Pose Estimation for Contact Manipulation with Manifold Particle Filters

Michael C. Koval Mehmet R. Dogar Nancy S. Pollard Siddhartha S. Srinivasa
{mkoval,mdogar,nsp,siddh}@cs.cmu.edu
The Robotics Institute, Carnegie Mellon University

***Abstract*— We investigate the problem of estimating the state of an object during manipulation. Contact sensors provide valuable information about the object state during actions which involve persistent contact, e.g. pushing. However, contact sensing is very discriminative by nature, and therefore the set of object states that contact a sensor constitutes a lower-dimensional manifold in the state space of the object. This causes stochastic state estimation methods, such as particle filters, to perform poorly when contact sensors are used. We propose a new algorithm, the *manifold particle filter*, which uses dual particles directly sampled from the contact manifold to avoid this problem. The algorithm adapts to the probability of contact by dynamically changing the number of dual particles sampled from the manifold. We compare our algorithm to the conventional particle filter through extensive experiments and we show that our algorithm is both faster and better at estimating the state. Unlike the conventional particle filter, our algorithm's performance improves with increasing sensor accuracy and the filter's update rate. We implement the algorithm on a real robot using commercially available tactile sensors to track the pose of a pushed object.**

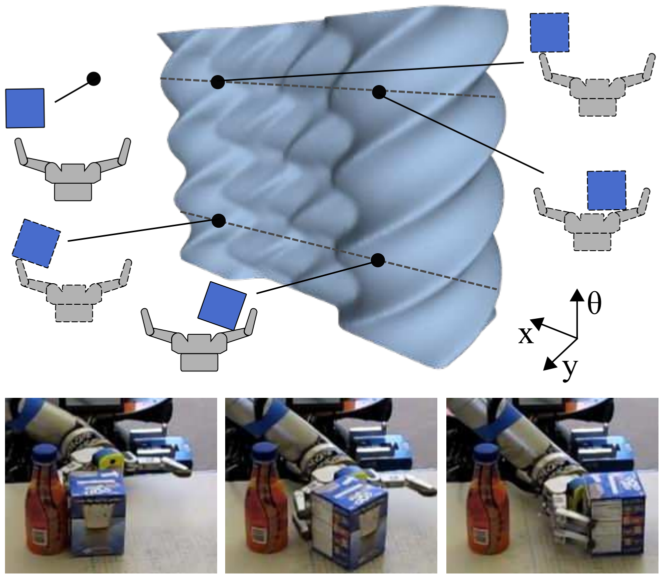

Fig. 1. Top: The contact states constitute a lower-dimensional manifold in the object's state space. Bottom: Example manipulation of a box with persistent contact.

I. Introduction

In this paper, we study *contact manipulation*, where a robot makes *persistent* contact with the object it is manipulating. Imagine reaching into a high cabinet to feel around for the salt shaker, or a robot push-grasping an object into its hand (Fig. 1-Bottom). The persistence of contact in these actions makes contact sensors, like strain gauges, force-torque sensors, and tactile pads a rich source of information about the object pose's during manipulation.

Prior research on pose estimation for contact manipulation has focused on using simple analytical motion models derived from the physics of pushing to build analytical state estimators that track the pose of the object from contact positions on the hand [1]. However, in reality there is much uncertainty both in the motion and observation models in the real world: physical parameters—like friction, mass, and pressure distributions—are hard to measure and variable. Contact sensors are noisy and potentially have low spatial resolution. This naturally leads to probabilistic formulations that lend themselves to Bayesian analysis, like the particle filter [2, 3].

However, we observed that the conventional particle filter [4] suffers from a startling problem when applied to contact manipulation: *they systematically perform worse as the sensor resolution increases.*

The problem arises because contact sensing is highly *discriminative* between contact and no-contact states: if a particle (i.e. a hypothesized object pose) is infinitesimally close to the robot hand but not touching it, then contact sensors will not discriminate between it and another particle which is much farther away from the hand. Topologically, the observation space of a contact sensor constitutes a lower dimensional manifold in the configuration space of the object (Fig. 1-Top). Particles sampled from the state space during contact have low probability of falling into the observation space. This results in *particle starvation* in the vicinity of the true state. Artificially introducing noise into the observation model sidesteps this problem, but comes at the expense of losing precious information [5].

We address this problem by deriving the manifold particle filter (MPF) for state estimation on multiple manifolds of possibly different dimensions. The gist of the algorithm is quite simple: we factorize belief into the marginal probability of being on a manifold and the probability of the current state conditioned on that manifold. We first sample a manifold, then a particle from that manifold.

Our factorization has two key consequences.

First, we can use a *different* sampling technique for each manifold. This allows us to avoid particle starvation on the contact manifold by using the *dual proposal distribution* [5] to sample directly from the observation model.

Second, the marginal *adaptively* and automatically adjusts the number of particles on each manifold. When there is no contact, most of the particles are concentrated in the ambient space. Once contact occurs, the marginal shifts the focus onto the contact manifold.

Choosing a manifold requires evaluating a marginal of the current belief. We subvert this race condition by exploiting the discriminative nature of the observation model to approximate the marginal.

The MPF is not only theoretically sound, but also practically useful. We demonstrate:

Better state estimation. Through extensive simulation experiments, we show that the MPF's state estimate becomes more accurate as we increase the resolution of the contact sensors. On the contrary, the conventional particle filter becomes less accurate because higher-resolution sensors further shrink the size of the observation space.

Faster performance. The MPF requires fewer particles and is orders of magnitude faster than the conventional particle filter. The increase in speed is critical as it enables state estimation to occur in real-time.

Real robot implementation. Finally, we contribute an implementation of our algorithm on a real robot, Andy, equipped with an array of tactile sensors that covers the interior surface of its hand. These experiments confirm that the MPF successfully tracks the pose of an object using commercially available tactile sensors [6].

We also discuss several limitations of our work. Key among them is scope: The MPF exploits a *discriminative* observation model that accurately identifies which manifold the state is evolving on. It is also designed for *persistent* contact and is unlikely to outperform a conventional particle filter when there is *intermittent* contact, like when tracking a billiard ball bouncing on a pool table.

II. Related Work

We borrow the concept of the *dual proposal distribution* from mobile robot localization literature [5]. Particle filters in this domain suffer from a similar particle deprivation problem if a robot uses very high-accuracy depth rangefinders or cameras. The dual proposal distribution solves the problem by sampling *dual particles* directly from the observation model. This is possible because the vision and depth sensors used on mobile robots provide high-accuracy readings independent of the true state. Conversely, contact sensors only provide accurate readings when the object is in contact with the sensor. Therefore, normal particles are necessary for periods of no contact and dual particles are ideal during contact. Our algorithm adaptively varies the number of dual particles sampled from the contact manifold according to the probability of contact.

Our focus is on state estimation during contact manipulation and, particularly, pushing actions. Pushing enables robots to perform a wide variety of tasks that are not possible through pick-and-place manipulation: pushing can move objects that are too large or heavy to be grasped [7], is effective at manipulating objects under uncertainty [8, 9], and can be used as *pre-grasp manipulation* to bring objects to configurations where they can be easily grasped [10, 11]. Additionally, pushing has been used to simultaneously move multiple objects [12]. Since pushing offers such a dramatic expansion of manipulation skills, there have been extensive research on the fundamental mechanics of pushing [13–16] and on the planning of pushing operations [16, 17]. Recently, there has been interest in generating push trajectories using sampling based planners [18, 19] and learning methods [20].

Most of the work described above employs pushing as an open-loop operation. Conversely, closed-loop actions that use contact sensors for feedback allows the robot to adapt in real-time and achieve success in more scenarios. One approach of using sensor feedback is to create a feedback controller that directly maps sensor readings to actions [21–23]. These controllers have been shown to be effective for specific tasks, such as locally refining the quality of a grasp [22], but do not generalize to general contact manipulation. Our method explicitly estimates the state of the object, which can then be used by a higher-level planning algorithm to achieve an arbitrary goal.

Another, separate, large body of work uses probabilistic methods for the tactile localization of *immovable* objects [24–26]. These systems produce a series of discrete move-until-touch actions that provide information about the object pose. In contrast, our system uses the persistent contact between the hand and the object during manipulation.

The most closely related prior work [2, 3] uses a conventional particle filter to track the pose of an object during persistent contact with environmental fixtures. In this paper we show that the MPF formulation leads to faster and more accurate state estimation when compared with the conventional formulation.

III. Conventional Particle Filter

In this section, we formalize the problem of state estimation for contact manipulation. We introduce the particle filter as a potential solution and provide insight into why its conventional implementation degenerates with contact sensors.

A. Pose Estimation for Contact Manipulation

Let $x \in S$ be the state of a dynamical system which evolves over time under actions $u \in A$ and emits observations $z \in Z$. The state estimation problem addresses the computation of the *belief state*, which is a probability distribution over the current state x_t

$$b(x_t) = p(x_t | z_{1:t}, u_{1:t}) \tag{1}$$

given the past history of actions $u_{1:t}$ and observations $z_{1:t}$.

In our problem, the state is the pose x of the manipulated object (Fig. 2(a)) and an action u (Fig. 2(b)) is a relative motion of the hand. During contact, the object moves according to the function $x_t = f_\Phi(x_{t-1}, u_t)$ that encodes the physics of the object's motion in response to the pushing action u_t. The parameter Φ includes any properties of the hand, object, or environment that may affect the object's motion. Unfortunately, the exact value of Φ is rarely known with certainty. Instead, we assume that there is a distribution $p(\Phi)$ over the parameters. This distribution, together with

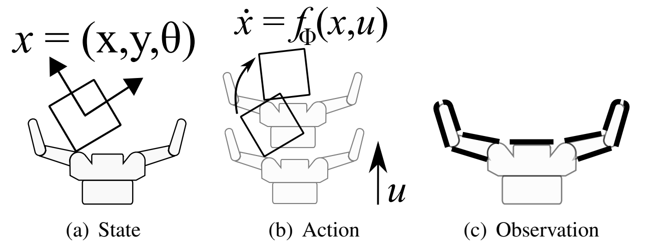

Fig. 2. Examples illustrating the (a) state, (b) action, and (c) observation for the state estimation for contact manipulation problem.

physics model f_Φ, defines the stochastic *transition model* $p(x_t|x_{t-1}, u_t)$.

We implement f_Φ using a quasistatic pushing simulation [12, 21]. The quasistatic assumption states that the object moves without accelerating; i.e. it comes to rest as soon as the robot stops exerting force. This approximation is accurate for the objects and end-effector velocities that are common during the manipulation tasks that we are concerned with. As a result, our formulation of the state consists of the object's pose, but not its velocity or acceleration. The parameter $\Phi = (\mu, c)$ consists of the hand-object coefficient of friction μ and the radius c of the object's pressure distribution.

Contact sensors provide observations z about where the object touches the hand during manipulation. We assume that the sensors accurately discriminate between contact and no-contact, but are noisy and potentially have low spatial resolution. For example, in Fig. 2(c), the nine contact sensors (drawn as bold line segments) cannot differentiate between different points of contact within their boundaries.

B. Bayes Filter

The *Bayes filter* is the most general algorithm for filtering a belief state given a sequence of actions and observations [4]. The Bayes filter recursively constructs $b(x_t)$ from $b(x_{t-1})$ using the update

$$b(x_t) = \eta \, p(z_t|x_t, u_t) \int_S p(x_t|x_{t-1}, u_t) b(x_{t-1}) dx_{t-1} \quad (2)$$

where η is a normalization factor. The terms $p(z_t|x_t, u_t)$ and $p(x_t|x_{t-1}, u_t)$ are, respectively, the observation and transition models defined above. The recursion is initialized with a prior belief $b(x_0)$ provided by task-specific knowledge or other sensors (e.g. an object recognition system).

The Bayesian update (Eq. (2)) is derived from the definition of the belief state (Eq. (1)) by applying Bayes' rule and assuming that the state satisfies the *Markov assumption* $x_t \perp (u_{1:t-1}, z_{1:t-1}) \,|\, x_{t-1}$. The Markov assumption holds when x is *complete*, i.e. is "memoryless," which is true of our formulation.

C. Conventional Particle Filter

There are a variety of techniques for representing the belief state and implementing the Bayes filter. In our case, the motion and observation models are highly non-linear and lack analytic derivatives. Even worse, the belief state is non-Gaussian and may be multi-modal. For example, the belief becomes bimodal in the trivial case where the hand sweeps through the center of the prior distribution without contacting the object. Together, these challenges preclude us from using the Kalman filter or its extended and unscented variants. Instead, we can track the belief state using a particle filter.

Algorithm 1 Particle Filter

```
Input: $X_{t-1}$, previous particles
Output: $X_t$, particles sampled from $b(x_t)$
1: $X_t \leftarrow \emptyset$
2: for all $x_{t-1}^{[i]} \in X_{t-1}$ do
3:     Sample $x_t^{[i]} \sim p(x_t|x_{t-1}^{[i]}, u_t)$
4:     $w_t^{[i]} \leftarrow p(z_t|x_t^{[i]}, u_t)$
5:     $X_t \leftarrow \{x_t^{[i]}\} \cup X_t$
6: $X_t \leftarrow \text{Resample}(X_t)$
```

The *particle filter* [4] is a non-parametric formulation of the Bayes filter that represents the belief state with a discrete set of samples. The samples $X_t = \{x_t^{[i]}\}_{i=1}^n$ are called *particles* and are distributed according to the belief state $x_t^{[i]} \sim b(x_t)$. The particle filter implements the Bayesian update (Eq. (2)) by recursively constructing X_t from X_{t-1} using *importance sampling*.

The conventional realization of the particle filter (CPF) is summarized in Alg. 1. The key insight behind this realization is that it is difficult to directly sample from the *target distribution* Eq. (2), but it is relatively easy to sample from the transition model. Therefore, we sample $x_t^{[i]}$ from the *proposal distribution* $\int_S p(x_t|x_{t-1}, u_t) b(x_{t-1}) dx_{t-1}$ (line 3) by forward-simulating X_{t-1} to X_t using the motion model. Next, we compute an *importance weight* $w_t^{[i]} = \eta \, p(z_t|x_t, u_t)$ for each forward-simulated particle (line 4).

The importance weights result from dividing the target distribution by the proposal distribution. The samples $x_t^{[i]}$, along with their importance weights $w_t^{[i]}$, are distributed according to the target distribution $b(x_t)$. Intuitively, the weighting step incorporates the observation model into the update by assigning higher weight to particles that are consistent with z_t.

Periodically, the particle filter *resamples* the set of weighted particles (line 6) with replacement to distribute X_t according to the desired posterior $b(x_t)$. Frequent resampling is necessary to prevent the weights from degenerating over time [4].

IV. Degeneracy of the Proposal Distribution

The particle filter, as described above, is agnostic to the observation model and has been applied to a variety of application domains [2, 27]. However, contact sensors are unique because they operate in two discrete states: contact and no contact. During periods of contact, observations are discriminative and the states for which $p(z_t|x_t, u_t)$ is peaked form a lower-dimensional manifold that includes the true state (Fig. 1). Conversely, during periods of no contact, $p(z_t|x_t, u_t)$ is nearly uniform and provides little useful information. This property makes contact sensors

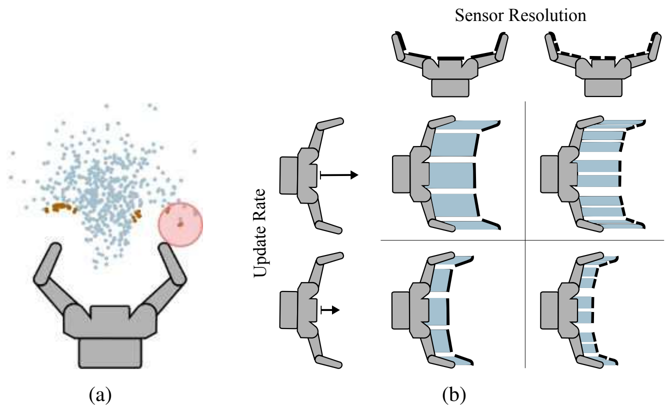

Fig. 3. (a) Illustration of particle deprivation. (b) The swept volume of contact sensors shrinks as the update rate or resolution of the sensors increases.

fundamentally different than the cameras and depth sensors, which have relatively smooth observation models, typically used in mobile robot localization [27].

In practice, particle filters update in discrete steps. The hand moves a non-zero distance during each step and the swept volume of the contact sensor gains full dimensionality. As such, particle filters are not completely ineffective at estimating the state. Instead, they require a large number of particles to increase the probability that some fall into the small swept volume of each sensor. We illustrate this in Fig. 3(a) for a hand pushing a cloud of 500 particles that represent the center of a 7 cm diameter bottle. Of the 500 particles (light blue), only 17 (dark orange) contact the hand during a 1 cm step. This causes particle deprivation and may result in there being no particles in the vicinity of the true state during periods of contact.

Surprisingly, this effect causes the particle filter to *perform worse as sensor resolution or the update rate increases*. We illustrate the reason in Fig. 3(b). As sensor resolution increases, the swept volume of each sensor becomes narrower. As the update rate increases, the distance traveled by the hand between updates decreases, and the swept volume becomes shorter. As a result, the particle filter requires a large number of particles to successfully track the state.

V. Manifold Particle Filter

We have shown that the conventional particle filter is poorly suited for contact sensors because the state evolves on a lower-dimensional manifold. In this section, we derive the manifold particle filter (MPF) to solve this problem and show how it can be applied to contact manipulation.

A. Formulation

Suppose the state space S is partitioned into m disjoint components $M = \{M_i\}_{i=1}^m$, where $M_1, \ldots, M_{m-1} \subseteq S$ are manifolds and $M_m = S \setminus \cup_{i=1}^{m-1} M_i$ is the remaining free space. We factor the belief space as

$$b(x_t) = \sum_{i=1}^{m} b(x_t|M_i)\,\Pr(x_t \in M_i) \tag{3}$$

Algorithm 2 Manifold Particle Filter

```
Input: $X_{t-1}$, previous particles
Input: $k$, number of particles to sample
Output: $\bar{X}_t$, particles sampled from $b(x_t)$
1: $\bar{X}_{M_i} \leftarrow \emptyset$ for $i = 1, \ldots, m$
2: for $1, \ldots, k$ do
3:     Sample $M_i \sim \Pr(x_t \in M_i)$
4:     if $M_i \neq M_m$ then
5:         Sample $\bar{x}_t^{[i]} \sim \frac{p(z_t|x_t,u_t)}{p(z_t|u_t)}$
6:         $\bar{w}_t^{[i]} = p(z_t|u_t) \cdot \text{EstimateDensity}(X_{t-1}, u_t, \bar{x}_t^{[i]})$
7:         $\bar{X}_{M_i} \leftarrow \{(\bar{x}_t^{[i]}, \bar{w}_t^{[i]})\} \cup \bar{X}_{M_i}$
8:     end if
9: $\bar{X}_{M_m} \leftarrow \text{ConventionalProposal}(X_{t-1}, u_t, z_t) \cap M_m$
10: $\bar{X}_t \leftarrow \text{Resample}(\sum_{i=1}^m P(x_t \in M_i)\bar{X}_{M_i})$
```

where $b(x_t|M_i)$ is the belief over M_i.

Our algorithm, summarized in Alg. 2, represents the belief using particles. For each particle, we first choose which manifold to sample from according to $M_i \sim \Pr(x_t \in M_i)$. Then, we sample the particle $\bar{x}_t^{[i]}$ from that manifold according to $\bar{x}_t^{[i]} \sim b(x_t|M_i)$ using an appropriate sampling technique.

Ideally, we would compute $\Pr(x_t \in M_i)$ as

$$\Pr(x_t \in M_i) = \int_{M_i} b(x_t)\, dx_t \tag{4}$$

by marginalizing over x_t. Unfortunately, doing so requires knowing $b(x_t)$, which is precisely the distribution that we are trying to estimate.

Instead, we must approximate $\Pr(x_t \in M_i)$. Using the previous belief state to compute $\int_{M_i} b(x_{t-1})\, dx_{t-1}$ might seem like a good approximation, but in fact it does *not* work. To see why, consider an update step at which the filter receives an observation which suggests $x_t \in M_i$ for the first time. If we approximate $b(x_t) \approx b(x_{t-1})$, then $\Pr(x_t \in M_i)$ will remain low and we will not sample from M_i. This problem persists even if we receive repeated observations that suggest $x_t \in M_i$ because we will always be using the belief from the previous step.

Hence, we approximate (4) using only the most recent observation

$$\Pr(x_t \in M_i) \approx \int_{M_i} \frac{p(z_t|x_t, u_t)}{p(z_t|u_t)}\, dx_t, \tag{5}$$

where $p(z_t|u_t) = \int_{x_t} p(z_t|x_t, u_t)\, dx_t$ is the prior probability of receiving observation z_t. This is not, in general, equivalent to Eq. (4). However, Eq. (5) is a good approximation when $p(z_t|x_t, u_t)$ accurately discriminates between the manifolds. In the limit, when observations perfectly discriminate between manifolds, $p(z_t|x_t, u_t)$ becomes binary and this technique directly samples from the target distribution. This approximation performs well for our purposes since contact sensors accurately discriminate between contact and no-contact.

Finally, we sample a particle $\bar{x}_t^{[i]}$ according to the belief distribution over the chosen manifold $b(x_t|M_i)$. Our key

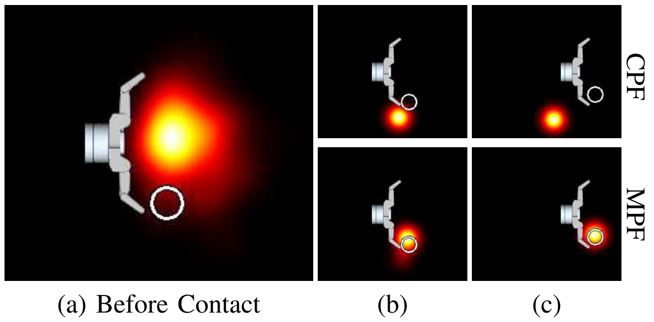

Fig. 4. Belief state estimated by the CPF and MPF for BarrettHand pushing a cylindrical bottle. (a) Particles that are in the swept volume of the hand are eliminated through no-contact observations. (b) Contact occurs. (c) The CPF has converged to an inaccurate state estimate, whereas the MPF successfully tracks the object.

insight is that we can apply a different sampling technique for each M_i that is specifically designed to take advantage of the structure of the manifold. For the manifolds $\{M_i : i < m\}$, we sample from the dual proposal distribution as described below. In the case of the free space M_m, we sample $\bar{x}_t^{[i]}$ with the conventional technique and reject any $\bar{x}_t^{[i]} \in \cup_{i=1}^{m-1} M_i$. This rejection sampling step is necessary to avoid biasing the estimate of $b(x_t)$ towards the manifolds.

B. Dual Proposal Distribution

If we choose to sample from M_i for $i < m$, then the conventional sampling technique will be ineffective. Instead, we sample from the observation model and compute the importance weights using the motion model. First, we sample from the *dual proposal distribution* [5]

$$\bar{x}_t^{[i]} \sim \frac{p(z_t|x_t, u_t)}{p(z_t|u_t)}, \tag{6}$$

which assign high probability to particles that are consistent with z_t (line 5). Next, we compute the importance weight

$$\bar{w}_t^{[i]} = \eta\, p(z_t|u_t) \int_S p(\bar{x}_t^{[i]}|x_{t-1}, u_t) b(x_{t-1}) dx_{t-1} \tag{7}$$

using the motion model (line 6). Just as before, the importance weight is found by dividing the target distribution (Eq. (2)) by the proposal distribution (Eq. (6)) [5].

The conventional proposal distribution forward-predicts using the motion model and computes importance weights using the observation model. Conversely, the dual proposal distribution samples particles from the observation model and weights them by how well they agree with the prediction of the motion model. This is the logical inverse of the conventional proposal distribution and has complementary strengths and weaknesses: the dual distribution performs best with a discriminative observation model that is tightly peaked around the true state [5].

C. Mixture Proposal Distribution

Just as how the conventional proposal distribution performs poorly with accurate sensors, the dual proposal distribution tends to be disrupted by observation noise. The MPF uses the dual proposal distribution to sample from the manifolds and, as result, shares the same weakness.

We use a *mixture proposal distribution* [5] to mitigate this effect by combining both sampling techniques. Instead of sampling all of the particles from the MPF, we sample n particles from the CPF and d particles from the MPF. We then combine the two sets of particles with the weighted sum $(1-\phi)\tilde{X}_t = X_t + \phi\bar{X}_t$ before resampling. The *mixing rate* $0 \le \phi \le 1$ is a parameter that allows the algorithm to smoothly transition from the CPF ($\phi = 0$) to the MPF ($\phi = 1$).

Intuitively, $d = |\bar{X}_t|$ is the number of particles necessary to cover manifolds and $n = |X_t|$ is the number of additional particles necessary to represent $b(x_t)$ over the entire state space.

D. Manifold Particle Filter for Contact Manipulation

In contact manipulation, $m = 2$ and there is a single *contact manifold* M_1 that contains the set of all states that are in non-penetrating contact with the hand. The free space M_2 contains the remaining states that are not in contact with the hand. Using the manifold particle filter for contact manipulation requires two extra capabilities: (1) sampling from the dual proposal distribution and (2) weighting with the transition model.

First, we must sample from the dual proposal distribution. In the general case, where the object and hand have arbitrary three-dimensional geometry, we represent M_1 as a pre-computed set of discrete samples that are in non-penetrating contact with the hand. This representation is tractable because M_1 is a two-dimensional manifold that can be densely covered using a relatively small number of samples. We sample from Eq. (6) by assigning each sample a weight equal to likelihood $p(z_t|x_t, u_t)$. Then, we sample $\bar{x}_t^{[i]}$ with replacement from this weighted set.

It is often possible to use a more compact representation of M_1 if there is special structure in the observation model or object-hand geometry. For example, we can pre-compute a set of samples consistent with each observation if the sensors are binary. In another case, we can approximate M_1 as a polyhedron if the object's and hand's geometry can be decomposed into a set of extruded convex polygons [28].

Second, we must compute the dual importance weights given in Eq. (7). Importance weights are defined up to a scale factor, so we drop the constant term $\eta\, p(z_t|u_t)$. The remaining term $\int_S p(x_t|x_{t-1}, u_t) b(x_{t-1}) dx_{t-1}$ is equal to the belief at time t before incorporating observation z_t. We evaluate this term by forward-simulating the previous set of particles X_{t-1} to time t using $p(x_t|x_{t-1}, u_t)$ and approximating the value of this distribution at $\bar{x}_t^{[i]}$ using kernel density estimation. Kernel density estimation is a non-parametric technique for estimating a continuous probability distribution that generated a set of samples.

Note that forward-simulating X_{t-1} to time t is necessary for both the CPF and MPF. As a result, the MPF incurs minimal computational overhead over the CPF. Additionally

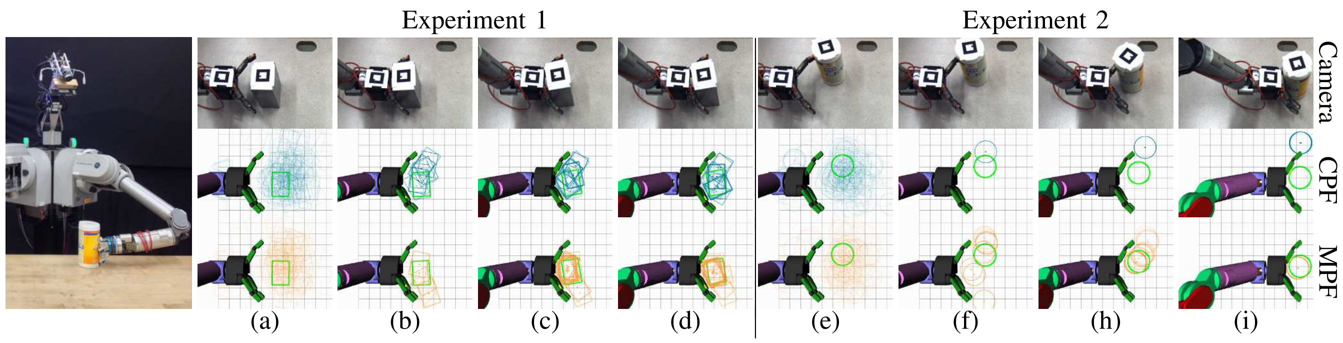

Fig. 5. Andy pushing a box (a)–(d) and cylinder (e)–(i) across the table. The top row shows a video of the experiment from an overhead camera. The bottom two rows show the belief state estimated by the CPF (middle, dark blue) and MPF (bottom, light orange) as a cloud of particles. Ground truth is shown as a thick green outline. In both cases, the belief state estimated by the MPF is more accurate than that estimated by the CPF.

it is possible to re-use this computation when the CPF and MPF are combined in a mixture proposal distribution.

Figure 4 shows a comparison of the CPF and MPF with simulated BarrettHand pushing a cylindrical bottle from left-to-right. Each link is equipped with a binary contact sensor and the $b(x_t)$ is shown as a colormap. The CPF (Fig. 4-Top) fails to track the state because there are no particles in the vicinity of the contact observation. The MPF (Fig. 4-Bottom) avoids this problem by sampling particles from the dual proposal distribution.

VI. Real-robot Experiments

We evaluated the estimators on Andy, a bimanual manipulator developed for the DARPA ARM-S competition. Andy used a Barrett WAM arm equipped with the i-HY [29] end-effector to push an object across a table. The i-HY's palm (48 tactels), interior of the proximal links (12 tactels each), interior of the distal links (6 tactels each), and fingertips (2 tactels each) were equipped with an array of tactile sensors [6] based on MEMS barometer technology. The tactels were grouped into 39 vertical stripes to compensate for dead tactels and to simplify the observation model.

Figure 5 shows the two representative runs of the state estimator on Andy. The ground-truth pose of the object was tracked by an overhead camera using a visual fiducial. Both filters were run with 250 particles, with $n = 250$ for the CPF and $n = 225$, $d = 25$, $\phi = 0.1$ for the MPF, and were updated after each 5 mm of end-effector motion. With the speed of the arm, this corresponded to an update rate of approximately 5–15 Hz.

In Experiment 1, Andy pushed a metal box that made initial contact with the right proximal link (b) and rolled into the palm (c). The CPF did not have any particles in the small observation space and, thus, failed to track the box as it rolled into the palm (d). The MPF successfully tracked the box by sampling particles that agree with the observation. Note that the MPF was able to exploit the observation of simultaneous contact on the palm and distal link to correctly infer the orientation of the box.

In Experiment 2, Andy pushed a cylindrical container that made initial contact with its left fingertip (e). The cylinder rolled down the distal (f) and proximal (h) links to finally settle in the palm (i). Both the CPF and MPF made use of the initial contact observation to localize the container near the robot's left fingertip. However, the CPF's few remaining particles incorrectly rolled off of the fingertip and outside the hand. The MPF avoided particle starvation near the true state and was able to successfully track the container for the duration of contact.

Please see the accompanying video for more experiments.

VII. Simulation Experiments

We have qualitatively shown that the MPF outperforms the CPF when used on a real robot. In this section, we verify those properties in simulation and show that these differences are statistically significant.

We will verify the following hypotheses:

H1 *The error of the CPF and MPF are similar before contact.*
H2 *The MPF has lower error than the CPF after contact.*
H3 *Improving resolution increases the error of the CPF.*
H4 *Improving resolution reduces the error of the MPF.*

Additionally, we demonstrate that the MPF achieves lower error than the CPF when provided with a fixed computational budget.

A. Experiment Design

We implemented both filters in the OpenRAVE [30] simulation environment and evaluated the algorithms with a simulated BarrettHand. The hand was equipped with binary contact sensors of a fixed size, ranging from 1 cm to 5 cm, uniformly distributed across the front surface of the hand. Observations were randomly perturbed with a 5% probability to simulate sensor error.

In each trial, the hand moved in a straight line at a constant speed of 5 cm/s for 10 s and received observations at 10 Hz. The initial pose of the object was drawn from a Gaussian prior distribution with a randomly chosen mean and a standard deviation of 10 cm. The object's orientation was uniformly sampled within 0.4 rad of its true value. Contact was simulated using three-dimensional mesh collision checks. For efficiency, the pushing simulation was implemented using an approximate, cached collision checker.

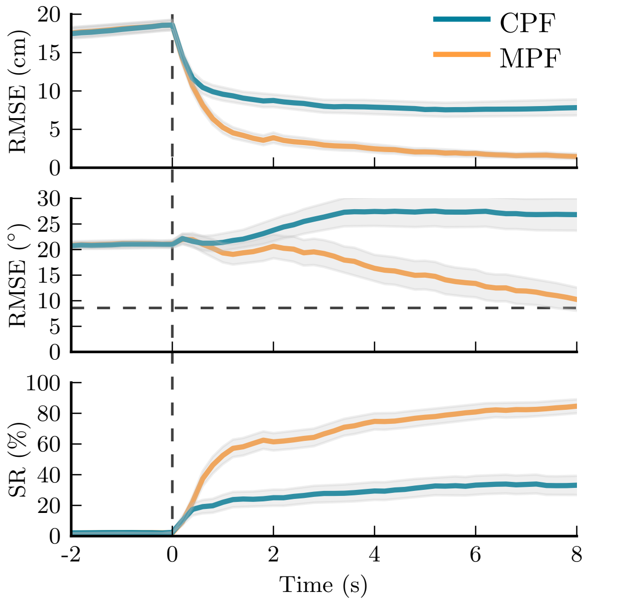

Fig. 6. RMSE and SR of the CPF ($n = 500$) and MPF ($n = 450, d = 50, \phi = 0.1$) plotted over time along with their 95% confidence intervals (gray shaded regions). Contact occurs at $t = 0$ and is denoted by the vertical dashed line. The horizontal dashed line in the middle figure marks the angular resolution of our approximate physics model.

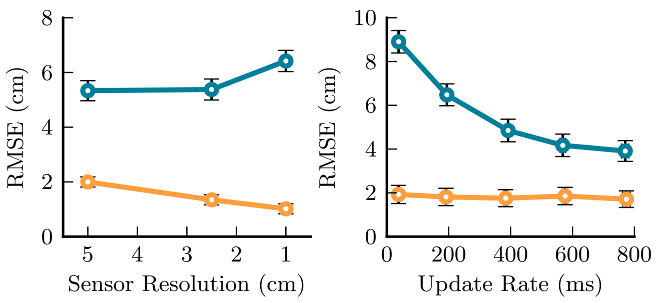

Fig. 7. (a) Effect of sensor resolution on RMSE. The MPF monotonically decreases in RMSE as resolution increases. Conversely, the accuracy of the CPF degrades with high resolution sensors. (b) Real-time performance of the filters. The MPF achieves lower RMSE than the CPF given a fixed computational budget. In both cases, the error bars denote a 95% confidence interval.

B. Metrics

We will compare the performance of the algorithms with the following metrics:

- *Root-mean-square error* (RMSE) between the particles and the true state.
- *Success rate* (SR), the probability mass of particles within 2 cm of the true state
- *Update rate* (UR), the time required to perform a single action-observation update

These three metrics capture different properties of the filters. RMSE is the traditional error metric used in the localization literature [5, 27] and measures how closely the particles track the true state. Similarly, SR acts as a proxy of how well the filter would perform while performing a task that succeeds under bounded uncertainty. UR directly measures the real-time performance of each filter and acts as a proxy for its computational complexity.

C. Estimation Error

We tracked the RMSE (Fig. 6-Top, Fig. 6-Middle) and SR (Fig. 6-Bottom) over 250 experiments. In all cases, the x-axes of the plots were aligned such that contact occurs at $t = 0$. The hand was outfitted with 1 cm resolution sensors and pushed a 11.5 cm $\times$ 1.7 cm $\times$ 17.2 cm rectangular box.

Before contact, both filters behave similarly and there not a significant difference in RMSE ($t(4066) = 0.20$, $p = .844$). After contact, the MPF achieves 6.4 cm less RMSE than the CPF ($t(15630) = 87.30$, $p < .001$). These results confirm hypotheses H1 and H2: the MPF achieves significantly lower error than the CPF. Note that position and orientation errors are coupled: once the transients of initial contact fade, further improvement in positional accuracy comes from better knowledge of the object's orientation.

D. Sensor Resolution

We additionally evaluated the effect of sensor resolution on the RMSE error of both filters over 250 experiments. To simplify the analysis, we pushed a 3.5 cm radius cylindrical bottle in this set of experiments. Figure 7(a) shows the average RMSE error during contact as the sensor resolution was varied from cell sizes of 1 cm to 5 cm. Smaller sizes correspond to a higher sensing resolution and, ideally, lower error. As expected, the MPF outperforms the CPF at all sensor resolutions.

In both cases, an ANOVA showed that sensor resolution was a significant main effect for both the particle filter ($F(3, 17900) = 316.83$, $p < .001$) and the MPF ($F(3, 17900) = 105.72$, $p < .001$) with a negative correlation for the CPF and a positive correlation for the MPF. This confirms hypotheses H3 and H4. Unlike the MPF, the performance of the CPF degrades as the sensor resolution increases. As described in Section IV, this occurs because it becomes progressively less likely to sample particle with high $p(z_t|x_t, u_t)$ during periods of contact. The MPF does not suffer from this problem and improves with sensor resolution.

E. Realtime Performance

These filters are intended to be used for real-time feedback. Therefore, it is important that the filters achieve acceptable error at real-time update rates. Figure 7(b) shows RMSE during contact as a function of update rate using the same parameters as in Section VII-D. The update rate was indirectly manipulated by varying the number of particles from 100 to 2000 and measuring the time required for each filter to execute a single update step. All measurements were taken using a single core of a 2.53 GHz Intel Xeon processor and were averaged over 250 experiments.

The MPF achieved acceptable accuracy (e.g. < 2.5 cm RMSE) with several hundred particles and URs of 20+ Hz.

Conversely, the CPF only was able to achieve 4 cm RMSE with 2000 particles and an UR approximately 1.25 Hz. These results confirm that CPF requires a huge number of particles to accurately track the state and is ill-suited for real-time use. Conversely, the MPF is fast enough to be used as real-time feedback.

VIII. Discussion and Future Work

We made several simplifying assumptions to find a real-time solution to the state estimation problem during contact manipulation.

First, we implicitly assume that the hand can only contact the object that we are manipulating. This may not be possible in highly cluttered environments where we must contact multiple objects to achieve the desired task [12]. In future work, we hope to explore methods of generalizing the MPF to environments with multiple—both static and movable—objects. We believe it is possible to do so by factoring the belief state (e.g. through Rao-Blackwellization) to avoid requiring exponentially more particles.

Second, our implementation assumes $X = SE(2)$. This is sufficient for planar manipulation, but the state space becomes larger if objects are articulated, can topple, or can roll. In particular, sampling $\bar{x}_t^{[i]}$ according to Eq. (6) may become challenging. For most observation models, this is done by representing M_1 with a finite set of samples. However, the number of samples required to densely represent M_1 grows exponentially with the dimension of the manifold. We plan to address this by replacing the importance sampling step with a more efficient Markov chain Monte Carlo sampling technique.

Third, we can improve our algorithm by recognizing that tracking the state on different manifolds may have very different computational costs. They are different during pushing: tracking particles on the contact manifold requires making a physics-based motion prediction, whereas tracking particles in the free space is almost free since those particles do not move. Similarly, different manifolds may require different number of particles. For example, fewer particles may sufficient to track the state on lower dimensional manifolds. The performance of our algorithm can be improved by taking into account such manifold characteristics.

In summary, we have shown that contact sensors fundamentally differ from the vision and depth sensors traditionally used for state estimation. Using this insight, we formulated the MPF and demonstrated that contact sensors provide useful information during manipulation both in simulation and on a real robot. We believe that the state estimation technique described in this paper could be used to create robust closed-loop actions that use real-time contact feedback to deal with high amounts of uncertainty.

ACKNOWLEDGMENT

This material is based on work supported by the DARPA Autonomous Robotic Manipulation Software Track (ARM-S) program. We would also like to thank the members of the Personal Robotics Lab for their helpful input.

References

[1] Y. bin Jia and M. Erdmann, “Pose and motion from contact,” *IJRR*, 1999.

[2] L. Zhang and J. C. Trinkle, “The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing,” in *IEEE ICRA*, 2012.

[3] L. Zhang, S. Lyu, and J. Trinkle, “A dynamic bayesian approach to simultaneous estimation and filtering in grasp acquisition,” in *IEEE International Conference on Robotics and Automation*, 2013.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[5] S. Thrun, D. Fox, and W. Burgard, “Monte Carlo localization with mixture proposal distribution,” in *AAAI*, 2000.

[6] Y. Tenzer, L. P. Jentoft, and R. D. Howe, “Inexpensive and easily customized tactile array sensors using MEMS barometers chips.” IEEE, 2012.

[7] M. Dogar and S. Srinivasa, “A framework for push-grasping in clutter,” in *RSS*, 2011.

[8] R. C. Brost, “Automatic grasp planning in the presence of uncertainty,” *IJRR*, vol. 7, no. 1, pp. 3–17, 1988.

[9] M. Dogar and S. Srinivasa, “Push-grasping with dexterous hands: Mechanics and a method,” in *IEEE/RSJ IROS*, 2010.

[10] L. Chang, S. Srinivasa, and N. Pollard, “Planning pre-grasp manipulation for transport tasks,” in *IEEE ICRA*, 2010.

[11] D. Kappler, L. Y. Chang, M. Przybylski, N. Pollard, T. Asfour, and R. Dillmann, “Representation of Pre-Grasp Strategies for Object Manipulation,” in *IEEE-RAS Humanoids*, December 2010.

[12] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, “Physics-based grasp planning through clutter,” in *RSS*, 2012.

[13] M. T. Mason, “Mechanics and Planning of Manipulator Pushing Operations,” *IJRR*, vol. 5, no. 3, pp. 53–71, 1986.

[14] K. Lynch and M. T. Mason, “Pulling by pushing, slip with infinite friction, and perfectly rough surfaces,” in *IJRR*, vol. 14, no. 1. Cambridge, MA: MIT Press Journals, April 1995, pp. 174–183.

[15] R. D. Howe and M. R. Cutkosky, “Practical Force-Motion Models for Sliding Manipulation,” *IJRR*, vol. 15, no. 6, pp. 557–572, 1996.

[16] K. M. Lynch and M. T. Mason, “Stable Pushing: Mechanics, Controllability, and Planning,” *IJRR*, vol. 15, no. 6, pp. 533–556, 1996.

[17] S. Akella and M. T. Mason, “Posing polygonal objects in the plane by pushing,” *IJRR*, vol. 17, no. 1, pp. 70–88, January 1998.

[18] M. Lau, J. Mitani, and T. Igarashi, “Automatic learning of pushing strategy for delivery of irregular-shaped objects,” in *IEEE ICRA*, 2011.

[19] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, “Push planning for object placement on cluttered table surfaces,” in *IEEE/RSJ IROS*, 2011.

[20] C. Zito, Stolkin, M. R. Kopicki, and J. L. Wyatt, “Two-level RRT planning for robotic push manipulation,” in *IEEE/RSJ IROS*, 2012.

[21] K. M. Lynch, H. Maekawa, and K. Tanie, “Manipulation and active sensing by pushing using tactile feedback,” in *IEEE/RSJ IROS*, 1992.

[22] R. Platt, A. H. Fagg, and R. A. Grupen, “Null-space grasp control: theory and experiments,” *IEEE Trans. on Robotics*, 2010.

[23] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, “Online movement adaptation based on previous sensor experiences,” in *IEEE/RSJ IROS*, 2011, pp. 365–371.

[24] K. Hsiao, “Relatively robust grasping,” Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[25] S. Javdani, M. Klingensmith, J. A. Bagnell, N. S. Pollard, and S. S. Srinivasa, “Efficient touch based localization through submodularity,” in *IEEE ICRA*, 2013.

[26] A. Petrovskaya and O. Khatib, “Global localization of objects via touch,” *IEEE Trans. on Robotics*, 2011.

[27] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” in *IJCAI*, 2003.

[28] T. Lozano-Pèrez, “Spatial planning: A configuration space approach,” *Computers, IEEE Transactions on*, 1983.

[29] L. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, “A compliant, underactuated hand for robust manipulation,” *CoRR*, 2013.

[30] R. Diankov and J. Kuffner, “OpenRAVE: A planning architecture for autonomous robotics,” Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, 2008.