian importance function generation method to increase the accuracy of Gaussian importance functions. The computational efficiency is enhanced by the inverse formulation of the Jacobian calculation, template resizing, and use of parent-child particles. We have also extended the SL(3) particle filtering framework to affine motion tracking on the group Aff(2). The improved performance of our probabilistic tracking framework and superiority of our Gaussian importance functions to other importance functions have been experimentally demonstrated. Finally, it has been shown that our framework yields superior tracking results to several state-of-the-art tracking methods via experiments using the publicly available benchmark data set.

Acknowledgments
Frank C. Park was supported by the Center for Advanced Intelligent Manipulation, BK21+, and SNU-IAMD.

References


Frak C. Park received the BS degree in electrical engineering from MIT in 1985, and the PhD degree in applied mathematics from Harvard University in 1991. From 1991 to 1995, he was an assistant professor of mechanical and aerospace engineering at the University of California, Irvine. Since 1995, he has been a professor of mechanical and aerospace engineering at Seoul National University. From 2009 to 2012, he was an adjunct professor in the Department of Interactive Computing at the Georgia Institute of Technology, and in 2001-2002 was a visiting professor at the Courant Institute of Mathematical Sciences at New York University. His research interests are in robot mechanics, planning and control, visual tracking, and related areas of applied mathematics. In 2007-2008, he was an IEEE Robotics and Automation Society (RAS) Distinguished Lecturer, and has served as secretary of RAS from 2009-2010 and 2012-2013. He has served as senior editor of the *IEEE Transactions on Robotics*, an area editor of the *Springer Handbook of Robotics and Advanced Tracts in Robotics (STAR)*, and as an associate editor of the *ASME Journal of Mechanisms and Robotics*. He is a fellow of the IEEE, and an incoming editor-in-chief of the *IEEE Transactions on Robotics*.

Kyoung Mu Lee received the BS and MS degrees in control and instrumentation engineering from Seoul National University (SNU), Seoul, Korea in 1984 and 1986, respectively, and the PhD degree in electrical engineering from the University of Southern California (USC), Los Angeles, California in 1993. He has been awarded the Korean Government Overseas Scholarship during the PhD courses. From 1993 to 1994, he was a research associate in the Signal and Image Processing Institute (SIPI) at USC. He was with the Samsung Electronics Co. Ltd. in Korea as a senior researcher from 1994 to 1995. On August 1995, he joined the Department of Electrical and Computer Engineering at Seoul National University as a professor and the director of the Automation Systems and Research Institute. His primary research interests include object recognition, MRF optimization, tracking, and visual navigation. He is currently serving as an associate editor of the *IPSJ Transactions on CVA, MVA, the Journal of HMSP*, and *IEEE Signal Processing Letters*. He also has served as a program chair and area chairs of ACCV, CVPR, and ICCV. He received the Okawa Foundation Research Grant Award in 2006, Honorable Mention Award at the ACCV2007, the Most Influential Paper over the Decade Award at IAPR MVA2009, and the Outstanding Research Award by the Asia-Pacific Signal and Information Processing Association (APSIPA) for 2012-2013. He has (co)authored more than 150 publications in refereed journals and conferences including *PAMI, IJCV, CVPR, ICCV* and *ECCV*. More information can be found on his homepage http://cv.snu.ac.kr/kmlee.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.