Robust 3D Visual Tracking Using Particle Filtering on the Special Euclidean Group: A Combined Approach of Keypoint and Edge Features

Changhyun Choi and Henrik I. Christensen
Robotics & Intelligent Machines, College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{cchoi,hic}@cc.gatech.edu

December 30, 2011

Abstract

We present a 3D model-based visual tracking approach using edge and keypoint features in a particle filtering framework. Recently, particle filtering based approaches have been proposed to integrate multiple pose hypotheses and have shown good performance, but most of the work has made an assumption that an initial pose is given. To ameliorate this limitation, we employ keypoint features for initialization of the filter. Given 2D-3D keypoint correspondences, we randomly choose a set of minimum correspondences to calculate a set of possible pose hypotheses. Based on the inlier ratio of correspondences, the set of poses are drawn to initialize particles. After the initialization, edge points are employed to estimate inter-frame motions. While we follow a standard edge-based tracking, we perform a refinement process to improve the edge correspondences between sampled model edge points and image edge points. For better tracking performance, we employ a first order autoregressive state dynamics, which propagates particles more effectively than Gaussian random walk models. The proposed system re-initializes particles by itself when the tracked object goes out of the field of view or is occluded. The robustness and accuracy of our approach is demonstrated using comparative experiments on synthetic and real image sequences.

1 Introduction

From robotic manipulation to augmented reality, estimating poses of objects is a key task. Since Harris (1992) proposed his early system which tracks an object by projecting a 3D CAD model into a 2D image and aligning the projected model edges to image edges, there have been active efforts to enhance the early edge-based tracking system at general scale (Drummond and Cipolla 2002, Comport et al. 2004) and at micro or nano scales (Yesin and Nelson 2005, Kratochvıl et al. 2009, Tamadazte et al. 2010). Edges are employed because they are easy to compute and invariant to illumination and pose changes. However, a critical disadvantage of using edges in visual tracking is that they look similar to each other. In general, edge correspondences are determined by local search based on a prior pose estimate. So the tracking performance of an edge-based tracker directly depends on correct pose priors. To improve the pose priors, there have been various attempts to enhance the pose accuracy by incorporating interest points (Vacchetti et al. 2004, Rosten and Drummond 2005, Pressigout and Marchand 2006) or employing additional sensors (Klein and Drummond 2004). Since interest points and their rich descriptors (Lowe 2004, Bay et al. 2008) can be extracted and matched well under illumination, scale, and rotation changes, keypoint features complement edges in an elegant manner (Kyrki and Kragic 2005).

Considering multiple edge correspondences was another interest in edge-based tracking. Since edges are ambiguous and false edge correspondences directly lead the tracker to false pose estimates, some approaches have considered multiple edge correspondences (Vacchetti et al. 2004, Kemp and Drummond 2005). However, their work was still limited because only one or two hypotheses were maintained from the multiple correspondences during tracking.

Multiple hypotheses tracking has been implemented using a particle filtering framework. Isard and Blake (1998) applied a particle filter to 2D visual edge-based tracking and have shown great potential. Affine 2D visual trackers have also been proposed in a particle filter framework with incremental measurement learning (Ross et al. 2008, Kwon and Park 2010). Among them, Kwon and Park (2010) proposed a particle filter on a 2D affine Lie group, \text{Aff}(2), in a coordinate-invariant way. For 3D visual tracking, Pupilli and Calway (2006) have shown the possibility of applying a particle filter to 3D edge-based tracking. While they demonstrated the tracking of simple 3D objects, Klein and Murray (2006) implemented a particle filtering approach which tracks a complex full 3D object in real-time by exploiting the GPU. Mörwald et al. (2010) also used the parallel power of GPU to implement a fast model-based 3D visual tracker. With edges from a 3D CAD model, they also employed edges from texture which possibly contributes to avoid false edge correspondences as well as to enhance the accuracy of pose estimates. Teulière et al. (2010) recently addressed a similar problem by maintaining multiple hypotheses from low-level edge correspondences.

With a few exceptions (Kyrki and Kragic 2005, Mörwald et al.
most of the work has made an assumption in which trackers start from a given pose. Several efforts (Klein and Murray 2006, Pupilli and Calway 2006) used annealed particle filters to find the true pose from scratch without performing an appropriate initialization, but the search space might be too large to converge to the true pose in reasonable time, and it might not converge to the pose after enough time elapses. It is thus more desirable to employ other information for initialization. The BLORT (Mörgwald et al. 2010) employed SIFT keypoints (Lowe 2004) to recognize objects and used them for particle initialization.

In this paper we utilize a particle filtering technique on the $SE(3)$ group that is based on the work of Kwon et al. (2007). For robust 3D visual tracking, we employ keypoint features in initialization and edges in the calculation of measurement likelihoods. Like Klein and Murray (2006), our system can track complex objects by performing a self-occlusion test. By maintaining multiple pose hypotheses, our algorithm can reliably track an object on challenging image sequences that have complex background and heavy clutter. Our key contributions are as follows:

- We employ keypoint features as additional visual cues. While Klein and Murray (2006), Pupilli and Calway (2006) have used annealing particle filter to find the initial pose, we initialize particles to highly probable states based on pose estimates calculated from keypoint correspondences so that initialized particles tend to converge faster than the usual annealed particle filtering.

- We refine edge correspondences between the projected model edges and the image edges via a RANSAC (Fischler and Bolles 1981). Most of the edge-based tracking approaches have used the nearest edge correspondences without performing a refining process (Harris 1992, Drummond and Cipolla 2002, Comport et al. 2004, Choi and Christensen 2010), except a few work in Armstrong and Zisserman (1995), Teulières et al. (2010). Considering the edge correspondences directly affect the measurement likelihood and thus entire tracking performance, we employ a RANSAC approach to ensure consistent edge data associations. While Armstrong and Zisserman (1995) applied a RANSAC on each 2D line segments individually, we perform that on 3D sampled points and their corresponding 2D closest edge points.

- While previous edge-based trackers (Klein and Murray 2006, Teulières et al. 2010) have employed random walk models as a motion model, we apply a first-order autoregressive (AR) state dynamics on the $SE(3)$ group to guide particles more effectively.

- To be fully automatic and reliable in practical settings, our approach monitors the number of effective particles and use the value to decide when the tracker requires re-initialization.

This paper is organized as follows. In Section 2, we define the monocular camera model used in this paper and explain 3D object models with the automatic salient edge selection process. In Section 3.1 and 3.2, we introduce a particle filtering framework with state and measurement equations. The AR state dynamics is then presented in Section 3.3. After explaining how particles are initialized and their likelihoods are evaluated in Section 3.4 and 3.5, respectively, the optimization performed in each particle is explained in Section 3.6. Lastly, the re-initialization scheme is presented in 3.7. Experimental results on various image sequences are shown in Section 4.

2 Camera and Object Models

2.1 Camera Model

Our system employs a calibrated monocular camera so that we have the intrinsic and lens distortion parameters as known a priori. We rectify input images in order to remove the lens distortion effect. Thus, as a camera model we use the standard pin-hole model given by:

$$p = \text{Project}(K, X_t, P^O) = K \begin{pmatrix} x_c \cr y_c \cr 1 \end{pmatrix}$$

$$P^C = X_t P^O$$

where $p = (u, v)^T$ is the projected 2D image coordinates, $P^O = (x^O, y^O, z^O, 1)^T$ and $P^C = (x^C, y^C, z^C, 1)^T$ are the 3D homogeneous coordinates of a point in object and camera coordinate systems, respectively, and $X_t \in SE(3)$ is the pose of the camera at time $t$ or the extrinsic matrix. The matrix $K$ represents the intrinsic parameters of the camera:

$$K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix}$$

where $f_u$ and $f_v$ are the focal length in pixel dimensions, and $u_0$ and $v_0$ represent the position of the principal point.

2.2 3D Object Model and Salient Edge Selection

Since most objects which exist in our daily environment are manufactured, their CAD models might be available, and such models provide helpful information for robotic manipulation. Although there are various formats for CAD models, most of them...
can be represented as a polygon mesh. A polygon mesh is usually composed of vertices, edges, faces, polygons and surfaces. To estimate the pose difference between two consecutive frames, we employ edge features in images coming from a monocular camera. So we should determine which edges in the CAD model of a targeted object would be visible in images. Here we make an assumption that sharp edges in the model are more likely to be salient in images. To identify the sharp edges, we use the face normal vectors from the model. As illustrated in Figure 2, if the face normal vectors of two adjacent faces are close to perpendicular, the edge shared by the two faces is regarded as a sharp edge. Similarly, if two face normal vectors are close to parallel, the edge is regarded as a dull edge. For the decision, we employ a simple thresholding scheme with the value of the inner product of two normal vectors. More formally, we can define an indicator function with respect to the edges in the model by:

$$I(e_i) = \begin{cases} 1 & \text{if } \mathbf{n}_1^T \mathbf{n}_2^T \leq \tau_s \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (4)$$

where $\mathbf{n}_1^T$ and $\mathbf{n}_2^T$ are the face normal unit vectors of the two adjacent faces which share the $i^{th}$ edge, $e_i$. We found the threshold $\tau_s = 0.3$ is a reasonable value. This salient edge selection is performed fully automatically offline. The original 3D CAD model edges and the selected edges are displayed in Figure 1. In general, the salient edges are only considered in edge-based tracking, but when dull edges constitute the object’s boundary as the “Cup” object, they are also considered. To determine these boundary edges, we find edges shared by a front face and a back face. Testing front or back faces is done by calculating inner products of the face normal vectors and the z-axis of the camera. Testing boundary of the dull edges is performed at run-time, and thus it is desirable to avoid when the target object does not have many dull boundary edges.

3 Particle Filter on the $SE(3)$ Group

In 3D visual tracking, a state represents a 6-DOF pose of a tracked object, and tracking estimates time-varying change of coordinates. It is well known that the trajectory is not on general vector space, rather it is on Lie groups – in general, the Special Euclidean group $SE(3)$ and the affine group $Aff(2)$ in 3D and 2D visual tracking, respectively. Since the trajectory we want to estimate is on a Lie group, the particle filter should be applied on the Lie group. Monte Carlo filtering on Lie groups is explicitly addressed in (Chiuso and Soatto 2000, Kwon et al. 2007, Kwon and Park 2010). It is well known that if a local coordinate system, an ad-hoc representation of motions (e.g. Euclidean embedding), is employed, the same perturbation on different states often results in different motions. Thus filtering performance and noise distribution of local coordinate-based particle filtering approaches are dependent on the choice of the local coordinates, while particle filtering on Lie groups is coordinate-invariant. This coordinate-invariance issue is well addressed in Kwon et al. (2007) and Kwon and Park (2010).

3.1 State and Measurement Equations

From the continuous general state equations on the $SE(3)$ group, discrete system equations is acquired via the first-order exponential Euler discretization (Kwon et al. 2007):

$$X_t = X_{t-1} \cdot \exp(A(X, t) \Delta t + \mathbf{dW}_t \sqrt{\Delta t}),$$  \hspace{1cm} (5)$$

$$\mathbf{dW}_t = \sum_{i=1}^{6} \epsilon_{t,i} \mathbf{E}_i,$$

$$\epsilon_t = (\epsilon_{t,1}, \ldots, \epsilon_{t,6})^T \sim \mathcal{N}(0_{6 \times 1}, \Sigma_w)$$

where $X_t \in SE(3)$ is the state at time $t$, $A : SE(3) \rightarrow se(3)$ is a possibly nonlinear map, $\mathbf{dW}_t$ represents the Wiener process noise on $se(3)$ with a covariance $\Sigma_w \in \mathbb{R}^{6 \times 6}$, and $\mathbf{E}_i$ are the $i^{th}$
basis elements of $se(3)$:

$$
\begin{align*}
E_1 &= \begin{pmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
E_2 &= \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
E_3 &= \begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
E_4 &= \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
E_5 &= \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
E_6 &= \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}.
\end{align*}
$$

Note that the stochastic state equation in (5) is equivalent to a convolution of probability densities (Park et al. 2008, Wang and Chirikjian 2006), except that the latter does not assume Gaussian noise.

The corresponding measurement equation is then:

$$
Z_t = g(X_t) + n_t, \quad n_t \sim \mathcal{N}(0_{N_t \times 1}, \Sigma_n) \tag{7}
$$

where $g : SE(3) \to \mathcal{R}^{N_t}$ is a nonlinear measurement function and $n_t$ is a Gaussian noise with a covariance $\Sigma_n \in \mathcal{R}^{N_t \times N_t}$.

### 3.2 Particle Filter

In a generic particle filtering framework, the posterior density function $p(X_t | Z_{1:t})$ is represented as a set of weighted particles by

$$
S_t = \{(X_t^{(1)}, \pi_t^{(1)}), \ldots, (X_t^{(N)}, \pi_t^{(N)})\} \tag{8}
$$

where the particles $X_t^{(n)} \in SE(3)$ represent samples of the current state $X_t$, the normalized weights $\pi_t^{(n)}$ are proportional to the likelihood function $p(Z_t | X_t^{(n)})$, and $N$ is the number of particles. The current state $X_t$ could be estimated by the weighted particle mean:

$$
X_t = \mathcal{E}[S_t] = \sum_{n=1}^{N} \pi_t^{(n)} X_t^{(n)} \tag{9}
$$

or the mode:

$$
X_t = \mathcal{M}[S_t] = X_t^{(j)}, \quad j = \arg \max_j \pi_t^{(j)}. \tag{10}
$$

When we apply the mean, however, there is a problem where the average of $X_t^{(n)}$ is not valid in the $SE(3)$. More specifically, let $R_t^{(n)} \in SO(3)$ be the rotation part of the $X_t^{(n)}$. Then the arithmetic mean $\overline{R}_t = \frac{1}{N} \sum_{n=1}^{N} R_t^{(n)}$ is not usually on the $SO(3)$ group. As an alternative, Moakher (2003) showed that a valid average of a set of rotations can be calculated by the orthogonal projection of $\overline{R}_t$ onto $SO(3)$ as

$$
R_t = \begin{cases}
VU^T & \text{when } \det(\overline{R}_t) > 0 \\
VHU^T & \text{otherwise},
\end{cases} \tag{11}
$$
Most of the particle filter-based trackers assumes that an initial state is given. In practice, the initial particle is crucial to ensure convergence to the true state. Several trackers (Klein and Murray 2006, Pupilli and Calway 2006) search for the true state from scratch, but it is desirable to initialize particle states by using other information. Using keypoint allows for direct estimation of 3D pose, but due to the need for a significant number of correspondences it is either slow or inaccurate. As such, keypoint correspondences are well suited for filter initialization.

For initialization, we employ so-called keyframes $\mathcal{F} = \{(p_1, P_1), \ldots, (p_F, P_F)\}$ which are composed of $F$ sets of SURF keypoints (Bay et al. 2008) coordinates in 2D and 3D. These keyframes are saved manually by user input. Since appearance of SURF keypoints are not invariant enough to viewpoint variations, it requires to capture multiple keyframes covering different views of an object. The keypoint coordinates in 2D $(p_i)$ is easily determined from SURF keypoints extraction, while 3D coordinates $(P_i)$ is determined by back-projecting 2D points to surfaces of the tracked object. For the back-projection, we need to know the exact pose of the tracked object. It could be possible to get the pose using a calibrated turn table, but we used our particle filter-based tracking started from a known pose.

At runtime, an input image $I$ coming from a monocular camera is matched with the saved keyframes by extracting keypoints $p_i$ from $I$ and comparing them with $\mathcal{F}$. To find keypoint correspondences $\mathcal{C} = \{\hat{p}_i, \hat{P}_j\}$ efficiently, we employ the Best-Bin-First (BBF) algorithm using a kd-tree data structure (Beis and Lowe 1997) that allows execution of the search in $O(n \log n)$. As described in Lowe (2004), the ratio test is then performed to find distinctive feature matches $\hat{C}$ with the threshold $\tau_r = 0.7$.

While we employed RANSAC (Fischler and Bolles 1981) after determining putative correspondences in our previous work (Choi and Christensen 2010), we skip this procedure because in the particle filter framework we can initialize particles in an alternative way in which the basic idea is similar to RANSAC, but it considers multiple pose hypotheses. Instead of explicitly performing RANSAC, we randomly select a set of correspondences $\hat{C}$ from the given putative correspondences $\mathcal{C}$ having a maximum number of correspondences over keyframes $\mathcal{F}$ and estimate a possible set of poses $X^{*}(n)$ from $\hat{C}$. Since we maintain 3D coordinates $P_f$ of keypoints in keyframes, we can get 2D-3D correspondences from the matching process described above. So we can regard this problem as the Perspective-n-Point (PnP) problem, in which the pose of a calibrated monocular camera is estimated from $n$ 2D-3D point correspondences. To find a pose from the correspondences, we use the EPnP algorithm (Lepetit et al. 2009) that provides a $O(n)$ time non-iterative solution for the PnP problem.

After a particle pose $X^{*}(n)$ is initialized from randomly selected minimum correspondences, all 3D points $P_f$ from the putative correspondences $\mathcal{C}$ are projected into 2D points $\hat{p}$ and compared with 2D keypoints $p_i$. We then count the number of inlier correspondences $N_i$ whose Euclidean distances between $\hat{p}$ and $\hat{p}_i$ is within the threshold $\tau_e = 20.0$. For $n = 1, \ldots, N$ where
\( N \) is the number of particles, the weights of particles are assigned by the ratio of the number of putative correspondences \( N_{pi} \), which is the number of entries in \( \hat{C} \), and the number of inlier correspondences \( N_i \), as:

\[
\hat{\pi}^{*}(n) \propto p(Z_i | X^{*}(n)) \propto \exp(-\lambda \frac{N_{pi} - N_i}{N_{pi}})
\]  

(15)

where \( \lambda \) is a parameter. Then the weights \( \hat{\pi}^{*}(n) \) are normalized by:

\[
\hat{\pi}^{*}(n) = \frac{\hat{\pi}^{*}(n)}{\sum_{i=1}^{N} \hat{\pi}^{*}(i)}.
\]  

(16)

After normalization, particles are randomly drawn with probability proportional to these weights. Since weights are proportional to \( N_i \), pose hypotheses having more inliers are likely to survive in the random sampling. This initialization process is similar to RANSAC, but it maintains multiple hypotheses of the initial poses while the RANSAC only finds a best hypothesis.

The initialization process is presented in Algorithm 2 where referred work and equations are cited as (·) in the comments area. Given an input image \( I \) and keyframes \( F \), the algorithm returns the set of initialized particle states \( \hat{S} \). Among the parameters, \( \tau_n \) is the minimum required number of keypoint correspondences for the initialization, \( m \) is the minimum number of correspondences to perform EPnP, and \( K \) is the intrinsic parameters of the camera in (3). We found that 7 and 9 are well suited to \( m \) and \( \tau_n \), respectively.

### 3.5 Edge-based Measurement Likelihood

![Residual determination for calculating the likelihood](image)

Figure 3. Residual determination for calculating the likelihood. Residual errors between projected model (bright lines) and extracted image edges (dark tortuous lines) are calculated. Sample points \( p_i \) are generated along the model per a fixed distance and are matched to image edge points \( q_i \) by performing 1D search along the direction orthogonal to the model edge.

Once each particle is initialized and propagated according to the AR state dynamics and Gaussian noise, it has to be evaluated based on its measurement likelihood. In general edge-based tracking, a 3D wireframe model is projected into a 2D image according to a pose hypothesis \( X^{*}(n) \). Then a set of points is sampled along edges in the wireframe model per a fixed distance. As some of sampled points are occluded by the object itself, a visibility test is necessary. While Drummond and Cipolla (2002) used a BSP tree for hidden line removal, OpenGL occlusion query is an easy and efficient alternative in which the sampled points are tested whether occluded or not (Klein and Murray 2006). The visible sampled points are then matched to edge points, which are obtained by using the Canny edge detector (Canny 1986), from the input image by performing 1D perpendicular search (Drummond and Cipolla 2002, Choi and Christensen 2010). In the matching, most approaches have tried to match the sampled points to closest edges points without examining their orientation characteristics. However, it is well known that using edge orientation significantly enhances the quality of edge correspondences (Olson and Huttenlocher 2002, Liu et al. 2010). Especially when the object contains relatively complex textures on its surfaces or it is located in cluttered environments, erroneous edge correspondences are often obtained from the textures or the background edges. These false correspondences result in a bad state hypothesis, and thus it is natural to exclude the edge correspondences having significant differences in orientation. This can be achieved by defining an indication function as

\[
I(p_i, q_i) = \begin{cases} 
1 & \text{if } \left| \theta_{m}(p_i) - \theta_{e}(q_i) \right| \leq \tau_{\theta} \\
0 & \text{otherwise} 
\end{cases}
\]  

(17)

where \( \theta_{m} \) and \( \theta_{e} \) return the orientation of the model edge to which the sample point \( p_i \) belongs and of the image edge point \( q_i \), respectively. Note that the computation burden is not significant since orientations of sampled points on the model edge and their corresponding edge points are only required to be determined.

After the orientation testing, the residual \( r_i \) which is Euclidean distance between \( p_i \) and \( q_i \) is calculated. By stacking all of the residual of visible sample points, the residual vector \( r \in \mathbb{R}^{N_e} \) is obtained as follows

\[
r = (r_1, r_2, \ldots, r_{N_e})^T
\]  

(18)

where \( N_e \) is the number of valid sample points (i.e. visible sample points correspond to the image edges). Along the residual \( r \), the unit normal vectors \( n_i \in \mathbb{R}^2 \) of the residual are also saved to be used in the optimization explained in the next section:

\[
\{n_1, n_2, \ldots, n_{N_e}\}.
\]  

(19)

Once we have edge correspondences, we refine them using RANSAC (Fischler and Bolles 1981). Although we discard matches having significant orientation differences, it is often that there are false matches having similar orientation. Some efforts tried to enhance these matches through maintaining multiple low-level edge clusters (Teulière et al. 2010) or applying a RANSAC on each 2D line segments (Armstrong and Zisserman 1995). One drawback of both work is the possibility of inconsistent refinement because edge or line segments are individually corrected. Another drawback of them is that their methods can be applied only to line segments and if the model of an object is composed of an amount of small line segments the effect of correction would be negligible or wrong. For consistent refinement of the edge correspondences, we perform a RANSAC on 3D sampled points \( P \) and their corresponding 2D closest edge points \( p \). Our approach consistently discard outliers by estimating the best 3D pose containing large number of inliers \( \hat{H} \). The RANSAC algorithm is presented in Algorithm 3 which finds the refined edge correspondences \( \hat{Z} = \{\hat{p}, \hat{P}\} \) given the current pose hypothesis \( X \) and the
original edge correspondences \( Z = \{ p, P \} \). Among the parameters, \( i_{\text{max}} \) is the maximum number of iterations, again \( m \) is the minimum number of correspondences required for EPnP, and \( \rho \) is the probability in which at least one set of randomly sampled \( m \) correspondences are from inliers. The \( \rho \) typically set as 0.99 is used to estimate the required number of iterations \( \kappa \) which is adaptively adjusted in the iteration.

Figure 3 illustrates the residual calculation in which each residual arrow represents \( r \). Note that the second edge correspondence \( q_2 \) is wrong since \( q_2 \) comes from background clutter, but it is soon discarded via the RANSAC refinement. The fifth edge correspondence is also erroneous because \( q_5 \) does not belongs to edges from the object, but it is excluded via the orientation comparison.

After the edge correspondences are refined, the measurement likelihood can be calculated from the ratio between the number of matched sample points \( N_z \) which are survived after the RANSAC and the number of visible sample points \( N_z \) which pass a self-occlusion test as:

\[
p(Z_t | X_t) \propto \exp\left( -\lambda_v \frac{(N_z - N_{\hat{z}})}{N_z} \right)
\]

where \( \lambda_v \) is a parameter to be tuned. This likelihood has been similarly used in Klein and Murray (2006). Another choice is employing \( \bar{r} \) which is an arithmetic average of the residual \( r \) (Teuli`ere et al. 2010):

\[
p(Z_t | X_t) \propto \exp\left( -\lambda_r \bar{r} \right)
\]

where \( \lambda_r \) is also a parameter. We noticed that both likelihoods are valid, and we empirically found that using both terms shows better results. Therefore, in our approach the measurement likelihood is evaluated as:

\[
p(Z_t | X_t) \propto \exp\left( -\lambda_v \frac{(N_z - N_{\hat{z}})}{N_z} \right) \exp\left( -\lambda_r \bar{r} \right).
\]
3.6 Optimization using IRLS

One of the challenges in particle filtering for 3D visual tracking is the large state space, and hence a large number of particles is usually required for reliable tracking performance. To reduce the number of particles, Klein and Murray (2006) has used an annealed particle filter, while Bray et al. (2004) and Teuli`ere et al. (2010) have selectively employed local optimizations in a subset of particles. For more accurate results, we optimize states of particles as well, for which Iteratively Reweighted Least Squares (IRLS) is employed (Drummond and Cipolla 2002, Choi and Christensen 2010). From IRLS, the optimized particle \( \hat{X}_t^{* (n)} \) is calculated as follows:

\[
\hat{X}_t^{* (n)} = X_t^{* (n)} \cdot \exp\left( \sum_{i=1}^{6} \hat{\mu}_i E_i \right) \tag{22}
\]

where \( \hat{\mu} \in \mathbb{R}^6 \) is the motion velocity that minimizes the residual vector \( \hat{r} \in \mathbb{R}^{6N_t} \), \( J \in \mathbb{R}^{6N_t \times 6} \) is a Jacobian matrix of \( n_i^T p_i \) with respect to \( \mu \) obtained by computing partial derivatives at the current pose, and \( W \in \mathbb{R}^{6N_t \times 6} \) is a weighted diagonal matrix. The derivation of the Jacobian matrix \( J \) and more detailed information about IRLS are explained in Appendix.

Note that the measurement likelihood in (21) is calculated before the IRLS optimization. To assign weights of particles, we have to evaluate the likelihood again with the optimized state \( \hat{X}_t^{* (n)} \). However, computing the likelihood again is computationally expensive because it requires the self-occlusion test and 1D perpendicular search for each particle. As an alternative, we can note that IRLS is a local optimization, and hence it highly depends on the previous state. Thus particles having higher likelihood...
hood tend to exhibit still higher likelihood after IRLS. But lower likelihood particles are likely to be stuck in local minima due to erroneous or insufficient edge correspondences. Therefore, we can approximate $p(Z_t | X_t^{(n)})$ as:

$$\hat{\pi}_t^{(n)} \propto p(Z_t | X_t^{*}) \approx p(Z_t | X_t^{(n)})$$  \hspace{1cm} (24)

It might not be an ideal way but it is an efficient alternative especially when computation time is considered. After assigning weights of the particles, the weight $\hat{\pi}_t^{(n)}$ is then normalized to $\pi_t^{(n)}$ by (16).

3.7 Re-initialization based on $\hat{N}_{eff}$

Ideally a tracked object should be visible during an entire tracking session. In reality, however, it is quite common that the object goes out of frame or is occluded by other objects. In these cases, the tracker is required to re-initialize the tracking. In general sequential Monte Carlo methods, the effective particle size $N_{eff}$ has been introduced as a suitable measure of degeneracy (Doucet et al. 2000). Since it is hard to evaluate $N_{eff}$ exactly, an alternative estimate $\hat{N}_{eff}$ is defined in Doucet et al. (2000):

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (\hat{\pi}(i))^2}.$$  \hspace{1cm} (25)

Often it has been used as a measure to execute the resampling procedure. But, in our tracker we resample particles every frame, and hence we use $\hat{N}_{eff}$ as a measure to do re-initialization. When the number of effective particles is below a fixed threshold $N_{thres}$, the re-initialization procedure is performed.

The overall algorithm describing the particle filtering on the SE(3) group is shown in Algorithm 1 where referred algorithms and equations are cited as ⟨·⟩ and ⟨·⟩ in the comments area, respectively. It requires a sequence of images $I$ and the keyframes $F$ as an input and estimates the posterior density as a set of weighted particles $S$ in each time $t$. 

Figure 6. Tracking results showing the effectiveness of considering multiple pose hypotheses. Results of our approach maintaining 1 and 100 particles are depicted in dark and bright wireframes, respectively. For fair comparison, the same parameters are used except the number of particles. The complex background textures often cause false edge correspondences. Thus the single pose hypothesis tracker drifts during the entire tracking, while the tracker considering 100 pose hypotheses shows robust tracking results.
In this section, we validate our proposed particle filter-based tracker via comprehensive experiments. We compare the performance of our approach with that of the single pose hypothesis tracker (Choi and Christensen 2010) and a state-of-the-art tracker, BLORT (Mörwald et al. 2010). For the comparison, we obtained a set of real image sequences as well as synthetic image sequences for quantitative analysis.

To obtain the synthetic image sequences, we first tuned the projection matrix in OpenGL with the intrinsic camera parameters of our monocular camera. We then rendered the objects with texture mapping to visualize our target objects as realistic as possible. In addition, we prepared two backgrounds, simple white and complex textured, so that the performance comparison between two different backgrounds is possible. During rendering, we was continuously changing the camera position and orientation in OpenGL to simulate the real camera motion. The object poses during the rendering were saved to be used as known ground truth. For the real image sequences, we placed the monocular camera around the target objects and moved the camera.

Our system is composed of a standard desktop computer (Intel Core2 Quad CPU Q9300, 3.25G RAM, NVIDIA Quadro FX 570) and a Point Grey Research’s Flea 1394 camera (640 × 480 resolution). The CAD models of the “Teabox”, “Book” and “Cup” were designed by using Blender™ which is an open source 3D modeling tool. The “Car door” model was provided by an automobile company. We converted all of the models to the OBJ format to be used in our C++ implementation.

To verify our approach, we presents a series of comparative experiments:

1. Effectiveness of considering edge orientations
2. Effectiveness of considering multiple pose hypotheses
3. Effectiveness of performing RANSAC

4. Effectiveness of employing AR state dynamics

First, we examine the effect of considering edge orientations in Section 4.1 where our approach with only one particle is compared with the baseline in Choi and Christensen (2010). Secondly, the effectiveness of considering multiple pose hypotheses is verified through a comparison between trackers having single and multiple particles in Section 4.2. In Section 4.3, the effectiveness of the refinement process using RANSAC is scrutinized. Lastly, the benefit of employing the first-order AR state dynamics is discussed in Section 4.4. Each comparative experiment represents results of synthetic followed by real image sequences. For fair performance comparison, all parameters are the same except the compared parameter. After showing the comparative experiments, we show the re-initialization capability of our approach in Section 4.5. Finally, we compare the performance of our approach with that of a state-of-the-art tracker in Section 4.6.

4.1 Effectiveness of considering edge orientations

When we find edge correspondences, we consider orientations of the edge points. By looking up orientations, we exclude edge correspondences having large differences in orientation between the sample points on model edges and the corresponding edge points from an input image. For this orientation testing, a simple indication function was defined in (17). It seems a simple enhancement, but even this modification significantly enhance tracking performance.

To investigate the effectiveness of considering edge orientations, we run our approach with only one particle in the synthetic and real image sequences. Since only one particle is considered in here, Gaussian noise is not added in propagation. As a reference,
Figure 9. The 6-D pose and normalized residual plots of the results in Figure 8a. With the AR state dynamics, our approach shows better performance in terms of accuracy than the one without the dynamics. Especially, when the camera undergoes abrupt changes in position or orientation, the approach with the linear state dynamics exhibits agile responses, while the approach without the state dynamics results in bigger errors.

A similar single pose hypothesis tracker of Choi and Christensen (2010) that does not consider the edge orientations was also run in the same image sequences. The tracking results are presented in Figure 4 and Extension 1 where the results with and without considering edge orientations, which is equivalent to the baseline, are depicted in bright and dark wireframes respectively. Note that both of the pose results are depicted in the same image sequences to clearly show the difference of the two approaches. Although the proposed approach uses the limited number of particles, it shows good tracking performance in the simple background synthetic sequence (Figure 4a) and acceptable tracking performance in the cluttered background real sequence (Figure 4b). However, the baseline is frequently stuck in local minima, and hence it drifts over the textured regions of the object or the background clutter. To decompose pose results, 6-D pose and residual plots of the “Teabox” object are presented in Figure 5. According to the plots, we can easily see the difference where the proposed approach, PF \( (N = 1, \lambda_a = 0.0) \), shows much better results than the previous approach, Single IRLS. A quantitative analysis of this and following tests on synthetic sequences is presented in Table 1 which shows the root mean square (RMS) errors. For each image sequence, the upper row shows the RMS errors of the baseline and the lower five rows are the RMS errors of the proposed approach in various parameter settings for which the number of particles \( N \), the AR parameter \( \lambda_a \), and operation of RANSAC were altered.

The difference between the previous and the proposed approaches is mainly due to false edge correspondences in the baseline, while the proposed approach discards some portion of the false edge correspondences by comparing edge orientation.
4.2 Effectiveness of considering multiple pose hypotheses

As shown in Section 4.1, when the background is relatively simple, single pose hypothesis edge-based tracking shows reasonable performance. But it is quite challenging to reliably track an object when the background is complex or there is an amount of clutter. These challenging situations often make erratic edge correspondences, and hence single pose hypothesis tracking can be a fragile solution in these cases. To validate this argument, we compare two versions of our tracker using 1 and 100 particles. The comparative tracking results are shown in Figure 6 and Extension 2, where results of the proposed tracker with 1 and 100 particles are drawn in dark and bright wireframes, respectively. For clear visualization, we only display the mean of particles calculated via the valid mean of particles in (12). For fair comparison, the same parameters are used except the number of particles.

The complex background textures often cause false edge correspondences. Thus the single pose hypothesis tracker drifts during the entire tracking, while the tracker considering multiple pose hypotheses shows robust tracking results. Since our particle filter considers multiple pose hypotheses and resamples based on measurement likelihood, it is quite robust to false edge correspondences from which the single pose hypothesis tracker is often suffered.

4.3 Effectiveness of performing RANSAC

As comparing orientations of edge points, our approach discards some false edge correspondences. But it still tend to have false edge matches because it might happen to have similar orientations but false correspondences. Our approach takes an additional refinement process based on RANSAC (Fischler and Bolles 1981). In this section, we examine the advantage of performing RANSAC refinement. The results of our approach with \((N = 100, \lambda_a = 0.0, \text{RANSAC})\) and without \((N = 100, \lambda_a = 0.0)\) RANSAC are shown in bright and dark wireframes in Figure 7 and Extension 3 respectively.

In the RMS error calculation, it is straightforward to compare rotational errors except the “Cup” object because it is symmetric and thus an exact set of three orientations cannot be estimated. Instead of estimating the three rotations, we calculate the angle between unit vectors whose directions are coincide with the axis of symmetry of the object in the object coordinate system via

\[
c^u_y = \mathbf{C}_O \cdot \mathbf{O}^u_y
\]

\[
\hat{c}^u_y = \mathbf{C}_{\hat{O}} \cdot \mathbf{O}^u_y
\]

where \(\mathbf{C}_O \in SE(3)\) is the pose of the object with respect to the camera frame estimated by our visual tracker, \(\mathbf{C}_{\hat{O}} \in SE(3)\) is the pose of the rendered object with respect to the virtual camera frame saved in OpenGL rendering, and \(\mathbf{C}^u_y = (0, 1, 0, 0)^\top\) is the unit vector of the axis of symmetry. Note that the fourth element of \(\mathbf{C}^u_y\) is 0, and hence only rotation is considered. The angle between these unit vectors can be calculated by the inner product as

\[
\theta = \arccos (c^u_y \mathbf{C}^u_y).
\]

The rotational RMS errors of the “Cup” object are calculated with this angle in Table 1.

According to results in Figure 7 and Table 1, the one with RANSAC clearly exhibits better results. By performing RANSAC, some false edge correspondences which are inconsistent with the best pose hypothesis are discarded. Thus our approach with RANSAC tracks robustly, while the one without RANSAC is frequently misled by the complex background clutter.

4.4 Effectiveness of employing AR state dynamics

To verify the effect of the AR state dynamics, we execute the proposed approach with and without the AR state dynamics. To disable the dynamics, we set the parameter \(\lambda_a\) in the AR state dynamics equation (14) as 0 which is equivalent to a random walk model. For fair comparison, we use the same parameters except the AR parameter. We test in the synthetic and real image sequences of the “Car door” object. The tracking results are presented in Figure 8 and Extension 4.

Although both use the same number of particles, Gaussian noise, and measurement likelihood, the tracking performances are quite distinctive. This difference is mainly due to the AR state dynamics which propagates particles according to the camera motion.

4.5 Re-initialization

In our previous system (Choi and Christensen 2010), we used a simple heuristic in which the difference in position of the object between frames and the number of valid sample points are monitored to trigger re-initialization. While that heuristic works well when the pose hypothesis drifts fast, it might not always be the case when the hypothesis stuck in local minima. We propose another way for re-initialization by taking advantage of multiple
Table 1. RMS errors and computation time in synthetic image sequences (Baseline vs. PF)

<table>
<thead>
<tr>
<th>Objects</th>
<th>Mode</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
<th>Time*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teabox</td>
<td>Baseline (single IRLS)$\dagger$</td>
<td>51.131</td>
<td>44.509</td>
<td>263.258</td>
<td>26.280</td>
<td>24.330</td>
<td>48.323</td>
<td>20.62</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 1, \lambda_a = 0.0$)</td>
<td>5.234</td>
<td>3.323</td>
<td>16.034</td>
<td>3.555</td>
<td>4.626</td>
<td>6.727</td>
<td>18.11</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$)</td>
<td>4.616</td>
<td>2.498</td>
<td>13.623</td>
<td>2.558</td>
<td>3.940</td>
<td>5.867</td>
<td>450.57</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$)</td>
<td>4.272</td>
<td>2.255</td>
<td>12.430</td>
<td>2.375</td>
<td>3.882</td>
<td>3.628</td>
<td>449.39</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$, RANSAC)</td>
<td>3.503</td>
<td>1.930</td>
<td>8.917</td>
<td>1.936</td>
<td>3.227</td>
<td>2.597</td>
<td>459.49</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$, RANSAC)</td>
<td>3.234</td>
<td>1.829</td>
<td>8.062</td>
<td>1.843</td>
<td>3.057</td>
<td>2.530</td>
<td>464.69</td>
</tr>
<tr>
<td></td>
<td>Baseline (single IRLS)$\dagger$</td>
<td>63.074</td>
<td>27.075</td>
<td>83.377</td>
<td>25.028</td>
<td>38.839</td>
<td>62.695</td>
<td>23.05</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 1, \lambda_a = 0.0$)</td>
<td>20.209</td>
<td>21.130</td>
<td>55.466</td>
<td>35.613</td>
<td>28.153</td>
<td>61.006</td>
<td>22.61</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$)</td>
<td>4.521</td>
<td>8.357</td>
<td>13.974</td>
<td>4.431</td>
<td>7.717</td>
<td>5.593</td>
<td>471.78</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$)</td>
<td>3.028</td>
<td>4.846</td>
<td>11.739</td>
<td>3.313</td>
<td>5.965</td>
<td>3.637</td>
<td>504.13</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$, RANSAC)</td>
<td>3.125</td>
<td>4.398</td>
<td>10.207</td>
<td>2.712</td>
<td>4.976</td>
<td>3.324</td>
<td>497.05</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$, RANSAC)</td>
<td>2.795</td>
<td>4.953</td>
<td>9.503</td>
<td>2.410</td>
<td>4.854</td>
<td>3.400</td>
<td>519.03</td>
</tr>
<tr>
<td>Book</td>
<td>Baseline (single IRLS)$\dagger$</td>
<td>16.980</td>
<td>37.375</td>
<td>131.320</td>
<td>8.372</td>
<td>22.691</td>
<td>20.273</td>
<td>21.83</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 1, \lambda_a = 0.0$)</td>
<td>76.376</td>
<td>95.839</td>
<td>92.012</td>
<td>3.571</td>
<td>14.141</td>
<td>27.524</td>
<td>20.90</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$)</td>
<td>4.633</td>
<td>3.570</td>
<td>22.881</td>
<td>1.107</td>
<td>1.718</td>
<td>3.661</td>
<td>802.08</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$)</td>
<td>4.502</td>
<td>3.402</td>
<td>22.240</td>
<td>0.954</td>
<td>1.508</td>
<td>3.791</td>
<td>799.30</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$, RANSAC)</td>
<td>4.298</td>
<td>3.420</td>
<td>21.107</td>
<td>1.037</td>
<td>1.618</td>
<td>3.563</td>
<td>815.76</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$, RANSAC)</td>
<td>4.140</td>
<td>3.171</td>
<td>20.084</td>
<td>0.876</td>
<td>1.408</td>
<td>3.545</td>
<td>811.00</td>
</tr>
<tr>
<td>Cup$\dagger$</td>
<td>Baseline (single IRLS)$\dagger$</td>
<td>36.317</td>
<td>31.552</td>
<td>122.737</td>
<td>–</td>
<td>78.843</td>
<td>–</td>
<td>33.65</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 1, \lambda_a = 0.0$)</td>
<td>189.488</td>
<td>19.947</td>
<td>58.342</td>
<td>–</td>
<td>25.917</td>
<td>–</td>
<td>32.63</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$)</td>
<td>3.842</td>
<td>3.098</td>
<td>14.837</td>
<td>–</td>
<td>2.116</td>
<td>–</td>
<td>1913.49</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$)</td>
<td>3.574</td>
<td>3.008</td>
<td>13.820</td>
<td>–</td>
<td>1.960</td>
<td>–</td>
<td>1910.34</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$, RANSAC)</td>
<td>3.112</td>
<td>2.544</td>
<td>12.076</td>
<td>–</td>
<td>1.777</td>
<td>–</td>
<td>1928.42</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$, RANSAC)</td>
<td>2.933</td>
<td>2.422</td>
<td>11.063</td>
<td>–</td>
<td>1.515</td>
<td>–</td>
<td>1925.59</td>
</tr>
<tr>
<td>Car door</td>
<td>Baseline (single IRLS)$\dagger$</td>
<td>126.662</td>
<td>57.309</td>
<td>127.284</td>
<td>–</td>
<td>33.180</td>
<td>–</td>
<td>37.67</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 1, \lambda_a = 0.0$)</td>
<td>65.645</td>
<td>48.978</td>
<td>100.036</td>
<td>–</td>
<td>71.873</td>
<td>–</td>
<td>36.53</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$)</td>
<td>8.856</td>
<td>11.778</td>
<td>22.064</td>
<td>–</td>
<td>7.041</td>
<td>–</td>
<td>1913.69</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$)</td>
<td>8.312</td>
<td>11.417</td>
<td>20.045</td>
<td>–</td>
<td>6.186</td>
<td>–</td>
<td>1911.11</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.0$, RANSAC)</td>
<td>7.111</td>
<td>18.625</td>
<td>15.577</td>
<td>–</td>
<td>6.197</td>
<td>–</td>
<td>1936.90</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5$, RANSAC)</td>
<td>6.997</td>
<td>18.516</td>
<td>17.134</td>
<td>–</td>
<td>6.333</td>
<td>–</td>
<td>1949.41</td>
</tr>
</tbody>
</table>

$\dagger$ The results of the previous approach (Choi and Christensen 2010) are compared as a baseline.

* The error units of translation and rotation are millimeter and degree, respectively. Better results are indicated in bold type.

§ The unit of computation time is milliseconds per frame.

Since the “Cup” object is symmetric and thus exact orientation cannot be estimated, we calculate the angle of the axis of symmetry.
pose hypotheses. As in Algorithm 1, our system re-initializes when the number of effective particles $\hat{N}_{\text{eff}}$ is below a threshold.

To verify this method, we run the proposed tracker on two real image sequences. The tracking results are shown in Figure 11 and Extension 5, and the number of effective particles for the first image sequence (Figure 11a) is plotted over the frame numbers in Figure 10. When the tracked object goes out of the field of view or images are blurred because of camera shaking, the $\hat{N}_{\text{eff}}$ decreases significantly below the threshold value $N_{\text{thres}}$, and that triggers the re-initialization. This also works when the object is disappearing slowly. When the object gradually moves out of the field of view, the particles still follow the real trajectory of the object. After the object is completely disappeared, the particles have no visible sample points. Since it is impossible to evaluate the likelihood (21) without visible sample points and edge correspondences from them, the weight of the particle is set to zero.

Figure 11. Tracking results showing the capability of re-initialization. The set of dark wireframes represent 100 particles and the bright thick wireframe shows the mean of particles on each image. When the tracked object goes out of the field of view and images are blurred because of camera shaking, our tracker re-initializes by itself. At frame number $t = 0, 247, 462, 844$ at the “Book” and $t = 0, 314, 585, 743$ at the “Car door”, our approach is (re-)initialized. Since some keypoint correspondences were wrong, there are erroneous initial pose hypotheses. But the tracker quickly converges to the global optimum as the bad pose hypotheses are removed in the importance resampling process.
This reduces the number of effective particles, and thus it enables the re-initialization process. During re-initialization, the tracker matches keypoints until it has at least the minimum number of keypoint correspondences $\tau_n$. Once enough keypoint correspondences are acquired, the proposed system initializes particles as explained in Section 3.4 and Algorithm 2.

4.6 Comparison with BLORT tracker

So far we have shown a series of comparative experiments with Choi and Christensen (2010) or our proposed approach with several different parameters. However, it would be more convincing if we compare our proposed approach with the state-of-the-art tracker. To compare the performance of our approach with an existing solution, we chose BLORT tracker (Mörwald et al. 2010) because it combines state-of-the-art methods for object recognition and tracking as well as is publicly available.\footnote{BLORT - The Blocks World Robotic Vision Toolbox: http://users.acin.tuwien.ac.at/mzillich/?site=4} By combining SIFT-based object recognition and edge-based particle filtering, BLORT provides reliable solution for robotic research. For robustness, BLORT considers edges from surface texture of objects in particle likelihood evaluation. Time consuming parts, such as hidden face removal, image processing, texture mapping, and particle filtering, are efficiently processed by GPU.

We used default parameters provided in the BLORT software package except the number of particles $N$ and recursions $R$. Since BLORT employs the recursive particle filtering (Mörwald et al. 2009), the number of recursion also determines the robustness of tracking. Thus we changed these parameters in order to compare our approach with various settings of BLORT. Since BLORT cannot handle arbitrary complex shaped objects, we excluded “Car door” object in this experiments. We converted our
OBJ formatted 3D polygonal mesh models to PLY format, which BLORT accepts. In addition, texture images of these objects were learned to be used in the likelihood evaluation of its particle filtering, and SIFT features were also saved to be used in the SIFT-based object recognition. For our tracker, 100 particles, the AR state dynamics, and RANSAC refinement were employed (i.e. \( N = 100, \lambda_a = 0.5, \) RANSAC).

Our approach and BLORT were tested in the synthetic image sequences. While we fixed the parameters of our approach, we tried several different numbers of particles (\( N = 100 \) or \( N = 200 \)) and recursions (\( R = 2 \) or \( R = 4 \)) for BLORT so that we can compare our approach with various settings of BLORT.\(^2\) Selected tracking results in the “Book” and the “Teabox” synthetic image sequences are shown in Figure 12 and Extension 6 where results of our approach and BLORT are represented in bright and dark wireframes respectively. Both trackers follow the true pose of the objects, but BLORT exhibits abrupt jitter noise. This noise can be clearly visible in 6-D pose and residual plots in Figure 13, where the tracked pose results in the “Book” simple background image sequence are represented in pose and residual plots. With the sufficient number of particles and recursions (\( N = 200, R = 4 \)), BLORT reports similar performance to our particle filter. BLORT even shows slightly better results in \( z \) translation possibly due to considering edges from texture of the object. However, BLORT severely suffers from jitter noise in rotations.

These selected results in Figure 12 are the best results from BLORT. In many cases, BLORT lost tracking in the middle of image sequences. In Table 2, RMS errors and computation time of all experiments for the synthetic image sequences are presented. As shown in the table, BLORT reports poor accuracy, especially

---

**Figure 13.** The 6-D pose and normalized residual plots of the results in Figure 12a. With the sufficient number of particles and recursions (\( N = 200, R = 4 \)), BLORT reports similar performance to our particle filter. BLORT even shows slightly better results in \( z \) translation possibly due to considering edges from texture of the object. However, BLORT severely suffers from jitter noise in rotations.

\(^2\)The default number of particles and recursions in BLORT are \( N = 200 \) and \( R = 2 \) respectively.
in complex background sequences (e.g. "Book" and "Cup"). While our approach outperforms BLORT in terms of accuracy in many cases, BLORT shows comparable results in "Teabox" object. The reason why BLORT gives better results for "Teabox" object is possibly due to the amount of texture. Since BLORT takes advantage of edges coming from surface texture in its particle filtering, sufficient texture information is crucial to a robust tracking. "Teabox" object has relatively plentiful textures compared to "Book" and "Cup" objects. Thus BLORT reports less RMS errors even in the "Teabox" complex background sequence, while it even shows large errors in "Cup" simple background sequence. On the contrary, our particle filter relies on sharp edges in the 3D polygonal model, and thus our approach is independent of textures. Better accuracy of our tracker is further achieved via IRLS optimization, AR state dynamics, and RANSAC refinement. Nor surprisingly, accuracy of BLORT gets better as the number of particles $N$ or recursions $R$ increases, but at the same time the computation time increases. As BLORT exploited the parallel power of GPU, it shows higher frame rates than our approach.

Similar comparative experiments are done in the real image sequences. The results in the "Book" and "Cup" image sequences are presented in Figure 14 and Extension 7. While our particle filter (bright wireframes) well tracks the target objects, BLORT (dark wireframes) loses the targets in the middle of the sequences. Once BLORT failed to track the target, it remained as lost without any successful recovery until the end of sequence. Even before the tracking failures, pose results from BLORT are not well aligned to the global optimum.

BLORT reported good performance in simple background image sequences with well textured objects. However, it did poorly perform in relatively complex background with less textured objects, and its tracking was not as agile as our tracker. The pose results of BLORT showed severe jitter noise, especially in rotation, which is not desirable for robotic applications. We note the high frame rates of BLORT due mainly to the parallelization of the algorithm in GPU. It might imply that exploiting the parallel power from GPU would be worthwhile to accelerate the process-

<table>
<thead>
<tr>
<th>Objects</th>
<th>Mode</th>
<th>RMS Errors*</th>
<th>Time†</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>X  Y  Z Roll Pitch Yaw</td>
<td></td>
</tr>
<tr>
<td>Teabox</td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>3.234 1.829 8.062 1.843 3.057</td>
<td>2.530 464.69</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>6.344 3.570 6.825 1.915</td>
<td>8.753 4.146</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>4.747 3.164 5.878 1.528</td>
<td>9.689 1.946</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>3.897 3.169 5.605 1.511</td>
<td>6.127 9.689</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 4$)</td>
<td>3.684 2.289 4.821 1.276</td>
<td>5.572 6.127</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>2.795 4.953 9.503 2.410 4.854</td>
<td>3.400 519.03</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>12.639 5.808 18.893 3.559</td>
<td>16.471 11.607</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>42.314 25.175 110.392 4.246</td>
<td>28.717 10.883</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>5.101 2.630 7.608 2.550</td>
<td>7.379 8.062</td>
</tr>
<tr>
<td>Book</td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>4.140 3.171 20.084 0.876 1.408</td>
<td>3.545 811.00</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>6.651 6.194 29.169 2.439</td>
<td>11.056 14.747</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>4.736 5.989 22.240 1.904</td>
<td>7.898 11.589</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>3.962 6.119 21.459 1.999</td>
<td>5.777 9.689</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 4$)</td>
<td>3.643 5.893 18.073 1.132</td>
<td>4.346 8.780</td>
</tr>
<tr>
<td>Cup†</td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>3.195 3.843 8.239 1.589 1.979</td>
<td>3.697 819.85</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>73.011 79.695 262.610 27.580</td>
<td>47.031 38.527</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>73.198 90.931 216.233 44.918</td>
<td>17.872 23.865</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>73.824 86.040 232.869 38.145</td>
<td>31.934 23.138</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 4$)</td>
<td>59.819 76.913 229.491 56.344</td>
<td>37.451 29.128</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>2.933 2.422 11.063 – 1.515†</td>
<td>1925.59</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>41.878 24.509 187.267 – 57.789</td>
<td>– 36.91</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>30.419 26.715 93.366 – 83.586</td>
<td>– 63.46</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>56.216 45.414 610.823 – 55.184</td>
<td>– 52.97</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 4$)</td>
<td>3.979 5.006 27.852 – 5.091</td>
<td>– 222.57</td>
</tr>
<tr>
<td></td>
<td>PF ($N = 100, \lambda_a = 0.5, \text{RANSAC}$)</td>
<td>6.997 18.516 17.134 – 6.333†</td>
<td>1949.41</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 2$)</td>
<td>86.748 75.334 226.736 – 69.987</td>
<td>– 70.07</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 2$)</td>
<td>79.777 35.124 114.818 – 69.422</td>
<td>– 56.70</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 100, R = 4$)</td>
<td>36.628 73.228 179.439 – 85.200</td>
<td>– 77.02</td>
</tr>
<tr>
<td></td>
<td>BLORT ($N = 200, R = 4$)</td>
<td>64.853 52.676 132.043 – 105.518</td>
<td>– 98.50</td>
</tr>
</tbody>
</table>

* The error units of translation and rotation are millimeter and degree, respectively. Better results are indicated in bold type.
† The unit of computation time is milliseconds per frame.
§ Since the "Cup" object is symmetric and thus exact orientation cannot be estimated, we calculate the angle of the axis of symmetry.
Conclusions

We have presented an approach to 3D visual object tracking based on a particle filtering algorithm on the $SE(3)$ group. For fast particle convergence, we employed keypoint features and initialized particles by solving the PnP problem. Particles are propagated by the state dynamics which is given by the AR process on the $SE(3)$, and the state dynamics distributed particles more effectively. Edge correspondences were first enhanced by considering orientations of the edge points, and they were further refined through the RANSAC process. Measurement likelihood was calculated from both the residual and the number of valid sample points of the edge correspondences. During the tracking, the proposed system appropriately re-initialized by itself when the number of effective particles was below a threshold. Our approach has been tested via various experiments in which our multiple pose hypotheses tracker has shown notable performance on challenging background and clutter.

Although our approach is not yet capable of real-time tracking, our algorithm can be further improved by exploiting the parallel power of GPU. The main bottleneck in our algorithm is the likelihood evaluation and the IRLS optimization, and these tasks are needed to execute per particle. Since our current implementation does not take advantage of parallel power, we expect that our algorithm can be accelerated with GPU (Klein and Murray 2006, Michel et al. 2007, Mörwald et al. 2010).

ACKNOWLEDGMENTS

This work was partially funded and initiated under a Collaborative Research Project between Georgia Tech and the General Motors R&D, Manufacturing Systems Research Laboratory. The work was further developed under a joint contract with Boeing Research and Technology. The support from General Motors and
the Boeing Corporation is gratefully acknowledged.

A Appendix A: Index to Multimedia Extensions

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

Table 3. Table of Multimedia Extension

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>Tracking results, considering edge orientation (Figure 4)</td>
</tr>
<tr>
<td>2</td>
<td>Video</td>
<td>Tracking results, considering multiple pose hypotheses (Figure 6)</td>
</tr>
<tr>
<td>3</td>
<td>Video</td>
<td>Tracking results, performing RANSAC (Figure 7)</td>
</tr>
<tr>
<td>4</td>
<td>Video</td>
<td>Tracking results, employing AR state dynamics (Figure 8)</td>
</tr>
<tr>
<td>5</td>
<td>Video</td>
<td>Tracking results, capability of re-initialization (Figure 11)</td>
</tr>
<tr>
<td>6</td>
<td>Video</td>
<td>A comparison (PF vs. BLORT), synthetic image sequences (Figure 12)</td>
</tr>
<tr>
<td>7</td>
<td>Video</td>
<td>A comparison (PF vs. BLORT), real image sequences (Figure 14)</td>
</tr>
</tbody>
</table>

B Appendix B: IRLS and its Jacobian derivation

Given the current pose hypothesis \( X_t \), the residual vector \( \hat{r} \in \mathbb{R}^N \) which represents Euclidean distances between the 2D projected sampled points \( p \) and their corresponding nearest edge points \( q \), we would like to find \( \hat{\mu} \in \mathbb{R}^N \) which minimizes the residual \( \hat{r} \) as follows:

\[
\hat{\mu} = \arg \min_\mu \sum_{i=1}^{N_i} ||\hat{r}_i||^2 
\]

(29)

\[
= \arg \min_\mu \sum_{i=1}^{N_i} ||q_i - p_i||^2. 
\]

(30)

Note that each element of \( \hat{\mu} \) is the coefficient of each basis of the Lie algebra \( \mathfrak{se}(3) \) in (6). Thus, the optimized pose hypothesis \( \hat{X}_t \) can be presented as:

\[
\hat{X}_t = X_t \cdot \exp(\sum_{i=1}^{N} \hat{\mu}_i E_i) 
\]

(31)

where \( \exp : \mathfrak{se}(3) \to SE(3) \) is the exponential map. From the camera model in (1), the projected point \( p_i \) in (30) of the \( i^{th} \) 3D sampled point in object coordinates \( P_i^O = (x_i^O, y_i^O, z_i^O, 1)^T \) is presented by

\[
p_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \text{Project}(K, \hat{X}_t, P_i^O) 
\]

(32)

where \( K \) is the intrinsic parameter matrix (3).

Given \( p_i \) and its unit normal vector \( n_i \), we can build a Jacobian matrix \( J \in \mathbb{R}^{N_i \times 6} \) by computing the partial derivative of \( n_i^T p_i \) in the direction \( \mu \) at \( \mu = 0 \):

\[
J_{ij} = \frac{\partial n_i^T p_i}{\partial \mu} \bigg|_{\mu=0} 
\]

(33)

\[
= n_i^T \frac{\partial}{\partial \mu} \left( \begin{array}{c} u_i \\ v_i \end{array} \right) \bigg|_{\mu=0} 
\]

(34)

\[
= n_i^T \frac{\partial}{\partial \mu} \left( \text{Project}(K, \hat{X}_t, P_i^O) \right) \bigg|_{\mu=0} 
\]

(35)

in order to find \( \hat{\mu} \) which minimizes the residual \( \hat{r} \) by solving the following equation:

\[
J \hat{\mu} = \hat{r}. 
\]

(36)

For convenience, we can split \( \text{Project}() \) in the camera model (1) into two parts as follows:

\[
\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}_i \\ \hat{v}_i \\ 1 \end{pmatrix} 
\]

(37)

\[
\begin{pmatrix} \hat{u}_i \\ \hat{v}_i \end{pmatrix} = \begin{pmatrix} \frac{e^x}{z_i} \\ \frac{e^y}{z_i} \end{pmatrix}. 
\]

(38)

Their corresponding Jacobian matrices are then obtained as:

\[
J_K = \begin{pmatrix} \frac{\partial u_i}{\partial x_i} & \frac{\partial u_i}{\partial y_i} & \frac{\partial u_i}{\partial z_i} \\ \frac{\partial v_i}{\partial x_i} & \frac{\partial v_i}{\partial y_i} & \frac{\partial v_i}{\partial z_i} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -x_i^2 \\ 0 & 1 & -y_i^2 \end{pmatrix}. 
\]

(39)

\[
J_P = \begin{pmatrix} \frac{\partial x_i}{\partial x_i} & \frac{\partial x_i}{\partial y_i} & \frac{\partial x_i}{\partial z_i} \\ \frac{\partial y_i}{\partial x_i} & \frac{\partial y_i}{\partial y_i} & \frac{\partial y_i}{\partial z_i} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -x_i^2 \\ 0 & 1 & -y_i^2 \end{pmatrix}. 
\]

(40)

If we calculate the partial derivative of the \( \hat{X}_t \) in the direction \( \mu_j \) at \( \mu = 0 \), we have the following from the equation of optimized pose in (31):

\[
\frac{\partial \hat{X}_t}{\partial \mu_j} \bigg|_{\mu=0} = X_t \frac{\partial}{\partial \mu_j} \exp(\sum_{i=1}^{N} \hat{\mu}_i E_i) \bigg|_{\mu=0} = X_t E_j. 
\]

(41)

(42)

We then obtain the partial derivative of the \( P_i^C \) in the direction \( \mu_j \) at \( \mu = 0 \) via the transformation (2) from object to camera coordinate frames:

\[
\frac{\partial P_i^C}{\partial \mu_j} \bigg|_{\mu=0} = \frac{\partial \hat{X}_t}{\partial \mu_j} \bigg|_{\mu=0} P_i^O = X_t E_j P_i^O. 
\]

(43)
With the two Jacobian matrices in (39), (40) and the partial derivative of the $P^i_i$ in (43), we finally obtain the Jacobian matrix $J$ from the intermediate result in (35) as

$$J_{ij} = n_i^T \frac{\partial}{\partial \mu_j} \left( \text{Project} \left( K, \hat{X}^i, P^i_i \right) \right) \Big|_{\mu=0} \tag{44}$$

$$= n_i^T J_K \left( \begin{bmatrix} J_p & 0 \\ 0 & 0 \end{bmatrix} \frac{\partial P^i_i}{\partial \mu_j} \right)_{\mu=0} \tag{45}$$

$$= n_i^T J_K \left( \begin{bmatrix} J_p & 0 \\ 0 & 0 \end{bmatrix} \right) X_i E_i \hat{X}^i \tag{46}.$$

We could solve the equation (36) using the general pseudoinverse of $J$:

$$\hat{\mu} = (J^T J)^{-1} J^T \hat{r}. \tag{47}$$

Above equation, however, tends to be sensitive to erroneous measurements. As alternatives, M-estimators (Huber 1981) or RANSAC (Fischler and Bolles 1981) are preferable. Especially, M-estimator, in which a robust penalty function to the residuals is applied, is preferred to RANSAC when the majority of measurements are inliers. For a M-estimator, it is common to define a function $\Psi(r) = \frac{1}{\lambda v} \exp \left( \frac{r}{\lambda v} \right)$ which penalizes for the residual errors $r$ and $\lambda v$ is a parameter which is usually set to one standard deviation of the inliers. With this function, we can have a diagonal matrix $W = \text{diag}(\Psi(r_1), \Psi(r_2), \cdots, \Psi(r_{N_h})) \in \mathbb{R}^{N_h \times N_h}$. Then $\hat{\mu}$ can be estimated via weighted least squares as:

$$\hat{\mu} = (J^T W J)^{-1} J^T W \hat{r}. \tag{48}$$

Drummond and Cipolla (2002) introduced this as Iteratively Reweighted Least Squares (IRLS) in which the penalty function is recursively updated by computing the weighted least squares problem. They mentioned performing a single iteration for each image frame was enough to converge to the true pose.

References


