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 which 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 particle filter through extensive experiments and we show that our algorithm is both faster and better at estimating the state. Our algorithm's performance improves with increasing sensor accuracy and the filter's update rate. We implement the algorithm on a real robot using a force/torque sensor and strain gauges to track the pose of a pushed object.**

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 makes contact sensors, like strain gauges, force-torque sensors, and tactile pads, a rich source of information for object pose estimation 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 to track the pose of the object from contact positions on the hand [1].

Unfortunately, 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, and sensors are noisy. This naturally leads to probabilistic methods, like particle filters, for object pose estimation [2], with stochastic motion and observation models.

However, we observed that conventional particle filters [2, 3] suffer from a startling problem in contact manipulation: *they systematically perform worse as sensor resolution or sensor update rate 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 contact sensors constitute a lower dimensional manifold in the state space of the pose of the object (Fig. 1-Top). In practice, particles sampled from the state space during contact will have very low probability of falling into the observation space which will result in particle deprivation in the vicinity of the correct state. This results in *particle starvation*. Artificially introducing noise into the observation model sidesteps this problem but comes at the expense of losing precious information.

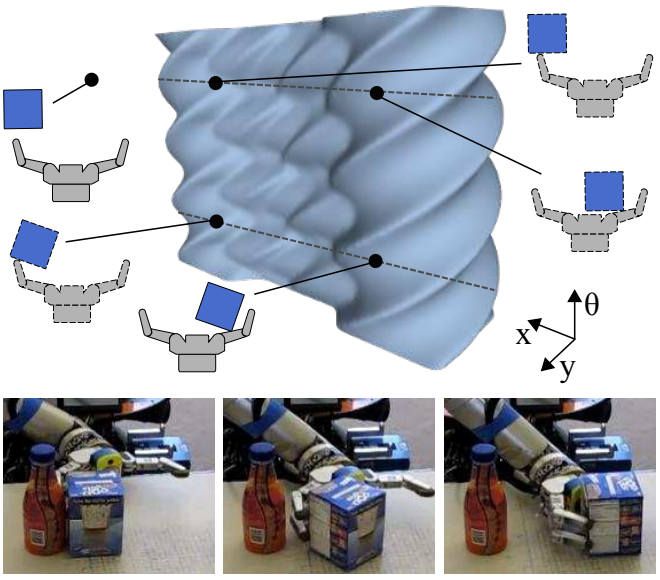


Fig. 1. Top: The contact states constitute a lower-dimensional manifold in the object's state space. Bottom: Example manipulation of a box with persistent contact.

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, we sample a particle from that manifold.

Our factorization has two key consequences.

First, we can now 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* [4], which samples from the observation model and computes importance weights using the motion model.

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

Computing the belief requires the marginal which requires 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 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 regular particle filter becomes less accurate because higher-resolution sensors further shrink the number of particles that agree with an observation.

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.

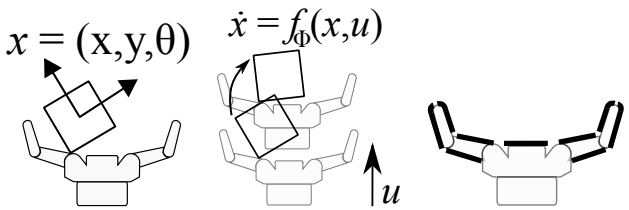
Real robot implementation. Finally, we contribute an implementation of our algorithm on a real robot, HERB [5], which uses a force-torque sensor and strain gauges to detect contact on different parts of its hand and estimates the pose of a pushed object.

We also discuss several limitations of our work. Key among them is scope. The MPF is designed for *persistent* contact, where contact evolves on manifolds. It is unlikely to outperform a conventional PF when there is very *intermittent* contact, like tracking a billiard ball bouncing on a table. The MPF is also designed to exploit a *discriminative* observation model. It is unlikely to outperform a non-adaptive dual particle filter when the observation model is not discriminative, like a mobile robot outfitted with very accurate LIDAR.

II. Related Work

We borrow the concept of dual particles from mobile robot localization literature [4]. Particle filters in this domain suffer from a similar particle deprivation problem if a robot uses very high-accuracy depth rangefinders or cameras. These systems, however, use a mixture proposal distribution with a fixed ratio of dual particles to normal particles. This is possible because vision and depth sensors provide high-accuracy readings independent of the actual state. Conversely, contact sensors provide accurate readings only when the object is in contact with the robot hand. Therefore, normal particles are necessary for periods of no contact and dual particles are ideal during contact. Our algorithm achieves this by varying the number of dual particles according to the contact probability.

Our focus is on state estimation during contact manipulation and particularly during pushing manipulation. 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 [6], is effective at manipulating objects under uncertainty [7, 8], and can be used as *pre-grasp manipulation* to bring objects to configurations where they can be easily grasped [9, 10]. Additionally, pushing has been used to simultaneously move multiple objects [11] and singulate an object from a pile [12, 13]. Since pushing offers such a dramatic expansion of manipulation skills, there have been extensive research on the fundamental mechanics of pushing [14–17] and on the planning of pushing operations [17, 18]. Recently, there has been interest in generating push trajectories using sampling based planners [19, 20] and learning methods [21].



(a) State (b) Action (c) Observation

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

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 [22–24]. These controllers have been shown to be effective for specific tasks, such as locally refining the quality of a grasp [23], 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.

Recent work uses probabilistic methods for the tactile localization of *immovable* objects [25–27]. These systems produce a number of distinct 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.

III. Conventional Particle Filter

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

A. Pose Estimation for Contact Manipulation

Let x be the state of a dynamical system which evolves under actions u and provides observations z. The state estimation problem addresses the computation of the *belief* which is a probability distribution over the state space

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

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

In our problem, the state is the pose $x \in SE(2)$ of the manipulated object (Fig. 2(a)). Actions are motions of the

hand, given by the velocity twist $u \in se(2)$. During contact, the object moves with a velocity $f_\Phi(x, u)$ where the function f encodes the physics of the object motion in response to pushing actions (Fig. 2(b)). The parameter Φ includes environmental properties such as the coefficient of friction between the object and the underlying surface, the coefficient of friction between the robot hand and the object, the mass distribution of the object, and the pressure distribution of the object.

Contact sensors provide observations z about where the object touches the hand during manipulation. In Fig. 2(c) we illustrate an example distribution of nine contact sensors (shown as bold line segments) on the hand. We assume that the sensors are not only noisy but also potentially have low spatial resolution, much like contact sensors in real life. Hence, we assume that a sensor cannot discriminate between different contact points in its boundary.

B. Bayes Filter

The *Bayes filter* is the most general algorithm of filtering a belief state given a sequence of actions and observations. The Bayes filter update works recursively:

$$bel(x_t) = \eta\, p(z_t|x_t, u_t) \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{2}$$

where η is a normalization factor. $p(z_t|x_t, u_t)$ is called the *observation model* and denotes the probability of making a certain observation given the current state and action. $p(x_t|x_{t-1}, u_t)$ is called the *motion model* and denotes the probability of reaching state x_t from state x_{t-1} with action u_t. The Bayes filter update Eq. (2) can be derived from the definition of belief Eq. (1) by applying the Bayes' rule and assuming that the state is *complete*, i.e. it satisfies the *Markov assumption* $x_t \perp (u_{1:t-1}, z_{1:t-1}) \,|\, x_{t-1}$. The Markov assumption states that x_t is independent of all previous actions and observations given x_{t-1}.

In our case the observation model $p(z_t|x_t, u_t)$ strongly discriminates between states which result in contact and no contact. This is true by definition for contact sensors such as tactile arrays, force/torque sensors, and strain gauges.

We build the motion model by computing f_Φ using a quasi-static simulation of pushing [11, 22]. Instead of assuming exact values for the parameters Φ, we assume that they are drawn from a known prior distribution and are stationary. By doing so, we can predict the effect of our actions on the state using the stochastic motion model $p(x_t|x_{t-1}, u_t)$.

Since the quasi-static analysis does not allow accelerations, our formulation of the state x as the pose—and not the velocity or acceleration—of the object satisfies the Markov assumption.

The Bayes filter is recursive and, therefore, requires an initial belief $bel(x_0)$. In manipulation, we can initialize the belief using task-specific knowledge or using other sensors, e.g. cameras.

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 common case where the finger 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 $bel(x_t)$

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: $X_t \leftarrow \text{Resample}(X_t)$

C. Particle Filter

The *particle filter* [3] 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 $x_t^{[i]} \sim bel(x_t)$. When the particle filter receives a new action and observation, it recursively constructs a new set of particles X_t from the previous set of particles X_{t-1} by applying the motion and observation models.

Algorithm 1 shows an overview of the particle filter. The particle filter samples $x_t^{[i]}$ from the *proposal distribution* $x_t^{[i]} \sim \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$ (line 3) by forward-simulating X_{t-1} to X_t using the motion model. Next, the algorithm uses the observation model to compute an *importance weight* $w_t^{[i]} = \eta\, p(z_t|x_t, u_t)$ for each forward-simulated particle (line 4). The weighting step assigns higher probability to particles that are consistent with our current observation; i.e. agree with our contact sensors. Finally, the algorithm *resamples* the set of weighted particles (line 6) to distribute X_t according to the desired posterior $bel(x_t)$.

IV. Degeneracy of the Proposal Distribution

The particle filter described above is agnostic to the observation model and has been applied to a variety of application domains [2, 28]. 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 around 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 fundamentally different than the cameras and depth sensors—sensors with relatively smooth observation models—typically used in mobile robot localization [28].

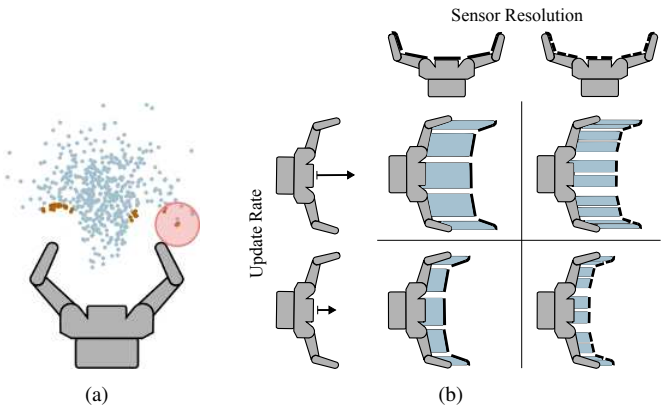


Fig. 3. (a) Illustration of particle deprivation. (b) The swept volume of contact sensors shrinks as the update rate or resolution of the sensors increases.

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 do not completely fail 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 bottle. Of the 500 particles (light blue), only 17 (dark orange) contact the hand during a 1 cm step. This causes particle deprivation around the true state during periods of contact: the proposal distribution poorly approximates the target distribution and many particles are wasted in low probability regions of the state space.

Surprisingly, this 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 MPF to solve this problem and show how it can be applied to contact manipulation.

A. Formulation

Suppose we divide a state space S into m disjoint components $M = \{M_i\}_1^m$, where $M_1, ..., M_{m-1}$ are manifolds and $M_m = S - \cup_{i=1}^{m-1} M_i$ is the remaining free space. Then we can write the belief in this space as:

$$bel(x_t) = \sum_i^m bel(x_t|M_i)\Pr(x_t \in M_i) \tag{3}$$

Our algorithm approximates this belief using particles. For each particle we first choose which manifold to sample from according to $\Pr(x_t \in M_i)$. Then, we sample the particle from that manifold according to $bel(x_t|M_i)$.

Algorithm 2 Manifold Particle Filter

Input: X_{t-1}, previous particles
Output: X_t, particles sampled from $bel(x_t)$

$X_t \leftarrow \emptyset$
for $i = 1, \ldots, |X_{t-1}|$ **do**
  Sample $M_i \sim \Pr(x_t \in M_i)$
  if $M_i \neq M_m$ **then**
    Sample $x_t^{[i]} \sim \frac{p(z_t|x_t,u_t)}{\pi(z_t|u_t)}$
    $w_t^{[i]} = \pi(z_t|u_t) \cdot \text{EstimateDensity}(X_{t-1}, x_t^{[i]})$
  else
    $x_t^{[i]}, w_t^{[i]} \leftarrow \text{ConventionalSampling}(X_{t-1}, u_t, z_t)$
  end if
  $X_t \leftarrow \{x_t^{[i]}\} \cup X_t$
$X_t \leftarrow \text{Resample}(X_t)$

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

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

Unfortunately, computing this integral requires knowing $bel(x_t)$, which is precisely the distribution that we are trying to estimate.

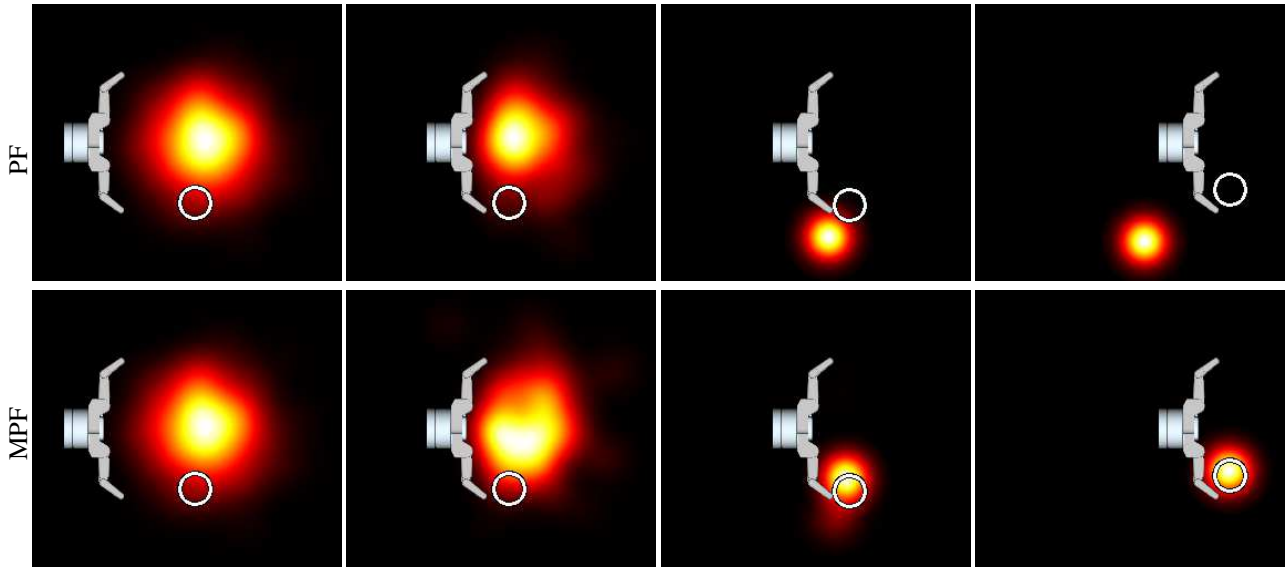
Instead, we approximate $\Pr(x_t \in M_i)$. Using the previous belief state to compute $\int_{M_i} bel(x_{t-1})\, dx_{t-1}$ might seem like a good approximation, but in fact it does *not* work. To see why, consider an update step at which the filter receives an observation which suggests the state to be on a manifold for the first time. If we use the previous belief, it will indicate a low probability for the state to be on the manifold and we will not pick that manifold for sampling. Even if we keep receiving observations that suggest the manifold, we will never be able to place particles on it since we will always be using the belief from the previous step.

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

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

where $\pi(z_t|u_t) = \int_{x_t} p(z_t|x_t, u_t)\, dx_t$ 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 sampling technique introduces no bias. This approximation performs well for our purposes since contact sensors accurately discriminate between contact and no-contact.

Finally, we sample a particle $x_t^{[i]}$ according to the belief distribution on the chosen manifold $bel(x_t|M_i)$. Our key insight is that we can apply a different sampling technique for each manifold that is specifically designed to take advantage of the structure of M_i. In the case of the free space M_m, we can sample $x_t^{[i]}$ from S using the conventional technique and



(a) Prior Belief (b) Before Contact (c) After Contact (d) Final Belief

Fig. 4. Belief state estimated by the PF and MPF for a hand pushing a bottle. (a) $p(x_0)$ is a Gaussian prior. (b) Particles that touch the hand are eliminated through a series of "no contact" observations. (c) and (d) As soon as contact is established the belief state collapses to the true state for the MPF.

reject any $x_t^{[i]} \in \cup_{i=1}^{m-1} M_i$. In the case of a contact manifold, we use *dual particles* as described in the next section.

B. Manifold Particle Filter for Contact Manipulation

In contact manipulation, there is a single *contact manifold* C that contains the set of all states in non-penetrating contact with the hand. The free space $\bar{C}$ contains the remaining poses that are not in contact with the hand. Algorithm 2 summarizes the application of the manifold particle filter to state estimation for contact manipulation problem.

If we choose to sample from C, then the conventional sampling technique will be ineffective. Instead, we importance sample using the *dual proposal distribution*, which samples from the observation model and computes importance weights using the motion model. Formally, we sample

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

from the observation model (line 5). The corresponding importance weight

$$w_t^{[i]} = \eta\, \pi(z_t|u_t) \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{7}$$

incorporates the motion model (line 6) and is found by dividing the target distribution Eq. (2) by the proposal distribution Eq. (6) [4].

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 [4].

Figure 4 shows a comparison of the PF and MPF for a hand pushing a bottle from left-to-right. Each link is equipped with a binary contact sensor and the belief is rendered using kernel density estimation. The PF (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 dual particles from the observation model.

C. Sampling from the Observation Model

One can use different methods to sample states from the observation model. Since the contact manifold is a two-dimensional manifold embedded in $SE(2)$, we can approximate C for an arbitrary object using a relatively small set of pre-computed hand-relative object poses. For each pre-computed sample, we compute its probability using Eq. (6) with the current u_t and z_t. Then, we sample $x_t^{[i]}$ from this weighted set.

We can further simplify this computation if we assume additional structure in the object model. For example, if the object and hand are extruded polygons, then we can avoid pre-computing samples of C by using an analytic representation of the contact manifold.

D. Weighting using the Motion Model

We must compute importance weights for the particles that we sampled from the dual proposal distribution. Equation (7) shows that the importance weights consist of three terms: (1) the normalization factor η, (2) the prior probability of the observation $\pi(z_t|u_t)$, and (3) the probability of the current state given our history $\int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$. The constant η appears in both the conventional and dual

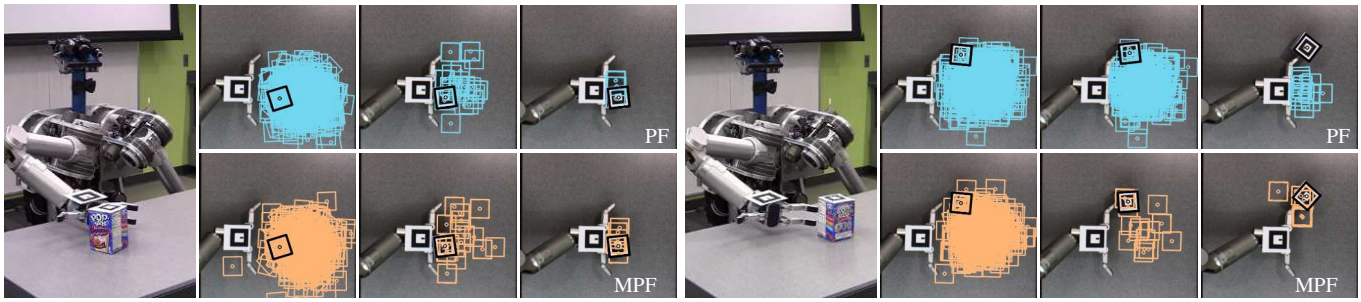


Fig. 5. HERB pushing a Pop-Tarts box across the table. The top and bottom rows show the belief as estimated by the PF (top, light blue) and the MPF (bottom, light orange) during different stages of manipulation. The PF and MPF perform similarly when the object contacts the large palm (left), but the MPF outperforms the PF when contact occurs with the small distal links (right).

importance weights and can be omitted. Similarly, $\pi(z_t|u_t)$ is the prior probability of receiving observation z_t and can be empirically estimated prior to running the algorithm.

Unfortunately, $\int p(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1}$ is difficult to evaluate: we have the ability to sample from this distribution using the motion model, but not evaluate it at an arbitrary value of x_t. Instead, we estimate the distribution by applying a density estimation algorithm. First, we forward-simulate the set of particles X_{t-1} to time t with the motion model [4]. Next, we use *kernel density estimation* to approximate $\int p(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1}$ from the forward-simulated samples. Kernel density estimation is a natural choice for this application because it is a non-parametric technique for estimating a probability distribution from a set of samples.

VI. Real-robot Experiments

We evaluated the PF and MPF on HERB, a bimanual mobile manipulator designed and built by the Personal Robotics Laboratory at Carnegie Mellon University [5]. HERB used a Barrett WAM arm equipped with a BarrettHand end-effector to push a 1.13 kg Pop-Tarts box across the table. We used the BarrettHand's finger strain gauges to detect binary contact on the distal links and a wrist-mounted force/torque sensor to detect binary contact on the rest of the hand at approximately 10 Hz. We assume that there is a single point contact, so the combination of those sensors uniquely localizes the point of contact in one of nine zones pictured in Fig. 2(c).

Figure 5 shows the two representative runs of the state estimator on HERB. The pose of the Pop-Tarts box relative to the hand was tracked by an overhead camera using a visual fiducial and is drawn as a black box. The conventional particle filter was run with 500 particles and the MPF was initialized with 450 conventional and 50 mixed particles. As described above, the mixed particles are adaptively sampled using whichever proposal distribution is best suited for the current state. These parameters were selected such that both algorithms took approximately 50 ms to update.

In Fig. 5-Left, the Pop-Tarts box initially contacted the palm and settled into persistent contact with the hand. The palm has a large swept volume, so both the PF (Fig. 5-Top-Left) and MPF (Fig. 5-Bottom-Left) successfully tracked the state. In both cases, the distribution quickly converged to the true state after contact is observed.

In Fig. 5-Right, the Pop-Tarts contacted the hand's left distal link and slips off the finger during the duration of the push. In this case, the distal link had a small swept volume and the PF incorrectly converged to the palm. As expected, the MPF successfully traced the state through the duration of the contact. This problem would be even more pronounced if HERB's observation model included high-resolution data, e.g. tactile arrays.

Please see the accompanying video[1] for more examples.

VII. Simulation Experiments

We have qualitatively shown that the MPF outperforms the particle filter when used for state estimation 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:

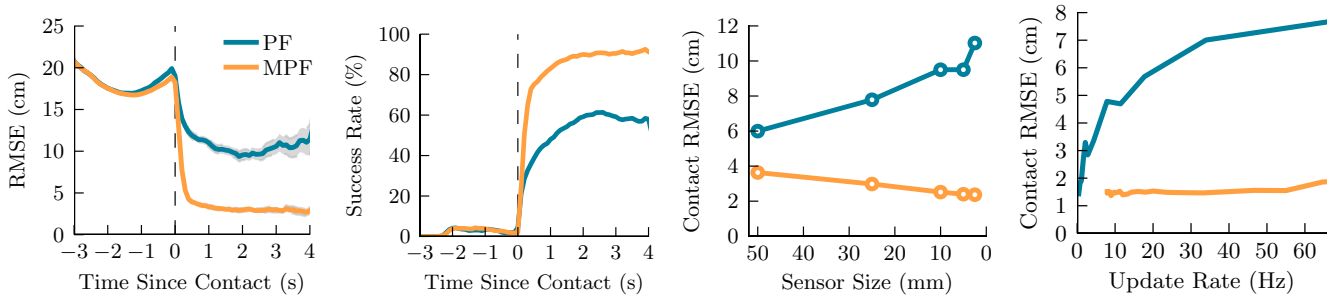
H1 *The error of the PF and MPF are similar before contact.*
H2 *The MPF has lower error than the PF after contact.*
H3 *Improving resolution increases the error of the PF.*
H4 *Improving resolution reduces the error of the MPF.*

Additionally, we demonstrate that the MPF achieves lower error than the MPF for a fixed amount of processing time.

A. Experiment Design

We implemented both filters in the OpenRAVE [29] simulation environment and evaluated the algorithms with a BarrettHand pushing a bottle. In each simulation, the hand moves in a straight line at a constant speed of 10 cm/s for 5 s and receives sensor updates from binary contact sensors at 10 Hz. The initial pose of the bottle was drawn from a Gaussian prior distribution with a randomly chosen mean and a standard deviation of 10 cm. We distributed contact sensors of a fixed size uniformly across the front surface of the hand. Observations from these sensors were simulated using full three-dimensional mesh collision checks between the object and sensors. For efficiency, the pushing simulation was implemented using an approximate, cached collision checker with a 1 mm resolution. Sensing errors were simulated by randomly perturbing the observation with a 5% probability.

[1] Video is available online at http://youtu.be/iCfApA39PiU.



(a) RMSE (b) Success Rate (c) Sensor Resolution (d) Update Rate

Fig. 6. (a) and (b) RMSE and SR of the PF and MPF plotted over time for 500 particles. Contact occurs at $t = 0$ and is denoted by the dashed line. (c) Effect of sensor resolution on RMSE. The MPF monotonically decreases in RMSE as resolution increases. Conversely, the accuracy of the PF degrades with high resolution sensors. (d) Real-time performance of the filters. The MPF achieves lower RMSE than the PF given a fixed computational resources. Additionally, the MPF achieves acceptable error with real-time update rates of 60+ Hz. The gray shaded region in (a) denotes the 95% confidence interval. This is omitted from (b), (c), and (d) because the confidence intervals are of a negligible size.

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 percent of time during which mean of the particles is within 2 cm of the true state.
- *Update rate* (UR), the maximum rate at which particle filter updates step can be executed.

These three metrics capture different properties of the filters. RMSE is the traditional error metric used in the localization literature [4, 28] 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 computational complexity and real-time performance of each filter.

C. Localization Error

We tracked the RMSE (Fig. 6(a)) and SR (Fig. 6(b)) of the two filters over time using a hand with 1 cm resolution contact sensors. The x-axes of the plots are aligned such that contact occurs at $t = 0$. Before contact, both filters behave similarly and there is a negligible difference in RMSE (3.67 mm $<$ 5 mm, $t(24192) = 4.3050$, $p < .0001$). After contact, the MPF achieves 7 cm less RMSE than the PF ($t(35556) = 74.86$, $p < .0001$). These results confirm hypotheses H1 and H2: the MPF achieves significantly lower error than the PF. We present example simulated runs of MPF and PF in Fig. 4.

D. Sensor Resolution

We additionally evaluated the effect of sensor resolution on the RMSE error of both filters. Figure 6(c) shows the average RMSE error during contact as the sensor resolution is varied from cell sizes of 2 mm to 5 cm. Smaller sizes correspond to a higher sensing resolution and, ideally, lower error. As expected, the MPF outperforms the PF at all sensor resolutions.

In both cases, an ANOVA showed that sensor resolution was a significant main effect for both the particle filter ($F(4, 134674) = 146.462$, $p < .0001$) and the MPF ($F(4, 134674) = 137.211$, $p < .0001$) with a negative correlation for the PF and a positive correlation for the MPF. In both cases, all but one pairwise tests of sensor resolutions were significant. This confirms hypotheses H3 and H4. Unlike the MPF, the performance of the PF 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 6(d) shows RMSE during contact as a function of update rate. The update rate was indirectly manipulated by varying the number of particles from 50 to 10,000 and measuring the time require 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.

The MPF achieves acceptable accuracy (e.g. $<$ 2 cm RMSE) with several hundred particles and URs of 60+ Hz. Conversely, the PF only achieves comparable accuracy when run with 10,000 particles and has an UR of approximately 1 Hz. These results confirm that PF 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

A. Limitations

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 [11]. 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, we assume $x_t \in SE(2)$. This is sufficient for planar manipulation, but the state space becomes larger if objects are articulated, can topple, or can roll. In particular, sampling $x_t^{[i]}$ according to Eq. (6) may become challenging. For most observation models, this is done by representing C with a finite set of samples. However, the number of samples required to densely represent C grows exponentially with the dimension of the state space. 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 be enough to track the state on lower dimensional manifolds. The performance of our algorithm can be improved by taking into account such manifold characteristics.

Finally, in future work, we plan to use the state estimate provided by the MPF as real-time feedback for physics-based actions. For example, this could include a *closed-loop push grasp* that uses constant feedback to react in real-time.

B. Contributions and Implications

Contact sensors provide useful information during object manipulation. 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.

In this paper, we have shown that contact sensors fundamentally differ from the vision and depth sensors traditionally used in state estimation (Section IV). Using this insight, we formulated the MPF (Section V) and demonstrated an implementation on a real robot (Section VI) using a simple sensor model. We evaluated the same implementation in a suite of simulations (Section VII) and showed that the MPF outperforms the conventional particle filter in both accuracy and speed.

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

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

[5] S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. Dogar, A. Dragan, R. Knepper, T. Niemueller, K. Strabala, and M. Vande Weghe, "HERB 2.0: Lessons learned from developing a mobile manipulator for the home," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 1–19, 2012.

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

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

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

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

[10] 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.

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

[12] D. F. Lillian Chang, Joshua R. Smith, "Interactive singulation of objects from a pile," in *IROS 2011 PR2 Workshop*, 2011.

[13] M. Gupta and G. S. Sukhatme, "Using manipulation primitives for brick sorting in clutter," in *IEEE ICRA*. IEEE, 2012, pp. 3883–3889.

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

[15] 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.

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

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

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

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

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

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

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

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

[24] 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.

[25] K. Hsiao, "Relatively robust grasping," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

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

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

[28] 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.

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