Rao-Blackwellized Particle Filtering for Probing-Based 6-DOF Localization in Robotic Assembly

Yuichi Taguchi*, Tim K. Marks*, and Haruhisa Okuda†
*Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
†Advanced Technology R&D Center, Mitsubishi Electric Corporation, Amagasaki, Japan
*{taguchi, tmarks}@merl.com †Okuda.Haruhisa@ct.MitsubishiElectric.co.jp

***Abstract*— This paper presents a probing-based method for probabilistic localization in automated robotic assembly. We consider peg-in-hole problems in which a needle-like peg has a single point of contact with the object that contains the hole, and in which the initial uncertainty in the relative pose (3D position and 3D angle) between the peg and the object is much greater than the required accuracy (assembly clearance). We solve this 6 degree-of-freedom (6-DOF) localization problem using a Rao-Blackwellized particle filter, in which the probability distribution over the peg's pose is factorized into two components: The distribution over position (3-DOF) is represented by particles, while the distribution over angle (3-DOF) is approximated as a Gaussian distribution for each particle, updated using an extended Kalman filter. This factorization reduces the number of particles required for localization by orders of magnitude, enabling real-time online 6-DOF pose estimation. Each measurement is simply the contact position obtained by randomly repositioning the peg and moving towards the object until there is contact. To compute the likelihood of each measurement, we use as a map a mesh model of the object that is based on the CAD model but also explicitly models the uncertainty in the map. The mesh uncertainty model makes our system robust to cases in which the actual measurement is different from the expected one. We demonstrate the advantages of our approach over previous methods using simulations as well as physical experiments with a robotic arm and a metal peg and object.**

I. Introduction

This paper presents a probing-based method for probabilistic localization in automated robotic assembly. We consider peg-in-hole problems in which the peg is needle-like (has a single point of contact when probing) and in which the initial estimate of the pose of the peg with respect to the object may be quite inaccurate. Pose uncertainty of the peg with respect to the object may greatly exceed the assembly clearance (the desired accuracy) in all 6 degrees of freedom (6-DOF), comprising 3-DOF uncertainty in the peg's position and 3-DOF uncertainty in the angle of the peg with respect to the object. We use as measurements contact positions obtained by repositioning a robot arm that holds the peg and then moving the peg in the general direction of the object until there is contact. We assume that other than contact detection and robot arm encoders, there are no other sensors (such as cameras), though of course another sensor could be used to bring the peg into the vicinity of the part before implementing our algorithm. The goal of our algorithm is to deal with levels of uncertainty for which spiral search and other neighborhood search strategies [1], [2] would fail due to large initial uncertainty, too many degrees of freedom, and the existence of numerous local minima. We therefore address the localization problem (determining the 6-DOF relative pose of the peg and object) rather than the dynamics of peg insertion.

A. Relation to Previous Work

Here we describe previous work on particle-based Monte-Carlo localization for assembly using probing. Chhatpar and Branicky presented a localization method using probing and particle filtering for lock-key assembly [3] (which is quite similar to the needle-like peg-in-hole problem that we address) and round/square peg-in-hole problems [4]. They first exhaustively probe every x, y location of the object with the peg to generate a contact configuration-space map, which describes all possible transformations in which the peg has a contact with the object. After this preprocessing step of dense probing, they perform particle filtering by sequentially probing the object and using the contact positions as the observations.

Thomas et al. [5] similarly describe an exhaustive preprocessing step, densely probing an object with a force/torque sensor to generate a force/torque map that consists of contact force and torque at every possible contact pose. They also estimated a force-torque map directly from a CAD model, but this estimated force-torque map was not as accurate as the map acquired by probing, so they did not use the CAD-model-based map for localization. Thomas et al. [5] also used particle filtering to match each force/torque observation to the map and to incorporate observations from a camera.

Since the number of particles required for standard particle filtering increases roughly exponentially with the number of dimensions in the search space, the aforementioned previous methods, which use particle filtering for all dimensions, are not well-suited for full 6-DOF localization. Although the formulations of these previous methods are described for 6-DOF uncertainty, in practice they were only used for localization in lower-dimensional search spaces (2- or 3-DOF), such as the 2-DOF case in which uncertainty only exists in x, y translation.

In this paper, we solve the full 6-DOF localization problem using a Rao-Blackwellized particle filter (RBPF) [6], in which the probability distribution over position (3-DOF) is represented using particles, and the distribution over angle

(3-DOF) is approximated by a Gaussian distribution conditioned on the position of each particle. This factorization greatly improves the efficiency of the algorithm, resulting in an orders-of-magnitude reduction in the number of particles and computational resources required as compared to standard particle filtering. We use particles to represent the position of the peg with respect to the object, because given weak prior information and only a small amount of evidence, the posterior distribution over peg position tends to be multimodal. Given the position of the peg (at the current and previous time steps), however, the posterior distribution over angle is more well-behaved, and thus a Gaussian approximation is suitable. In our RBPF approach, at each time step we first use particle filtering to update the probability distribution over the position of the peg. We then use extended Kalman filtering (EKF) for each particle to update the particle's weight and probability distribution over angle, conditioned on the particle's position.

We use as the map a mesh model of the object that is being probed, which we generate before probing the object using a CAD model as well as any prior knowledge about the uncertainty of the CAD model. The mesh model explicitly models uncertainty about the position of its faces, edges, and vertices. This probabilistic map of the object's surface enables our algorithm to succeed in situations in which measurements are not consistent with the CAD model. Such differences can arise from measurement errors due to numerous factors such as slipping and sensor imprecision, as well as from differences between the CAD model and the object that can arise from causes such as manufacturing limitations, undocumented design changes, and reverse-engineered (approximate) CAD models.

Prior to the actual pose estimation, previous approaches include an exhaustive preprocessing step in which the object is probed with the peg at every possible contact location or pose, and the contact position or force/torque values are measured. The resulting detailed map is later used to accurately evaluate the measurement probability for pose estimation. Our new mesh representation, which explicitly accounts for varying levels of uncertainty in the map, is designed to allow robust localization with only an approximate map, thus obviating the need both for a time-consuming map measurement preprocessing step and for storing a large amount of detailed (dense) map data.

We describe our system model and RBPF inference algorithm in Section II. Then in Section III, we demonstrate our algorithm's marked improvement in efficiency over previous methods using physical experiments (with a robot arm and a metal peg and object) and simulations (using a virtual model of the metal object as well as a more complex simulated object). We conclude with a brief discussion in Section IV.

II. Localization Method

Our problem is to find the pose of a needle-like peg with respect to an object, by probing the object with the peg. The 6-DOF uncertainty between the peg and the object is represented as $(\boldsymbol{s}, \boldsymbol{\theta})$, where $\boldsymbol{s} = (x, y, z)^{\mathrm{T}}$ and $\boldsymbol{\theta} = (\alpha, \beta, \gamma)^{\mathrm{T}}$ are relative positions and angles, respectively, between the peg and the object. The angles (α, β, γ) are defined as angles of rotation around each of the (x, y, z) axes, respectively.

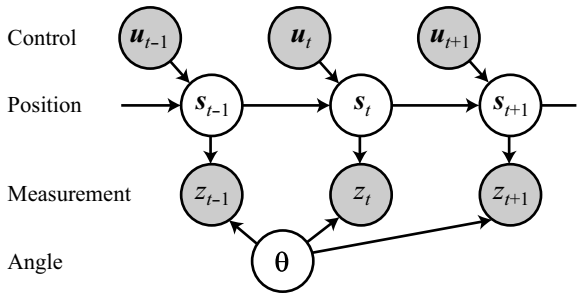


Fig. 1. Graphical model showing the probabilistic dependencies of our localization method. The angle variables $\boldsymbol{\theta}$ can be considered as analogous to the map variables in typical methods for SLAM in mobile robotics. The shaded nodes represent variables that are measured, while the white nodes represent variables that are inferred.

The key to our approach is to factorize the probability distribution over pose separately into two parts: 3D position $\boldsymbol{s}$ and 3D rotation $\boldsymbol{\theta}$. The probability distribution over position is represented by particles, which enables our system to represent multimodal distributions for position. The probability distribution over rotation angles is represented as a Gaussian distribution for each particle, conditioned on the current and previous positions of the particle.

The graphical model in Fig. 1 shows the probabilistic dependencies of our system. Given the sequence of motion commands from time 1 to t, denoted $\boldsymbol{u}_{1:t}$, and the sequence of observations, $z_{0:t}$, our goal is to infer the posterior distribution over the position and angle, $p(\boldsymbol{s}_{0:t}, \boldsymbol{\theta} | z_{0:t}, \boldsymbol{u}_{1:t})$. We factorize this posterior probability as follows:

$$p(\boldsymbol{s}_{0:t}, \boldsymbol{\theta} | z_{0:t}, \boldsymbol{u}_{1:t}) = p(\boldsymbol{s}_{0:t} | z_{0:t}, \boldsymbol{u}_{1:t}) p(\boldsymbol{\theta} | \boldsymbol{s}_{0:t}, z_{0:t}). \quad (1)$$

This factorization enables us to separately estimate the posterior distribution over position and angle using a Rao-Blackwellized particle filter [6], as follows. We first update the position of each particle using the motion model and compute the particle's weight using the measurement model and the particle's previous Gaussian distribution over angles. We then update the particle's Gaussian estimate of angles by using extended Kalman filtering (EKF).

In our framework, therefore, each particle maintains a 3D position estimate $\boldsymbol{s}_t$ and a 3D Gaussian distribution over angle, which is represented using the sufficient statistics, mean $\boldsymbol{\mu}_t$ and covariance $\boldsymbol{\Sigma}_t$. Note that the observation variable is binary, $z_t \in \{1, 0\}$, where $z_t = 1$ if at time t the peg has a contact with the object at position $\boldsymbol{s}_t$, and $z_t = 0$ otherwise. However, since we update the state only at the time instant when the robot senses a contact, $z_t = 1$ at every time step t. We therefore simplify the notation $p(z_t = 1)$ using the shorthand $p(z_t)$ throughout this paper.

The factorization (1) is well-studied for simultaneous localization and mapping (SLAM) problems in mobile robotics [7], in which the pose of a mobile robot is estimated using particle filtering, and each particle's map is independently updated using EKF. Our approach to localization in robotic assembly can be viewed as analogous to SLAM in mobile

robotics, as follows: Our position vector $\boldsymbol{s}$ is analogous to the pose of a mobile robot, and our angle vector $\boldsymbol{\theta}$ is analogous to the mobile robot's map of the environment. When RBPF is applied to the SLAM problem [7], each particle maintains its own estimate of the map that depends on that particle's pose history. Analogously, in our system each particle maintains its own estimate of (distribution over) angle that depends on that particle's position history.

A. Coordinate Transformations

The position of the peg at time t, denoted $\boldsymbol{s}_t$, is represented in a frame of reference that we call the *base* coordinate system. The transformation between the *robot* coordinate system (the frame of reference of the robot arm) and the base coordinate system can be selected arbitrarily; here we assume it is represented by a translation, without rotation. (Because our base and robot coordinate systems differ by a pure translation, any motion of the peg is represented by the same translation in the base coordinate system as in the robot coordinate system.) There is also an *object* coordinate system, which is fixed with respect to the object that contains the hole. Whereas $\boldsymbol{s}_t$ represents the peg's position at time t in the base coordinate system, we denote the peg's position in the object coordinate system at time t as $\boldsymbol{s}^o_t$.

The base coordinate system and the object coordinate system are related by a 3D rotation of $\boldsymbol{\theta}$ about the initial position of the peg, $\boldsymbol{s}_0$, as illustrated in Figure 2. Thus, the initial position of the peg is the same in both the base and object coordinate systems: $\boldsymbol{s}_0 = \boldsymbol{s}^o_0$. In our setup, $\boldsymbol{\theta}$ is the unknown relative rotation between the robot and the object.

B. Motion Model

At each time step, the robot arm moves the peg from one contact position (one point of contact between the peg and the object) to another. The position of the peg in the robot coordinate system is obtained from the robot arm's internal encoders. As we described above, the translation of the peg from time $t-1$ to time t, which we denote $\boldsymbol{u}_t$, is identical in the robot coordinate system and the base coordinate system. The motion of the peg in the base coordinate system, from $\boldsymbol{s}_{t-1}$ to $\boldsymbol{s}_t$, is therefore given by

$$\boldsymbol{s}_t = \boldsymbol{s}_{t-1} + \boldsymbol{u}_t + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(\boldsymbol{0}, \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{pmatrix}\right), \quad (2)$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

In our experiments, we used $\sigma_x = \sigma_y = \sigma_z = 0.1$ mm. Since our robot arm has accurate control, the motion error is in reality much smaller even than this small amount of noise that we assume. We nonetheless include this small noise term in our motion model to reduce the particle deprivation (particle impoverishment) problem. That is, none of the particles will have an exactly correct (perfect) estimate of the initial contact position $\boldsymbol{s}_0$, and including a small noise term in the motion model enables the particle filter to compensate for this inaccuracy over time.

To determine $\boldsymbol{s}^o_t$, the position of the peg at time t in the object coordinate system, we compute a rotation matrix $\boldsymbol{R}$

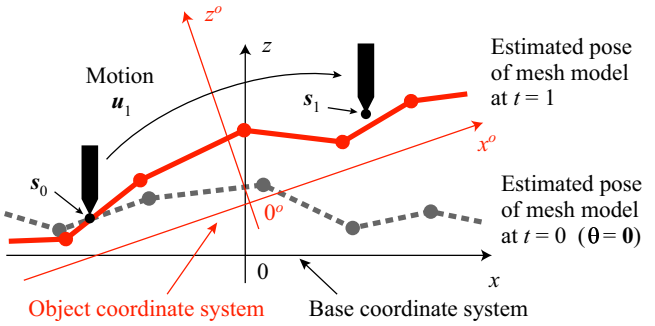


Fig. 2. Visualization of the *base* and *object* coordinate systems and of updates to position and angle estimates. The surface of the object is represented by gray dashed lines (in the base coordinates system) or by red lines (in the object coordinate system). The position of the peg at time t, denoted $\boldsymbol{s}_t$, is defined in the base coordinate system. The object coordinate system (red axes) is rotated with respect to the base coordinate system (black axes) by the angle $\boldsymbol{\theta}$ about the initial position of the peg, $\boldsymbol{s}_0$. The mesh model representation (and the measurement probability computation) resides in the object coordinate system. At time $t = 0$, each particle's estimate of the relative angle $\boldsymbol{\theta}$ between robot and object coordinates is a Gaussian distribution with mean $\boldsymbol{0}$ (this value $\boldsymbol{\theta} = \boldsymbol{0}$ corresponds to the object and base coordinate systems being identical). At time $t = 1$, the particle's estimate of $\boldsymbol{\theta}$ is updated, causing a corresponding rotation of the its estimate of the pose of the mesh model (i.e., a rotation of the particle's estimate of the object coordinate frame).

using the angle $\boldsymbol{\theta} = (\alpha, \beta, \gamma)^{\mathrm{T}}$, as follows:

$$\boldsymbol{R}(\boldsymbol{\theta}) = \begin{pmatrix} c_\gamma c_\beta & c_\gamma s_\beta s_\alpha - s_\gamma c_\alpha & c_\gamma s_\beta c_\alpha + s_\gamma s_\alpha \\ s_\gamma c_\beta & s_\gamma s_\beta s_\alpha + c_\gamma c_\alpha & s_\gamma s_\beta c_\alpha - c_\gamma s_\alpha \\ -s_\beta & c_\beta s_\alpha & c_\beta c_\alpha \end{pmatrix}, \quad (3)$$

where s_α and c_α are shorthand for $\sin\alpha$ and $\cos\alpha$, respectively. According to the definition of our coordinate systems, the peg's position in object coordinates is given by

$$\boldsymbol{s}^o_t = \boldsymbol{R}(\boldsymbol{\theta})(\boldsymbol{s}_t - \boldsymbol{s}_0) + \boldsymbol{s}_0. \quad (4)$$

Since the contact position in the robot coordinate system does not depend on the angle of the peg, our motion model (2) only includes 3D translation. Hence, in our experiments we held the angle of the peg fixed (to the $-z$ direction) in the robot coordinate system. Our inference algorithm nonetheless performs full 6-DOF localization, because the peg's position in the *object* coordinate system depends not only on the 3D translation (motion control signal) but also on the 3D rotation angle $\boldsymbol{\theta}$.

C. Map Representation and Measurement Model

We use as the map of the object a mesh model consisting of vertices, edges, and triangular faces, all of which we define as features in the model. The mesh model can be generated from a CAD model of the object. To deal with measurement errors as well as differences between the CAD model and the actual object, we model measurement uncertainty using a Gaussian probability density function (pdf) of the distance between the contact position and the features of the mesh model. For each feature (face, edge, or vertex) f_k, where $k = \{1, \ldots, K\}$, the standard deviation of that feature's measurement uncertainty is denoted σ_k, as shown in Fig. 3.

The motivation behind this representation of map uncertainty is that different features can exhibit different types

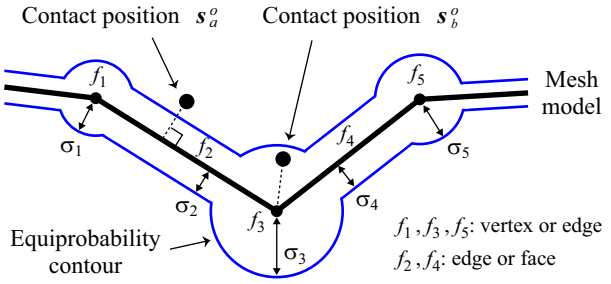


Fig. 3. Our map representation. We define every face, edge, and vertex in the mesh model as a feature f_k, which has uncertainty σ_k. For a given contact position in object coordinates, $\boldsymbol{s}^o$, the contact feature is defined as the feature f_k whose distance to the contact position, normalized by σ_k, is smallest. (For example, f_2 and f_3 are the contact features for peg positions $\boldsymbol{s}_a^o$ and $\boldsymbol{s}_b^o$, respectively). Measurement probability is defined as a Gaussian function (with standard deviation σ_k) of the distance from the contact point to the contact feature f_k.

of measurement errors. As evidenced in the video that accompanies this paper, in our experiments the peg is more likely to slip or bend when it contacts faces that are steeply slanted (faces that form a small angle with the probing direction of the peg). For this reason, when generating the map from a CAD model, we evaluate the normals of each face and assign a larger uncertainty (a larger value of σ_k) to faces f_k for which the probing direction forms a small angle with the plane of the face (according to the mean of the initial distribution over angle, which in our experiments was $\boldsymbol{\theta} = \mathbf{0}$). Another typical source of error is differences between the CAD model and the actual object. For instance, when the CAD model dictates that adjacent faces meet at a sharp angle, these corners where the faces meet may in fact be rounded due to limits of the manufacturing process. For this reason, we assign larger uncertainty to edges and vertices at the intersection of planes whose normals form large angles with each other. (More precisely, we compute the maximum angle between the normals of all possible pairs of the faces that border each edge/vertex, and assign larger uncertainty to those edges/vertices that have larger angles.)

Note that our mesh model does not require the use of the particular heuristics described above. It only requires that some uncertainty value be assigned to every feature. This enables users to incorporate knowledge of their particular industrial settings or manufacturing processes into the uncertainty maps. In fact, there is no requirement that the uncertainty model be Gaussian. It could be defined by any of a wide range of probability fields or energy functions.

In our implementation, the measurement probability is computed based on the distance between the mesh model and the peg's position in object coordinates, $\boldsymbol{s}^o$, as well as the measurement uncertainty of the contact feature. The distance from the peg's position to each feature f_k is denoted $d(\boldsymbol{s}^o, f_k)$, where the function d computes the Euclidean distance between the position $\boldsymbol{s}^o$ and feature f_k. (Note that the distance to a face or edge is only defined if the perpendicular projection of the point to the corresponding plane or line lies within that face or edge.) We compute the distance to the peg's contact location from all features and select as the contact feature the one whose distance, normalized by the corresponding standard deviation, is smallest. The index of the contact feature, k_c, is therefore given by

$$k_c = \arg\min_k \frac{d(\boldsymbol{s}^o, f_k)}{\sigma_k}. \tag{5}$$

Since the peg's position in object coordinates, $\boldsymbol{s}^o$, is computed from $\boldsymbol{s}$ (the peg's position in base coordinates) and $\boldsymbol{\theta}$ (the angle between base coordinates and object coordinates) using (4), the distance measure h can be expressed equivalently in either object coordinates (as a function of $\boldsymbol{s}^o$) or in base coordinates (as a function of $\boldsymbol{s}$ and $\boldsymbol{\theta}$):

$$h_{k_c}(\boldsymbol{s}, \boldsymbol{\theta}) = d(\boldsymbol{s}^o, f_{k_c}). \tag{6}$$

Our measurement probability at time t is computed using this distance measure and the uncertainty of the mesh model:

$$p(z_t | \boldsymbol{s}_t, \boldsymbol{\theta}) = \mathcal{N}(h_{k_c}(\boldsymbol{s}_t, \boldsymbol{\theta}); 0, \sigma_{k_c}^2). \tag{7}$$

D. Inference Algorithm

As described at the beginning of Section II, our inference algorithm capitalizes on factorization (1) of the posterior probability by using an RBPF [6] to infer the relative pose of the peg and the object. We estimate the 3D position of the peg at time t in base coordinates, $\boldsymbol{s}_t$, using particles, where each particle represents a single discrete position estimate. Each particle also maintains a Gaussian probability distribution over the 3D rotation, $\boldsymbol{\theta}$, from base coordinates to object coordinates (the relative rotation between the peg and object). Each time the peg is moved to a new contact position, our inference algorithm incorporates the new observation by first updating each particle's estimate of the peg position using the motion model. We then update each particle's distribution over angle, given that particle's position, using an EKF.

At time t, each particle j maintains a 3D point estimate, $\boldsymbol{s}_t^{[j]}$, of the position of the peg at time t, as well as a normal distribution over angle, $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{\theta}; \boldsymbol{\mu}_t^{[j]}, \boldsymbol{\Sigma}_t^{[j]}\big)$.

1) Initialization: We initially sample J particles from a uniform distribution over the position (x, y) in base coordinates, bounded by the maximum initial (x, y) uncertainty (which in our experiments is 30 mm square, as illustrated in Fig. 5(a)). The initial value of z for each particle j is determined from that particle's (x, y) values such that the particle position $\boldsymbol{s}_0^{[j]} = (x, y, z)$ is set on the surface of the map (of the mesh model). Note that the initial position $\boldsymbol{s}_0^{[j]}$ is also used as the center of rotation for particle j, which is constant across all time steps. Every particle's Gaussian distribution over the 3D angle $\boldsymbol{\theta}$ is initialized with mean $\boldsymbol{\mu}_0^{[j]} = \mathbf{0}$ and a diagonal covariance matrix $\boldsymbol{\Sigma}_0^{[j]}$ representing the initial angular uncertainty (in our experiments, we assumed an initial Gaussian uncertainty with standard deviation 10° about each rotation axis).

2) Particle Update at Each Time Step: When the robot arm moves the peg from one contact position to the next, we update the state of each particle $j = \{1, \ldots, J\}$ based on the motion $\boldsymbol{u}_t$ in base coordinates (obtained from the robot arm encoders) and the observation z_t that the robot detects a contact at that position. For the RBPF update, we use similar

techniques to those used in FastSLAM [7], [8]. Algorithm 1 gives a pseudocode summary of the update algorithm at each time step, which is detailed below. In Algorithm 1, $\mathcal{X}_t$ represents the collection of particles at time t.

Figure 2 illustrates the particle update from $t = 0$ to $t = 1$. Every particle's probability distribution over angle is initialized with mean $\boldsymbol{\mu}_0^{[j]} = \mathbf{0}$. At this mean value of $\boldsymbol{\theta} = \mathbf{0}$, the base coordinate system and the object coordinate system would be identical, which corresponds to the estimate of the mesh model at $t = 0$ shown in Fig. 2. After a new observation at time $t = 1$, the particle's estimate of position is updated from $\boldsymbol{s}_0^{[j]}$ to $\boldsymbol{s}_1^{[j]}$, according to the control signal $\boldsymbol{u}_1$ and the motion model, as described below in (8). Based on the particle's new position, the particle's estimate of $\boldsymbol{\theta}$ is updated to a new distribution. The particle's representation of the object coordinate system (and hence the mesh model) at $t = 1$ will be rotated about the point $\boldsymbol{s}_0^{[j]}$ by the angle $\boldsymbol{\theta}$ with respect to the base coordinate system, as indicated in Fig. 2. For illustration purposes, the figure shows a single value of the angle $\boldsymbol{\theta}$ at each time step, but in fact each particle maintains (and updates) an entire Gaussian distribution over the angle $\boldsymbol{\theta}$, as described below.

a) Position Update: The position $\boldsymbol{s}_t^{[j]}$ of particle j at time t is sampled from the proposal distribution given by the motion model (2), the particle's previous position $\boldsymbol{s}_{t-1}^{[j]}$, and the control $\boldsymbol{u}_t$:

$$\boldsymbol{s}_t^{[j]} \sim p\big(\boldsymbol{s}_t \big| \boldsymbol{s}_{t-1}^{[j]}, \boldsymbol{u}_t\big). \tag{8}$$

b) Angle Update: Based on each particle's updated position (in base coordinates), $\boldsymbol{s}_t^{[j]}$, and the observation z_t that there was a contact between the peg and the object, we compute the particle's posterior distribution over the angle $\boldsymbol{\theta}$ using an EKF update. To do so, we express the measurement probability as a function of $\boldsymbol{\theta}$ and linearize that measurement probability about $\boldsymbol{\theta} = \boldsymbol{\mu}_{t-1}^{[j]}$, the mean of the particle's previous estimate of the angular distribution.

We first use the value $\boldsymbol{\theta} = \boldsymbol{\mu}_{t-1}^{[j]}$ in (4) to compute the predicted particle position in the object coordinate system, $\hat{\boldsymbol{s}}_t^{o[j]}$:

$$\hat{\boldsymbol{s}}_t^{o[j]} = \boldsymbol{R}\big(\boldsymbol{\mu}_{t-1}^{[j]}\big)\big(\boldsymbol{s}_t^{[j]} - \boldsymbol{s}_0^{[j]}\big) + \boldsymbol{s}_0^{[j]}. \tag{9}$$

We then use (5) to determine the index of the contact feature, k_c, based on this predicted particle position.

We use the measurement probability (7) defined with respect to the contact feature, f_{k_c}, to update the posterior probability over angle, which is the second factor of (1). This posterior further factorizes as

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{s}_{0:t}, z_{0:t}) \;&\propto\; p(z_t|\boldsymbol{\theta}, \boldsymbol{s}_{0:t}, z_{0:t-1})\, p(\boldsymbol{\theta}|\boldsymbol{s}_{0:t}, z_{0:t-1}) \\ &= \underbrace{p(z_t|\boldsymbol{s}_t, \boldsymbol{\theta})}_{\mathcal{N}\left(h_{k_c}(\boldsymbol{s}_t^{[j]}, \boldsymbol{\theta}); 0, \sigma_{k_c}^2\right)} \underbrace{p(\boldsymbol{\theta}|\boldsymbol{s}_{0:t-1}, z_{0:t-1})}_{\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{t-1}^{[j]}, \boldsymbol{\Sigma}_{t-1}^{[j]}\right)}. \end{aligned} \tag{10}$$

The second factor in (10), the previous posterior $p(\boldsymbol{\theta}|\boldsymbol{s}_{0:t-1}, z_{0:t-1})$, is represented by a Gaussian with mean $\boldsymbol{\mu}_{t-1}^{[j]}$ and covariance $\boldsymbol{\Sigma}_{t-1}^{[j]}$, but the first factor in (10) is not a Gaussian distribution over $\boldsymbol{\theta}$ since the distance measure (6) is not linear in $\boldsymbol{\theta}$. Nonetheless, we can use (10) to approximate the posterior probability of the angle estimate as a Gaussian distribution. This update is performed using EKF by linearizing the measurement function about $\boldsymbol{\theta} = \boldsymbol{\mu}_{t-1}^{[j]}$:

$$\begin{aligned} h_{k_c}\big(\boldsymbol{s}_t^{[j]}, \boldsymbol{\theta}\big) &\approx h_{k_c}\big(\boldsymbol{s}_t^{[j]}, \boldsymbol{\mu}_{t-1}^{[j]}\big) + \tfrac{\partial h_{k_c}}{\partial \boldsymbol{\theta}}\big(\boldsymbol{s}_t^{[j]}, \boldsymbol{\mu}_{t-1}^{[j]}\big)\big(\boldsymbol{\theta} - \boldsymbol{\mu}_{t-1}^{[j]}\big) \\ &= \hat{h}_{k_c,t}^{[j]} + \boldsymbol{H}_t^{[j]}\big(\boldsymbol{\theta} - \boldsymbol{\mu}_{t-1}^{[j]}\big), \end{aligned} \tag{11}$$

where $\hat{h}_{k_c,t}^{[j]} = d\big(\hat{\boldsymbol{s}}_t^{o[j]}, f_{k_c}\big)$ is the distance measure computed at the predicted position $\hat{\boldsymbol{s}}_t^{o[j]}$, and $\boldsymbol{H}_t^{[j]}$ is Jacobian of the distance measure with respect to $\boldsymbol{\theta}$ computed at the predicted position and angle:

$$\boldsymbol{H}_t^{[j]} = \left(\frac{\partial h_{k_c}}{\partial \alpha}, \frac{\partial h_{k_c}}{\partial \beta}, \frac{\partial h_{k_c}}{\partial \gamma}\right)\Bigg|_{\left(\boldsymbol{s}, \boldsymbol{\theta}\right) = \left(\boldsymbol{s}_t^{[j]}, \boldsymbol{\mu}_{t-1}^{[j]}\right)}. \tag{12}$$

The particle's posterior distribution over angle is then computed using the standard EKF measurement update rule:

$$\boldsymbol{K}_t^{[j]} \;=\; \boldsymbol{\Sigma}_{t-1}^{[j]} \boldsymbol{H}_t^{[j]^{\mathrm{T}}} \big(\boldsymbol{H}_t^{[j]} \boldsymbol{\Sigma}_{t-1}^{[j]} \boldsymbol{H}_t^{[j]^{\mathrm{T}}} + \sigma_{k_c}^2\big)^{-1} \tag{13}$$

$$\boldsymbol{\mu}_t^{[j]} \;=\; \boldsymbol{\mu}_{t-1}^{[j]} - \boldsymbol{K}_t^{[j]} \hat{h}_{k_c,t}^{[j]} \tag{14}$$

$$\boldsymbol{\Sigma}_t^{[j]} \;=\; \big(\boldsymbol{I} - \boldsymbol{K}_t^{[j]} \boldsymbol{H}_t^{[j]}\big) \boldsymbol{\Sigma}_{t-1}^{[j]} \tag{15}$$

c) Importance Weight Update and Resampling: Since we use the motion model (8) as the proposal distribution, the importance weight for each particle, $w_t^{[j]}$, is computed by marginalizing the measurement probability over the particle's previous angle estimate:

$$\begin{aligned} w_t^{[j]} &\propto w_{t-1}^{[j]} \cdot p\big(z_t \big| \boldsymbol{s}_t^{[j]}\big) \\ &= w_{t-1}^{[j]} \int p\big(z_t \big| \boldsymbol{s}_t^{[j]}, \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta} \big| \boldsymbol{s}_t^{[j]}\big) d\boldsymbol{\theta} \\ &= w_{t-1}^{[j]} \int \underbrace{p\big(z_t \big| \boldsymbol{s}_t^{[j]}, \boldsymbol{\theta}\big)}_{\mathcal{N}\left(h_{k_c}(\boldsymbol{s}_t^{[j]}, \boldsymbol{\theta}); 0, \sigma_{k_c}^2\right)} \underbrace{p\big(\boldsymbol{\theta} \big| \boldsymbol{s}_{0:t-1}^{[j]}, z_{0:t-1}\big)}_{\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{t-1}^{[j]}, \boldsymbol{\Sigma}_{t-1}^{[j]}\right)} d\boldsymbol{\theta}. \end{aligned} \tag{16}$$

The final integration in (16) is computed in closed form by using the same linear approximation that we used for the angle update, as follows [7]:

$$w_t^{[j]} \propto w_{t-1}^{[j]} \cdot \big(2\pi q_t^{[j]}\big)^{-1/2} \exp\Big(-\big(\hat{h}_{k_c,t}^{[j]}\big)^2 / 2 q_t^{[j]}\Big), \tag{17}$$

$$q_t^{[j]} = \boldsymbol{H}_t^{[j]} \boldsymbol{\Sigma}_{t-1}^{[j]} \boldsymbol{H}_t^{[j]^{\mathrm{T}}} + \sigma_{k_c}^2. \tag{18}$$

To maintain good particle diversity, at each time step we estimate the effective number of particles [9]:

$$J_{\text{eff}} = \frac{1}{\sum_{j=1}^{J} \big(w_t^{[j]}\big)^2}. \tag{19}$$

If $J_{\text{eff}} < J/2$, we perform resampling. Particles are resampled with probability proportional to their weights $w_t^{[j]}$; after resampling, all particle weights are reset to $w_t^{[j]} = 1/J$. Otherwise ($J_{\text{eff}} \geq J/2$), we do not resample, and all particles keep the current weights that were computed using (17).

Since our inference starts at $t = 0$ with global uncertainty, we initially use a relatively large number of particles. After several measurements, however, the number of particles required for localization decreases. We therefore reduce the total number of particles at each resampling step (whenever $J_{\text{eff}} < J/2$), until the number of particles reaches a predefined number, $J_{\min}$. Specifically, if the number of particles

Algorithm 1 Particle update algorithm

Particle Update $(\mathcal{X}_{t-1}, \boldsymbol{u}_t, z_t)$
$\mathcal{X}_t = \emptyset$ (the empty set)
for $j = 1$ to J **do**
  retrieve $\langle \boldsymbol{s}_0^{[j]}, \boldsymbol{s}_{t-1}^{[j]}, \boldsymbol{\mu}_{t-1}^{[j]}, \boldsymbol{\Sigma}_{t-1}^{[j]}, w_{t-1}^{[j]} \rangle$ from $\mathcal{X}_{t-1}$

  // Update position
  $\boldsymbol{s}_t^{[j]} \sim p\big(\boldsymbol{s}_t \big| \boldsymbol{s}_{t-1}^{[j]}, \boldsymbol{u}_t\big)$

  // Find contact feature with maximum likelihood
  $\hat{\boldsymbol{s}}_t^{o[j]} = R\big(\boldsymbol{\mu}_{t-1}^{[j]}\big)\big(\boldsymbol{s}_t^{[j]} - \boldsymbol{s}_0^{[j]}\big) + \boldsymbol{s}_0^{[j]}$
  for $k = 1$ to K **do**
    $\hat{h}_{k,t}^{[j]} = d\big(\hat{\boldsymbol{s}}_t^{o[j]}, f_k\big)$
  end for
  $k_c = \arg\min_k \hat{h}_{k,t}^{[j]} / \sigma_k$

  // Update angle and importance weight
  $\boldsymbol{H}_t^{[j]} = \frac{\partial h_{k_c}}{\partial \boldsymbol{\theta}}\big(\boldsymbol{s}_t^{[j]}, \boldsymbol{\mu}_{t-1}^{[j]}\big)$
  $q_t^{[j]} = \boldsymbol{H}_t^{[j]} \boldsymbol{\Sigma}_{t-1}^{[j]} \boldsymbol{H}_t^{[j]\mathrm{T}} + \sigma_{k_c}^2$
  $\boldsymbol{K}_t^{[j]} = \boldsymbol{\Sigma}_{t-1}^{[j]} \boldsymbol{H}_t^{[j]\mathrm{T}} / q_t^{[j]}$
  $\boldsymbol{\mu}_t^{[j]} = \boldsymbol{\mu}_{t-1}^{[j]} - \boldsymbol{K}_t^{[j]} \hat{h}_{k_c,t}^{[j]}$
  $\boldsymbol{\Sigma}_t^{[j]} = \big(\boldsymbol{I} - \boldsymbol{K}_t^{[j]} \boldsymbol{H}_t^{[j]}\big) \boldsymbol{\Sigma}_{t-1}^{[j]}$
  $w_t^{[j]} = w_{t-1}^{[j]} \cdot \big(2\pi q_t^{[j]}\big)^{-1/2} \exp\big(-\big(\hat{h}_{k_c,t}^{[j]}\big)^2 / 2 q_t^{[j]}\big)$

  add $\langle \boldsymbol{s}_{0,t}^{[j]}, \boldsymbol{\mu}_t^{[j]}, \boldsymbol{\Sigma}_t^{[j]}, w_t^{[j]} \rangle$ to $\mathcal{X}_t$
end for
normalize $w_t^{[j]}$ such that $\sum_j w_t^{[j]} = 1$

// Resampling
$J_{\text{eff}} = 1 \big/ \sum_j \big(w_t^{[j]}\big)^2$
if $J_{\text{eff}} < J/2$ **then**
  if $J > J_{\min}$ **then** $J = J/2$ **end if**
  $\mathcal{X}_t =$ resample J particles from $\mathcal{X}_t$ with probabilities $w_t^{[j]}$
  reset importance weights to $w_t^{[j]} = 1/J$
end if

J is greater than $J_{\min}$ at a resampling step, then we set $J = J/2$ (i.e., only the half number of particles is resampled from the current particle set). In future work, we may instead use more efficient strategies, such as KLD-sampling [10], to reduce the number of particles over time.

3) Convergence Check: For checking convergence, we compute the weighted average and weighted covariance of the particle positions in the object coordinate system:

$$\bar{\boldsymbol{s}}_t^o = \sum_{j=1}^J w_t^{[j]} \boldsymbol{s}_t^{o[j]} \tag{20}$$

$$\boldsymbol{\Psi}_t^o = \frac{\sum_{j=1}^J w_t^{[j]} (\boldsymbol{s}_t^{o[j]} - \bar{\boldsymbol{s}}_t^o)(\boldsymbol{s}_t^{o[j]} - \bar{\boldsymbol{s}}_t^o)^{\mathrm{T}}}{1 - \sum_{j=1}^J \big(w_t^{[j]}\big)^2}, \tag{21}$$

where $\boldsymbol{s}_t^{o[j]}$ is computed using (4) with the mean of each particle's posterior distribution over angle at time t:

$$\boldsymbol{s}_t^{o[j]} = \boldsymbol{R}\big(\boldsymbol{\mu}_t^{[j]}\big)\big(\boldsymbol{s}_t^{[j]} - \boldsymbol{s}_0^{[j]}\big) + \boldsymbol{s}_0^{[j]}. \tag{22}$$

We continue to probe the object with the peg (continue to collect observations for additional time steps) until the trace of the covariance matrix $\boldsymbol{\Psi}_t^o$ is less than some predetermined threshold (indicating that uncertainty among particle positions is small). Once this convergence condition is achieved, the peg is moved to the estimated position and angle of the hole. The estimated position of the hole is computed based on the weighted average of all particle positions, $\bar{\boldsymbol{s}}_t^o$. The estimated angle of the hole is computed based on a weighted average of particle angles, $\bar{\boldsymbol{\mu}}_t$, which we compute using a subset of particles (those with the largest weights, since we found that in practice, the particles with small weights can have very different angle estimates).

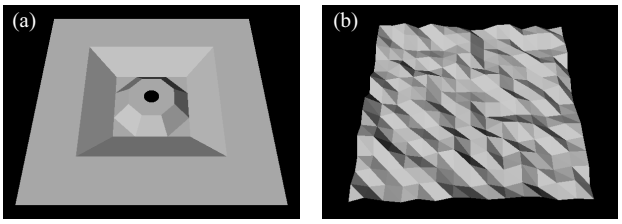


Fig. 4. Mesh models of: (a) the object with the hole pictured in Fig. 7(a), and (b) a randomly generated map. Model (a) spans 75 mm in each horizontal direction, with a 5 mm height difference between the uppermost and lowermost points on the surface. The random model (b) was generated by selecting z values uniformly from [0, 5] mm on a regular (x, y) grid, with interval 5 mm and total horizontal dimensions of 70 mm square.

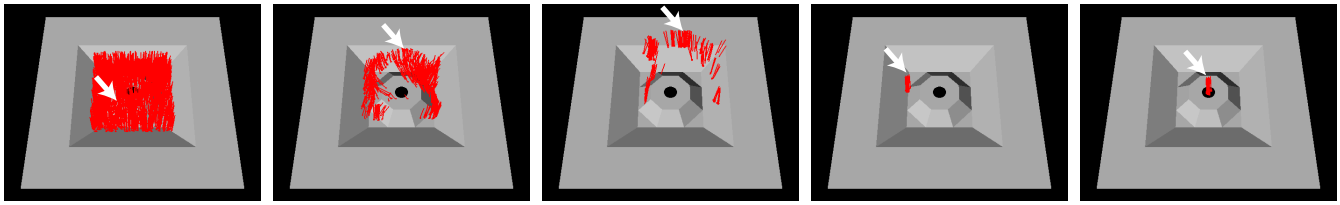
III. Experiments

In this section, we first show simulation results to compare our RBPF approach to standard particle filtering, which is the basis for previous methods [3], [4], [5]. The results demonstrate that for the 6-DOF localization problem, our system is orders of magnitude more efficient than previous approaches. We then describe physical experiments using a robot arm to insert a needle-like peg into a small hole in a metal object. The video that accompanies this paper shows an example sequence of localization and peg insertion using the robot arm.

A. Simulations

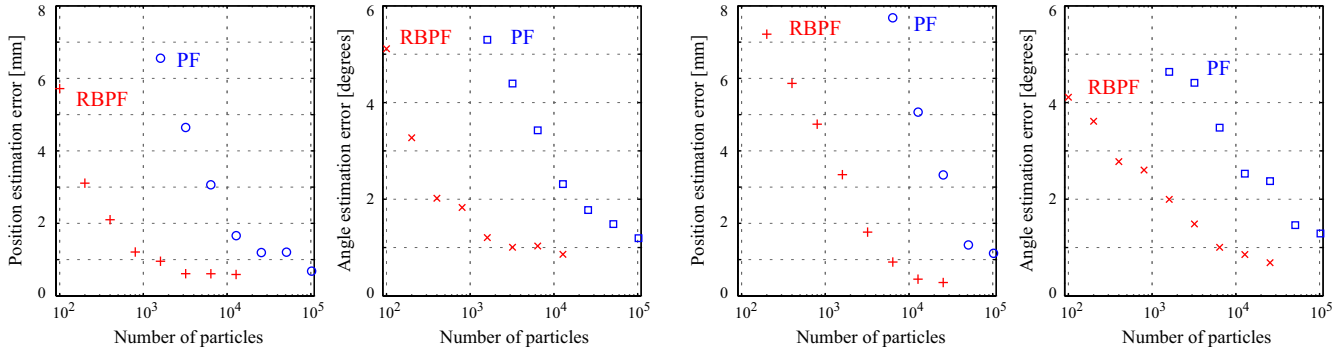
We used two different mesh models, shown in Fig. 4, to test our localization algorithm in simulations. The first mesh model, shown in Fig. 4(a), was generated from the CAD model of the actual physical object that we used for the robot experiments. The second, much more complex mesh model was the random surface shown in Fig. 4(b). In our simulations with the first mesh model, the goal of localization was to determine the pose (position and orientation) of the central hole. With the second model (random surface), the goal was to find the position and orientation of the center of the surface (the origin of the object coordinate system).

In each simulation trial, the (ground truth) initial position of the peg was randomly chosen from a uniform distribution 30 mm square (from $[-15, 15]$ mm in the x and y directions, with z coordinate given by the surface of the object), and the (ground truth) rotation angle between peg and object was randomly chosen from a uniform distribution from $[-10, 10]$ degrees about each axis. As described in Section II-D.1, particles were initialized with positions sampled from, and angle distributions covering, these same uncertainty regions. Each contact measurement was obtained by randomly moving the peg to a new horizontal position within a $[-15, 15]$ mm range of the peg's initial position, then moving the peg in the $-z$ direction (in base coordinates) until contacting the surface. Figure 5 shows the distribution of particles at different time steps of one simulation trial.



(a) $t = 0$ (b) $t = 2$ (c) $t = 4$ (d) $t = 7$ (converged) (e) Final insertion

Fig. 5. Distribution of particles at each time step of one simulation trial. Each particle is represented by a red line, indicating the particle's estimate of the peg position and the mean of the particle's distribution over the peg angle, both represented in the object coordinate system. At each time step, the white arrow indicates the ground-truth position of the peg. For the initialization (step 0), shown in (a), particle positions are sampled from the initial (30 mm $\times$ 30 mm) uncertainty region of (x, y), and particle angle distributions all have zero mean. The distribution of particles after update steps 2, 4, and 7 are shown in (b), (c), and (d), respectively. Once the algorithm determines convergence (d), it moves the peg to the inferred pose (position and angle) of the hole (e) based on a weighted mean across particles of the estimated pose.



(a) Results obtained using the object with the hole

(b) Results obtained using the random surface

Fig. 6. Comparison of our Rao-Blackwellized particle filter (RBPF) with a standard particle filter (PF), using as the map (a) the object pictured in Figs. 4(a) and 5, and (b) the random surface pictured in Fig. 4(b). In each part (a) and (b), the position error is shown in the graph on the left as Euclidean distance (in mm), while the angle error is shown on the right as the absolute angle (in degrees) between the correct peg insertion direction and the estimated peg insertion direction. The number of particles is shown in a logarithmic scale on the x-axis of each graph. The number of particles required by our RBPF approach is greatly reduced from the number required by previous PF approaches to this problem, with only a slight increase in the time required to update each particle.

Figure 6 compares our RBPF approach to a standard particle filter, which represents all 6-DOF of uncertainty using particles (corresponding to a 6-DOF version of previous approaches [3], [4]). For these simulations, we set the uncertainty in the mesh model (map) to $\sigma_k = 0.2$ mm for all features k. The performance is indicated in Fig. 6 by the final estimation error in position and angle. The position error is the Euclidean distance from the correct position, while the angle error is the magnitude of the angle between the ground truth z direction and the estimated z direction (between the correct and estimated peg insertion angle). Each data point in the figure represents the average of 100 trials.

Here the standard particle filter (PF) was implemented in all 6-DOF by estimating both the position and the angle of a particle using the method we described for position estimation alone in Section II, so that each particle maintains a point estimate for angle rather than a Gaussian distribution over angle. For the standard PF, we included angular noise in the motion model (2), which corresponds to perturbing each particle's angle estimate at every time step of the inference algorithm, with standard deviation of 0.5° about each axis.

As shown in Fig. 6, the standard PF requires a much larger number of particles (by orders of magnitude) than our RBPF approach to achieve the same accuracy. This is because our approach reduces the state space that must be sampled by particles from 6 down to just 3 dimensions. Furthermore, the time required per particle for our RBPF approach is only slightly slower than that required per particle for the standard PF: The average computation time to update 6400 particles (with resampling) was 0.95 sec for our RBPF algorithm versus 0.91 sec for the standard PF, on a 2.66 GHz PC with unoptimized code. This shows that our method greatly reduces computational time as compared to the standard PF.

B. *Experiments with a Robot Arm*

We performed physical experiments using a Mitsubishi MELFA RV-6SL 6-axis robot arm (see the accompanying video). As shown in Fig. 7(a), a needle-like peg was attached to the robot's end effector along the z-direction in the robot coordinate system, and the object was placed on an approximately horizontal table. The diameter of the peg is 2.5 mm, and the hole tapers from 5 mm diameter down to 3mm diameter (over a vertical distance of 3 mm). Insertion will succeed if the estimation error of the position is within the clearance range and the angular error is small.

At the start of the experiments, we first measured the ground truth position of the hole by moving the robot arm manually. We then started trials by randomly selecting the initial position of the peg from the [-15, 15] mm range in (x, y) direction around the hole (just as in simulation). For each trial, the particle positions are initially sampled from that same distribution. In our current implementation, the

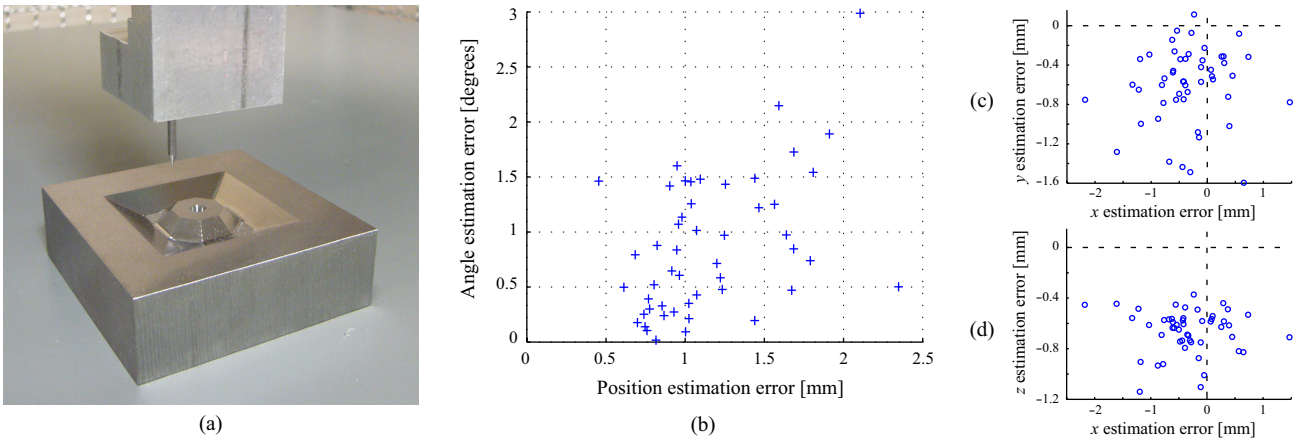


Fig. 7. (a) Experimental setup. (b) Error plot of the estimated positions and angles for 50 trials. The measures of position and angle errors are the same as in Fig. 6. To plot these results, we assume that the correct ground truth direction of insertion is parallel to the z-axis. In (c) and (d), estimation errors plotted in the (x, y) and (x, z) spaces show that the estimation is biased (especially in z direction) mainly because of a slight bending of the peg.

robot arm lifts the peg in the positive z-direction from its previous contact position, moves the peg to randomly chosen (x, y) coordinates (as in the simulation), then lowers the peg in the negative z direction until contact is detected. We used the robot arm's built-in impact detection function to detect contact, obtaining the contact position from the robot arm encoders. For these physical experiments, we assigned different measurement uncertainties to each feature of the mesh model, using the heuristics described in Section II-C. We used 6400 particles in our RBPF algorithm.

Figure 7(b) plots the error of the final estimated positions and angles in each of 50 trials, measured with respect to the ground truth pose of the hole in robot coordinates. All 50 trials resulted in a correct insertion (as can be seen in the figure, all the estimated positions were within the clearance range). Figures 7(c) and (d) detail the 3D position errors, illustrating a bias due to the fact that the peg we used had a tendency to bend in a particular direction. The number of contact measurements needed for convergence ranged from 6 to 15, with an average of 9.8.

IV. Conclusion

We have presented a factorization approach for localization in robotic assembly. Using Rao-Blackwellized particle filtering, we separate pose estimation into a particle-based estimator (which can easily represent multimodal distributions) for position, and a (unimodal) Gaussian estimator for angle conditioned on the particle position. This representation makes 6-DOF localization in peg-in-hole problems tractable by greatly reducing the number of particles required, resulting in a similar reduction in computational time.

We have also described a novel map representation to explicitly model the uncertainty of each feature in the mesh, without the need for a dense memory-intensive map representation. The explicit incorporation of uncertainty into our mesh model enables us to perform localization using a map that was directly obtained from a CAD model, without the need for a time-consuming preprocessing step (such as probing at every possible location).

As evidenced by both simulated and physical experimental results, our algorithm benefits from its sequential estimation approach, which makes it possible to efficiently solve the localization problem using a small number of measurements. In addition, the computational complexity of each sequential update is constant, independent of the total number of measurements. One limitation of our current approach is that we randomly select each probing position. In future work, we will modify the system to choose the next probing position based on the system's current estimate of the posterior distribution over pose, which will make the sequential estimation approach even more powerful. In addition, we plan to generalize our approach to peg-in-hole problems with more complex geometries.

References

[1] W. S. Newman, M. S. Branicky, H. A. Podgurski, S. Chhatpar, L. Huang, J. Swaminathan, and H. Zhang, "Force-responsive robotic assembly of transmission components," in *Proc. IEEE Int. Conf. Robotics Automation (ICRA)*, vol. 3, May 1999, pp. 2096–2102.

[2] S. R. Chhatpar and M. S. Branicky, "Search strategies for peg-in-hole assemblies with position uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems (IROS)*, vol. 3, Oct. 2001, pp. 1465–1470.

[3] ——, "Localization for robotic assemblies using probing and particle filtering," in *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics (AIM)*, July 2005, pp. 1379–1384.

[4] ——, "Particle filtering for localization in robotic assemblies with position uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems (IROS)*, Aug. 2005, pp. 3610–3617.

[5] U. Thomas, S. Molkenstruck, R. Iser, and F. M. Wahl, "Multi sensor fusion in robot assembly using particle filters," in *Proc. IEEE Int. Conf. Robotics Automation (ICRA)*, Apr. 2007, pp. 3837–3843.

[6] A. Doucet, N. de Freitas, K. P. Murphy, and S. J. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proc. 16th Conf. Uncertainty in Artificial Intelligence*, June 2000, pp. 176–183.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2006, ch. 13.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI National Conf. Artificial Intelligence*, 2002.

[9] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, July 2000.

[10] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Int. J. Robotics Research*, vol. 22, no. 12, pp. 985–1003, Dec. 2003.