The International Journal of
Robotics Research
31(4) 498–519
© The Author(s) 2012
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364912437213
ijr.sagepub.com

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

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.

Keywords

Visual tracking, object recognition, pose estimation, keypoint, edge, particle filtering, Euclidean group, Lie group

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 the general scale (Drummond and Cipolla 2002; Comport et al. 2004) and at micro or nano scales (Yesin and Nelson 2005; Kratochvil 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)

Robotics and Intelligent Machines, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

Corresponding author:
Changhyun Choi, Robotics and Intelligent Machines, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA.
Email: cchoi@cc.gatech.edu

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, *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. 2010), 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 (Mörwald et al. 2010) employed scale-invariant feature transform (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) and Pupilli and Calway (2006) have used an 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 random sample consensus (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 for a few (Armstrong and Zisserman 1995; Teulière 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; Teulière 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 Sections 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 Sections 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

$$\mathbf{p} = \texttt{Project}(\mathbf{K}, \mathbf{X}_t, \mathbf{P}^{\mathcal{O}}) = \mathbf{K}\begin{pmatrix} \frac{x^{\mathcal{C}}}{z^{\mathcal{C}}} \\ \frac{y^{\mathcal{C}}}{z^{\mathcal{C}}} \\ 1 \end{pmatrix} \tag{1}$$

$$\mathbf{P}^{\mathcal{C}} = \mathbf{X}_t \mathbf{P}^{\mathcal{O}} \tag{2}$$

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

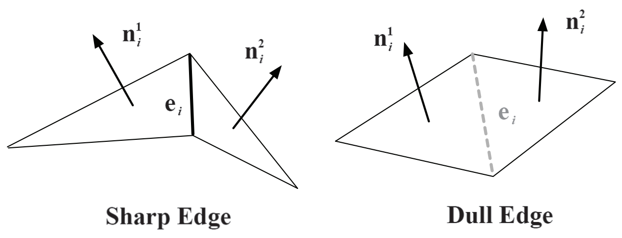


Fig. 1. Determining salient edges. We use the face normal vectors available in the model.

$$\mathbf{K} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \tag{3}$$

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 1, 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

$$I(\mathbf{e}_i) = \begin{cases} 1 & \text{if } \mathbf{n}_i^{1\mathsf{T}} \mathbf{n}_i^2 \le \tau_s \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{n}_i^1$ and $\mathbf{n}_i^2$ are the face normal unit vectors of the two adjacent faces which share the ith edge, $\mathbf{e}_i$. 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 2. 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 the boundary of the dull edges is performed at runtime, and thus it is desirable to avoid the test 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 six-degree-of-freedom (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), and 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):

$$\mathbf{X}_t = \mathbf{X}_{t-1} \cdot \exp(A(\mathbf{X}, t)\,\Delta t + \mathbf{dW}_t \sqrt{\Delta t}), \tag{5}$$
$$\mathbf{dW}_t = \sum_{i=1}^{6} \epsilon_{t,i} \mathbf{E}_i,$$
$$\boldsymbol{\epsilon}_t = (\epsilon_{t,1}, \ldots, \epsilon_{t,6})^{\mathsf{T}} \sim \mathcal{N}(\mathbf{0}_{6\times1}, \Sigma_w)$$

where $\mathbf{X}_t \in SE(3)$ is the state at time t, $A : SE(3) \mapsto \mathfrak{se}(3)$ is a possibly nonlinear map, $\mathbf{dW}_t$ represents the Wiener process noise on $\mathfrak{se}(3)$ with a covariance $\Sigma_w \in \Re^{6\times6}$, and $\mathbf{E}_i$ are the ith basis elements of $\mathfrak{se}(3)$:

$$\mathbf{E}_1 = \begin{pmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \quad \mathbf{E}_2 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \quad \mathbf{E}_3 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix},$$
$$\mathbf{E}_4 = \begin{pmatrix} 0&0&0&0 \\ 0&0&-1&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{pmatrix}, \quad \mathbf{E}_5 = \begin{pmatrix} 0&0&1&0 \\ 0&0&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \quad \mathbf{E}_6 = \begin{pmatrix} 0&-1&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}. \tag{6}$$

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.

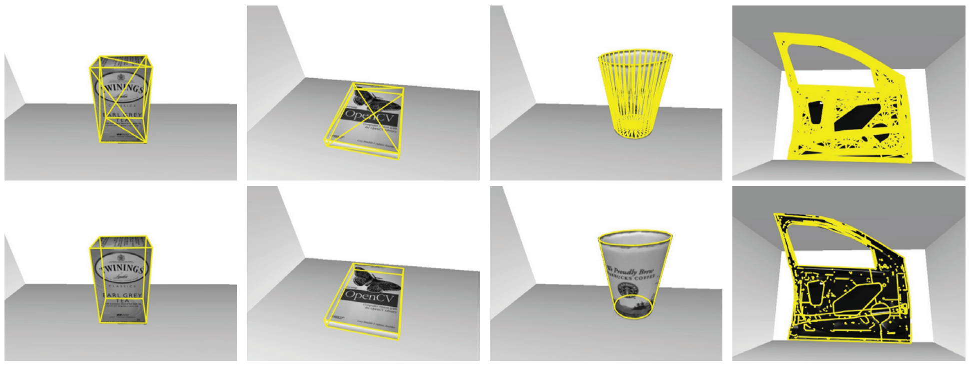


Fig. 2. Original CAD models (upper row) and selected salient edges (lower row) of our target objects. Our approach automatically selects sharp edges which are likely to be visible in real images. From left to right, ‘Teabox’, ‘Book’, ‘Cup’, and ‘Car door’.

Algorithm 1: Particle filtering on the $SE(3)$ group.

```
Data: $\mathcal{I} = \{\mathcal{I}_0, \mathcal{I}_1, \dots, \mathcal{I}_I\}, \mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \dots, (\mathbf{p}_F, \mathbf{P}_F)\}$
Result: $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_I\}$
Params: $N, \mathbf{\Sigma}_w, \lambda_a, \lambda_v, \lambda_r, N_{thres}$
1:  $t \leftarrow 0$
2:  $init \leftarrow 1$
3:  $\mathbf{A}_0 \leftarrow \mathbf{0}_{4\times4}$
4:  while $\mathcal{I}_t \neq 0$ do
5:      if $init = 1$ then
6:          $\mathcal{S}_t \leftarrow$ InitParticle($\mathcal{I}_t, \mathcal{F}$)                                   ⟨2⟩
7:          if $\mathcal{S}_t \neq \{\phi\}$ then
8:              $init \leftarrow 0$
        else
9:          for $n \leftarrow 1$ to $N$ do
10:             $\mathbf{X}_t^{*(n)} \leftarrow$ Propagate($\mathbf{X}_{t-1}^{(n)}, \mathbf{A}_{t-1}^{(n)}, \mathbf{\Sigma}_w$)          (13)
11:             $\mathbf{A}_t^{*(n)} \leftarrow$ AR_vel($\mathbf{X}_t^{*(n)}, \mathbf{X}_{t-1}^{*(n)}, \lambda_a$)                 (14)
12:             $\mathbf{Z}_t^{*(n)} \leftarrow$ Measurement($\mathbf{X}_t^{*(n)}, \mathcal{I}_t$)                       (7)
13:             $\widehat{\mathbf{Z}}_t^{*(n)} \leftarrow$ RANSAC($\mathbf{Z}_t^{*(n)}$)                                  ⟨3⟩
14:             $\tilde{\pi}_t^{*(n)} \leftarrow$ Likelihood($\mathbf{Z}_t^{*(n)}, \widehat{\mathbf{Z}}_t^{*(n)}, \lambda_v, \lambda_r$)       (21)
15:             $\acute{\mathbf{X}}_t^{*(n)} \leftarrow$ IRLS($\mathbf{X}_t^{*(n)}, \widehat{\mathbf{Z}}_t^{*(n)}$)                   (22)(23)
16:         $\pi_t^* \leftarrow$ Normalize($\tilde{\pi}_t^*$)                                       (16)
17:         $\widehat{N_{eff}} \leftarrow$ Neff($\pi_t^*$)                                           (25)
18:         if $\widehat{N_{eff}} \geq N_{thres}$ then
19:             $\mathcal{S}_t \leftarrow$ Resampling($\mathcal{S}_t^*$)
            else
20:             $init \leftarrow 1$
21:     $t \leftarrow t + 1$
```

Algorithm 2: InitParticle($\mathcal{I}, \mathcal{F}$).

```
Data: $\mathcal{I}, \mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \dots, (\mathbf{p}_F, \mathbf{P}_F)\}$
Result: $\mathcal{S} = \{(\mathbf{X}^{(1)}, \pi^{(1)}), \dots, (\mathbf{X}^{(N)}, \pi^{(N)})\}$
Params: $\tau_r, \tau_n, N, m, \mathbf{K}, \tau_\epsilon, \lambda_c$
1:  $\hat{l} \leftarrow 0$
2:  $\widehat{\mathcal{C}} \leftarrow \{\phi\}$
3:  $\mathbf{p}_i \leftarrow$ ExtractSURF($\mathcal{I}$)                                  (Bay et al. 2008)
4:  for $f \leftarrow 1$ to $F$ do
5:      $\{\acute{\mathbf{p}}_i, \acute{\mathbf{P}}_f\} \leftarrow$ BBF($\mathbf{p}_i, \mathbf{p}_f, \mathbf{P}_f, \tau_r$)           (Beis and Lowe 1997)
6:      $\mathcal{C} \leftarrow \{\acute{\mathbf{p}}_i, \acute{\mathbf{P}}_f\}$
7:      $\acute{\mathcal{C}} \leftarrow$ RatioTest($\mathcal{C}, \tau_r$)                             (Lowe 2004)
8:      $l \leftarrow$ length($\acute{\mathcal{C}}$)
9:      if $l > \hat{l}$ then
10:         $\hat{l} \leftarrow l$; $\widehat{\mathcal{C}} \leftarrow \mathcal{C}$
11: if $\hat{l} > \tau_n$ then
12:     $N_p \leftarrow \hat{l}$
13:     for $n \leftarrow 1$ to $N$ do
14:         $\widetilde{\mathcal{C}} \leftarrow$ RandomSample($\widehat{\mathcal{C}}, m$)
15:         $\mathbf{X}^{*(n)} \leftarrow$ EPnP($\mathbf{K}, \tilde{\mathbf{p}}_i, \widetilde{\mathbf{P}}_f$)                  (Lepetit et al. 2009)
16:         $\acute{\mathbf{p}} \leftarrow$ Project($\mathbf{K}, \mathbf{X}^{*(n)}, \widehat{\mathbf{P}}_f$)                        (1)(2)
17:         $\mathcal{H} \leftarrow \{\phi\}$
18:         for $s \leftarrow 1$ to $N_p$ do
19:             $\epsilon^{(s)} \leftarrow \|\hat{\mathbf{p}}_i^{(s)} - \acute{\mathbf{p}}^{(s)}\|_2$
20:             if $\epsilon^{(s)} < \tau_\epsilon$ then $\mathcal{H} \leftarrow \mathcal{H} \cup \{s\}$
21:         $N_i \leftarrow$ length($\mathcal{H}$)
22:         $\tilde{\pi}^{*(n)} \leftarrow$ KeypointLikelihood($N_p, N_i, \lambda_c$)                      (15)
23:     $\pi^* \leftarrow$ Normalize($\tilde{\pi}^*$)                                                (16)
24:     $\mathcal{S} \leftarrow$ Resampling($\mathcal{S}^*$)
    else
25:     $\mathcal{S} \leftarrow \{\phi\}$
```

The corresponding measurement equation is then

$$\mathbf{Z}_t = g(\mathbf{X}_t) + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}_{\mathsf{N_Z}\times 1}, \Sigma_n) \tag{7}$$

where $g : SE(3) \mapsto \mathfrak{R}^{\mathsf{N_Z}}$ is a nonlinear measurement function and $\mathbf{n}_t$ is a Gaussian noise with a covariance $\Sigma_n \in \mathfrak{R}^{\mathsf{N_Z}\times\mathsf{N_Z}}$.

3.2. *Particle filter*

In a generic particle filtering framework, the posterior density function $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is represented as a set of weighted particles by

$$\mathcal{S}_t = \{(\mathbf{X}_t^{(1)}, \pi_t^{(1)}), \dots, (\mathbf{X}_t^{(N)}, \pi_t^{(N)})\} \tag{8}$$

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

$$\mathbf{X}_t = \mathcal{E}[\mathcal{S}_t] = \sum_{n=1}^{N} \pi_t^{(n)} \mathbf{X}_t^{(n)} \tag{9}$$

or the mode

$$\mathbf{X}_t = \mathcal{M}[\mathcal{S}_t] = \mathbf{X}_t^{(j)}, \; j = \arg\max_j \pi_t^{(j)}. \tag{10}$$

When we apply the mean, however, there is a problem where the average of $\mathbf{X}_t^{(n)}$ is not valid in the $SE(3)$. More specifically, let $\mathbf{R}_t^{(n)} \in SO(3)$ be the rotation part of the $\mathbf{X}_t^{(n)}$. Then the arithmetic mean $\overline{\mathbf{R}}_t = \frac{1}{N}\sum_{n=1}^{N} \mathbf{R}_t^{(n)}$ 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

Algorithm 3: RANSAC($\mathbf{X},\mathbf{Z}$).

```
Data: $\mathbf{X}, \mathbf{Z} = \{\mathbf{p}, \mathbf{P}\}$
Result: $\widehat{\mathbf{Z}} = \{\hat{\mathbf{p}}, \hat{\mathbf{P}}\}$
Params: $i_{max}, m, \mathbf{K}, \tau_\epsilon, \rho$
1: $i \leftarrow 0$
2: $\hat{n} \leftarrow 0$
3: $\kappa \leftarrow \infty$
4: $\widehat{\mathcal{H}} \leftarrow \mathcal{H} \leftarrow \{\phi\}$
5: $S \leftarrow$ length(p)
6: while $i < \kappa$ and $i < i_{max}$ do
7:     $\tilde{\mathbf{Z}} \leftarrow$ RandomSample($\mathbf{Z}, m$)
8:     $\tilde{\mathbf{X}} \leftarrow$ IRLS($\mathbf{X}, \tilde{\mathbf{Z}}$)                 (22)(23)
9:     $\acute{\mathbf{p}} \leftarrow$ Project($\mathbf{K}, \tilde{\mathbf{X}}, \mathbf{P}$)            (1)(2)
10:    $\mathcal{H} \leftarrow \{\phi\}$
11:    for $s \leftarrow 1$ to $S$ do
12:        $\epsilon^{(s)} \leftarrow \|\mathbf{p}^{(s)} - \acute{\mathbf{p}}^{(s)}\|_2$
13:        if $\epsilon^{(s)} < \tau_\epsilon$ then $\mathcal{H} \leftarrow \mathcal{H} \cup \{s\}$
14:    $n \leftarrow$ length($\mathcal{H}$)
15:    if $n > \hat{n}$ then
16:        $\hat{n} \leftarrow n$; $\widehat{\mathcal{H}} \leftarrow \mathcal{H}$
17:        $\kappa \leftarrow \log(1-\rho)/\log(1-(\hat{n}/S)^m)$
18:    $i \leftarrow i + 1$
19: $\widehat{\mathbf{Z}} \leftarrow \mathbf{Z}^{(\widehat{\mathcal{H}})}$
```

by the orthogonal projection of $\overline{\mathbf{R}}_t$ onto $SO(3)$ as

$$\mathbf{R}_t = \begin{cases} \mathbf{V}\mathbf{U}^\mathsf{T} & \text{when } \det(\overline{\mathbf{R}}_t^\mathsf{T}) > 0 \\ \mathbf{V}\mathbf{H}\mathbf{U}^\mathsf{T} & \text{otherwise,} \end{cases} \tag{11}$$

where $\mathbf{U}$ and $\mathbf{V}$ are estimated via the singular value decomposition of $\overline{\mathbf{R}}_t^\mathsf{T}$ (i.e. $\overline{\mathbf{R}}_t^\mathsf{T} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$) and $\mathbf{H} = \text{diag}[1, 1, -1]$. Therefore, the valid arithmetic mean of the particles can be determined as

$$\mathbf{X}_t = \mathcal{E}_{SE(3)}[\mathcal{S}_t] = \begin{pmatrix} \mathbf{R}_t & \mathbf{T}_t \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{12}$$

where $\mathbf{T}_t = \frac{1}{N}\sum_{n=1}^{N} \mathbf{T}_t^{(n)}$ and $\mathbf{T}_t^{(n)} \in \Re^3$ is the translation part of $\mathbf{X}_t^{(n)}$. A more detailed discussion of mean and covariance on $SE(3)$ is presented in Wang and Chirikjian (2008).

3.3. AR state dynamics

A dynamic model for state evolution is an essential part that has a significant impact on tracking performance. However, many particle filter-based trackers have been based on a random walk model because of its simplicity (Klein and Murray 2006; Teulière et al. 2010). The first-order AR state dynamics is a good alternative since it is flexible, yet simple to implement. In the state equation (5), the term $A(\mathbf{X}, t)$ determines the state dynamics. A trivial case, $A(\mathbf{X}, t) = 0$, is a random walk model. Kwon and Park (2010) modeled this via the first-order AR process on the $Aff(2)$ as

$$\mathbf{X}_t = \mathbf{X}_{t-1} \cdot \exp(\mathbf{A}_{t-1} + \mathbf{dW}_t\sqrt{\Delta t}), \tag{13}$$

$$\mathbf{A}_{t-1} = \lambda_a \log(\mathbf{X}_{t-2}^{-1}\mathbf{X}_{t-1}) \tag{14}$$

where λ_a is the AR process parameter. Since the $SE(3)$ is a compact connected Lie group, the AR process model also holds on the $SE(3)$ group (Xavier and Manton 2006).

3.4. Particle initialization using keypoint correspondences

Most of the particle filter-based trackers assume 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 keypoints 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 $\mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \ldots, (\mathbf{p}_F, \mathbf{P}_F)\}$ which are composed of F sets of SURF keypoints (Bay et al. 2008) coordinates in two and three dimensions. These keyframes are saved manually by user input. Since the appearance of SURF keypoints are not invariant enough to viewpoint variations, it is necessary to capture multiple keyframes covering different views of an object. The keypoint coordinates in 2D ($\mathbf{p}_f$) are easily determined from SURF keypoints extraction, while 3D coordinates ($\mathbf{P}_f$) are 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 starting from a known pose. At runtime, an input image $\mathcal{I}$ coming from a monocular camera is matched with the saved keyframes by extracting keypoints $\mathbf{p}_i$ from $\mathcal{I}$ and comparing them with $\mathcal{F}$. To find keypoint correspondences $\mathcal{C} = \{\acute{\mathbf{p}}_i, \acute{\mathbf{P}}_f\}$ 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 $\acute{\mathcal{C}}$ 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 $\widetilde{\mathcal{C}}$ from the given putative correspondences $\widehat{\mathcal{C}}$ having a maximum number of correspondences over keyframes $\mathcal{F}$ and estimate a possible set of poses $\mathbf{X}^{*(n)}$ from $\widetilde{\mathcal{C}}$. Since we maintain 3D coordinates $\mathbf{P}_f$ 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 $\mathbf{X}^{*(n)}$ is initialized from randomly selected minimum correspondences, all 3D points $\widehat{\mathbf{P}}_f$ from the putative correspondences $\widehat{\mathcal{C}}$ are projected into 2D points $\acute{\mathbf{p}}$ and compared with 2D keypoints $\hat{\mathbf{p}}_i$. We then count the number of inlier correspondences N_i whose Euclidean distances between $\acute{\mathbf{p}}$ and $\hat{\mathbf{p}}_i$ are within the threshold $\tau_\epsilon = 20.0$. 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_p, which is the number of entries in $\widehat{\mathcal{C}}$, and the number of inlier correspondences N_i as

$$\tilde{\pi}^{*(n)} \propto p(\mathbf{Z}_t|\mathbf{X}_t^{*(n)}) \propto \exp^{(-\lambda_c \frac{N_p - N_i}{N_p})} \tag{15}$$

where λ_c is a parameter. Then the weights $\tilde{\pi}^{*(n)}$ are normalized by

$$\pi^{*(n)} = \frac{\tilde{\pi}^{*(n)}}{\sum_{i=1}^{N} \tilde{\pi}^{*(i)}}. \tag{16}$$

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 $(\cdot)$ in the comments area. Given an input image $\mathcal{I}$ and keyframes $\mathcal{F}$, the algorithm returns the set of initialized particle states $\mathcal{S}$. Among the parameters, τ_n is the minimum required number of keypoint correspondences for the initialization, m is the minimum number of correspondences to perform EPnP, and $\mathbf{K}$ is the intrinsic parameters of the camera in (3). We found that 7 and 9 are well suited to m and τ_n, respectively.

3.5. *Edge-based measurement likelihood*

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 $\mathbf{X}_t^{*(n)}$. 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 they are 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). In particular, 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

$$I(\mathbf{p}_i, \mathbf{q}_i) = \begin{cases} 1 & \text{if } |\theta_m(\mathbf{p}_i) - \theta_e(\mathbf{q}_i)| \le \tau_\theta \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where θ_m and θ_e return the orientation of the model edge to which the sample point $\mathbf{p}_i$ belongs and of the image edge point $\mathbf{q}_i$, 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 $\mathbf{p}_i$ and $\mathbf{q}_i$ is calculated. By stacking all of the residual of visible sample points, the residual vector $\mathbf{r} \in \Re^{\mathsf{N_z}}$ is obtained as follows

$$\mathbf{r} = (r_1, r_2, \ldots, r_{N_z})^{\mathsf{T}} \tag{18}$$

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

$$\{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_{N_z}\}. \tag{19}$$

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 segment (Armstrong and Zisserman 1995). One drawback of both of these approaches is the possibility of inconsistent refinement because edge or line segments are corrected individually. 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 $\mathbf{P}$ and their corresponding 2D closest edge points $\mathbf{p}$. Our approach consistently discard outliers by estimating the best 3D pose containing large number of inliers $\widehat{\mathcal{H}}$. The RANSAC algorithm is presented in Algorithm 3 which finds the refined edge correspondences $\widehat{\mathbf{Z}} = \{\hat{\mathbf{p}}, \widehat{\mathbf{P}}\}$ given the current pose hypothesis $\mathbf{X}$ and the original edge correspondences $\mathbf{Z} = \{\mathbf{p}, \mathbf{P}\}$.

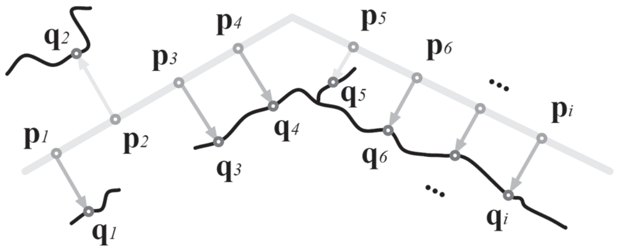


Fig. 3. Residual determination for calculating the likelihood. Residual errors between projected model (bright lines) and extracted image edges (dark tortuous lines) are calculated. Sample points $\mathbf{p}_i$ are generated along the model per a fixed distance and are matched to image edge points $\mathbf{q}_i$ by performing 1D search along the direction orthogonal to the model edge.

Among the parameters, i_{max} is the maximum number of iterations, again m is the minimum number of correspondences required for EPnP, and ρ is the probability in which at least one set of randomly sampled m correspondences are from inliers. The ρ typically set as 0.99 is used to estimate the required number of iterations κ which is adaptively adjusted in the iteration.

Figure 3 illustrates the residual calculation in which each residual arrow represents $r_i\mathbf{n}_i$. Note that the second edge correspondence of $\mathbf{p}_2$ and $\mathbf{q}_2$ is wrong since $\mathbf{q}_2$ comes from background clutter, but it is soon discarded via the RANSAC refinement. The fifth edge correspondence is also erroneous because $\mathbf{q}_5$ does not belong 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 $N_{\hat{z}}$ which are survived after the RANSAC and the number of visible sample points N_z which pass a self-occlusion test as

$$p(\mathbf{Z}_t|\mathbf{X}_t) \propto \exp^{(-\lambda_v \frac{(N_z - N_{\hat{z}})}{N_z})} \tag{20}$$

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

$$p(\mathbf{Z}_t|\mathbf{X}_t) \propto \exp^{(-\lambda_r \bar{\mathbf{r}})}$$

where λ_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

$$p(\mathbf{Z}_t|\mathbf{X}_t) \propto \exp^{(-\lambda_v \frac{(N_z - N_{\hat{z}})}{N_z})} \exp^{(-\lambda_r \bar{\mathbf{r}})}. \tag{21}$$

3.6. Optimization using iteratively reweighted least squares

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 Teulière 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 $\acute{\mathbf{X}}_t^{*(n)}$ is calculated as follows:

$$\acute{\mathbf{X}}_t^{*(n)} = \mathbf{X}_t^{*(n)} \cdot \exp\left(\sum_{i=1}^{6} \hat{\mu}_i \mathbf{E}_i\right) \tag{22}$$

$$\hat{\boldsymbol{\mu}} = (\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J})^{-1}\,\mathbf{J}^\mathsf{T}\mathbf{W}\hat{\mathbf{r}} \tag{23}$$

where $\hat{\boldsymbol{\mu}} \in \Re^6$ is the motion velocity that minimizes the residual vector $\hat{\mathbf{r}} \in \Re^{N_{\hat{z}}}$, $\mathbf{J} \in \Re^{N_{\hat{z}} \times 6}$ is a Jacobian matrix of $\mathbf{n}_i^\mathsf{T}\mathbf{p}_i$ with respect to $\boldsymbol{\mu}$ obtained by computing partial derivatives at the current pose, and $\mathbf{W} \in \Re^{N_{\hat{z}} \times N_{\hat{z}}}$ is a weighted diagonal matrix. The derivation of the Jacobian matrix $\mathbf{J}$ and more detailed information about IRLS are explained in Appendix A.

Note that the measurement likelihood in (21) is calculated before the IRLS optimization. To assign weights to particles, we have to evaluate the likelihood again with the optimized state $\acute{\mathbf{X}}_t^{*(n)}$. 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 is highly dependent on the previous state. Thus, particles having higher likelihood 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 $p(\mathbf{Z}_t|\acute{\mathbf{X}}_t^{*(n)})$ as

$$\tilde{\pi}_t^{*(n)} \propto p(\mathbf{Z}_t|\acute{\mathbf{X}}_t^{*(n)}) \approx p(\mathbf{Z}_t|\mathbf{X}_t^{*(n)}) \tag{24}$$

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

3.7. Re-initialization based on $\widehat{N_{eff}}$

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 $\widehat{N_{eff}}$ is defined in Doucet et al. (2000):

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N}(\tilde{\pi}^{(i)})^2}. \tag{25}$$

Often it has been used as a measure to execute the resampling procedure. However, in our tracker we resample particles every frame, and hence we use $\widehat{N_{eff}}$ 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 $\langle\cdot\rangle$ and $(\cdot)$ in the comments area, respectively. It requires a sequence of images $\mathcal{I}$ and the keyframes $\mathcal{F}$ as an input and estimates the posterior density as a set of weighted particles $\mathcal{S}$ in each time t.

4. Experimental 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 background is possible. During rendering, we were 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, 3.25 GB 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 present a series of comparative experiments:

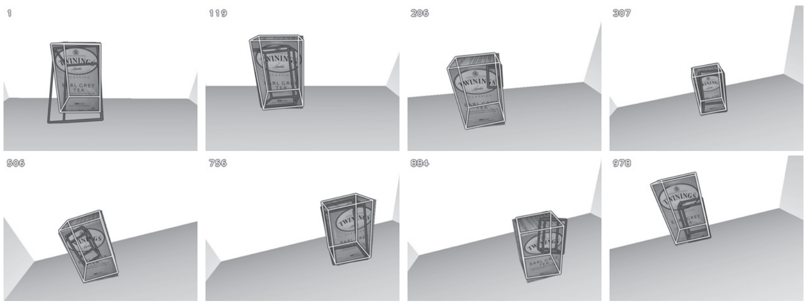
1. effectiveness of considering edge orientations;
2. effectiveness of considering multiple pose hypotheses;
3. effectiveness of performing RANSAC;
4. effectiveness of employing AR state dynamics.

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). Second, 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 for 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.

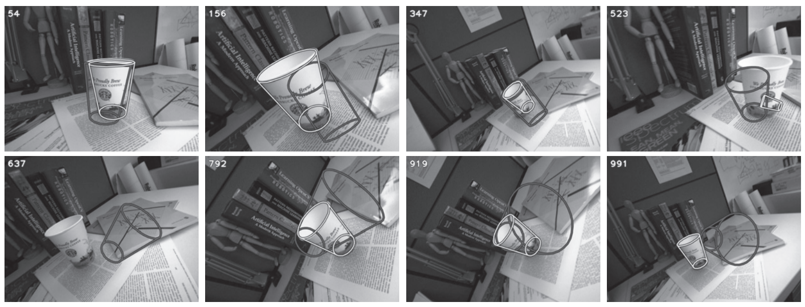
4.1. *Effectiveness of considering edge orientations*

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 enhances 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, 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 a limited number of particles, it shows good tracking performance in the simple background synthetic sequence (Figure 4(a)) and acceptable tracking performance in the cluttered background real sequence (Figure 4(b)). 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, 6D 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 fives rows are the RMS errors of the proposed approach in various parameter settings for which the



(a) The synthetic image sequence of the 'Teabox' object: simple background case



(b) The real image sequence of the 'Cup' object

Fig. 4. Tracking results showing the effectiveness of considering edge orientation. Results of our approach with only one particle (i.e. $N = 1$) are depicted in bright wireframes. As a baseline, tracking results of Choi and Christensen (2010) are shown in dark wireframes. In spite of the limited number of particles, the proposed approach shows good tracking performance in the simple background sequence and acceptable tracking performance in the cluttered background real sequence, while the baseline is frequently stuck in local minima, and hence it drifts over the textured regions of the object or the background clutter. This difference 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.

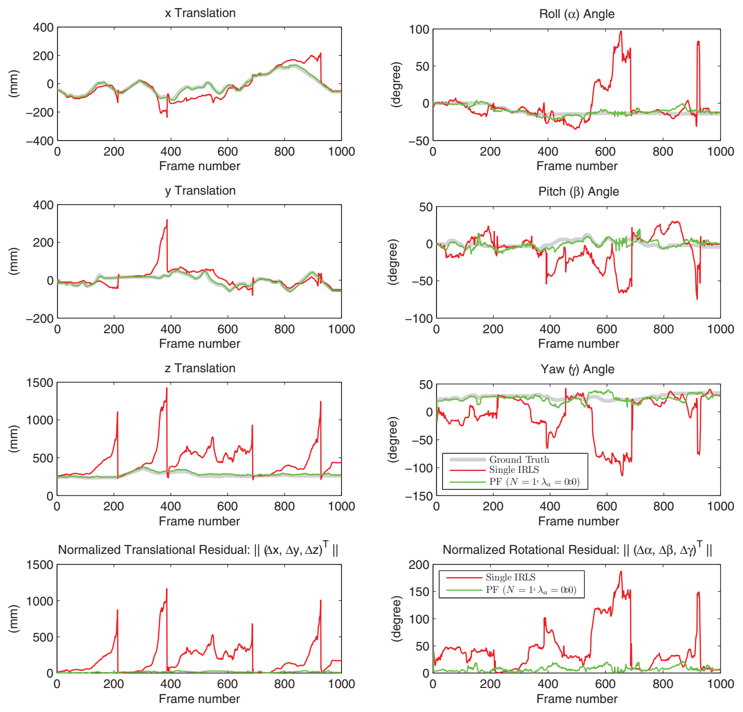


Fig. 5. The 6D pose and normalized residual plots of the results in Figure 4(a). The proposed approach, particle filer ($N = 1, \lambda_a = 0.0$), shows reasonable accuracy, while the baseline from Choi and Christensen (2010) has significant errors. The baseline recovers from local minima via the re-initialization described in Choi and Christensen (2010), but it soon drifts again.

number of particles N, the AR parameter λ_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. These experimental results clearly support the argument that considering edge orientations enhances the quality of the matching between edge points.

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. However, 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 were 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 we are comparing orientations of edge points, our approach discards some false edge correspondences. However, it still tends 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 ($N = 100$, $\lambda_a = 0.0$, 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 for 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 coincide with the axis of symmetry of the object in the object coordinate system via

$$^{\mathcal{C}}\mathbf{u}_y = {}^{\mathcal{C}}\mathbf{T}_{\mathcal{O}} \cdot {}^{\mathcal{O}}\mathbf{u}_y \tag{26}$$

$$^{\mathcal{C}}\hat{\mathbf{u}}_y = {}^{\mathcal{C}}\widehat{\mathbf{T}}_{\mathcal{O}} \cdot {}^{\mathcal{O}}\mathbf{u}_y \tag{27}$$

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

$$\theta = \arccos({}^{\mathcal{C}}\mathbf{u}_y^\top \, {}^{\mathcal{C}}\hat{\mathbf{u}}_y). \tag{28}$$

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 approach 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 approach without RANSAC is frequently misled by the complex background clutter.

4.4. Effectiveness of employing AR state dynamics

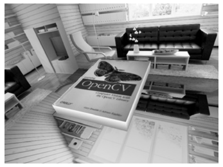
To verify the effect of the AR state dynamics, we execute the proposed approach with and without the AR state dynamics. To disable the dynamics, we set the parameter λ_a in the AR state dynamics equation (14) as 0 which is equivalent to a random walk model. For fair comparison, we use the same parameters 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. (See also Figure 9.)

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

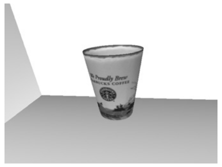
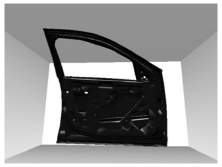
In our previous system (Choi and Christensen 2010), we used a simple heuristic in which the difference in position of the object between frames and the number of valid sample points are monitored to trigger re-initialization. While that heuristic works well when the pose hypothesis drifts fast, it might not always be the case when the hypothesis stuck in local minima. We propose another way for re-initialization by taking advantage of multiple pose hypotheses. As in Algorithm 1, our system re-initializes when the number of effective particles $\widehat{N_{eff}}$ is below a threshold.

Table 1. RMS errors and computation time in synthetic image sequences (baseline versus PF).

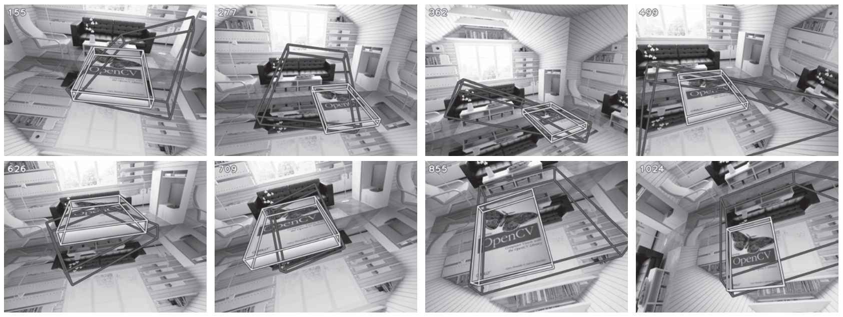
Objects		Mode	RMS Errors*						Time§
			X	Y	Z	Roll	Pitch	Yaw	
Teabox		Baseline (single IRLS)†	51.131	44.509	263.258	26.280	24.330	48.323	20.62
		PF ($N = 1, \lambda_a = 0.0$)	5.234	3.323	16.034	3.555	4.626	6.727	18.11
		PF ($N = 100, \lambda_a = 0.0$)	4.616	2.498	13.623	2.558	3.940	3.867	450.57
		PF ($N = 100, \lambda_a = 0.5$)	4.272	2.255	12.430	2.375	3.882	3.628	449.39
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	3.503	1.930	8.917	1.936	3.227	2.597	459.49
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**3.234**	**1.829**	**8.062**	**1.843**	**3.057**	**2.530**	464.69
		Baseline (single IRLS)†	63.074	27.075	83.377	25.028	38.839	62.695	23.05
		PF ($N = 1, \lambda_a = 0.0$)	20.209	21.130	55.466	35.613	28.153	61.006	22.61
		PF ($N = 100, \lambda_a = 0.0$)	4.521	8.357	13.974	4.431	7.717	5.593	471.78
		PF ($N = 100, \lambda_a = 0.5$)	3.028	4.846	11.739	3.313	5.965	3.637	504.13
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	3.125	**4.398**	10.207	2.712	4.976	**3.324**	497.05
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**2.795**	4.953	**9.503**	**2.410**	**4.854**	3.400	519.03
Book		Baseline (single IRLS)†	16.980	37.375	131.320	8.372	22.691	20.273	21.83
		PF ($N = 1, \lambda_a = 0.0$)	76.376	95.839	92.012	3.571	14.141	27.524	20.90
		PF ($N = 100, \lambda_a = 0.0$)	4.633	3.570	22.881	1.107	1.718	3.661	802.08
		PF ($N = 100, \lambda_a = 0.5$)	4.502	3.402	22.240	0.954	1.508	3.791	799.30
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	4.298	3.420	21.107	1.037	1.618	3.563	815.76
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**4.140**	**3.171**	**20.084**	**0.876**	**1.408**	**3.545**	811.00
		Baseline (single IRLS)†	242.794	71.234	168.683	35.455	42.016	18.776	25.61
		PF ($N = 1, \lambda_a = 0.0$)	53.528	61.926	177.906	13.346	15.161	17.974	25.03
		PF ($N = 100, \lambda_a = 0.0$)	11.663	14.519	28.967	3.276	2.771	4.189	771.41
		PF ($N = 100, \lambda_a = 0.5$)	10.343	15.288	27.654	4.208	2.881	3.904	816.39
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	4.554	4.751	9.156	1.772	**1.967**	3.746	819.32
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**3.195**	**3.843**	**8.239**	**1.589**	1.979	**3.697**	819.85

(Continued)

Table 1. (Continued)

Objects		Mode	RMS Errors*						Time§
			X	Y	Z	Roll	Pitch	Yaw	
Cup‡		Baseline (single IRLS)†	36.317	31.552	122.737	–	78.843‡	–	33.65
		PF ($N = 1, \lambda_a = 0.0$)	189.488	19.947	58.342	–	25.917	–	32.63
		PF ($N = 100, \lambda_a = 0.0$)	3.842	3.098	14.837	–	2.116	–	1915.49
		PF ($N = 100, \lambda_a = 0.5$)	3.574	3.008	13.820	–	1.960	–	1910.34
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	3.112	2.544	12.076	–	1.777	–	1928.42
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**2.933**	**2.422**	**11.063**	–	**1.515**	–	1925.59
		Baseline (single IRLS)†	126.662	57.309	127.284	–	33.180‡	–	37.67
		PF ($N = 1, \lambda_a = 0.0$)	65.645	48.978	100.036	–	71.873	–	36.53
		PF ($N = 100, \lambda_a = 0.0$)	8.856	11.778	22.064	–	7.041	–	1913.69
		PF ($N = 100, \lambda_a = 0.5$)	8.312	**11.417**	20.045	–	**6.186**	–	1911.11
		PF ($N = 100, \lambda_a = 0.0$, RANSAC)	7.111	18.625	**15.577**	–	6.197	–	1936.90
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**6.997**	18.516	17.134	–	6.333	–	1949.41
Car door		Baseline (single IRLS)†	6.446	8.435	9.615	0.398	0.771	1.529	33.50
		$N = 1, \lambda_a = 0.0$	5.206	6.821	11.921	0.333	0.974	1.523	40.93
		$N = 100, \lambda_a = 0.0$	4.645	6.437	11.129	0.292	0.841	1.209	2699.23
		$N = 100, \lambda_a = 0.5$	4.049	6.003	11.113	**0.254**	0.797	1.114	2702.93
		$N = 100, \lambda_a = 0.0$, RANSAC	4.388	6.166	**9.189**	0.318	0.718	0.967	2691.41
		$N = 100, \lambda_a = 0.5$, RANSAC	**3.831**	**5.778**	9.281	0.281	**0.658**	**0.944**	2676.74
		Baseline (single IRLS)†	93.991	70.093	323.827	16.753	7.972	60.336	37.16
		$N = 1, \lambda_a = 0.0$	12.550	29.420	80.789	1.636	5.055	14.382	45.46
		$N = 100, \lambda_a = 0.0$	9.872	14.614	28.212	0.913	3.257	5.960	2796.06
		$N = 100, \lambda_a = 0.5$	7.488	11.584	24.185	0.731	1.865	4.096	2793.18
		$N = 100, \lambda_a = 0.0$, RANSAC	6.167	11.721	17.653	0.822	2.384	4.417	2733.86
		$N = 100, \lambda_a = 0.5$, RANSAC	**4.787**	**9.137**	**15.942**	**0.617**	**1.400**	**2.758**	2737.89

*The error units of translation and rotation are millimeters and degrees, respectively. Better results are indicated in bold type.
§The unit of computation time is milliseconds per frame.
†The results of the previous approach (Choi and Christensen 2010) are compared as a baseline.
‡Since the 'Cup' object is symmetric and thus exact orientation cannot be estimated, we calculate the angle of the axis of symmetry.


(a) The synthetic image sequence of the 'Book' object: complex background case

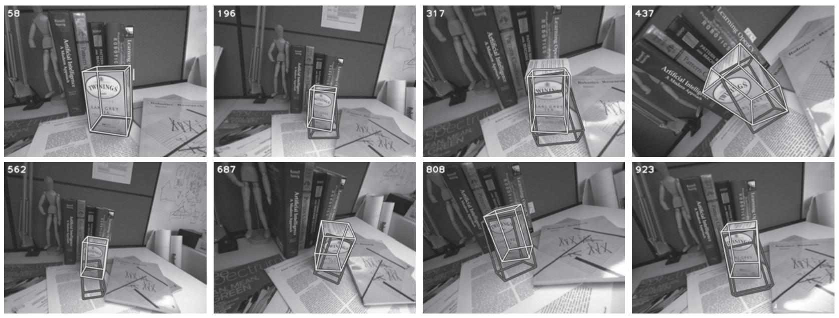
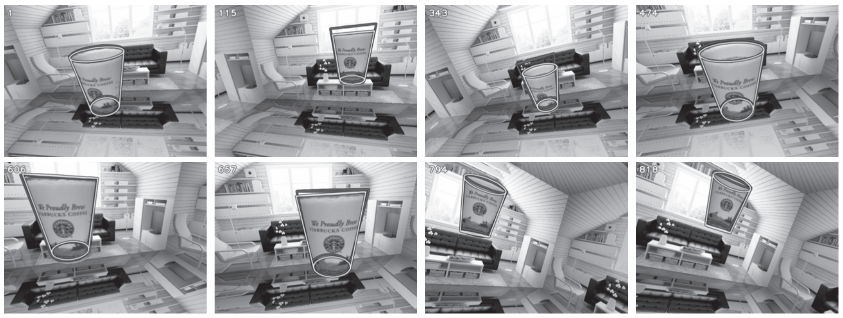

(b) The real image sequence of the 'Teabox' object

Fig. 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 for 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.


(a) The synthetic image sequence of the 'Cup' object: complex background case

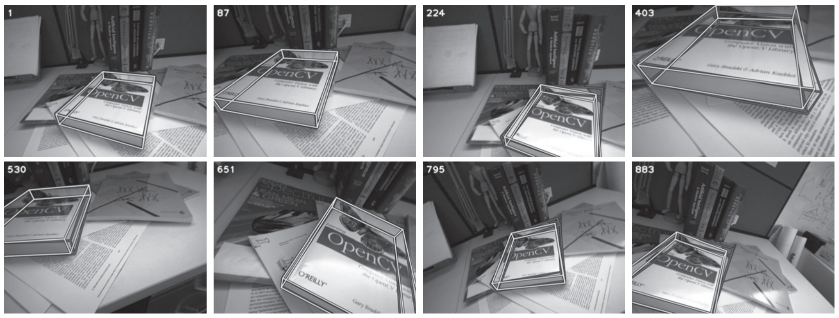
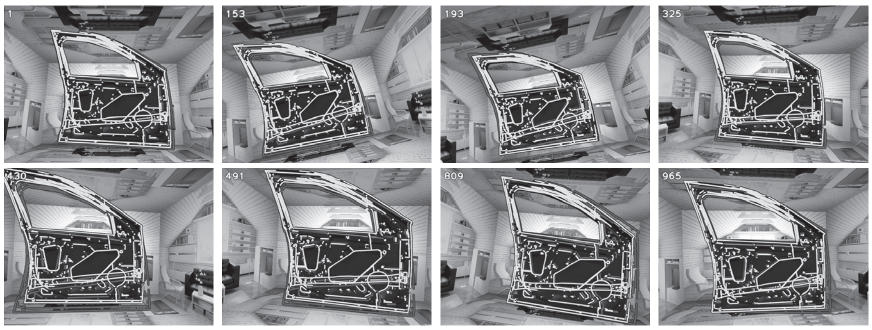
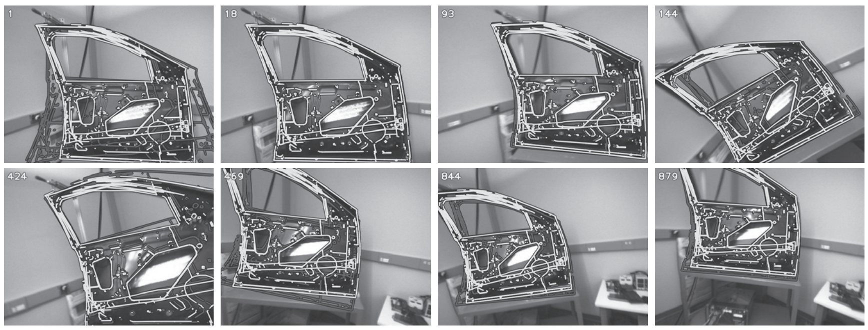

(b) The real image sequence of the 'Book' object

Fig. 7. Tracking results showing the effectiveness of performing RANSAC. The bright wireframes represent estimated poses of our approach with RANSAC refinement; the dark wireframes are the results of the same approach without RANSAC. Both use the same number of particles and the AR process parameter ($N = 100,\ \lambda_a = 0.0$). While both show good tracking results, 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.



(a) The synthetic image sequence of the 'Car door' object: complex background case



(b) The real image sequence of the 'Car door' object

Fig. 8. Tracking results showing the effectiveness of employing AR state dynamics. To investigate the advantage of incorporating the AR state dynamics in our framework, we run our approach with ($\lambda_a = 0.5$) and without ($\lambda_a = 0.0$) the dynamics. Results of our approach with and without AR state dynamics are depicted in bright and dark wireframes, respectively. Both maintain 100 particles and perform RANSAC. The only difference is the AR parameter λ_a. As the linear state dynamics propagates particles with respect to their velocities, the particle filtering evolves effectively, and hence the mean of particles represented in the bright wireframes follows the global optimum. However, the one without AR state dynamics which is equivalent to a random walk model is occasionally stuck in local minima.

To verify this method, we run the proposed tracker on two real image sequences. The tracking results are shown in Figure 10 and Extension 5, and the number of effective particles for the first image sequence (Figure 10(a)) is plotted over the frame numbers in Figure 11. When the tracked object goes out of the field of view or images are blurred because of camera shaking, the $\widehat{N_{eff}}$ decreases significantly below the threshold value N_{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. 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 τ_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 the 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 a state-of-the-art tracker. To compare our approach with an existing solution, we chose the BLORT tracker (Mörwald et al. 2010) because it combines state-of-the-art methods for object recognition and tracking and is publicly available.[1] By combining SIFT-based object recognition and edge-based particle filtering, BLORT provides a reliable vision 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

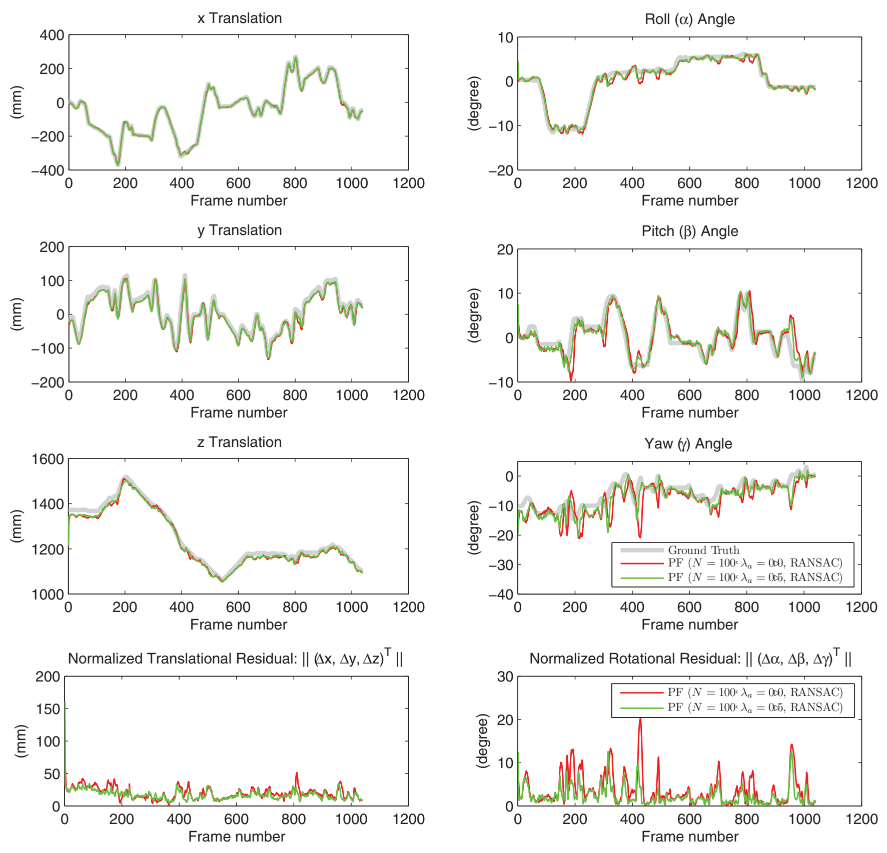
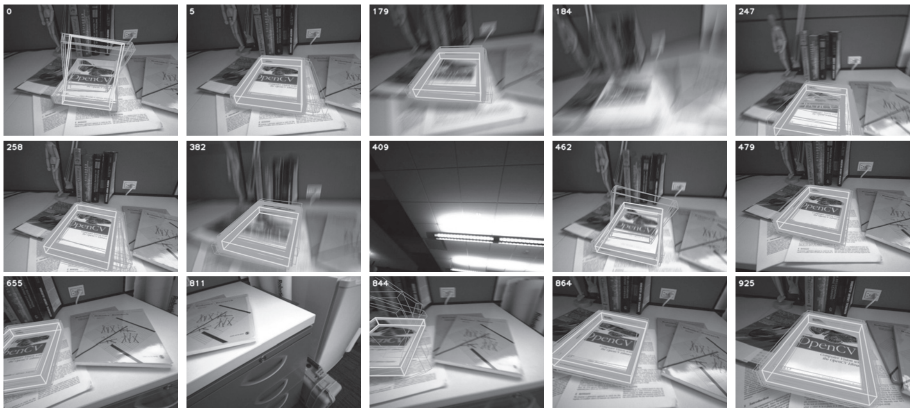


Fig. 9. The 6D pose and normalized residual plots of the results in Figure 8(a). With the AR state dynamics, our approach shows better performance in terms of accuracy than the one without the dynamics. In particular, when the camera undergoes abrupt changes in position or orientation, the approach with the linear state dynamics exhibits agile responses, while the approach without the state dynamics results in bigger errors.

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 $N = 4$) 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 the 6D pose and residual plots in Figure 13, where the tracked pose results of 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 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 the 'Teabox' object. The reason why BLORT gives better results for the 'Teabox' object is possibly due to the abundant 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


(a) The ‘Book’ object

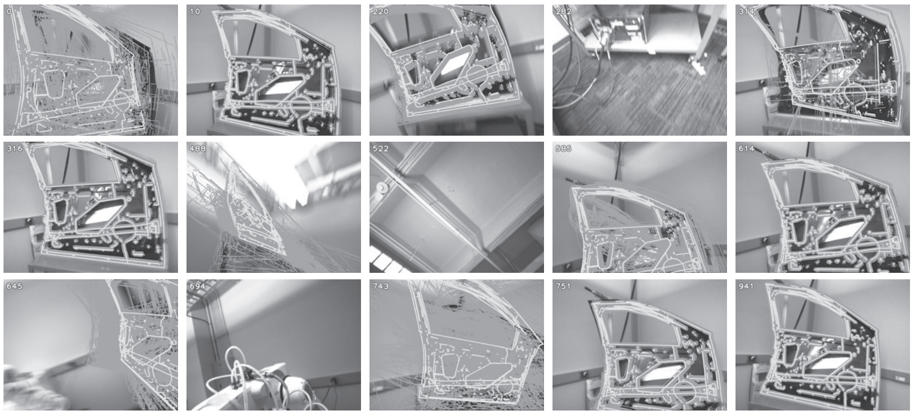

(b) The ‘Car door’ object

Fig. 10. 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. However, the tracker quickly converges to the global optimum as the bad pose hypotheses are removed in the importance resampling process.

with ‘Book’ and ‘Cup’ objects. Thus, BLORT reports less RMS errors even in the ‘Teabox’ complex background sequence, while it shows large errors in ‘Cup’ simple background sequence. In contrast, 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, the 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 the sequence. Even before the tracking failures, pose results from BLORT are not well aligned to the global optimum.

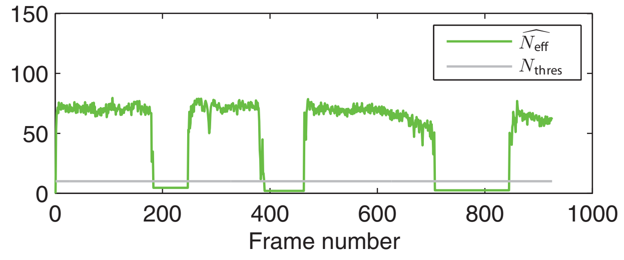
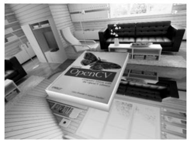
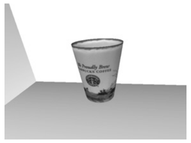


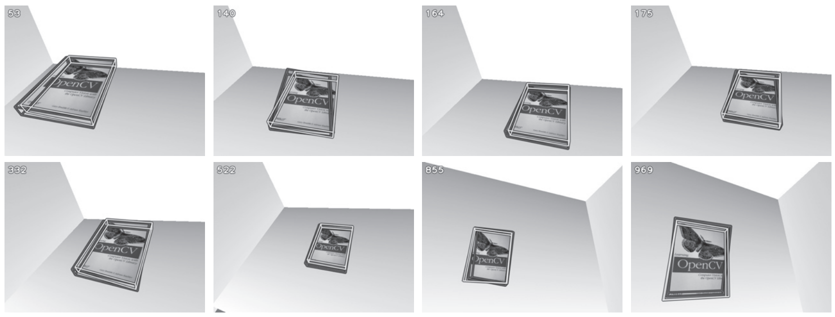
Fig. 11. The $\widehat{N_{eff}}$ plot of the results in Figure 10(a). When the number of effective particles is under the threshold N_{thres}, our system re-initializes by itself.

BLORT reported good performance in the 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

Table 2. RMS errors and computation time in synthetic image sequences (PF versus BLORT).

Objects		Mode	RMS Errors*						Time§
			X	Y	Z	Roll	Pitch	Yaw	
Teabox		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**3.234**	**1.829**	8.062	1.843	**3.057**	2.530	464.69
		BLORT ($N = 100, R = 2$)	6.344	3.570	6.825	1.915	8.753	4.146	99.72
		BLORT ($N = 200, R = 2$)	4.747	3.164	5.878	1.528	6.879	3.228	174.36
		BLORT ($N = 100, R = 4$)	3.897	3.169	5.605	1.511	6.127	3.353	173.96
		BLORT ($N = 200, R = 4$)	3.684	2.289	**4.821**	**1.276**	5.572	**1.946**	325.03
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**2.795**	4.953	9.503	**2.410**	**4.854**	**3.400**	519.03
		BLORT ($N = 100, R = 2$)	12.639	5.808	18.893	3.559	16.471	11.607	104.37
		BLORT ($N = 200, R = 2$)	42.314	25.175	110.392	4.246	28.717	10.883	101.15
		BLORT ($N = 100, R = 4$)	5.101	**2.630**	**7.608**	2.550	7.379	3.836	190.70
		BLORT ($N = 200, R = 4$)	6.411	3.180	16.986	3.745	6.537	3.744	331.18
Book		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	4.140	**3.171**	20.084	**0.876**	**1.408**	**3.545**	811.00
		BLORT ($N = 100, R = 2$)	6.651	6.194	29.169	2.439	11.056	14.747	85.71
		BLORT ($N = 200, R = 2$)	4.736	5.989	22.240	1.904	7.898	11.589	147.94
		BLORT ($N = 100, R = 4$)	3.962	6.119	21.459	1.399	5.777	9.643	150.55
		BLORT ($N = 200, R = 4$)	**3.643**	5.893	**18.073**	1.132	4.346	8.780	275.86
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**3.195**	**3.843**	**8.239**	**1.589**	**1.979**	**3.697**	819.85
		BLORT ($N = 100, R = 2$)	73.011	79.695	262.610	27.580	47.031	38.527	37.32
		BLORT ($N = 200, R = 2$)	73.198	90.931	216.233	44.918	17.872	23.865	51.41
		BLORT ($N = 100, R = 4$)	73.824	86.040	232.869	38.145	31.934	23.138	48.50
		BLORT ($N = 200, R = 4$)	59.819	76.913	229.491	56.344	37.451	29.128	75.40
Cup†		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**2.933**	**2.422**	**11.063**	–	**1.515**†	–	1925.59
		BLORT ($N = 100, R = 2$)	41.878	24.509	187.267	–	57.789	–	36.91
		BLORT ($N = 200, R = 2$)	30.419	26.715	93.366	–	83.586	–	63.46
		BLORT ($N = 100, R = 4$)	56.216	45.414	610.823	–	55.184	–	52.97
		BLORT ($N = 200, R = 4$)	3.979	5.006	27.852	–	5.091	–	222.57
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	**6.997**	**18.516**	**17.134**	–	**6.333**†	–	1949.41
		BLORT ($N = 100, R = 2$)	86.748	75.334	226.736	–	69.987	–	37.03
		BLORT ($N = 200, R = 2$)	79.777	35.124	114.818	–	69.422	–	56.70
		BLORT ($N = 100, R = 4$)	36.628	73.228	179.439	–	85.200	–	77.02
		BLORT ($N = 200, R = 4$)	64.853	52.676	132.043	–	105.518	–	98.50

*The error units of translation and rotation are millimeters and degrees, 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.


(a) The synthetic image sequence of the 'Book'object: simple background case

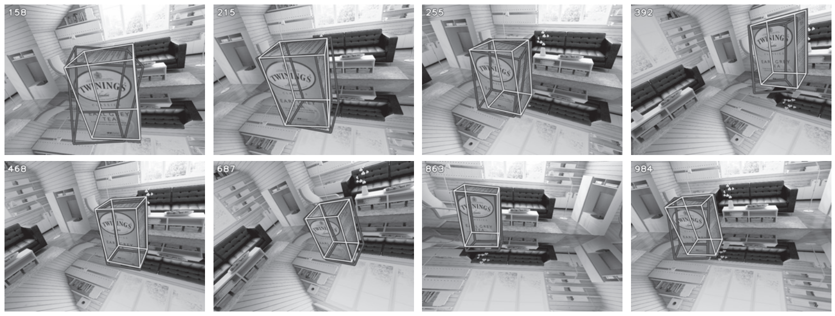

(b) The synthetic image sequence of the 'Teabox' object: complex background case

Fig. 12. A comparison between the tracking results of our particle filter (bright wireframes) and BLORT (dark wireframes) on synthetic image sequences. To compare the performance of our approach with an existing solution, BLORT (Mörwald et al. 2010) was executed in the same image sequences. In both image sequences, our particle filter employed 100 particles, the AR state dynamics, and RANSAC refinement (i.e. $N = 100,\ \lambda_a = 0.5$, RANSAC). For BLORT, we used default parameters provided in the software package except the number of particles N and recursions R. Tracking results of BLORT ($N = 200,\ R = 4$) and ($N = 100,\ R = 4$) are depicted in the dark wireframes in (a) and (b), respectively.

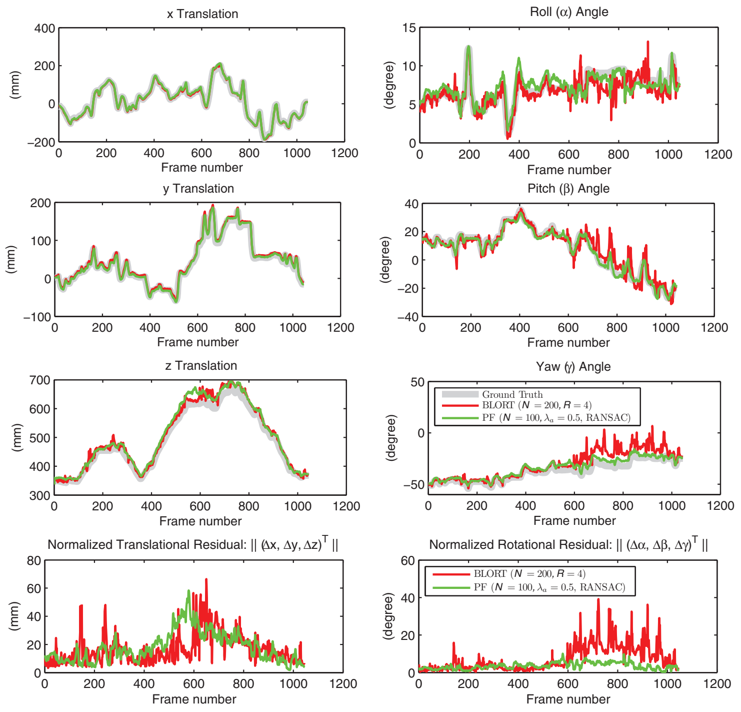
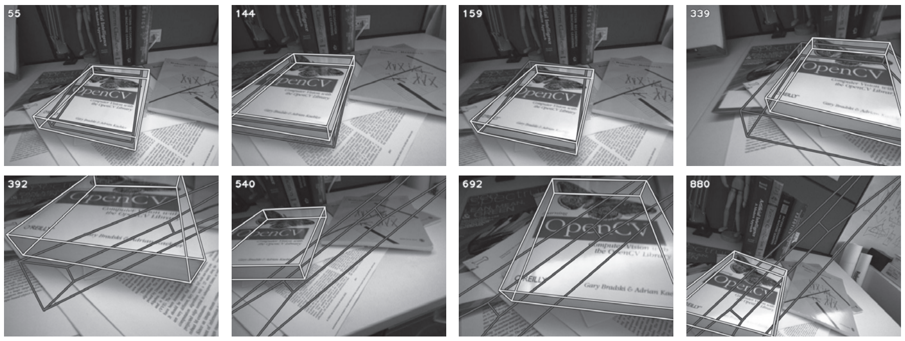
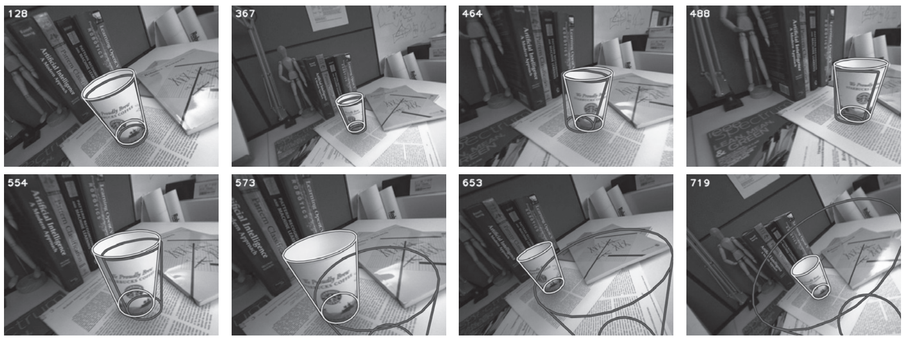


Fig. 13. The 6D pose and normalized residual plots of the results in Figure 12(a). With the sufficient number of particles and recursions ($N = 200,\ R = 4$), BLORT reports similar performance to our particle filter. BLORT even shows slightly better results in z translation possibly due to considering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



(a)The real image sequence of the 'Book' object



(b) The real image sequence of the 'Cup' object

Fig. 14. A comparison between the tracking results of our particle filter (bright wireframes) and BLORT (dark wireframes) on real image sequences. In both image sequences, our particle filter ($N = 100$, $\lambda_a = 0.5$, RANSAC) is compared with BLORT ($N = 200$, $R = 4$). While our particle filter tracks the target objects well, BLORT loses the targets in the middle of the sequences. Even before the tracking failures, pose results from BLORT are not well aligned to the global optimum.

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 processing time of our approach.

5. 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 were 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).

Notes

1. BLORT - The Blocks World Robotic Vision Toolbox: http://users.acin.tuwien.ac.at/mzillich/?site=4.
2. The default number of particles and recursions in BLORT are $N = 200$ and $R = 2$, respectively.

Funding

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.

References

Armstrong M and Zisserman A (1995) Robust object tracking. In *Asian Conference on Computer Vision*, vol. 1, pp. 58–61.

Bay H, Ess A, Tuytelaars T and Gool LV (2008) Speeded-up robust features (SURF). *Computer Vision and Image Understanding* 110: 346–359.

Beis J and Lowe D (1997) Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1000–1006.

Bray M, Koller-Meier E and Gool LV (2004) Smart particle filtering for 3D hand tracking. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 675–680.

Canny J (1986) A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8: 679–698.

Chiuso A and Soatto S (2000) Monte Carlo filtering on Lie groups. In *IEEE Conference on Decision and Control*, vol. 1, 304–309.

Choi C and Christensen HI (2010) Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pp. 4048–4055.

Comport AI, Marchand E and Chaumette F (2004) Robust model-based tracking for robot vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1.

Doucet A, Godsill S and Andrieu C (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* 10: 197–208.

Drummond T and Cipolla R (2002) Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24: 932–946.

Fischler MA and Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24: 381–395.

Harris C (1992) *Tracking with Rigid Objects*. Cambridge, MA: MIT Press.

Huber PJ (1981) *Robust Statistics*. New York: Wiley-Interscience.

Isard M and Blake A (1998) Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision* 29: 5–28.

Kemp C and Drummond T (2005) Dynamic measurement clustering to aid real time tracking. In *IEEE International Conference on Computer Vision*, pp. 1500–1507.

Klein G and Drummond T (2004) Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing* 22: 769–776.

Klein G and Murray D (2006) Full-3D edge tracking with a particle filter. In *British Machine Vision Conference*.

Kratochvil BE, Dong L and Nelson BJ (2009) Real-time rigid-body visual tracking in a scanning electron microscope. *The International Journal of Robotics Research* 28: 498–511.

Kwon J, Choi M, Park FC and Chun C (2007) Particle filtering on the Euclidean group: framework and applications. *Robotica* 25: 725–737.

Kwon J and Park FC (2010) Visual tracking via particle filtering on the affine group. *The International Journal of Robotics Research* 29: 198.

Kyrki V and Kragic D (2005) Integration of model-based and model-free cues for visual object tracking in 3D. In *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1566–1572.

Lepetit V, Moreno-Noguer F and Fua P (2009) EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision* 81: 155–166.

Liu MY, Tuzel O, Veeraraghavan A and Chellappa R (2010) Fast directional chamfer matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1696–1703.

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60: 91–110.

Michel P, Chestnut J, Kagami S, Nishiwaki K, Kuffner J and Kanade T (2007) GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 463–469.

Moakher M (2003) Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications* 24: 1–16.

Mörwald T, Zillich M and Vincze M (2009) Edge tracking of textured objects with a recursive particle filter. In: *International Conference on Computer Graphics and Vision (Graphicon)*, pp. 96–103.

Mörwald T, Prankl J, Richtsfeld A, Zillich M and Vincze M (2010) BLORT - the blocks world robotic vision toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)*.

Olson CF and Huttenlocher DP (2002) Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing* 6: 103–113.

Park W, Liu Y, Zhou Y, Moses M and Chirikjian GS (2008) Kinematic state estimation and motion planning for stochastic nonholonomic systems using the exponential map. *Robotica* 26: 419–434.

Pressigout M and Marchand E (2006) Real-time 3D model-based tracking: Combining edge and texture information. In *IEEE International Conference on Robotics and Automation*, pp. 2726–2731.

Pupilli M and Calway A (2006) Real-time camera tracking using known 3D models and a particle filter. In *IEEE International Conference on Pattern Recognition*, vol. 1.

Ross DA, Lim J, Lin RS and Yang MH (2008) Incremental learning for robust visual tracking. *International Journal of Computer Vision* 77: 125–141.

Rosten E and Drummond T (2005) Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, vol. 2.

Tamadazte B, Marchand E, Dembélé S and Fort-Piat NL (2010) CAD model-based tracking and 3D visual-based control for MEMS microassembly. *The International Journal of Robotics Research* 29: 1416–1434.

Teulière C, Marchand E and Eck L (2010) Using multiple hypothesis in model-based tracking. In *IEEE International Conference on Robotics and Automation*.

Vacchetti L, Lepetit V and Fua P (2004) Combining edge and texture information for real-time accurate 3D camera tracking. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 48–56.

Wang Y and Chirikjian GS (2006) Error propagation on the Euclidean group with applications to manipulator kinematics. *IEEE Transactions on Robotics* 22: 591–602.

Wang Y and Chirikjian GS (2008) Nonparametric second-order theory of error propagation on motion groups. *The International Journal of Robotics Research* 27: 1258.

Xavier J and Manton JH (2006) On the generalization of AR processes to Riemannian manifolds. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Yesin KB and Nelson BJ (2005) A CAD model based tracking system for visually guided microassembly. *Robotica* 23: 409–418.

Appendix A: IRLS and its Jacobian derivation

Given the current pose hypothesis $\mathbf{X}_t$, the residual vector $\hat{\mathbf{r}} \in \mathfrak{R}^{N_{\hat{z}}}$ which represents the Euclidean distances between the 2D projected sampled points $\mathbf{p}$ and their corresponding nearest edge points $\mathbf{q}$, we would like to find $\hat{\boldsymbol{\mu}} \in \mathfrak{R}^6$ which minimizes the residual $\hat{\mathbf{r}}$ as follows:

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \sum_{i=1}^{N_{\hat{z}}} \|\hat{r}_i\|^2 \tag{29}$$

$$= \arg\min_{\boldsymbol{\mu}} \sum_{i=1}^{N_{\hat{z}}} \|\mathbf{q}_i - \mathbf{p}_i\|^2. \tag{30}$$

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

$$\acute{\mathbf{X}}_t = \mathbf{X}_t \cdot \exp(\sum_{i=1}^{6} \hat{\mu}_i \mathbf{E}_i) \tag{31}$$

where $\exp : \mathfrak{se}(3) \mapsto SE(3)$ is the exponential map. From the camera model in (1), the projected point $\mathbf{p}_i$ in (30) of the ith 3D sampled point in object coordinates $\mathbf{P}_i^{\mathcal{O}} = (x_i^{\mathcal{O}}, y_i^{\mathcal{O}}, z_i^{\mathcal{O}}, 1)^{\mathsf{T}}$ is presented by

$$\mathbf{p}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \texttt{Project}(\mathbf{K}, \acute{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \tag{32}$$

where $\mathbf{K}$ is the intrinsic parameter matrix (3).

Given $\mathbf{p}_i$ and its unit normal vector $\mathbf{n}_i$, we can build a Jacobian matrix $\mathbf{J} \in \mathfrak{R}^{N_{\hat{z}} \times 6}$ by computing the partial derivative of $\mathbf{n}_i^{\mathsf{T}} \mathbf{p}_i$ in the direction μ_j at $\boldsymbol{\mu} = \mathbf{0}$:

$$J_{ij} = \left. \frac{\partial \mathbf{n}_i^{\mathsf{T}} \mathbf{p}_i}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \tag{33}$$

$$= \mathbf{n}_i^{\mathsf{T}} \frac{\partial}{\partial \mu_j} \left. \begin{pmatrix} u_i \\ v_i \end{pmatrix} \right|_{\boldsymbol{\mu}=\mathbf{0}} \tag{34}$$

$$= \mathbf{n}_i^{\mathsf{T}} \frac{\partial}{\partial \mu_j} \left. \Big(\texttt{Project}(\mathbf{K}, \acute{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \Big) \right|_{\boldsymbol{\mu}=\mathbf{0}} \tag{35}$$

in order to find $\hat{\boldsymbol{\mu}}$ which minimizes the residual $\hat{\mathbf{r}}$ by solving the following equation:

$$\mathbf{J}\hat{\boldsymbol{\mu}} = \hat{\mathbf{r}}. \tag{36}$$

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

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \\ 1 \end{pmatrix} \tag{37}$$

$$\begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix} = \begin{pmatrix} \frac{x_i^{\mathcal{C}}}{z_i^{\mathcal{C}}} \\ \frac{y_i^{\mathcal{C}}}{z_i^{\mathcal{C}}} \end{pmatrix}. \tag{38}$$

Their corresponding Jacobian matrices are then obtained as

$$\mathbf{J}_K = \begin{pmatrix} \frac{\partial u_i}{\partial \tilde{u}_i} & \frac{\partial u_i}{\partial \tilde{v}_i} \\ \frac{\partial v_i}{\partial \tilde{u}_i} & \frac{\partial v_i}{\partial \tilde{v}_i} \end{pmatrix} = \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \tag{39}$$

$$\mathbf{J}_P = \begin{pmatrix} \frac{\partial \tilde{u}_i}{\partial x_i^{\mathcal{C}}} & \frac{\partial \tilde{u}_i}{\partial y_i^{\mathcal{C}}} & \frac{\partial \tilde{u}_i}{\partial z_i^{\mathcal{C}}} \\ \frac{\partial \tilde{v}_i}{\partial x_i^{\mathcal{C}}} & \frac{\partial \tilde{v}_i}{\partial y_i^{\mathcal{C}}} & \frac{\partial \tilde{v}_i}{\partial z_i^{\mathcal{C}}} \end{pmatrix} = \begin{pmatrix} \frac{1}{z_i^{\mathcal{C}}} & 0 & -\frac{x_i^{\mathcal{C}}}{(z_i^{\mathcal{C}})^2} \\ 0 & \frac{1}{z_i^{\mathcal{C}}} & -\frac{y_i^{\mathcal{C}}}{(z_i^{\mathcal{C}})^2} \end{pmatrix}. \tag{40}$$

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

$$\left. \frac{\partial \acute{\mathbf{X}}_t}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} = \mathbf{X}_t \frac{\partial}{\partial \mu_j} \exp(\Sigma_{i=1}^{6} \mu_i \mathbf{E}_i) \Big|_{\boldsymbol{\mu}=\mathbf{0}} \tag{41}$$

$$= \mathbf{X}_t \mathbf{E}_j. \tag{42}$$

We then obtain the partial derivative of the $\mathbf{P}_i^{\mathcal{C}}$ in the direction μ_j at $\boldsymbol{\mu} = \mathbf{0}$ via the transformation (2) from object to camera coordinate frames:

$$\left. \frac{\partial \mathbf{P}_i^{\mathcal{C}}}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} = \left. \frac{\partial \acute{\mathbf{X}}_t \mathbf{P}_i^{\mathcal{O}}}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}}$$

$$= \left. \frac{\partial \acute{\mathbf{X}}_t}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \mathbf{P}_i^{\mathcal{O}}$$

$$= \mathbf{X}_t \mathbf{E}_j \mathbf{P}_i^{\mathcal{O}}. \tag{43}$$

With the two Jacobian matrices in (39), (40) and the partial derivative of the $\mathbf{P}_i^{\mathcal{C}}$ in (43), we finally obtain the Jacobian matrix $\mathbf{J}$ from the intermediate result in (35) as

$$J_{ij} = \mathbf{n}_i^{\mathsf{T}} \frac{\partial}{\partial \mu_j} \left. \Big(\texttt{Project}(\mathbf{K}, \acute{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \Big) \right|_{\boldsymbol{\mu}=\mathbf{0}} \tag{44}$$

$$= \mathbf{n}_i^{\mathsf{T}} \mathbf{J}_K \begin{pmatrix} \mathbf{J}_P & \begin{matrix} 0 \\ 0 \end{matrix} \end{pmatrix} \left. \frac{\partial \mathbf{P}_i^{\mathcal{C}}}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \tag{45}$$

$$= \mathbf{n}_i^{\mathsf{T}} \mathbf{J}_K \begin{pmatrix} \mathbf{J}_P & \begin{matrix} 0 \\ 0 \end{matrix} \end{pmatrix} \mathbf{X}_t \mathbf{E}_j \mathbf{P}_i^{\mathcal{O}}. \tag{46}$$

We could solve Equation (36) using the general pseudo-inverse of **J**:

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

The above equation, however, tends to be sensitive to erroneous measurements. As alternatives, M-estimators (Huber 1981) or RANSAC (Fischler and Bolles 1981) are preferable. In particular, the M-estimator approach, in which a robust penalty function to the residuals is applied, is preferred to RANSAC when the majority of measurements is inliers. For an M-estimator, it is common to define a function $\Psi(r) = \frac{1}{\lambda_w + |r|}$ which penalizes for the residual errors r and λ_w is a parameter which is usually set to one standard deviation of the inliers. With this function, we can have a diagonal matrix $\mathbf{W} = \mathrm{diag}[\Psi(r_1), \Psi(r_2), \ldots, \Psi(r_{N_{\hat{z}}})] \in \mathfrak{R}^{N_{\hat{z}} \times N_{\hat{z}}}$. Then $\hat{\boldsymbol{\mu}}$ can be estimated via weighted least squares as

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

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

Appendix B: Index to Multimedia Extensions

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

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Tracking results, considering edge orientation (Figure 4)
2	Video	Tracking results, considering multiple pose hypotheses (Figure 6)
3	Video	Tracking results, performing RANSAC (Figure 7)
4	Video	Tracking results, employing AR state dynamics (Figure 8)
5	Video	Tracking results, capability of re-initialization (Figure 10)
6	Video	A comparison (PF versus BLORT), synthetic image sequences (Figure 12)
7	Video	A comparison (PF versus BLORT), real image sequences (Figure 14)