Robust Manipulation via Contact Sensing

Michael C. Koval

September 7, 2016

CMU-RI-TR-16-54

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Siddhartha S. Srinivasa, CMU RI (co-chair)
Nancy S. Pollard, CMU RI (co-chair)
Geoffrey J. Gordon, CMU MLD
Tomás Lozano-Pérez, MIT CSAIL

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Abstract

Humans effortlessly manipulate objects in cluttered and uncertain environments. In contrast, most robotic manipulators are limited to carefully engineered environments to circumvent the difficulty of manipulation under uncertainty. Contact sensors can provide robots with the feedback vital to addressing this limitation.

This thesis proposes a framework for using feedback from contact sensors to reliably manipulate objects under uncertainty. We formalize manipulation as a partially observable Markov decision process that includes object pose uncertainty, proprioceptual error, and kinematic constraints. Our algorithms exploit the structure of contact to efficiently estimate state and plan with this model.

First, we introduce the manifold particle filter as a principled method of estimating object pose and robot configuration. This algorithm avoids degeneracy by drawing samples from the lower-dimensional manifold of states induced by contact.

Next, we introduce two belief space planning algorithms that seek out contact with sensors when doing so is necessary to achieve the goal. One algorithm harnesses the decoupling effect of contact to share computation between problem instances. The second leverages lower-dimensional structure to plan around kinematic constraints.

Finally, we evaluate the efficacy of our approach in real-robot and simulation experiments. The results show that our state estimation and planning algorithms consistently outperform those that are not tailored to manipulation with contact sensing.

Acknowledgements

I could not ask for better advisors than Sidd Srinivasa and Nancy Pollard. Thank you both for teaching me how to write, present, and do research. And for putting up with my side-projects on HERB.

I am grateful to the members of my committee, Geoff Gordon and Tomás Lozano-Pérez, for their input. Geoff, thank you for sharing your bottomless knowledge of machine learning. Tomás, thank you for sharing your experience and helping me focus on important problems. I would also like to thank Drew Bagnell, David Hsu, and Joelle Pineau for our discussions: I have learned a lot from you.

I have had the joy—and, sometimes, anguish—of working on several robots. It takes a team to run a robot, so thanks to everyone who made this possible. On HERB, special thanks to Mike Vande Weghe, Michael Dawson-Haggerty, and Clint Liddick for the thankless task of keeping him running. On Andy, special thanks to Moslem Kazemi and Jean-Sébastien Valois for bringing levity to even the most stressful demonstrations. On Robonaut, special thanks to Julia Badger for being my research collaborator and Allison Thackston for teaching me how R2 works. At CMU, I would like to thank Jean Harpley, Suzanne Lyons Muth, and Keyla Cook for all of your help.

Working in the Personal Robotics Lab has been an unforgettable experience. Thanks to Aaron J., Aaron W., Gilwoo, Henny, J.S. Lee, Laura, Matt K., Mehmet, Pyry, Rachel, Shervin, Shushman, Stefanos, and Zita for making the lab awesome. Special thanks to Vikram for being a great minion and to Anca, Chris, Jen K., and Liz for the Caribbean—and non-Caribbean—fun. And thanks to Pras for being a partner on our numerous side-projects.

Thanks to my friends for keeping me sane. Nitish, thank you for being a great roommate and friend for the past seven years. Arun, thank you for being my roommate, a friend, and for teaching me all I know about machine learning. Akshara and Will, thank you for the dinner conversations and all of our adventures. Thanks to Cody, Jeff, Jen P., John, Jordan, Owen, and Nat ("the suite") for keeping in touch.

My final thanks goes to my parents, Cap and Karen Koval, who have always supported me. Thank you for everything: none of this would have been possible without you.

Funding

This thesis contains work supported by a NASA Space Technologies Research Fellowship Program (grant number NNX13AL62H); the Defense Advanced Research Projects Agency (grant number DARPA-BAA-10-28); the National Science Foundation (grant numbers NSF-IIS-1218182, NSF-IIS-1409003 and NSF-IIS-1426703) the U.S. Office of Naval Research (grant number N000141210613); and the Toyota Motor Corporation.

Contents

List of Figures

1
Introduction

Humans effortlessly manipulate their environment. We are capable of grasping our glasses from our nightstand in the dark, blindly plugging a cable into the back of our computer, and grasping our coffee mug without glancing away from our computer monitor. In the process, we fearlessly push, pull, and slide objects as necessary to complete the task. We use proprioception and tactile sensing to manipulate with confidence despite the variety and complexity of these tasks.

In contrast, most robotic manipulators perform only *pick-and-place* actions in factories. First, the robot perceives an object and chooses a grasp. Then, it uses a *motion planner* to generate a collision-free trajectory to the grasp. Finally, the robot closes its fingers and—hopefully—achieves the desired grasp. This approach ignores the fact that manipulation is a noisy physical process.

Pick-and-place works in industrial applications because the robot has clear access to the target object, is meticulously calibrated, and operates in an environment specifically engineered for a robot to autonomously perform its task. For example, industrial robots frequently are equipped with task-specific end-effectors, track visual fiducials, rely on off-board perception systems, and use fixtures to precisely position the target object.

In many domains it is not possible to engineer the environment in these ways. Robots like HERB [Srinivasa et al., 2012], Andy [Bagnell et al., 2012], and Robonaut 2 [Diftler et al., 2011] operate in environments designed for humans (fig. 1.1). Human environments are cluttered and fundamentally *uncertain*. These robots rely on on-board perception, suffer from occlusion, and rarely have a complete model of their environment (fig. 1.2a). Additionally, robots designed for human environments typically use low-gain, inaccurate, controllers to safely work in proximity to humans (fig. 1.2b). As a result, open-loop actions are brittle and often fail.

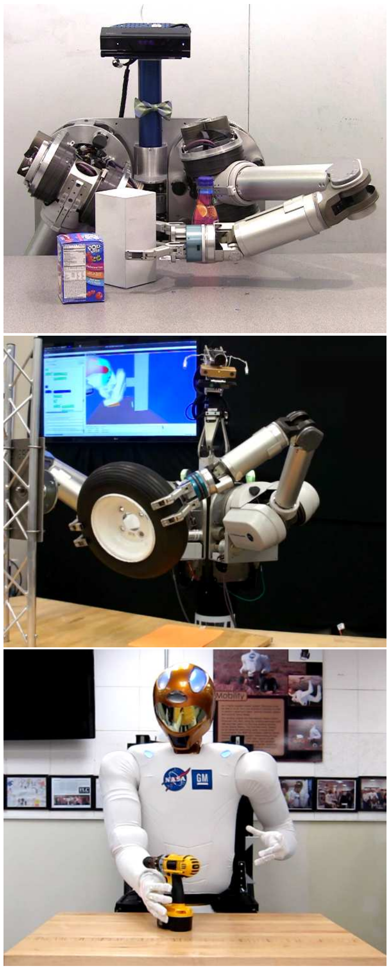
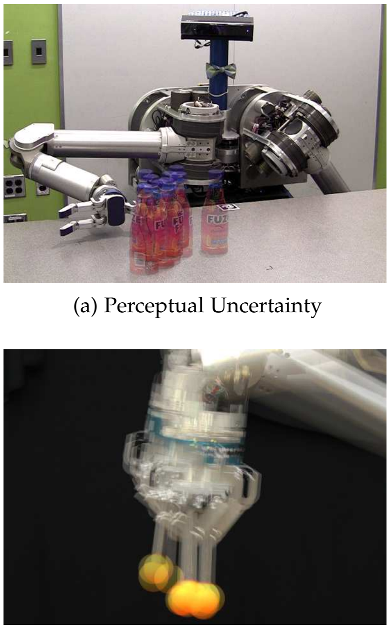


Figure 1.1: HERB (top), Andy (middle), and Robonaut 2 (bottom) manipulating objects in human environments. All three of these robots rely on on-board sensing to cope with uncertainty.

Contact sensors provide vital feedback that can be used to reduce uncertainty. Contact is intimately linked to manipulation: a robot can only apply force to an object while in contact with it. Vision and depth sensors must infer contact from object pose and motion. This is challenging in manipulation because the hand often occludes the object of interest. In contrast, contact sensors directly observe interaction with the object and are unaffected by occlusion.


(a) Perceptual Uncertainty

(b) Proprioceptive Uncertainty

Figure 1.2: Robots suffer form uncertainty in both (a) perception and (b) proprioception. Bottom figure courtesy of Klingensmith et al. [2013].

However, contact sensors have two key limitations. First, contact sensors do not directly estimate object pose. Second, contact sensors provide little information while out of contact with an object. As a result of these limitations, contact sensors are often neglected as a source of real-time feedback during manipulation.

> *This thesis proposes a framework for using real-time feedback from contact sensors to reliably manipulate objects under uncertainty.*

To address this problem in depth, this thesis focuses on the manipulation of a single object using feedback exclusively from proprioception and contact sensors. We assume that the goal can be expressed in configuration space and that some—possibly uncertain—model of the robot, object, and environment are available.

We do not specifically consider planning multi-stage tasks, but anticipate that our framework could be used as a robust primitive action in multi-stage planning algorithms. We also do not consider feedback from other sensing modalities; e.g. vision. However, as we discuss in section 8.2.5, it is relatively straightforward to incorporate these observations into our state estimate at runtime.

1.1 *Challenges with Contact Sensing for Manipulation*

There are three principal challenges that make it difficult to use contact sensing for closed-loop manipulation. We survey them here.

Challenge 1: Contact is Inherently Discontinuous

Contact is binary: two rigid bodies are either in contact or they are not. If they are in contact, then the contact may transmit force. If not—even if the bodies are infinitesimally close together—no force is transmitted. As a result, changes in contact are inherently *discontinuous*. The robot may make or break contact by moving an infinitesimal distance. Similarly, an object's motion may change discontinuously.

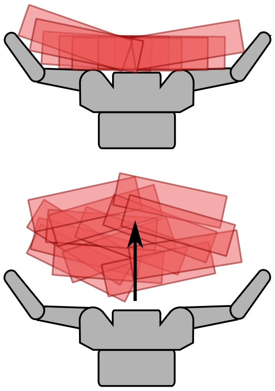
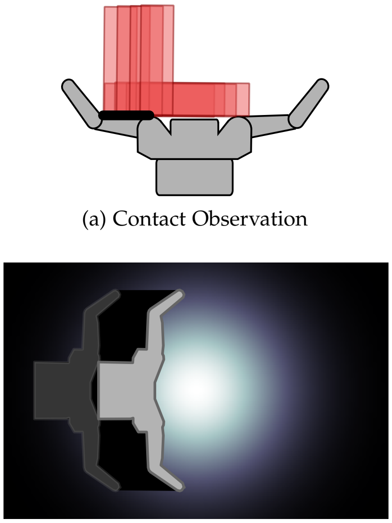


Figure 1.3: Pushing produces a discontinuous belief state where the pose of the object is constrained to a line.

Contact sensors accurately *discriminate* between contact and no-contact. As a result, contact sensors inherit the discontinuous nature of contact. The output of a contact sensor changes discontinuously when the robot makes or breaks contact with an object.

Discontinuities introduced by contact accumulate over time into discontinuous belief states. Figure 1.3 shows an example of a pushing action producing a belief state that is constrained to a line. Observations from contact sensors introduce additional discontinuities into the belief state. Figure 1.4 shows an example where receiving a sequence of (a) contact or (b) no-contact observations produces a discontinuous belief state. In fig. 1.4b, the initial belief state was a Gaussian distribution centered in the figure. The posterior belief state assigns zero-probability to the swept volume of the hand because any object in this region would have generated a contact observation.

Planning in this domain is challenging. An infinitesimal change in a policy may result in a discontinuous change in its quality. Policies that exhibit this property are likely to fail when noisily executed on a real system. Our goal is to generate robust policies that succeed despite uncertainty during execution. Additionally, the presence of discontinuous belief states makes it challenging to use planning techniques that rely on discretization or require a compact, parametric representation of belief states.


(a) Contact Observation

(b) No-Contact Observation

Figure 1.4: Receiving (a) contact or (b) no-contact observations may produce discontinuous belief states.

Challenge 2: Contact Sensors Require Active Sensing

Contact sensors provide rich information while in contact with an object, but little information otherwise. For example, a contact sensor may return a contact position and normal vector during contact. We can use this information to partially infer the object's pose. This is not possible while out of contact with the sensor: we can only infer that the object is not touching the sensor.

Fully utilizing contact sensors requires *active sensing*. The robot must take *information-gathering actions* to force the object into contact with its sensors (fig. 1.5). Information-gathering actions may not directly work towards achieving the goal. Instead, they generate vital observations that reduce uncertainty. Using these observations the robot can robustly achieve the goal.

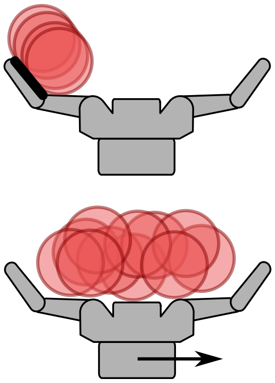


Figure 1.5: The robot moves right to force the bottle into contact with a sensor on its fingertip. This action localizes the bottle and allows the robot to achieve the goal.

Synthesizing information-gathering actions requires reasoning about uncertainty during planning. Planning under uncertainty requires planning in *belief space*, the infinite-dimensional space of probability distributions over state. Belief space planning is computationally expensive even in small, discrete domains.

Belief space planning is particularly hard for manipulation. First, manipulation is continuous and high-dimensional. Second, the robot requires multiple observations from contact sensors to infer the full state of the environment. The planner must generate a sophisticated policy that seeks out and combines information from multiple observations to achieves success.

These challenges are not unique to contact sensing. Different sensing modalities require differing amounts of information-gathering (fig. 1.6). Long range sensors, like LIDAR (left), are nearly equally accurate throughout the robot's entire workspace. Near-touch sensors (middle) only observe nearby objects. Contact sensors (right) take this to the extreme by only observing objects touching the sensor. We focus on contact sensing in this thesis, but believe that our framework is applicable to other sensing modalities (section 8.2.5).

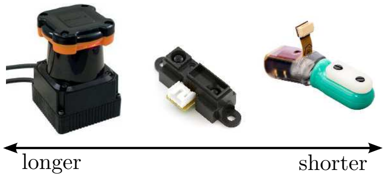


Figure 1.6: Different sensing modalities have different maximum ranges. LIDAR (left) has a long range, near-touch sensors (middle) have a short range, and contact sensors (right) have zero range.

Challenge 3: Manipulation is a Physical Process

Making contact with an object also allows the robot to apply force to it. This force may push the object out of the way or—if the object is heavy—significantly affect the motion of the robot. The outcome of the interaction is governed by physics.

Physics can be thought of as a *non-holonomic constraint* on the evolution of the environment's state. The presence of this constraint means that it is difficult to solve the *two-point boundary value problem* in state space; i.e to find a sequence of actions that moves the environment from an initial state to a desired final state. Planning in this type of domain is challenging even when state is fully observed. Most motion planning algorithms gain efficiency by performing a bi-directional search [Kuffner and LaValle, 2000] or building a graph-like structure in configuration space [Kavraki et al., 1996, Karaman and Frazzoli, 2011, Gammell et al., 2015], both of which require solving the two-point boundary value problem.

Operation	Time (ms)	Percent
Physics Model	130.37	80.47%
Sensor Model	29.28	18.07%
Other	2.37	1.46%

Table 1.1: Time spent performing each operation in Koval et al. [2015b]. Times are reported for one update of the filter.

Additionally, planning with physics requires a *physics model* to predict the outcome of actions. Physics models suffer from two key problems. First, simulating physics is computationally expensive and dominates planning time (table 1.1). Second, physics models depend on a variety of physical properties; e.g. those listed in table 1.2. We rarely know the value of these properties, so every action we take introduces additional uncertainty into the state of the environment.

Parameter	Dim.	Units
Center of Mass	3	m
Mass	1	kg
Inertia Tensor	6	kg·m^2
Friction Coeffient	1	–
Restitution Coefficient	1	–
Geometry	∞	–

Table 1.2: Properties necessary to simulate a rigid body in the DART [Liu et al., 2016] physics simulator.

1.2 *Key Insights to Our Approach*

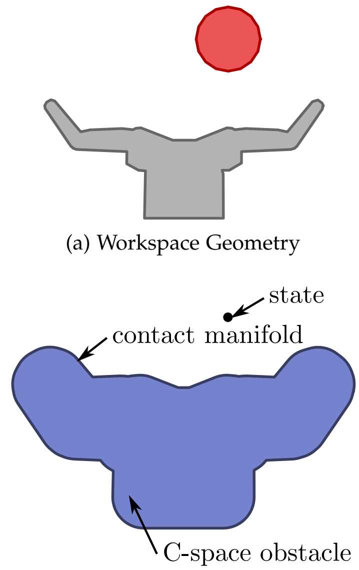
The general problem of planning under control and sensing uncertainty is intractable. In this section, we identify the structure unique to manipulation and contact sensing that allows us to plan efficiently.

Insight 1: Contact Constrains State to a Manifold

Contact is a constraint on configuration. Configurations can be partitioned into free space (no contact), an invalid region (penetrating contact), and a lower-dimensional *contact manifold* (non-penetrating contact). See fig. 1.7 for a graphical depiction of the contact manifold for a radially symmetric object and fig. 1.8 for an asymmetric object.

The lower-dimensional nature of the contact manifold introduces the discontinuity described in challenge 1. However, we can also exploit this structure to our benefit. If we receive a contact observation, then we can infer that an object lies on the lower-dimensional contact manifold of the sensor. This observation eliminates one dimension of uncertainty. Each independent contact further reduces the dimensionality of unobserved state by one.

This thesis uses the contact manifold in several ways. Chapter 4 uses an explicit representation of the contact manifold to efficiently estimate the pose of an object relative to the end-effector. Chapter 5 uses an implicit representation of the contact manifold to estimate the configuration of the robot in a known environment. Finally, chapter 6 discretizes the surface of the contact manifold to apply discrete planning methods to a continuous problem.



(a) Workspace Geometry

(b) C-space Geometry

Figure 1.7: The (a) workspace geometry of a robot pushing an object induces (b) the lower-dimensional contact manifold in the configuration space of the object.

Insight 2: Contact Decouples the Optimal Policy

Observing contact implies that the true state of the environment lies on the contact manifold. As a result, we can infer that any hypothesized state that is not on the contact manifold is incorrect. A contact observation can be thought of as a "funnel" that collapses a large set of initial belief states into a smaller set of belief states, each with support restricted to the contact manifold.

This funneling behavior partially decouples the optimal pre-contact policy from the optimal post-contact policy. We expect scenarios that result from the same contact observation to be solved by similar post-contact policies. This allows us to efficiently plan long horizon policies that gather information (challenge 2).

Chapter 6 exploits this structure by decomposing a policy into an open-loop pre-contact trajectory followed by a closed-loop post-contact policy. This method plans the post-contact policy offline, once per object class. Then, when confronted with a problem instance, the planner performs an online search to find a pre-contact trajectory.

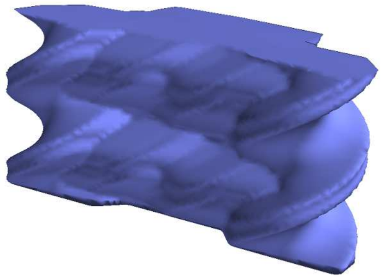


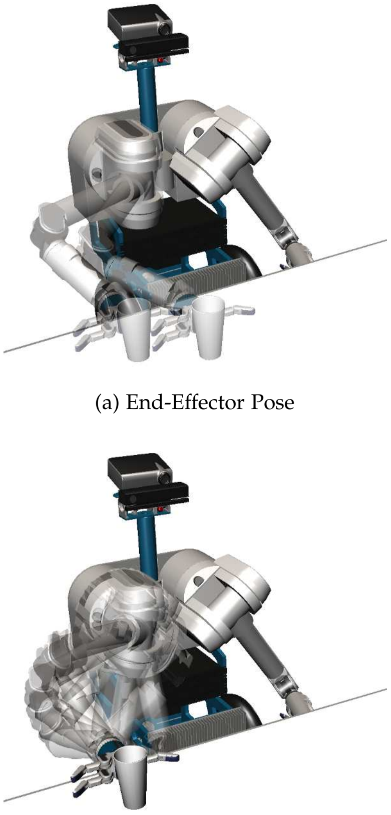
Figure 1.8: For the planar manipulation of non-radially symmetric objects, such as the box shown in fig. 1.3, the contact manifold is a two dimensional structure embedded in $SE(2)$.

Insight 3: Manipulation Occurs in a Lower-Dimensional Space

Finally, we note that the motion of an object often does not depend on the full state of the environment. Instead, we can write the motion of the object as a function of only the pose and motion of the robot's links that are in contact with it. Often, we only consider the pose of the object relative to the end-effector (fig. 1.9a).

Solving this lower-dimensional problem may be significantly easier than solving the full problem, i.e. may require fewer evaluations of the physics model (challenge 3). We can project from state to end-effector pose using the robot's forward kinematics. Similarly, we can use inverse kinematics to project an end-effector pose back into state (fig. 1.9b). However, this conversion is lossless only when the motion of the robot is subject to no *kinematic constraints*, e.g. kinematic reachability constraints, joint limits, or collision.

Chapters 4 and 6 gain efficiency by, respectively, estimating and planning for the pose of an object relative to the end-effector. These techniques perform well when the robot is in an uncluttered environment that subjects it to few kinematic constraints. Chapter 7 exploits this structure in a more nuanced way by using a hand-relative policy to guide a planner through the robot's configuration space.



(a) End-Effector Pose

(b) Kinematic Redundancy

Figure 1.9: Interaction between the object and end-effector is invariant to (a) the end-effector pose and (b) the null space of inverse kinematics solutions.

1.3 Contributions

This thesis develops a framework that allows robots to use real-time feedback from contact sensors to reliably manipulate objects under uncertainty. Specifically, we contribute:

- A formulation of manipulation as a partially observable Markov decision process (POMDP, Smallwood and Sondik [1973], Kaelbling et al. [1998]) that includes object pose uncertainty, proprioceptual error, and kinematic constraints (chapter 3).
- The *manifold particle filter* as a method of estimating the pose of an object relative to the end-effector using real-time feedback from contact sensors. (chapter 4, Koval et al. [2013b,a, 2016c, 2015b]).
- An *implicit manifold representation* that enables the manifold particle filter to scale up to estimating robot configuration in known, static environments (chapter 5, Klingensmith et al. [2016]).
- A planner for hand-relative manipulation that *decomposes the policy into pre- and post-contact phases* to find long-horizon policies that actively gather information (chapter 6, Koval et al. [2014, 2016d]).
- An online planner that *uses heuristics derived from a hand-relative policy* to cope with both kinematic constraints and object pose uncertainty (chapter 7, Koval et al. [2016a,b]).

Finally, chapter 8 concludes by discussing the limitations of our approach and outlining potential directions for future research.

2
Background

This thesis builds on prior work on non-prehensile manipulation (section 2.1), contact sensing for manipulator control (section 2.2), and contact sensing for state estimation (section 2.3). Similar to some recent work work on motion planning under uncertainty (section 2.4), we formulate manipulation as a POMDP (section 2.5).

2.1 Open-Loop Pushing Manipulation under Uncertainty

Early work in manipulation addressed the part alignment problem. In this problem, a robot plans an open-loop trajectory that reconfigures an object despite uncertainty in its initial pose [Brokowski et al., 1993, Erdmann and Mason, 1988, Brokowski et al., 1993]. More recently, the same approach has been applied to the problems of grasping [Dogar and Srinivasa, 2010] and rearrangement planning [Dogar and Srinivasa, 2012, Koval et al., 2015a] under the quasistatic assumption [Lynch et al., 1992].

These techniques all consider non-deterministic uncertainty [LaValle and Hutchinson, 1998] in object pose and use worst-case analysis to guarantee success. For example, the push-grasp uses a long, straight-line pushing action to funnel the object into the hand before closing the fingers to achieve a stable grasp [Dogar and Srinivasa, 2010].

Our approach also makes the quasistatic assumption and—when necessary to achieve the goal—generates uncertainty-reducing actions that resemble the push-grasp. However, our approach uses real-time feedback from contact sensors to estimate the pose of the object and achieve the goal. This allows us to achieve the success more quickly and under larger amounts of uncertainty than open-loop strategies.

2.2 Contact Sensing for Manipulator Control

One method of achieving success under uncertainty is to use real-time feedback from contact sensors by directly mapping observations to actions. Prior work has developed controllers that can locally refine the quality of a grasp [Platt et al., 2010a] or achieve a desired tactile sensor reading [Zhang and Chen, 2000, Li et al., 2013]. These techniques achieve real-time control rates of up to 1.9 kHz [Li et al., 2013] and impressive performance in controlled environments. However—unlike our approach—these algorithms require a high-level planner to analyze the scene and provide a set point to the controller.

It is possible to subvert this problem by directly learning a robust control policy. This has been done by learning a model of expected sensor observations from past experience [Pastor et al., 2011] and using perturbed rollouts to evaluate the effect of uncertainty on candidate policies [Stulp et al., 2011]. These approaches have been shown to perform well in practice, but policies learned in this way do not easily generalize new tasks or robots. Our approach can be applied to any task or robot for which transition, observation, and reward functions are available.

2.3 *Contact Sensing for State Estimation*

An alternative use of contact sensing is to estimate the state of the environment during manipulation. Prior work has used contact sensors to predict grasp stability [Dang et al., 2011] and object identity [Xu et al., 2013, Schneider et al., 2009].

Approaches dating back to the 1970s [Šimunović, 1979] have formulated manipulation under uncertainty as a Bayesian estimation problem. Recently, there has been renewed interest in using contact sensors in the particle filter to track the pose [Zhang and Trinkle, 2012] and physical parameters [Zhang et al., 2013] of an object being pushed in the plane. Other, closely related work, used a particle filter to track a hybrid discrete-continuous probability distribution over the discrete contact formation [Xiao, 1993] and continuous pose of the object [Gadeyne et al., 2005, Meeussen et al., 2007].

An alternative approach recursively applies local optimization techniques, like iterative closest point [Besl and McKay, 1992], to track robot configuration and object pose with a depth sensor [Klingensmith et al., 2013, Schmidt et al., 2014]. These techniques have been extended to penalize inter-penetration and disagreement with contact observations [Bimbo et al., 2013, Schmidt et al., 2015].

State estimation is an important part of successfully manipulating objects under uncertainty. We introduce the manifold particle as a Bayesian method of estimating object pose (chapter 4) and robot configuration (chapter 5) from contact sensor observations. Unlike previous techniques, the manifold particle filter draws samples from the contact manifold to avoid particle deprivation during contact.

The implicit representation of the contact manifold that we introduce in chapter 5 closely resembles the technique used to incorporate contact sensors into an optimization-based tracker. However, unlike those techniques, the manifold particle filter implements a true Bayes update and, thus, provides an estimate of the full belief state.

2.4 *Motion Planning under Uncertainty*

Several motion planning algorithms generate closed-loop policies that achieve a goal under uncertainty. These algorithms include low-level controllers (e.g. those cited in section 2.2), high-level symbolic planners [Smith and Weld, 1998, Hyafil and Bacchus, 2003], and hybrid task planners [Kaelbling and Lozano-Pérez, 2013]. In this thesis, we specifically consider the problem of generating a low-level policy. These policies can be used as primitive actions inside a high-level planning framework.

Other work has solved the motion planning under uncertainty problems under the linear-quadratic-Gaussian (LQG) assumptions [Athans, 1971]. In this case, it is efficient to plan by generating and testing candidate trajectories [van den Berg et al., 2010], building a roadmap in state space [Agha-mohammadi et al., 2011, van den Berg et al., 2011], or planning with the maximum-likelihood hypothesis [Platt et al., 2010b]. These techniques have been extended to a variety of application domains. Unfortunately, the belief states encountered during contact manipulation are non-Gaussian and quickly become multi-modal (challenge 3). This precludes us from using techniques that assume that the belief state remains Gaussian.

The idea of planning with the maximum-likelihood hypothesis has also been applied to manipulation [Platt et al., 2011]. This approach uses trajectory optimization to plan a trajectory that either: (1) achieves the goal for a nominal hypothesis or (2) receives observations that invalidate that hypothesis. In the latter case, the algorithm is guaranteed to converge to the goal in a finite number of re-planning steps [Platt. et al., 2012]. Unfortunately, theses techniques aim for feasibility, not optimality. In contrast, our approach optimizes a reward function that drives the robot to quickly achieve the goal.

2.5 *Planning for Manipulation under Uncertainty*

The work described above solves the general problem of motion planning under uncertainty. In this thesis, we specifically consider planning for manipulation tasks using feedback from contact sensors. Physics-based manipulation under uncertainty is particularly challenging because the observations generated by contact sensors are inherently discontinuous (challenge 1), we must generate information-gathering actions (challenge 2), and we must plan with a physics model (challenge 3).

Early work on robotic assembly used feedback from force sensors to perform fine manipulation [Šimunović, 1979]. A common strategy is to use guarded moves; i.e. move-until-touch actions, to localize the manipulator relative to the environment [Will and Grossman, 1975]. Guarded moves were constructed by hand [Bolles and Paul, 1973] and, later, synthesized automatically [Lozano-Pérez et al., 1984]. Recent work has considered the problem of tactile localization [Petrovskaya and Khatib, 2011, Hebert et al., 2013, Javdani et al., 2013], where the robot plans a sequence of guarded moves to localize an object.

These techniques split the policy into an information-gathering stage, which attempts to localize the object, followed by a goal-directed stage. An alternative approach is to switch between executing information-gathering and goal-directed trajectories depending upon the amount of uncertainty [Nikandrova et al., 2014]. We propose to eliminate the need for an explicit information-gathering stage by naturally gathering information during execution when doing so is necessary to achieve the goal.

We trade-off between information gathering and goal directed behavior by formulating contact manipulation as a POMDP [Kaelbling et al., 1998]. Hsiao et al. first formulated grasping as a POMDP by decomposing the continuous state space into a discrete set of cells [Hsiao et al., 2007] and, later, by selecting trajectories from a set of candidates [Hsiao et al., 2008]. Both of these approaches assume that the object does not significantly move when touched [Hsiao, 2009]. We consider the case of planar pushing, where motion of the object is critical to achieving the goal.

More recent work has used SARSOP [Kurniawati et al., 2008] to synthesize an efficient policy that grasps a lug nut under uncertainty [Horowitz and Burdick, 2013]. That work explicitly models the motion of the lug nut and introduces the concept of an *interactivity-based state space* that densely discretizes states that are near contact.

Chapter 6 introduces a planner that also uses SARSOP to find a post-contact policy and discretizes the state space. However, our planner differs in three key ways: it (1) uses a pre-contact search to extend the planning horizon, (2) re-uses a post-contact policy across problem instances, and (3) discretizes the contact manifold to preserve the discontinuity of contact. Chapter 7 introduces an online planner based on DESPOT [Somani et al., 2013] that, unlike past techniques, considers kinematic constraints during planning.

3 Approach

We consider the problem of a robot manipulating a movable object in a static environment into a goal region. The robot has a *configuration space* Q and the movable object has pose $x \in X = \mathrm{SE}(3)$. We represent the state of the system as a point in the *state space* $S = Q \times X$.

State space consists of *immovable obstacles* $S_{\mathrm{obs}} \subseteq S$ and *free space* $S_{\mathrm{free}} = S \setminus S_{\mathrm{obs}}$. We define S_{free} to include its boundary to allow non-penetrating contact between the robot, the object, and immovable obstacles in the environment. Our goal is to manipulate the movable object into a *goal region* $G \subseteq S_{\mathrm{free}}$ defined in this space.

We model manipulation as a stochastic, discrete-time, dynamic system. The robot begins in an *initial state* $s_0 \in S$ at time $t = 0$. At each time step, the robot executes an *action* $a_t = (\xi, \Delta t) \in A$ that is a configuration-space trajectory $\xi : [0, \Delta t] \to Q$ with duration Δt. State evolves from s_{t-1} to s_t according to a stochastic *transition model*

$$T(s_{t-1}, a_t, s_t) = p(s_t | s_{t-1}, a_t)$$

that includes the motion of the robot, collision with immovable obstacles, and contact with the movable object.

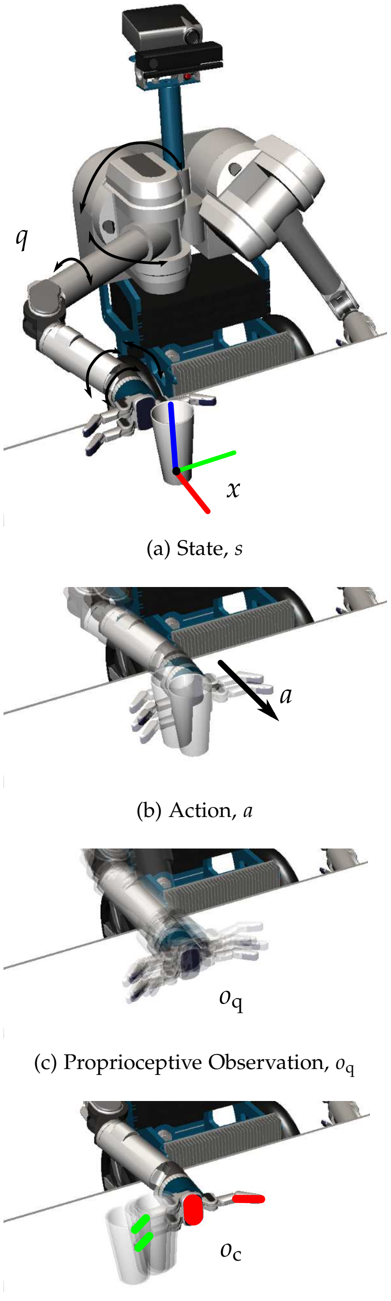
After transitioning to state s_t, the robot receives an *observation* $o_t = (o_{\mathrm{q}}, o_{\mathrm{s}}) \in O$ consisting of a *proprioceptive observation* $o_{\mathrm{q}} \in O_{\mathrm{q}}$ from the robot's internal sensors and a *contact sensor observation* $o_{\mathrm{s}} \in O_{\mathrm{s}}$ from its contact sensors. We assume that observations are generated according to a stochastic *observation model*

$$\Omega(o_t, s_t, a_t) = p(o_t | s_t, a_t) = p(o_{\mathrm{q}} | s_t, a_t) p(o_{\mathrm{s}} | s_t, a_t),$$

which factors into two terms because proprioceptive and contact sensor observations are conditionally independent given state.

The robot does not know the true state s_t. Instead, it uses the transition and observation models to track the *belief state*

$$b(s_t) = p(s_t | a_{1:t}, o_{1:t})$$



(a) State, s

(b) Action, a

(c) Proprioceptive Observation, o_{q}

(d) Contact Sensor Observation, o_{s}

Figure 3.1: (a) State, (b) an action, and (c)–(d) observations.

over time. The belief state $b(s_t)$ is the probability distribution over the state s_t at time t given the history of past actions $a_{1:t} = a_1, \ldots, a_t$ and observations $o_{1:t} = o_1, \ldots, o_t$ (fig. 3.2). Each belief state is an element of the infinite-dimensional simplex Δ called *belief space*.

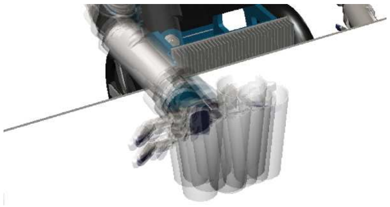


Figure 3.2: A belief state is a probability distribution over state given the history of actions and observations.

Our goal is to find a policy $\pi : \Delta \to A$ over belief space that quickly reaches the goal. A *policy* chooses which action $\pi[b]$ to execute in a belief state b. We encode our desire to reach the goal into a *value function* $V^\pi : \Delta \to \mathbb{R}$ that defines the utility of starting in belief state b and following policy π. Our goal is to find an *optimal policy*

$$\pi^* = \arg\max_\pi V^\pi[b(s_0)]$$

that achieves the highest value on the *initial belief state* $b(s_0)$.

The remainder of this section defines the transition model (section 3.1), observation model (sections 3.2 and 3.3), and value function (section 3.4) in detail. We conclude by defining three simplified models that prove to be useful throughout this thesis (section 3.5).

3.1 Transition Model

The transition model encodes the physics of manipulation. We restrict ourselves to the quasistatic manipulation of objects resting on a planar support surface. The *quasistatic assumption* states that friction is high enough to neglect acceleration of the object; i.e. an object stops moving as soon as it leaves contact with its pusher [Mason, 1986]. This assumption has been shown to be an accurate approximation of many household manipulation tasks [Dogar, 2013].

We implement the transition model using an analytic [Lynch et al., 1992] or numerical [Catto, 2010] rigid body simulator. We model the simulator as a deterministic transition function $T^\theta_{\text{det}} : S \times A \to S$ parameterized by *static properties* $\theta \in \Theta$. The properties may include those listed in table 1.2, e.g. geometry and friction coefficients.

In practice, θ is rarely known with certainty (challenge 3). We model θ as a random variable with distribution $p(\theta)$ and define

$$T(s_{t-1}, a_t, s_t) = \int_\theta \delta_{T^\theta_{\text{det}}(s_{t-1}, a_t)}(s_t)\, p(\theta) d\theta$$

where δ denotes the Dirac delta function. Marginalizing over θ introduces uncertainty into the transition model, while still guaranteeing that s_t is feasible [Duff et al., 2010, Duff, 2011].

As we explain in section 8.3.3, this approximation ignores correlation of error between time steps. We discuss how to address this limitation, by estimating θ as state, in section 8.2.4.

3.2 *Observation Model for Proprioception*

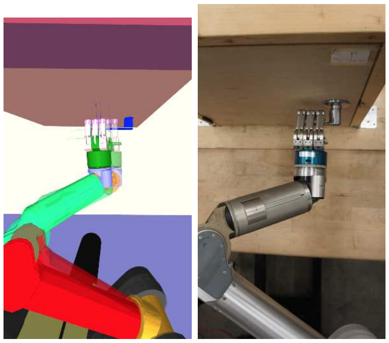


Figure 3.3: Proprioception error on the Barrett WAM arm due to uncertainty in the transmission between the arm's encoders and the joints. Courtesy of Klingensmith et al. [2013].

If a robot has accurate proprioception, then $p(o_q|q) = \delta_q(o_q)$ and $o_q = q$. Unfortunately, this is not true on the many robots that suffer from proprioceptive error.

Some robots, like the Barrett WAM [Salisbury et al., 1988], measure joint positions at the actuator instead of the joint. Uncertainty in the robot's transmissions introduces error into its proprioception [Klingensmith et al., 2013, Boots et al., 2014]. Figure 3.3 shows an example of the difference between the measured (solid render) and actual (semi-transparent render) configuration. Compliant mechanisms, such as the flexible rubber joints used in the iHY hand [Odhner et al., 2014], have an infinite number of unobservable degrees of freedom.

In general, we expect o_q to differ from q by the affine model

$$o_q = q + \delta q + \epsilon_q(s)$$

where δq is a history-dependent offset [Boots et al., 2014] and ϵ_q is state-dependent noise. We add δq as an auxiliary state variable and assume its dynamics $\delta q_t \sim p(\delta q_t|s_{t-1}, a_t)$ are known.

3.3 *Observation Model for Contact Sensors*

Contact sensors are unique because they provide a wealth of information while in contact, but little information otherwise. As in prior work [Javdani et al., 2013, Hebert et al., 2013, Petrovskaya and Khatib, 2011], we assume that contact sensors are *discriminative*; i.e. unlikely to generate false positive observations of contact.

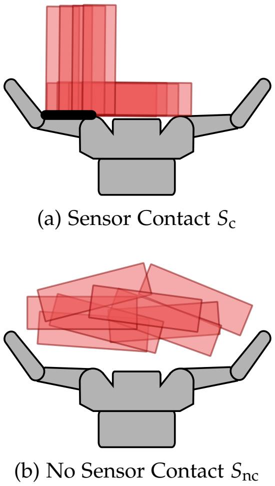
Consider a robot with n_s contact sensors. Each sensor i covers a portion $G_{s,i}(q) \subseteq \partial G_r(q)$ of the surface of the robot, where $G_r(q) \subseteq \mathbb{R}^3$ is the geometry of the robot in configuration q and ∂ denotes the boundary of a set. Define the *contact manifold of sensor i* to be

$$S_{c,i} = \{(q, x) \in S : G_{s,i}(q) \cap \partial G_o(x) \neq \emptyset\},$$

the set of states where the geometry of the movable object $G_o(x) \subseteq \mathbb{R}^3$ is in non-penetrating contact with the geometry of the sensor.

Each contact sensor also partitions the observation space. Let $O_{c,i} \subseteq O_s$ denote the set of contact sensor observations that indicate contact with sensor i. A sensor is *discriminative* if receiving an observation from that set implies that the state lies on the sensor's contact manifold with high probability; i.e. $\Pr(s_t \in S_{c,i}|o_t \in O_{c,i}) > 1 - \epsilon_s$.

When $o_s \in O_{c,i}$, sensor i may provide additional information about the state in the form of the likelihood $p(o_s|s, a)$ over $S_{c,i}$. For example, measurements of force/torque [Haidacher, 2004] and joint effort [Dogar et al., 2010, Manuelli and Tedrake, 2016] constrain the



(a) Sensor Contact S_c

(b) No Sensor Contact S_{nc}

Figure 3.4: The state space is partitioned into (a) states in contact with a sensor and (b) those that are not.

set of feasible contact points. We refer to the set of all such observations $O_c = \bigcup_{i=1}^{n_s} O_{c,i}$ as *contact observations* and use $\Lambda(o_s) \subseteq [1, n_s]$ to denote which sensors are active for an observation $o_s \in O_c$.

When $o_s \in O_{nc} = O_s \setminus O_c$, a contact sensor induces a uniform likelihood function $p(o_s|s,a)$ over S_{nc}. This property encodes the fact that receiving a *no-contact observation* provides little information.

In some cases, we ignore the distinction between contact sensors by allowing O_c to induce a likelihood over the *contact manifold*

$$S_c = \{(q,x) \in S : \partial G_r(q) \cap \partial G_o(x) \neq \emptyset\},$$

the set of states where the movable object is in non-penetrating contact with the robot. Discriminative sensors only assign non-zero likelihood to the *observable contact manifold* $S_{c,*} = \bigcup_{i=1}^{n_s} S_{c,i} \subseteq S_c$.

Chapter 6 uses this simplification. It also appeared in previous presentations of the manifold particle filter [Koval et al., 2013b,a, 2015b, 2016c].

This simplification ignores the fact that receiving a contact observation $o_s \in O_c$ constrains state to a set $S_{c,o_s} = \bigcap_{i \in \Lambda(o_s)} S_{c,i}$, which may be of lower dimension than S_c. We discuss the theory behind mixing probability distributions of different dimensionalities in section 8.3.1.

Each independent contact point reduces the dimensionality of S_{c,o_s} by one.

3.4 Value and Reward Functions

We encode our desire to reach the goal region (fig. 3.5) into a value function V^π that can be optimized. One intuitive choice of V^π is the negative time required for π to reach G with probability greater than $1 - \epsilon_g$. Unfortunately, this function is undefined for any π that is incapable of reaching G with the desired confidence level.

We negate the time required to reach G to convert the maximization of value into the minimization of time.

Another intuitive choice of V^π is the probability $\Pr(s_T \in G)$ of reaching G after taking T actions. This choice is flawed because there is no incentive for a policy to reach G more quickly than T time steps.

Instead, we encode our goal into a *reward function* $R : S \times A \to \mathbb{R}$ that encourages the robot to quickly reach G. We define

$$R(s_{t-1}, a_t) = \begin{cases} 0 & : s_{t-1} \in G \\ -\Delta t & : \text{otherwise} \end{cases}$$

by assigning zero reward to G and negative reward to other states.

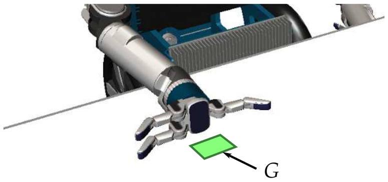


Figure 3.5: The reward function induces a value function that drives the robot towards the goal region $G \subseteq S$.

Our objective is to optimize the sum of expected future reward

$$V^\pi[b(s_0)] = E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(s_{t-1}, a_t)\right] \tag{3.1}$$

where the expectation $E[\cdot]$ is taken over the initial state $s_0 \sim b(s_0)$, transitions $s_t \sim p(s_t|s_{t-1}, a_t)$, and observations $o_t \sim p(o_t|s_t, a_t)$ assuming that actions are selected by $a_t = \pi[b(s_{t-1})]$. The discount factor $0 \leq \gamma < 1$ mediates between quickly reaching G with low probability and slowly reaching G with high probability.

Defining V^π as we did in eq. (3.1) makes our model a *partially observable Markov decision process* (POMDP) [Smallwood and Sondik, 1973, Kaelbling et al., 1998]. This allows chapters 6 and 7 to leverage the rich history of prior work on POMDP solvers, which would not be possible if V^π were not defined in terms of a reward function.

3.5 *Simplified Models*

Our problem formulation allows for a movable object, object pose uncertainty, and proprioceptive error. This thesis considers three simplified versions of the full problem:

- **Model 1:** *Hand-Relative* The robot is a lone end effector actuated by an incorporeal planar joint with perfect proprioception. This is equivalent to setting $S_{\text{obs}} = \emptyset$, $Q = \text{SE}(2)$, and $p(o_{\text{q}}|q) = \delta_q\left(o_{\text{q}}\right)$. We use this model to estimate object pose in chapter 4, plan end-effector motion in chapter 6, and to build a heuristic in chapter 7.

- **Model 2:** *Known, Static Environment* The robot moves in a known environment that contains no movable objects. This is equivalent to assuming that $b(x_0) = \delta_{x_{\text{static}}}(x_0)$ and $p(x_t|x_{t-1}, a_t) = \delta_{x_{t-1}}(x_t)$. for some static pose $x_{\text{static}} \in X$. We use this model to estimate the configuration of the robot in chapter 5.

- **Model 3:** *Perfect Proprioception* The robot has perfect proprioception, i.e. $p(o_{\text{q}}|q) = \delta_q\left(o_{\text{q}}\right)$. We use this model to plan with object pose uncertainty and kinematic constraints in chapter 7.

We discuss how to relax these assumptions in chapter 8.

4
Object Pose Estimation

This chapter is adapted from Koval et al. [2013b,a, 2016c, 2015b].

Estimating state is critical for a robot to reliably manipulate its environment. We specifically consider *recursive Bayesian estimation*, where we recursively construct the belief state

$$b(s_t) = \eta \, p(o_t | s_t, a_t) \int_S p(s_t | s_{t-1}, a_t) b(s_{t-1}) \, ds_{t-1} \tag{4.1}$$

from the previous belief state $b(s_{t-1})$ using the most recent action a_t and observation o_t [Thrun et al., 2005].

Implementing a *Bayes filter* of this form is possible because the stochastic dynamic system we defined in chapter 3 satisfies the *Markov property*. This property requires state to be a sufficient statistic for all previous actions and observations (fig. 4.1).

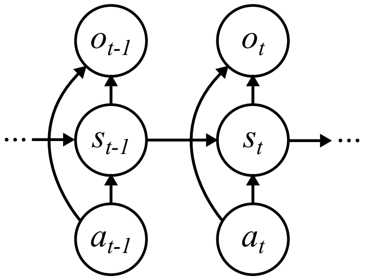


Figure 4.1: Bayesian network for the state estimation problem. The Markov property states that the future is independent of the past given the current state s_t.

This chapter introduces the *manifold particle filter* as an efficient implementation of the Bayes filter for contact sensing. Our key insight is to use the contact manifold (insight 1) to reduce the number of particles required to maintain an accurate state estimate.

We restrict ourselves to model 1 from section 3.5 by estimating the pose of an object relative to the end-effector. We extend the manifold particle filter to robot configuration estimation in chapter 5.

4.1 *Particle Filter*

Implementing a Bayes filter requires committing to a parameterization of the belief state that is closed under application of eq. (4.1).

The Kalman filter [Kalman, 1960] requires the belief state to be Gaussian, the transition model to be linear, and the observation model to be corrupted by additive Gaussian white noise. The extended and unscented [Julier and Uhlmann, 1997] Kalman filters relax the requirements on the transition and observation models, but still require the belief state to be Gaussian.

As a result, these filters are a poor fit for the complex belief states (challenge 1) found in manipulation. Instead, we use the *particle filter*

Algorithm 4.1 Conventional Particle Filter

```
Input: action $a_t \in A$ and observation $o_t \in O$
Input: particles $S_{t-1} = \{\langle s_{t-1}^{[i]}, w_{t-1}^{[i]} \rangle\}_{i=1}^{n}$ s.t. $S_{t-1} \sim b(s_{t-1})$
Output: particles $S_t = \{\langle s_t^{[i]}, w_t^{[i]} \rangle\}_{i=1}^{n}$ s.t. $S_t \sim b(s_t)$
$S_t \leftarrow \emptyset$
for $i = 1, \ldots, n$ do
    $s_t^{[i]} \sim p(s_t^{[i]} | s_{t-1}^{[i]}, a_t)$                ▷ eq. (4.3)
    $w_t^{[i]} \leftarrow w_{t-1}^{[i]} p(o_t | s_t^{[i]}, a_t)$       ▷ eq. (4.4)
    $S_t \leftarrow \{\langle s_t^{[i]}, w_t^{[i]} \rangle\} \cup S_t$
end for
$S_t \leftarrow$ RESAMPLE$(S_t)$
```

that represents the belief state as

$$b(s_t) \approx \sum_{i=1}^{n} w_t^{[i]} \delta_{s_t^{[i]}}(s_t),$$

the weighted sum of n probability point masses [Gordon et al., 1993]. The points, along with their weights, are called *particles* $S_t = \{\langle s_t^{[i]}, w_t^{[i]} \rangle\}_{i=1}^{n}$ and are chosen such that $S_t \sim b(s_t)$.

We say that S_t is "distributed according to $b(s_t)$" if eq. (4.2) is true for all functions f. We abuse notation to denote this by $S_t \sim b(s_t)$.

We can approximate the expectation of any function $f : S \to \mathbb{R}$ as

$$E_{s_t \sim b(s_t)}[s_t] = \int_S f(s_t) b(s_t) ds_t \approx \sum_{i=1}^{n} w_t^{[i]} f\left(s_t^{[i]}\right) \tag{4.2}$$

by replacing the integral over S with a summation over S_t. Critically, this statement holds when f is the value function used for planning.

Ideally, we would construct S_t by drawing samples from the *target distribution* given by eq. (4.1). Sampling from this distribution is challenging, so the particle filter instead draws samples from a *proposal distribution* $s_t^{[i]} \sim q(s_t)$. It corrects for the mismatch between $q(s_t)$ and $b(s_t)$ by computing an *importance weight* $w_t^{[i]} \sim b(s_t)/q(s_t)$ for each sample. This technique is known as *importance sampling*.

Finally, the particle filter periodically re-samples n particles with replacement from S_t with probability proportional to their importance weights. This step is known as *re-sampling* and is necessary to prevent growth in the variance of the importance weights.

4.1.1 Conventional Proposal Distribution

Importance sampling is valid for any choice of proposal distribution that satisfies $b(s_t) > 0 \implies q(s_t) > 0$. However, in practice, we must choose a proposal distribution that allows us to efficiently: (1) draw a sample $s_t^{[i]} \sim q(s_t)$ from the proposal distribution and (2) compute its importance weight $w_t^{[i]} = b(s_t)/q(s_t)$.

Conventionally we choose the proposal distribution to be

$$q(s_t) = \int_S p\left(s_t | s_{t-1}, a_t\right) b(s_{t-1}) ds_{t-1}, \tag{4.3}$$

the belief state after taking action a_t, but before receiving observation o_t. The importance weight for a sample $s_t^{[i]}$ drawn from eq. (4.3) is

$$w_t^{[i]} = \frac{b(s_t^{[i]})}{q(s_t^{[i]})} = \eta\, p(o_t | s_t^{[i]}, a_t), \tag{4.4}$$

which integrates observation o_t into the posterior belief state.

The *conventional particle filter* (CPF, algorithm 4.1) draws n samples from eq. (4.3) by propagating each particle from time $t-1$ through the transition model to produce a particle at time t (line 3). Next, it computes each particle's importance weight (line 4) by evaluating observation likelihood function. Finally, it periodically re-samples n particles with replacement (line 7) to normalize their weights.

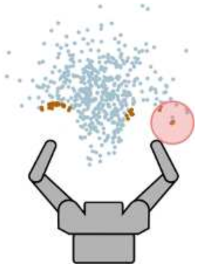


Figure 4.2: Only the small set of particles (dark orange) that are in the swept volume of the sensors generate contact observations. Most particles (light blue) generate no-contact observations and have low weight during contact.

4.1.2 *Degeneracy of the Conventional Proposal Distribution*

Unfortunately, importance sampling from the conventional proposal distribution performs poorly when using contact sensors. Contact sensors are discontinuous (challenge 1) and only respond to state on a lower-dimensional contact manifold. As a result, the conventional proposal distribution is a poor approximation of the target distribution when $o_s \in O_c$. This cause *particle deprivation*, a condition where no particles in S_t agree with o_t with high probability (fig. 4.2).

Surprisingly, this fact causes the CPF to *perform worse as the sensor resolution increases*. Figure 4.3 illustrates the reason for this unintuitive result. As sensor resolution increases (left-to-right), the swept volume of each sensor becomes narrower. As the update frequency increases (top-to-bottom), the distance traveled by the end-effector between updates decreases and the swept volume shrinks.

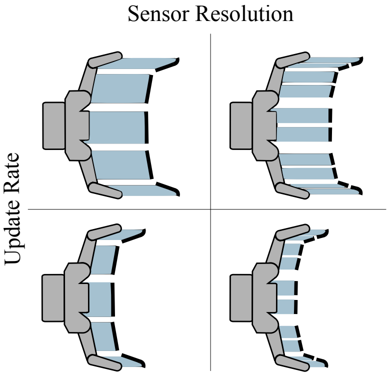


Figure 4.3: Increasing the sensor's resolution (left-to-right) or update rate (top-to-bottom) reduces the swept volume of the sensors. This exacerbates the problem of particle starvation.

4.1.3 *Dual Proposal Distribution*

Alternatively, we can draw samples from the *dual proposal distribution*

$$\bar{q}(s_t) = \frac{p(o_t | s_t, a_t)}{p(o_t | a_t)} \tag{4.5}$$

As in prior work, we assume that $p(o_t|a_t)$ is finite [Thrun et al., 2000b].

to generate a particle $\bar{s}_t^{[i]} \sim \bar{q}(s_t)$ that is consistent with the latest observation o_t. Equation (4.5) ignores $b(s_{t-1})$ and generates samples that are likely to produce observation o_t.

The corresponding *dual importance weight* is

$$\bar{w}_t^{[i]} == \frac{b(\bar{s}_t^{[i]})}{\bar{q}(\bar{s}_t^{[i]})} = \eta' \int_S p(\bar{s}_t^{[i]} | s_{t-1}, a_t) b(s_{t-1}) ds_{t-1} \tag{4.6}$$

Algorithm 4.2 Manifold Particle Filter

```
Input: number of dual particles $n_\text{d}$ and mixing rate $\phi$
Input: action $a_t \in A$ and observation $o_t \in O$
Input: particles $S_{t-1} = \{\langle s_{t-1}^{[i]}, w_{t-1}^{[i]} \rangle\}_{i=1}^{n}$
Output: particles $S_t$ s.t. $S_t \sim b(s_t)$
$S_{\text{cpf},t} \leftarrow \text{CPF}(S_{t-1}, a_t, o_t)$    ▷ algorithm 4.1
for $i = 1, \ldots, n_\text{d}$ do
    $\bar{s}_t^{[i]} \sim p(o_t | s_t^{[i]}, a_t)$    ▷ section 4.2.2
    $\bar{w}_t^{[i]} = \int_S p(\bar{s}_t^{[i]}, s_{t-1}, a_t) b(s_{t-1})\, ds_{t-1}$    ▷ section 4.2.3
    $\bar{S}_t \leftarrow \{\langle \bar{s}_t^{[i]}, \bar{w}_t^{[i]} \rangle\} \cup \bar{S}_t$
end for
$S_t \leftarrow (1-\phi) S_{\text{cpf},t} + [1 - \phi + \phi b(s_t \in S_\text{nc})](S_{\text{cpf},t} \cap S_\text{nc})$
        $+ \phi b(s_t \in S_\text{c})(\bar{S}_t \cap S_\text{c})$    ▷ section 4.2.4
$S_t \leftarrow \textsc{Resample}(S_t)$
```

with normalization factor η'. Dual importance weights incorporates the outcome of executing action a_t in belief state $b(s_{t-1})$ into $b(s_t)$.

4.2 *Manifold Particle Filter*

Sampling from the conventional proposal distribution performs well when the transition model is more informative than the observation model. Conversely, sampling from the dual proposal distribution performs best when the observation model is informative. Neither case directly applies to contact sensors.

Contact sensors provide a wealth of information while in contact, but little information while in free space (challenges 1 and 2). Our key insight is exploit this fact by factoring the belief state $b(s_t)$ into

$$b(s_t) = b(s_t \in S_\text{nc}) b(s_t | S_\text{nc}) + \sum_{o_\text{s} \in O_\text{c}} b(s_t \in S_{\text{c},o_\text{s}}) b(s_t | S_{\text{c},o_\text{s}}) \qquad (4.7)$$

where $b(s_t|S_\text{nc})$ is a probability distribution over free space and $b(s_t|S_{\text{c},o_\text{s}})$ is a probability distribution over the states in contact with all of the sensors $\Lambda(o_\text{s})$ active in observation o_s. The marginal probabilities $b(s_t \in S_\text{nc})$ and $b(s_t \in S_{\text{c},o_\text{s}})$ define the probability of the state residing in each of those sets.

The *manifold particle filter* (MPF, algorithm 4.2) samples in three steps. First, the MPF chooses a manifold in proportion to its marginal probability. Next, it uses importance sampling to draw a sample from $b(s_t)$ restricted to that set. We tailor our choice of proposal distribution to the current set: we choose the conventional proposal distribution for S_nc and the dual proposal distribution for all of the contact manifolds. Finally, we repeat this process n times such that the number of particles remains constant.

Algorithm 4.2 differs from this description of the MPF because it includes the optimization from section 4.2.4.

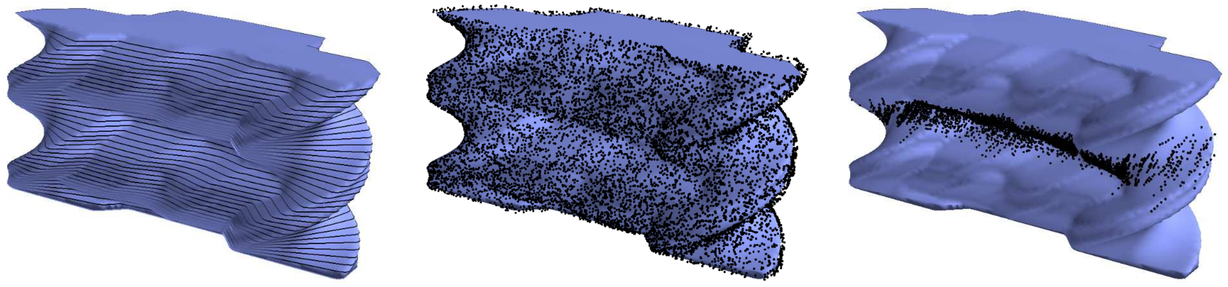

(a) Analytic (b) Rejection-Sampled (c) Trajectory Rollout

Figure 4.4: (a) Analytic, (b) rejection sampling, and (c) trajectory rollout representations of the contact manifold.

The MPF combines the advantages of the conventional and dual proposal distributions. However, there are three key challenges in implementing the MPF. First, we must estimate the marginal $b(s_t \in S_{c,o_s})$. Second, we must sample from the dual proposal distribution. Third, we must compute the dual importance weights. We address these challenges in the next three sections.

4.2.1 *Estimating the Probability of Contact*

Factoring $b(s_t)$ requires estimating the probability of contact $b(s_t \in S_{c,o_s})$ for each combination of sensors that may be active for any contact observation $o_s \in O_c$. We typically would compute

$$b(s_t \in S_{c,o_s}) = \int_{S_{c,o_s}} b(s_t)\, ds_t$$

by marginalizing the belief state over S_{c,o_s}. Unfortunately, computing this integral requires precisely the distribution we wish to estimate.

Given $b(s_t \in S_{c,o_s})$, we can compute $b(s_t \in S_{nc}) = 1 - \sum_{o_s \in O_c} b(s_t \in S_{c,o_s})$.

Instead, we set $b(s_t \in S_{c,o_{s,t}}) = 1$ for the most recent contact observation $o_{s,t}$ and $b(s_t \in S_{c,o_s}) = 0$ for all $o_s \neq o_{s,t}$. This assumption is correct when contact sensors are perfectly discriminative (section 3.3) and a good approximation of the true belief dynamics when false positives are unlikely, i.e. when ϵ_s is small.

4.2.2 *Sampling from the Dual Proposal Distribution*

The MPF samples from the dual proposal distribution, restricted to the contact manifold $S_c \subseteq S$, when contact is observed. This section introduces three methods of constructing *explicit representations* of S_c that are amenable to sampling. Chapter 5 describes an *implicit representation* of S_c that scales to high-dimensional state spaces.

This section simplifies notation by using S_c to refer to an arbitrary contact manifold. In practice, we build a separate representation of S_{c,o_s} for each contact observation $o_s \in O_c$.

In some cases, e.g. when the end-effector and object are polygons, we can compute an *analytic representation* of S_c. We use the Minkowski sum to compute the C-obstacle of the end-effector in the configuration space of the object at a fixed orientation [Lozano-Pérez, 1983]. We repeat this computation for many orientations of the object and approximate S_c as the union of the boundaries of those C-obstacles (fig. 4.4a). We draw a sample from S_c by first choosing an orientation, then choosing a position along the boundary of the corresponding C-obstacle.

It often is not feasible to represent S_c analytically. Instead, we can approximate S_c with a discrete set of states $\tilde{S}_c \subseteq S_c$ on the manifold (fig. 4.4b). The most straightforward way of building $\tilde{S}_c$ is through rejection sampling. *Rejection sampling* begins by drawing a candidate state $s \sim \text{uniform}(S)$ from the ambient space. If $\min_{s' \in S_c} ||s - s'|| \leq \epsilon_{\text{rs}}$, then s is added to $\tilde{S}_c$. Otherwise, it is rejected. This process repeats until $\tilde{S}_c$ is sufficiently large. The parameter $\epsilon_{\text{rs}} > 0$ trades off between the speed of building $\tilde{S}_c$ and its accuracy.

Ideally we would set $\epsilon_{\text{rs}} = 0$ and rejection sample from S_c. This is not possible because S_c has zero measure in the ambient space: there is zero probability of drawing such a sample.

Rejection sampling covers S_c with uniform density. As a result, much of $\tilde{S}_c$ may be found in regions of S_c that remain low probability during execution. We can perform *trajectory rollouts* from the initial belief state to approximate the distribution of states that will be encountered during execution. Each time we encounter a state $s \in S_c$ during a rollout, we add it to $\tilde{S}_c$. Just as with rejection sampling, we repeat this process until $\tilde{S}_c$ is sufficiently large.

4.2.3 *Computing Dual Importance Weights*

Once we have drawn a sample $\bar{s}_t^{[i]}$ from the dual proposal distribution, we must compute its importance weight $\bar{w}_t^{[i]}$. Recall from section 4.1.3 that the importance weight integrates $b(s_{t-1})$ and the effect of taking action a_t into the posterior [Thrun et al., 2000a].

We propagate each particle $s_t^{[i]}$ from time $t-1$ to time t using the transition model $s_t^{[i]} \sim p(s_t^{[i]} | s_{t-1}^{[i]}, a_t)$. This set of particles $S_{t-1}^+ = \{\langle s_t^{[i]}, w_{t-1}^{[i]} \rangle\}_{i=1}^n$ is distributed according to $b^+(s_{t-1})$, the belief state after taking action a_t, but before receiving observation o_t. The importance weight defined by eq. (4.6) can be re-written as $\bar{w}_t^{[i]} = b^+(\bar{s}_t^{[i]})$, the density of this belief state at particle i.

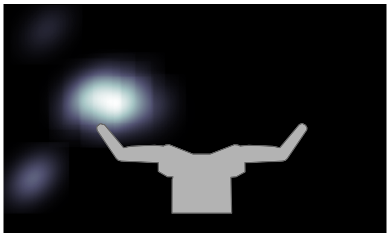


Figure 4.5: Kernel density estimate used to compute dual importance weights.

We use kernel density estimation [Rosenblatt, 1956] to estimate $b^+(\bar{s}_t^{[i]})$ by promoting S_{t-1}^+ into a probability density function

$$b^+(s) \approx \sum_{i=1}^n w_{t-1}^{[i]} K(s, s_t^{[i]}),$$

where $K : S \times S \to \mathbb{R}^+$ is a kernel. We use this estimate to compute the importance weight for each sample drawn from eq. (4.5).

We choose K to be a Gaussian kernel and select the bandwidth matrix using a multivariate generalization of Silverman's rule of thumb [Silverman, 1981]. Note that K will assign non-zero probability to S_{nc} because it is defined in the ambient space. This is not issue because $b^+(s)$ is only evaluated on samples drawn from $\bar{q}(s_t | S_c)$.

4.2.4 *Mixture Proposal Distribution*

There are two potential issues with the MPF as described above.

First, the conventional and dual proposal distributions have complementary strengths and weaknesses. Just as the how the conventional proposal distribution performs poorly with accurate sensors, the dual proposal distribution responds poorly to observation noise [Thrun et al., 2000a]. The MPF uses the dual proposal distribution during contact and, thus, inherits the same weakness.

Second, the kernel density estimate used to compute dual importance weights introduces variance into the posterior belief state. This is unavoidable: the technique replaces a point estimate of a probability distribution with a sum of kernel functions that has broader support. If left unchecked, this variance could grow over time.

We use a *mixture proposal distribution* [Thrun et al., 2000a] to mitigate these effects. Instead of sampling all of the particles from the MPF, some particles $S_{\text{cpf},t}$ from the CPF and the remainder of the particles $S_{\text{mpf},t}$ from the MPF. We combine the two sets of particles $S_t = (1-\phi)S_{\text{cpf},t} + \phi S_{\text{mpf},t}$ weighted by a *mixing rate* $0 \leq \phi \leq 1$.

We define the sum $aX + cY$ of the sets of particles $X = \{\langle x^{[i]}, w_x^{[i]}\rangle\}_{i=1}^{n_x}$ and $Y = \{\langle y^{[i]}, w_y^{[i]}\rangle\}_{i=1}^{n_y}$ with non-negative scale factors $a, c \in \mathbb{R}^{\geq 0}$ to be $aX + cY = \{\langle x^{[i]}, aw_x^{[i]}/W_x\rangle\}_{i=0}^{n_x} \cup \{\langle y^{[i]}, cw_y^{[i]}/W_y\rangle\}_{i=0}^{n_y}$. The variables $W_x = \sum_{i=1}^{n_x} w_x^{[i]}$ and $W_y = \sum_{i=1}^{n_y} w_y^{[i]}$ denote the total weight of X and Y.

Implementing the mixture proposal distribution typically incurs the computational expense of running both filters at once. We avoid this overhead by integrating the mixing step into the MPF. To show how this is possible, rewrite the mixture proposal distribution as

We use $S_{\text{cpf},t}$ to denote particles drawn from $q(s_t)$, S_t to denote particles drawn from $q(s_t|S_{\text{nc}})$, and $\bar{q}(s_t)$ to denote particles drawn from $\bar{q}(s_t|S_{\text{c}})$.

$$
\begin{aligned}
S_t &= (1-\phi)S_{\text{cpf},t} + \phi\left[b(s_t \in S_{\text{c}})\bar{S}_t + b(s_t \in S_{\text{nc}})S_t\right] \\
&= (1-\phi)S_{\text{cpf},t} + \phi\left[b(s_t \in S_{\text{c}})\bar{S}_t + b(s_t \in S_{\text{nc}})(S_{\text{cpf},t} \cap S_{\text{nc}})\right] \\
&= (1-\phi)(S_{\text{cpf},t} \cap S_{\text{c}}) + \left[1 - \phi + \phi\, b(s_t \in S_{\text{nc}})\right](S_{\text{cpf},t} \cap S_{\text{nc}}) \\
&\qquad + \phi\, b(s_t \in S_{\text{c}})\bar{S}_t
\end{aligned}
$$

by partitioning $S_{\text{cpf},t}$ into particles $S_{\text{cpf},t} \cap S_{\text{c}}$ on the contact manifold and those $S_{\text{cpf},t} \cap S_{\text{nc}}$ that are not.

Our key observation is that the MPF draws particles from free space using $q(s|S_{\text{nc}})$, the same distribution $q(s)$ used by the CPF, except restricted to S_{nc}. Instead of repeating this computation twice, we substitute $S_t = S_{\text{cpf},t} \cap S_{\text{nc}}$. We also reuse $S_{\text{cpf},t}$ to build S_{t-1}^{+}, which is required to compute the dual importance weights.

Instead of parameterizing the algorithm by a total number of particles n, we specify the numbers of particles $n_{\text{c}} = |S_{\text{cpf},t}|$ to draw from $q(s_t)$ and the number of particles $n_{\text{d}} = |S_{\text{mpf},t} \cap S_{\text{c}}|$ to draw from $\bar{q}(s_t|S_{\text{c}})$. These particles can be thought of as, respectively, the number of particles necessary to cover the high-probability regions of free space and the contact manifold. The mixing rate smoothly transitions between the CPF ($\phi = 0$) and the MPF ($\phi = 1$).

Despite this, the MPF maintains a single set of particles: it is not meaningful to identify whether a particle was sampled from the conventional or dual proposal distribution. After each time step, both sets of particles are seamlessly mixed into the posterior distribution.

4.3 Simulation Experiments

This section compares the MPF with the CPF at estimating object pose through simulation experiments. Because of the particle deprivation problem described in section 4.1.2, we hypothesize that:

H1. *The MPF will outperform the CPF after contact.*

Increasing the contact sensor resolution or update rate should exacerbate the particle deprivation. Therefore, we hypothesize:

H2. *The CPF will perform worse as sensor resolution increases; the MPF will perform better.*

H3. *The CPF will perform worse as the sensor update rate increases; the MPF will perform better.*

All of the above hypotheses should be true regardless of which representation of the contact manifold is used to implement the MPF.

Section 4.2.2 introduced the analytic (AM), rejection-sampled (RS), and trajectory rollout (TR) representations of S_c as tools for drawing samples from the dual proposal distribution. Since AM is the only representation that avoids discretization, we hypothesize that:

H4. *The MPF will perform best with the analytic representation.*

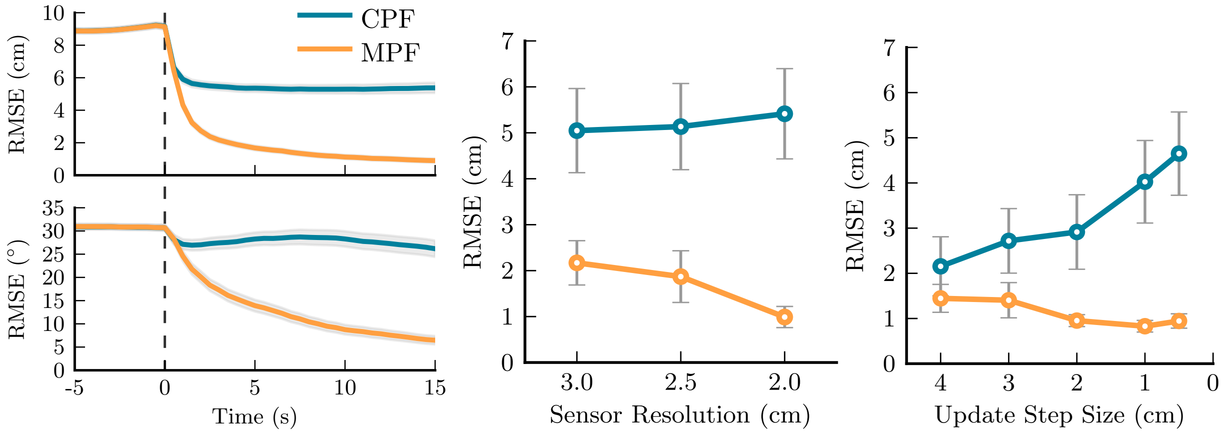
Surprisingly, our results suggest that H4 is false: MPF-TR outperforms MPF-AM. It does so by focusing samples on the subset of S_c that is reachable from the initial belief state.

Similar to AM, RS attempts to represent S_c with uniform resolution. Therefore, for the same reason as above, we hypothesize:

H5. *The MPF will perform better with the trajectory rollout representation than the rejection-sampled representation.*

4.3.1 Experimental Design

We implemented the CPF and MPF in a two-dimensional simulation environment with polygonal geometry. Each trial consisted of a simulated BarrettHand pushing a rectangular box in a straight line at a speed of 1 cm/s for 50 s. The initial belief state was set to $b(s_0) = \mathcal{N}(\bar{s}_0, \Sigma_0)$ with variance $\Sigma_0^{1/2} = \text{diag}[5\text{ cm}, 5\text{ cm}, 20°]$. The mean $\bar{s}_0 = (\bar{x}_0, \bar{y}_0, \bar{\theta}_0)$ was placed a fixed distance $\bar{x}_0 = 20$ cm from the end-effector and at lateral offset $\bar{y}_0 \sim \text{uniform}[-10\text{ cm}, 10\text{ cm}]$ and orientation $\bar{\theta}_0 \sim \text{uniform}[0°, 360°]$.



(a) Estimation Error

(b) Spatial Resolution

(c) Temporal Resolution

Figure 4.6: Comparison between the CPF and the MPF-AM in simulation. (a) The CPF and MPF perform identically before contact, but the MPF outperforms the CPF post-contact. (b) The MPF improves as spatial sensor resolution increases, whereas the CPF declines in performance. (c) Similarly, the MPF improves and the CPF declines when faced with a faster update frequency. Error bars indicate a 95% confidence interval.

We simulated the motion of the object using a penetration-based quasistatic physics model [Lynch et al., 1992] with a 1 mm step size. Before each step, the finger-object coefficient of friction μ_f and the radius of the object's pressure distribution c were sampled from the Gaussian distributions $\mu_f \sim \mathcal{N}(0.5, (0.2)^2)$ and $c \sim \mathcal{N}(5\text{ cm}, (1\text{ cm})^2)$, then truncated to enforce $\mu_f, c > 0$.

The same model simulated observations for contact sensors distributed uniformly across the front surface of the hand. The simulated contact sensors were perfectly discriminative, but had a 10% chance of generating an incorrect observation during contact.

We quantified accuracy by computing the *root mean square error*

$$\text{RMSE}(S_t, s_t^*) = \sqrt{\frac{\sum_{i=1}^{n}(s_t^{[i]} - s_t^*)^2 w_t^{[i]}}{\sum_{i=1}^{n} w_t^{[i]}}}$$

of the particles S_t with respect to the true state s_t^*. Instead of combining the position error (measured in centimeters) with the orientation error (measured in degrees), we report separate values for each.

4.3.2 *Conventional vs. Manifold Particle Filter (H1)*

We ran the CPF with $n = 100$ particles and the MPF with $n_c = 100$ conventional particles, $n_d = 25$ dual particles, and a mixing rate of $\phi = 0.1$. We intentionally chose the same value of $n = n_c$ for both algorithms—despite the addition of n_d dual particles for the MPF—because the dual sampling step adds negligible overhead to the runtime of the algorithm. The MPF sampled from an analytic representation of the contact manifold with a 1 mm linear and 1.15° angular resolution.

Figure 4.6a shows the performance of both filters averaged over 900 trials. These results show that—as expected—both filters behave

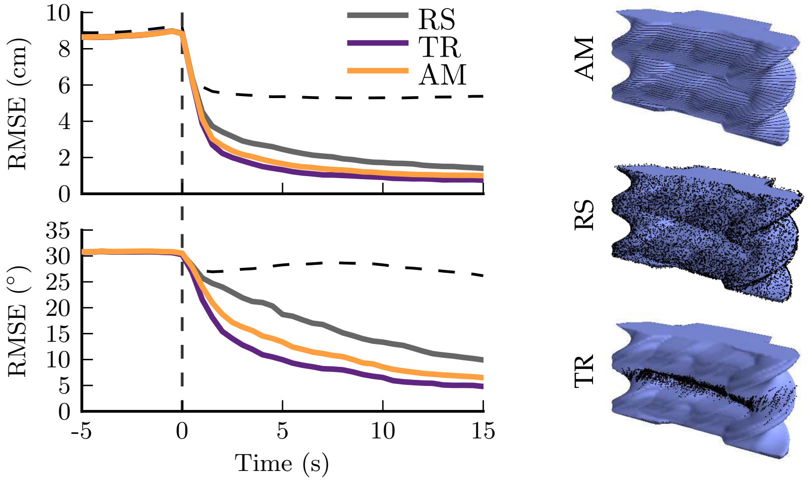


Figure 4.7: Performance of the MPF using the rejection-sampled (RS), trajectory-rollout (TR), and analytical (AM) manifold representations. The data is aligned such that contact occurs at $t = 0$. The performance of the CPF is drawn as a dashed lined.

similarly before contact ($t \leq 0$) and there was not a significant difference in RMSE. After contact ($t > 0$), the MPF quickly achieves 4.4 cm less RMSE than the CPF. This supports H1.

4.3.3 *Spatio-Temporal Sensor Resolution (H2 and H3)*

Next, we evaluated the effect of sensor resolution on estimation accuracy by varying the resolution of binary contact sensors. In all cases, the sensors are distributed uniformly over the front surface of the hand. Figure 4.6b shows the relative performance of the CPF and MPF for three different resolutions averaged over 95 trials. As expected, the CPF performs worse as the spatial sensor resolution increases. In contrast, the MPF performs better. This supports H2.

We also varied the distance traveled between sensor updates from 5 mm to 4 cm. Since the hand was moving at a constant velocity, this corresponds to changing the sensor's update frequency. Figure 4.6c shows the performance of the CPF and MPF averaged over 95 trials. As expected, the CPF performs worse as the update frequency increases. The MPF performs better, supporting H3.

4.3.4 *Contact Manifold Representation (H4 and H5)*

We also compared the performance of the MPF using the RS, TR, and AM representations of the contact manifold. The RS representation consisted of 10,000 samples that were held constant across all trials. The TR representation generated a different set 10,000 samples for each trial by collecting five samples each from 2000 trajectory rollouts using the same physics model as used during execution. We built the AM representation using the parameters described in section 4.3.1.

Figure 4.7 shows the performance of the three representations averaged over 900 trials. The MPF outperformed the CPF with all three representations. As expected, the results support H5: MPF-AM and MPF-TR both outperform MPF-RS. This occurs because RS attempts to cover S_c at uniform density. In contrast, TR focuses the same number of samples on the smaller region of S_c that is encountered during execution.

Surprisingly, H4 is not supported: MPF-AM did not outperform MPF-TR. This is partially explained by same reasoning as above: the TR was able to densely cover the reachable subset of S_c at a resolution indistinguishable from that of AM. Additionally, we know that every sample drawn from TR must be reachable from the initial belief state. MPF-TR does not waste samples from the dual proposal distribution in regions of S_c that are known to be unreachable.

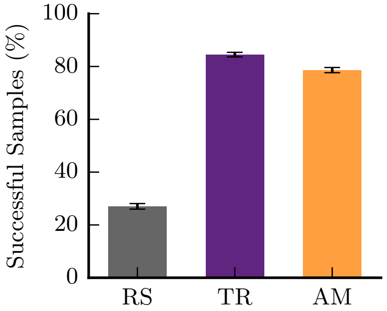


Figure 4.8: Percent of the time that the MPF succeeded at sampling from the dual proposal distribution during contact.

4.3.5 *Sampling Failures*

Our intuition is that the relatively poor performance of MPF-RS is caused by it frequently failing to sample from the dual proposal distribution. Sampling fails when all particles drawn from $\bar{q}(s_t|S_{c,o_{s,t}})$ have low probability $p(o_t|s_t, a_t)$ of generating o_t. In the case of binary contact sensors, a sampling failure typically occurs when several sensors are simultaneously active that were never observed to be simultaneously active while building the contact manifold representation.

Figure 4.8 shows the rate of sampling failures for MPF-AM, MPF-RS, and MPF-TR over 900 trials. We formally define a sampling failure as an update where $p(o_t|s_t, a_t) < 0.1$ for all samples from the dual proposal distribution. Under this metric, MPF-TR and MPF-AM fail to sample from the dual proposal distribution for $< 30\%$ of time steps. Conversely, the MPF-RS fails to sample $> 70\%$ of time steps.

When sampling fails the MPF behaves identically to the CPF. As a result, MPF-RS performs poorly compared to MPF-AM and MPF-TR.

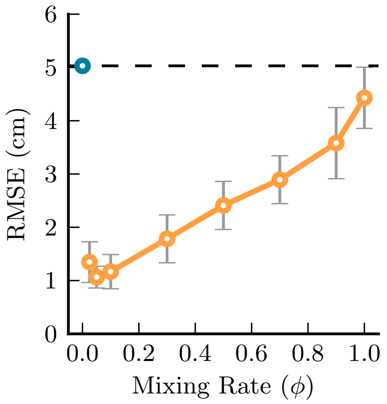


Figure 4.9: Performance of the MPF-AM as a function of the mixing rate $0 \le \phi \le 1$.

4.3.6 *Mixing Rate*

In addition to the choice of manifold representation, the MPF has a mixing rate parameter ϕ. We repeated the experiments described in section 4.3.2 while varying MPF-AM's mixing rate over the set $\phi = \{0.025, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. Note that $\phi = 1$ corresponds to the pure MPF. Figure 4.9 shows the post-contact performance of the MPF averaged over 150 trials and plotted as a function of ϕ. The performance of the CPF ($\phi = 0$) is plotted as a horizontal dotted line.

As expected, the MPF outperforms the CPF for all $\phi > 0$. Surprisingly, however, the optimal value of ϕ falls into the lowest range of values $0.025 \le \phi \le 0.1$ that we tested. Increasing ϕ out of this range

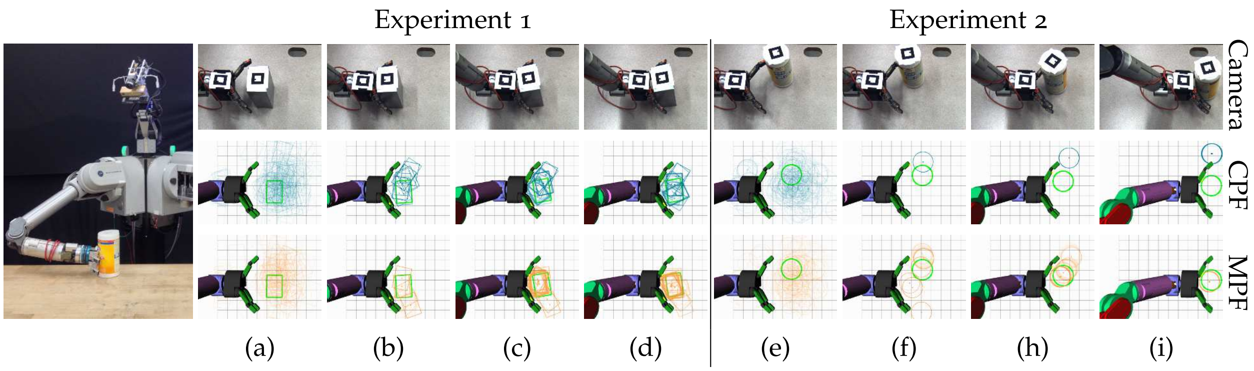


Figure 4.10: Andy pushing a box (a)–(d) and cylinder (e)–(i) across the table. The top row shows a video of the experiment from an overhead camera. The bottom two rows show the belief state estimated by the CPF (middle, dark blue) and MPF (bottom, light orange) as a cloud of particles. Ground truth is shown as a thick green outline.

leads to a predictable, linear increase in error. This occurs for two reasons. First, the dual proposal distribution performs poorly when there is observation noise [Thrun et al., 2000a]. Second, the MPF samples from an approximation of the dual proposal distribution that has higher variance than the true posterior belief. Reducing the mixing rate decreases the rate at which this variance grows.

4.4 *Real-Robot Experiments*

We evaluated the CPF and MPF on Andy [Bagnell et al., 2012], which used a Barrett WAM arm [Salisbury et al., 1988] equipped with the i-HY [Odhner et al., 2014] 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 the arrays of tactile sensors [Tenzer et al., 2014] shown in fig. 4.11. The tactels were grouped into 39 vertical stripes to to simplify the observation model.

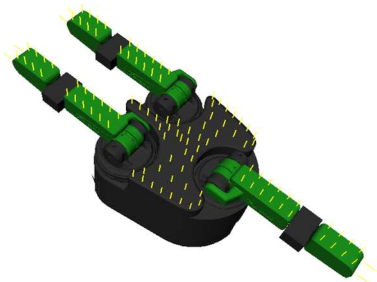


Figure 4.11: Tactile sensors (yellow lines) on the i-HY hand.

Figure 4.10 shows two representative runs of the state estimator on Andy. We tracked the ground-truth pose of the movable object with an overhead camera using a visual fiducial. Both filters were run with 250 particles, with $n = 250$ for the CPF and $n_c = 225$, $n_d = 25$, $\phi = 0.1$ 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 (b) made initial contact with the right proximal link and (c) rolled into the palm. The CPF did not have any particles consistent with the contact observation and, thus, failed to track the box as it (d) rolled into the palm. The MPF successfully tracked the box by sampling particles that agreed 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 (e) made initial contact with its left fingertip. The cylinder (f) rolled down the distal and (h) proximal links to finally (i) settle in the palm. 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 deprivation near the true state and was able to successfully track the container for the duration of contact.

4.5 *Discussion*

This chapter introduced the manifold particle filter as a Bayesian filter that exploits the contact manifold (insight 1) to cope with the discontinuity of contact (challenge 1). Our simulation (section 4.3) and real-robot (section 4.4) experiments show that the MPF outperforms the CPF at the task of estimating the pose of the movable object relative to the end-effector.

While the MPF is a principled realization of a Bayes filter, we had to make several approximations to implement it. We enumerate those assumptions here.

First, we replace the marginal probability of contact with an observation likelihood (section 4.2.1). This substitution is exact only when contact sensors are perfectly discriminative and is a good approximation when false positives are unlikely. Extending the MPF to noisy contact sensors would require another method of estimating contact state, e.g. using a contact state graph [Meeussen et al., 2007].

Second, we draw particles from an approximate representation of the contact manifold (section 4.2.2). This introduces error because the particles: (1) may lie slightly off the surface of the manifold, (2) are not uniformly distributed, and (3) are drawn from a subset of the manifold. Our experiments found (1) to be insignificant, (2) to be difficult to quantify due to the subtlety described in section 8.3.1, and (3) to be dominant. These errors are exacerbated when S is high-dimensional because fidelity of representation often is sacrificed for computational tractability. We ameliorate this in the next chapter.

Third, we approximate the dual importance weights using a kernel density estimate (section 4.2.3). Kernel density estimation increases the variance of the posterior distribution by promoting a probability distribution with finite support to one with broad support. When applied recursively, as in the MPF, this substitution causes the variance of the belief state to grow over time. We reduce the rate of, but can not eliminate, variance growth using the mixture proposal distribution (section 4.2.4). This growth in variance is the dominant source of error in the MPF and, as a result, our experiments show that a surprisingly low value of ϕ is optimal. Reducing this error requires reducing the variance of the kernel density estimate, e.g. through shrinkage [Liu and West, 2001], or eliminating the kernel density estimate entirely, e.g. using a Markov chain Monte Carlo method.

Finally, this chapter applied the MPF specifically to object pose estimation under model 1. The next chapter extends this algorithm to robot configuration estimation under model 2. We also discuss how the MPF could be extended to incorporate other modalities (section 8.2.5) estimate static parameters (section 8.2.4), and cope with un-modelled error (section 8.3.3) in future work.

5
Robot Configuration Estimation

This chapter is adapted from Klingensmith et al. [2016] and contains work done in collaboration with Matthew Klingensmith.

The manifold particle filter is a recursive Bayesian estimation technique that is applicable to any state space. However, our implementation of the algorithm in chapter 4 requires constructing an *explicit representation* of the contact manifold offline that is used to draw samples from the dual proposal distribution. Doing so is tractable for hand-relative manipulation (model 1 in section 3.5), where state consists only of the three-dimensional pose of the movable object.

However, as we described in chapter 1, many robots also suffer from proprioceptive uncertainty. On some robots, like the Barrett WAM arm [Salisbury et al., 1988] or a soft manipulator [Sanan et al., 2011], this error originates from the transmission that connects a position sensor to its joint. On others, like an under-actuated hand [Rig, 2016], there may be no mechanism of measuring position at all.

Unfortunately, it is often intractable to explicitly represent the contact manifold in a high-dimensional configuration space. Due to the curse of dimensionality, the number of samples required to represent the manifold scales exponentially with its dimension.

This chapter introduces an *implicit representation* of the contact manifold that enables the manifold particle filter to scale up. Our key insight is to use constraint projection to draw samples from the manifold that will be assigned high importance weights. We specifically consider model 2 from section 3.5 by estimating the configuration of a manipulator in a known, static environment.

5.1 *Implicitly Representing the Contact Manifold*

Recall that the contact manifold is the set of states where the robot is in non-penetrating contact with the object. We can write the contact manifold $S_{c,o}$ of observation o as the zero iso-contour

$$S_{c,o} = \{s \in S : f_o(s) = 0\}$$

of a constraint function $f_o : S \to \mathbb{R}^{n_\text{a}}$, where $n_\text{a} = |\Lambda(o)|$ is the number of sensors active in o. We choose this function to be

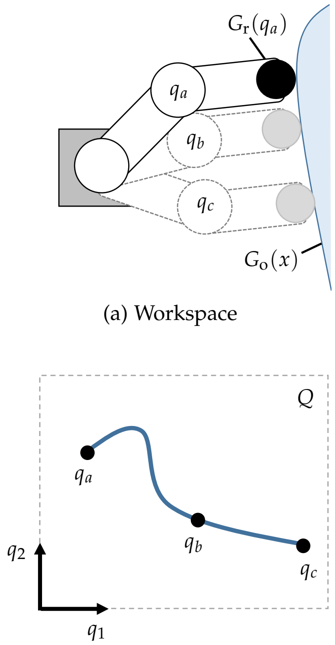
$$f_o(s) = \begin{bmatrix} \textsc{SDist}(G_{\text{s},i_1}(q), G_\text{o}(x)) \\ \textsc{SDist}(G_{\text{s},i_2}(q), G_\text{o}(x)) \\ \vdots \\ \textsc{SDist}(G_{\text{s},i_{n_\text{a}}}(q), G_\text{o}(x)) \end{bmatrix}, \tag{5.1}$$

where each entry is the signed distance between the geometry $G_{\text{s},i_j}(q) \subseteq \mathbb{R}^3$ of sensor $i_j \in \Lambda(o)$ in configuration q and the geometry $G_\text{o}(x) \subseteq \mathbb{R}^3$ of the movable object at pose x.

Equation (5.1) is defined in terms of the *signed distance function*

$$\textsc{SDist}(A, B) = \begin{cases} \text{dist}(A, B\ ) & : A \cap B = \emptyset \\ -\text{dist}(A, B^c) & : \text{otherwise} \end{cases} \tag{5.2}$$

that returns the positive distance $\text{dist}(A, B) = \min_{a \in A, b \in B} ||a - b||$ between two disjoint sets or the negative penetration depth between two intersecting sets.



(a) Workspace

(b) Configuration Space

Figure 5.1: (a) Three configurations of a robot that satisfy $f_o(s) = 0$ and, thus, (b) lie on the contact manifold.

5.2 *Sampling via Constraint Projection*

Implementing the MPF requires drawing samples from the dual proposal proposal restricted to $S_{\text{c},o}$. Given eq. (5.1) and its gradient, a root-finding algorithm can project a *seed state* $\hat{s}_t^{[i]} \in S$ from the ambient space onto $S_{\text{c},o}$ by finding a root of the equation $f_o(s) = 0$.

We use an iterative algorithm to use root-finding to generate a diverse set of samples from $S_{\text{c},o}$. Each iteration proceeds by: (1) generating a seed state $\hat{s}_t^{[i]}$ using a *projection strategy*, (2) using a root-finding algorithm to project $\hat{s}_t^{[i]}$ to a state $\bar{s}_t^{[i]} \in S_{\text{c},o}$ and (3) accepting the state if $f_o(\bar{s}_t^{[i]}) \leq \epsilon_\text{proj}$. We repeat this process until enough samples are available or the set of seed states is exhausted.

Projection may fail if no solution exists or the root-finding algorithm fails to find a solution in an acceptable time limit.

The uniformity of the samples drawn by this algorithm depends on the projection strategy. This thesis considers three strategies.

Uniform projection draws $\hat{s}_t^{[i]} \sim \text{uniform}\, S$. This strategy is unbiased with respect to S_{t-1} and has a non-zero probability of sampling from every point on $S_{\text{c},o}$. Unfortunately, a large number of samples may be required to cover $S_{\text{c},o}$ with adequate density when it is high-dimensional. This can lead to particle deprivation.

Particle projection tightly focuses samples near S_{t-1} by choosing the seed states at time t to be S_{t-1}^+, the set of particles from time $t-1$ after applying action a_t. This strategy avoids particle deprivation, but has two downsides: (1) S_{t-1}^+ may have a highly non-uniform distribution and (2) the finite set of n seed states be exhausted if the root-finding algorithm fails frequently.

Ball projection combines uniform and particle projection strategies by drawing $\hat{s}_t^{[i]} \sim \text{uniform}\, R(S_{t-1}^{+})$ where $R(S_{t-1}^{+}) = \bigcup_{s \in S_{t-1}^{+}} B(s, r_{\text{b}})$ is the subset of S within radius r_{b} of at least one particle propagated from time $t-1$. The notation $B(c, r) = \{p \in \mathbb{R}^3 : ||p - c|| < r\}$ denotes a ball with radius r centered at c.

Ball projection is equivalent to particle projection as $r_{\text{b}} \to 0$ and to uniform projection as $r_{\text{b}} \to \infty$. In practice, we choose r_{b} such that the normalized importance weight of any particle outside of $R(S_{t-1}^{+})$ is determined to be sufficienty small. This is possible because dual importance weights are computed by kernel denesity estimation.

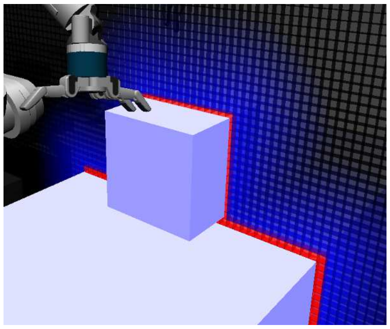


Figure 5.2: Voxel grid colored to show ■ large positive, 🟦 small positive, and 🟥 negative values of Φ. Best viewed in color.

5.3 *Computing Signed Distance*

We use a signed distance field to compute f_o and its gradient $\nabla_s f_o$ when $G_{\text{o}}(x) = G_{\text{const}}$ is known and static. The *signed distance field*

$$\Phi(p) = \text{SDIST}(\{p\}, G_{\text{const}})$$

returns the signed distance between a point $p \in \mathbb{R}^3$ and the environment. We approximate Φ by converting G_{const} to a voxel grid and evaluating its distance transform [Felzenszwalb and Huttenlocher, 2012]. We approximate the gradient $\nabla_p \Phi$, which is needed below, by the finite-difference of Φ between voxels.

Approximate the geometry of each sensor i as a collection of balls

$$G_{\text{s},i}(q) = \bigcup_{j=1}^{n_{\text{b},i}} B(c_{i,j}(q), r_{i,j}),$$

It is possible to approximate any solid object represented by triangular mesh as a union of balls [Wang et al., 2006]

where $c_{i,j} : Q \to \mathbb{R}^3$ defines the forward kinematics for the center of the j-th ball representing sensor i and $r_{i,j}$ is its radius.

Under this assumption, we write f_o and its gradient as

$$f_{o,i}(s) = \min_{j \in [1, n_{\text{b},i}]} \left[\Phi(c_{i,j}(q)) - r_{i,j}\right] \tag{5.3}$$

$$\nabla_s f_{o,i}(s) = \left[\nabla_p \Phi\left(c_{i,j^*}(q)\right)\right] J_{c_{i,j^*}}(q) \tag{5.4}$$

where j^* is the value of j that minimizes eq. (5.3) and $J_{c_{i,j^*}}$ is the Jacobian of c_{i,j^*}. This is equivalent to the linear component of the manipulator Jacobian of $c_{i,j^*}(q)$ as if it were rigidly attached to ball j^* of the i-th sensor active in observation o. This is the same method that CHOMP, a trajectory optimization algorithm for motion planning, uses to penalize collision [Zucker et al., 2013].

5.4 *Simulation Experiments*

This section compares the MPF with the CPF at estimating robot configuration. Just as with object pose estimation, we expect:

H1. *The MPF will outperform the CPF at estimating robot configuration.*

This chapter introduced an implicit representation of the contact manifold. Both implicit and explicit methods will saturate the contact manifold in low-dimensional spaces, so we expect that:

H2. *All implicit and explicit representations of the contact manifold will perform similarly when S is low-dimensional.*

Uniform projection (MPF-Uniform) attempts to cover all of S_c at uniform density. In contrast, particle (MPF-Particle) and ball (MPF-Ball) projection focus their samples on the subset of S_c that will be assigned high importance weight. Therefore, we hypothesize:

H3. *Particle and ball projection outperform uniform projection when S is high-dimensional.*

Unlike MPF-Particle, which may produce samples that are heavily biased towards S^+_{t-1}, MPF-Ball attempts to approximate a uniform distribution over the relevant portion of the contact manifold. Therefore, we expect that:

H4. *Ball projection will perform best when S is high-dimensional.*

Our results results indicate that all four hypotheses are true.

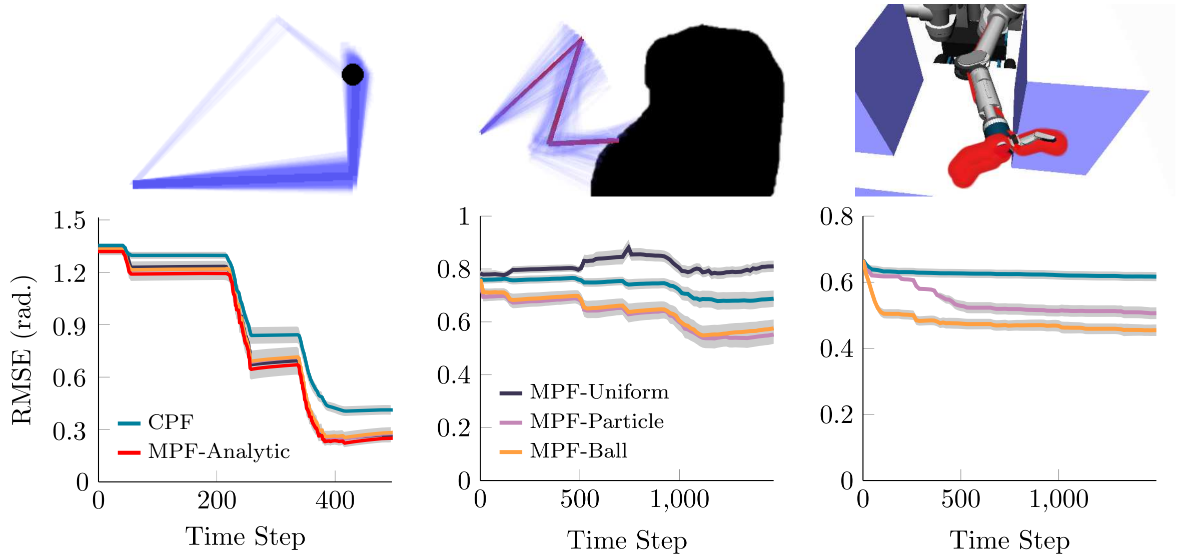
5.4.1 Experimental Design

We evaluate the CPF and MPF with $n = 250$ on three different simulated manipulators interacting with known, static environments. Each trial begins by choosing an initial proprioception offset $\delta q \sim \mathcal{N}(0, \Sigma_{\delta q})$ and executing a pre-defined sequence of actions. We model motion of the arm using a frictionless contact model that projects the robot out of inter-penetration. After each time step, we simulate a proprioceptive observation with noise $\epsilon_q \sim$ uniform $B(0, r_q)$.

First, we consider a two degree-of-freedom (2-DOF) manipulator in a two-dimensional workspace that contains a single point obstacle (fig. 5.3a). The robot has one binary contact sensor on the tip of its distal link. We set $\Sigma_{\delta q}^{1/2} = (2 \text{ rad})I$ and $r_q = 0.05$ rad.

Next, we consider a 3-DOF manipulator in a two-dimensional workspace that contains an unstructured obstacle (fig. 5.3b). The robot has 20 binary contact sensors spaced evenly along its two distal-most links. We set $\Sigma_{\delta q}^{1/2} = (0.8 \text{ rad})I$ and $r_q = 0.05$ rad.

Finally, we consider a 7-DOF Barrett WAM arm [Salisbury et al., 1988] with a BarrettHand [Townsend, 2000] end-effector in a three-dimensional workspace that contains several boxes (fig. 5.3c). The robot has binary contact sensors on its hand, wrist, and forearm. We set $\Sigma_{\delta q}^{1/2} = (0.5 \text{ rad})I$ and $r_q = 0.01$ rad.



(a) 2-DOF Manipulator

(b) 3-DOF Manipulator

(c) 7-DOF Manipulator

Figure 5.3: Performance of the MPF using different projection strategies on (a) 2-DOF, (b) 3-DOF, and (c) 7-DOF manipulators. Results are averaged over 100 experiments. All three plots show RMSE and the error bars denote a 95% confidence interval. Best viewed in color.

In all three cases, we implemented root-finding by using gradient descent to minimize the loss function $F_o(s) = \frac{1}{2}||f_o(s)||^2$ with gradient $\nabla F_o(s) = f_o(s)$. This function achieves its global minimum at $f_o(s) = 0$; i.e. when f_o has a root.

5.4.2 *Conventional vs. Manifold Particle Filter (H1 and H2)*

The contact manifold for the 2-DOF planar arm experiments consists of two points in configuration space corresponding the "elbow-up" and "elbow-down" configurations with the tip of the manipulator in contact with the point obstacle. We use this fact to implement MPF-Explicit by drawing samples from an explicit representation of the contact manifold derived from the robot's kinematics.

Figure 5.3a shows that the robot was able to reduce most of its uncertainty in two touches. The first touch ($t \approx 200$) constrained the robot to one of two configurations. The second touch ($t \approx 350$) disambiguated between those configurations.

The CPF performed poorly because particle deprivation caused the belief state to quickly collapse to one, possibly incorrect, configuration. The MPF performed better by drawing samples from both modes. All variants of the MPF performed similarly because the contact manifold was low-dimensional. These results support H1 and H2: all variants of the MPF outperform the CPF on low-dimensional problems.

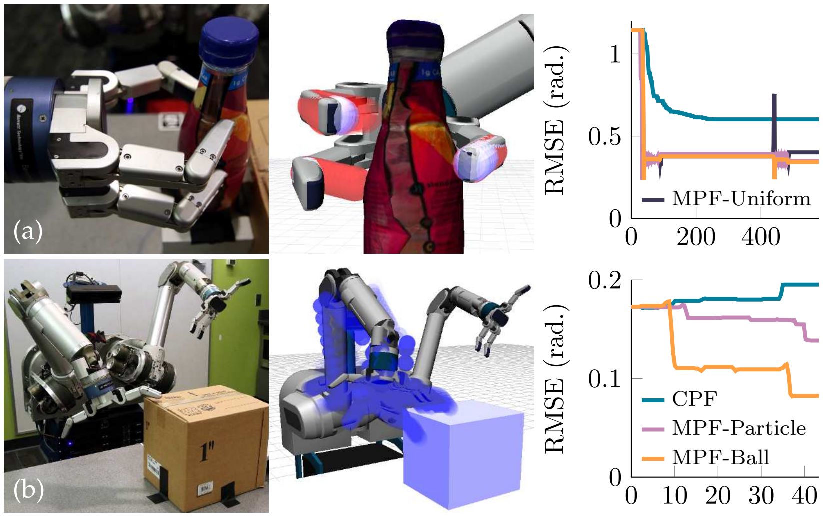


Figure 5.4: Estimating the configuration of a (a) BarrettHand end-effector and (b) Barrett WAM arm using the CPF and MPF. The middle figure shows a snapshot of the belief state with (a) the maximum-likelihood estimate in red and (b) each particle in blue. Data from these trials supports our simulation results: MPF outperforms CPF and MPF-Ball performs best in the WAM's 7-DOF configuration space. Best viewed in color.

5.4.3 *Projection Strategy (H3 and H4)*

We repeated the same experiment on a 3-DOF manipulator to evaluate how the projection strategies scale with dimensionality. Figure 5.3b shows that MPF-Particle and MPF-Ball both continued to outperform the CPF. However, MPF-Uniform suffered from particle deprivation and performed the worst. This supports H3: strategies that focus their samples outperform those that do not.

Figure 5.3c shows that increasing from a 3-DOF to a 7-DOF manipulator exacerbates this effect: MPF-Uniform performed so poorly that it is omitted from the plot to avoid distorting the scale. These results also support H4: MPF-Ball outperformed MPF-Particle by avoiding introducing bias into the belief state. Maintaining an unbiased estimate is critical for a 7-DOF manipulator due to the additional ambiguity introduced by kinematic redundancy.

5.5 *Real-Robot Experiments*

We implemented the CPF and MPF on HERB [Srinivasa et al., 2012] to estimate the configuration of a BarrettHand [Townsend, 2000] end-effector and the Barrett WAM arm [Salisbury et al., 1988]. In both cases, HERB used the BarrettHand's strain gauges to detect binary contact with the distal links of the fingers. We measured the actual configuration of the robot using optical joint encoders. These measurements were not available to the estimator. We specifically chose these tasks because ground truth data was available.

In the first experiment (fig. 5.4a), we held the arm in a fixed configuration and closed the end-effector around an object. We assumed that the proximal joint angles were known, but the distal joint angles were not. This was a simple estimation problem because each finger was an independent kinematic chain. Figure 5.4a shows data from two grasps: one from $t \approx 40$ to $t \approx 100$ and another from $t \approx 450$ to $t \approx 500$. Our results are consistent with the 2-DOF simulation results in section 5.4: all variants of the MPF outperformed the CPF.

In the second experiment (fig. 5.4b), we kept the hand open and considered the configuration of the arm to be uncertain. We teleoperated HERB to execute a trajectory in an environment that contained a large box on a table. We used the joint angles measured by encoders located before the cable drive transmission as o_q. We ran the CPF and MPF with $n = 100$, $\Sigma_{\delta q} = (0.1 \text{ rad})I$, and $r_\text{q} = 0.1$ rad. Figure 5.4b supports our simulation results: the MPF achieved lower error than the CPF and MPF-Ball performed best.

5.6 Discussion

This chapter introduced an implicit representation of the contact manifold (section 5.1) based on constraint projection (section 5.2) that scaled the MPF up to robot configuration estimation. Our experimental results, both in simulation (section 5.4) and on a real robot (section 5.5) showed that the MPF with the ball projection strategy outperformed the CPF across all experiments. However, as in chapter 4, we made several assumptions to implement the MPF.

First, we used the technique from Felzenszwalb and Huttenlocher [2012] to evaluate the signed distance function and its gradient. This technique is efficient under model 2 because a static environment allows us to re-use a signed distance field across multiple update steps. Extending our approach to a dynamic environment requires a method of computing signed distance, e.g. with Gilbert-Johnson-Keerthi algorithm [Gilbert et al., 1988] for distance and the expanding polytope algorithm [van den Bergen, 2001] for penetration depth.

Second, projecting samples drawn uniformly from the ambient space onto the contact manifold may yield a non-uniform distribution over the manifold. Quantifying this error is difficult because, a we discuss in section 8.3.3, a probability measure over the ambient space differs from a probability measure defined over the contact manifold. In practice, the error introduced by this discrepancy did not harm the MPF's ability to outperform the CPF.

This chapter highlights the ability of the MPF to generalize across tasks: we applied the same algorithm that we used to estimate object pose in chapter 4 to estimate robot configuration with only minor modifications. We exploited the idea of lazy evaluation (section 8.1.4) to implicitly represent the contact manifold. We exploit a similar idea in the next two chapters for planning.

6
Hand-Relative Planning

This chapter is adapted from Koval et al. [2014, 2016d].

State estimation is an important part of manipulation, but it fundamentally does not tell the robot which actions to execute. This chapter introduces a *belief space planner* that trades off between information-gathering and goal-directed actions to achieve the goal.

Finding an optimal policy for an arbitrary partially observable Markov decision process (POMDP, Smallwood and Sondik [1973], Kaelbling et al. [1998]) is intractable [Littman, 1996], so we use domain-specific knowledge to inform the design of our planner.

Our key insight is that contact decomposes a policy into pre- and post-contact stages (insight 2). We exploit this effect by planning the post-contact policy in an offline computation step and sharing it between problem instances. Then, when confronted with a new problem instance, we plan a pre-contact trajectory using an efficient online search. We prove that decomposing planning in this way has a bounded effect on the optimality of the overall policy.

These stages resemble the dichotomy between gross (pre-contact) and fine (post-contact) motion planning found in early manipulation research [Hwang and Ahuja, 1992].

This chapter specifically considers model 1 from chapter 3 by planning for a lone end-effector manipulating a movable object. Our planner ignores kinematic constraints during planning and, thus, the policy may not be feasible to execute in environments cluttered with kinematic constraints. We use the insights gained in this chapter to develop a planner that considers kinematic constraints in chapter 7.

6.1 *Policy Decomposition*

The discriminative nature of contact sensing causes policies to naturally decompose into pre- and post-contact stages (insight 2). Before observing contact, the robot executes an open-loop *pre-contact trajectory* $\zeta = (a_1, a_2, \ldots, a_t)$ and receives a series of no-contact observations $o_1, o_2, \ldots, o_{t-1} \in O_{\text{nc}}$. Once contact is observed $o_t \in O_{\text{c}}$, the robot switches to a *post-contact policy* π^{c} that incorporates real-time feedback from observations to achieve the goal.

This chapter uses o to refer to the contact sensor observation o_{s}. The proprioceptive observation o_{q} is unnecessary because model 1 asserts perfect proprioception.

Any policy π of this form has a value function

$$V^{\pi}[b] = R(b,a) + \gamma \int_{\Delta} T(b,a,b') \Bigg(\underbrace{\Omega(o_{\text{nc}}, b', a) V^{\pi}[b']}_{\text{pre-contact}} + \underbrace{\sum_{o \in O_{\text{c}}} \Omega(o, b', a) V^{\text{c}}[b']}_{\text{post-contact}} \Bigg) db' \qquad (6.1)$$

that decomposes into two terms based on the most recent observation. This equation assumes that the action $a = \pi[b]$ is selected by the policy and overloads the notation $R(b,a) = \int_S R(s,a)\, ds$, $T(b,a,b') = \iint_S T(s,a,s')b(s)b'(s')\, ds\, ds'$ and $\Omega(o,b,a) = \int_S \Omega(o,s,a)b(s)\, ds$.

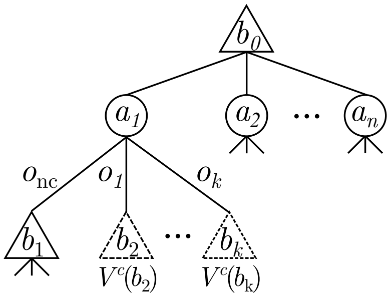


Figure 6.1: Online POMDP solvers must branch over both A and O. The pre-contact search only branches over A by evaluating all post-contact belief states with the post-contact value function V^{c}.

The *pre-contact term* includes the value $V^{\pi}[b']$ earned by following π given that contact has not yet been observed. Critically, evaluating this term does not require computing an expectation over O_{nc} because all such observations are indistinguishable. We use $o_{\text{nc}} \in O_{\text{nc}}$ denote an arbitrary observation from that set.

The *post-contact term* includes the value $V^{\text{c}}[b']$ earned by executing π^{c} once the robot observes contact. Critically, this term does not recursively include $V^{\pi}[b']$ because π^{c} is assumed to be fixed.

We leverage this decomposition to compute the post-contact policy, once for each object that the robot expects to encounter, in an offline pre-computation step (section 6.2). Finding such a policy is tractable because the set of belief states consistent with a contact observation is small. Then, when presented with an initial belief state, we plan a pre-contact trajectory (section 6.3). Doing so is efficient because, as fig. 6.1 shows, eq. (6.1) does not branch on observations.

Decomposing planning in this way has a bounded effect on the optimality of the resulting policy. We prove this for the case where the pre-contact trajectory is planned using a H-step lookahead:

Theorem 6.1. *Suppose that policy π executes a pre-contact trajectory ζ until contact is observed, then switches to following π^c. If ζ is found by performing a H-step lookahead on eq.* (6.1), *then*

$$||V^* - V^{\pi}||_{\infty} \leq \gamma^H \epsilon_{nc} + \frac{\gamma(1-\gamma^H)}{1-\gamma} p_{max} \epsilon_c$$

where $\epsilon_{nc} = ||V^* - V_0||_{\infty}$ is the sub-optimality of the evaluation function V_0 used to truncate the lookahead, $\epsilon_c = ||V^* - V^c||_{\infty}$ is the sub-optimality of π^c, and p_{max} is the maximum single step probability of observing contact.

Proof. Let V_h be the value function of a h-step lookahead policy. Since

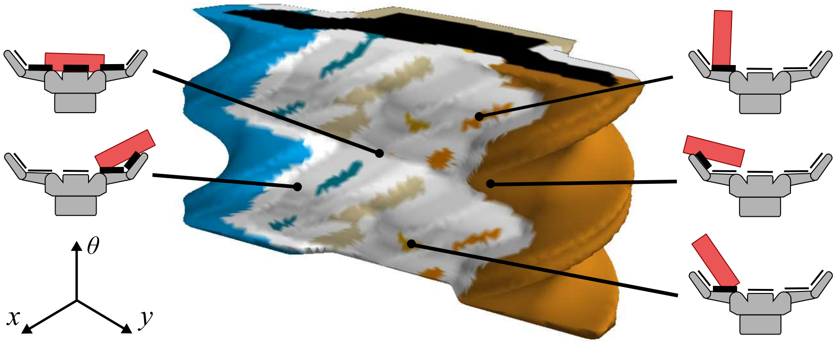


Figure 6.2: The contact manifold S_c for a hand manipulating a rectangular box. Each color indicates that the object is in contact with a particular sensor. White indicates contact with multiple sensors. Best viewed in color.

π is a H-step lookahead policy, we use the fact that $V^\pi = V_H$ to show

$$\begin{aligned}||V^* - V^\pi||_\infty &\leq \gamma||V^* - V_{H-1}||_\infty + \gamma p_{\max}||V^* - V^{c}||_\infty \\ &\leq \gamma^H||V^* - V_0||_\infty + \sum_{t=1}^{H}\gamma^t p_{\max}||V^* - V^{c}||_\infty \\ &\leq \gamma^H\epsilon_{nc} + \frac{\gamma(1-\gamma^H)}{1-\gamma}p_{\max}\epsilon_{c}.\end{aligned}$$

First, we distribute $||\cdot||_\infty$ using the triangle inequality and bound the single-step probability of observing contact by $p_{\max}$. Next, we recursively expand V_H in terms of V_{H-1} down to V_0. Finally, we evaluate the infinite sum over t as a geometric series.

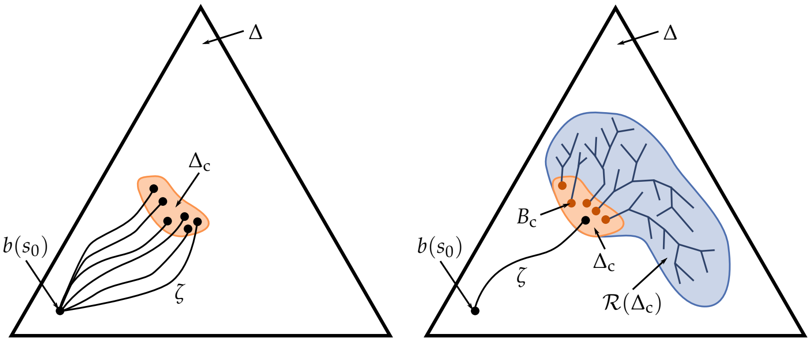
□ □

Theorem 6.1 shows that the sub-optimality of π comes from two sources. First, truncating the lookahead at depth H introduces the gap $\gamma^H\epsilon_{nc}$ characteristic of this type of policy [Ross et al., 2008]. This gap vanishes as $H \to \infty$ or $\epsilon_{nc} \to 0$. Second, the sub-optimality of the post-contact policy introduces a gap of $\frac{\gamma(1-\gamma^H)}{1-\gamma}p_{\max}\epsilon_c$. This gap does not vanish as $H \to \infty$ because π cannot deviate from π^c once contact is observed, even if it is sub-optimal. It does, however, vanish as π^c approaches an optimal policy, i.e. $\epsilon_c \to 0$.

6.2 *Post-Contact Policy*

Our planning method requires that π^c be known while planning the pre-contact trajectory. This section describes how we use an offline POMDP solver to find a near-optimal post-contact policy.

Suppose the robot is in belief state b while partway through its execution of ζ, takes action a, receives observation $o \in O_c$, and transitions to the posterior belief state b'. At this point, the robot switches to executing π^c.



(a) Pre-Contact Trajectory

(b) Post-Contact Policy

Figure 6.3: We decompose the policy π into a (a) pre-contact trajectory ζ and a (b) post-contact policy π^c. Pre- and post-contact are coupled by the small set of post-contact belief states $\Delta_c \subseteq \Delta$.

Since contact sensors are discriminative, we know that $s \in S_c$ with high probability. The set of belief states that satisfy this constraint

$$\Delta_c = \{b \in \Delta : b(s) = 0 \; \forall s \notin S_c\}$$

form the *post-contact belief space* (fig. 6.3a). The set Δ_c is small relative to Δ, which makes planning π^c amenable to a point-based method [Lee et al., 2007].

Point-based methods, first introduced by Pineau et al. [2003], are a class of offline POMDP planners that break the *curse of history* by performing backups at a finite set of *belief points* $B \subseteq \mathcal{R}(b_0)$. These methods perform well when the *reachable belief space* $\mathcal{R}(b_0) \subseteq \Delta$, the set of beliefs that are reachable from the initial belief b_0 given any sequence of actions and observations, is small [Pineau et al., 2003].

6.2.1 *Initial Belief Points*

Point-based methods typically require the initial belief state to be known. In our application, we only know that $b_0 \in \Delta_c$ and cannot initialize $B = \Delta_c$ because it is uncountably infinite.

Instead, we initialize the method with a finite set of points $B = B_c \subseteq \Delta_c$ selected to be representative of the post-contact belief space (fig. 6.3b). We may refine this set over time by adding to it any post-contact belief state that we encounter during execution.

Given B_c, the simplest way of planning π^c is to find a separate policy for each belief point and combine their α-vectors. Since each α-vector is a global lower bound on the optimal value function, their union provides a tighter lower bound than any one policy in isolation. However, this approach is inefficient because it does not share any information while planning the individual policies.

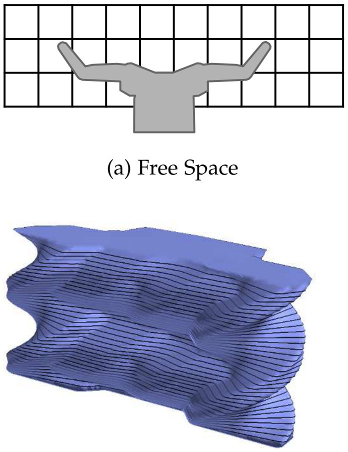
We leverage this structure by converting the POMDP with the set of initial belief states B_c into an augmented POMDP with one initial belief state. By doing so, we allow the point-based method to exploit its heuristics to simultaneously find a policy over all of Δ_c.

6.2.2 State Space Discretization

Most point-based methods—with few exceptions [Porta et al., 2006, Brunskill et al., 2008, Bai et al., 2011]—also require the state space to be discrete. The model we defined in chapter 3 has a continuous state space, so must discretize S to apply one of these methods.

Ideally, we would only discretize the subset of S required to support the belief states $\mathcal{R}^*(\Delta_c)$ reachable from Δ_c under an optimal policy. Unfortunately, finding $\mathcal{R}^*(\Delta_c)$ is as difficult as planning an optimal policy [Kurniawati et al., 2008]. Instead, we define a *trust region* $S_{\text{trust}} \subseteq S$ that we believe to over-approximate the support of $\mathcal{R}^*(\Delta_c)$, discretize S_{trust}, and use that discretization for planning.

There is a trade-off in choosing the size of S_{trust}: making it too small may disallow the optimal policy, while making it too large may make planning π^c intractable. In the case of quasistatic manipulation [Mason, 1986], we believe S_c to be relatively small because the optimal policy will not allow the object to stray far from the hand.



(a) Free Space

(b) Contact Manifold

Figure 6.4: Explicit discretization of (a) free space and (b) the contact manifold.

We compute the transition, observation, and reward functions for the discrete state space by taking an expectation over the corresponding continuous models. We approximate the expectation with Monte Carlo sampling by assuming a uniform distribution over the aggregation of continuous states that map to each discrete state.

Assuming there is a uniform distribution over the discrete state is motivated by the *principle of maximum entropy* [Jaynes, 1957]. We discuss drawbacks to this assumption in section 8.2.1.

Discretization at a uniform resolution poorly represents the structure of S_{trust}: two states in S may be arbitrarily close together, but behave differently due to the presence or absence of contact (challenge 1). Instead, we separately discretize $S_{\text{trust}} \cap S_{\text{nc}}$ into a uniform grid and $S_{\text{trust}} \cap S_c$ along the surface of an analytic representation of the contact manifold described in section 4.2.2.

6.3 Pre-Contact Trajectory

The belief dynamics are a deterministic function of the action given a fixed sequence of "no contact" observations. This allows us to find the optimal trajectory ζ by solving a shortest path problem over an augmented belief space (fig. 6.3b).

To show that value is additive over actions, a requirement of A*, we recursively expand eq. (6.1) to write the Bellman equation as:

$$V^*[b_0] = \max_{\zeta} \sum_{t=1}^{\infty} \left[\gamma^{t-1} \left(\prod_{i=1}^{t-1} \Omega(o_{\text{nc}}, b_i, a_i) \right) \left(R(b_{t-1}, a_t) + \sum_{o_t \in O_c} \Omega(o_t, b_t, a_t) V^c[b_t] \right) \right]. \tag{6.2}$$

Equation (6.2) defines the Bellman equation a policy of the form described in section 6.1. We abuse notation and denote this by V^*, even though there may exist a superior policy that is not of that form.

Each term in the outer summation corresponds to taking a single action in ζ. The product computes the probability of reaching time t

without having observed contact. We find the optimal value of ζ by solving a shortest path problem over a graph.

6.3.1 Graph Construction

Define the directed graph $G_\zeta = (V, E, c)$ where each vertex $v = (b, p_{\text{nc}}, t) \in V$ consists of a belief state b, the probability p_{nc} of having not yet observed contact, and the time t. An edge $(v, v') \in E$ between v and $v' = (b', p'_{\text{nc}}, t')$ represents the outcome of executing an action $a \in A$. Its successor vertex is given by $b'(s') = \eta\Omega(o_{\text{nc}}, s', a) \int_S T(s, a, s') b(s) ds$, $p'_{\text{nc}} = p_{\text{nc}}\Omega(o_{\text{nc}}, b', a)$, and $t' = t + 1$.

As in chapters 4 and 5, we use η to represent a normalization factor chosen such that $\int_S b'(s')\, ds' = 1$.

The cost of an edge $(v, v') \in E$ generated by action a is given by

$$c(v, v') = -\gamma^t p_{\text{nc}} \left(R(b, a) + \gamma \sum_{o \in O_{\text{c}}} \Omega(o, b', a) V^{\text{c}}[b'] \right),$$

precisely one term in the summation in eq. (6.2). This expression consists of two parts: (1) the immediate reward $R(b, a)$ and (2) the expected reward $\sum_{o \in O_{\text{c}}} \Omega(o, b', a) V^{\text{c}}[b']$ obtained by executing π^{c} if contact is observed. A minimum-cost path trades off between quickly making contact to reduce p_{nc} and passing through belief states that have high value under the post-contact policy to increase $V^{\text{c}}[b']$.

6.3.2 Search Algorithm

Finding the maximum ζ with lookahead H is equivalent to finding the shortest path in G_ζ from the start vertex $(b(s_0), 0, 0)$ to the set of goal vertices $\{(b, p_{\text{nc}}, t) \in v : t = H\}$ at time H. We use weighted A*, a heuristic search algorithm, to solve this shortest path problem [Pohl, 1977]. Weighted A* operates identically to A* but sorts the vertices in the frontier by priority function

Theorem 6.1 bounds the error introduced by terminating the A* search at depth H, instead of optimizing the infinite horizon value function.

$$g(v) = f(v) + \epsilon_{\text{w}} h(v)$$

where f is the cost-to-come, h is a heuristic estimate of the cost-to-go, and $\epsilon_{\text{w}} \geq 1$ is a heuristic inflation factor. We require h to be *admissible* by under-estimating the true cost-to-go [Pearl, 1984].

If $\epsilon_{\text{w}} = 1$, weighted A* is identical to A*. For $\epsilon_{\text{w}} > 1$, weighted A* may return a path of cost up to ϵ_{w} times that of an optimal path [Pohl, 1977]. However, weighted A* tends to expand far fewer vertices than A* when h is informative. This is beneficial for our application because expanding a vertex is computationally expensive.

6.3.3 Heuristic Function

We specifically constructed G_ζ such that the cost-to-go is $-V^*$. Therefore, we can find an admissible heuristic by computing an upper bound on the optimal value function.

First, we apply the *MDP approximation* [Ross et al., 2008]

$$V^*[b] \leq V^{\mathrm{MDP}}[b] = \int_S V^{\mathrm{MDP}}(s) b(s)\, ds$$

to bound V^* by optimal value function V^{MDP} of the Markov decision process (S, A, T, R). This is equivalent to assuming full observability.

The symbol V^{MDP} is overloaded to refer both to the MDP optimal value function $V^{\mathrm{MDP}} : S \to \mathbb{R}$ and the MDP approximation to the POMDP value function $V^{\mathrm{MDP}} : \Delta \to \mathbb{R}$. These uses are unambiguous because the two functions have different domains.

Next, we further relax the MDP approximation

$$V^{\mathrm{MDP}}(s) \leq \bar{V}^{\mathrm{MDP}}(s) = \sum_{t=1}^{t_{\min}(s)} \gamma^t R_{\max}$$

by allowing the movable object to pass through the robot. In this equation, $R_{\max} = \max_{s \in S \setminus G, a \in A} R(s, a)$ is the maximum reward obtainable while $s \notin G$ and $t_{\min}(s)$ is a lower bound on the number of steps required to place the movable object in the goal region.

Finally, we compute a lower bound on that number of steps

$$t_{\min}(s) \leq \left\lfloor \frac{\min_{s' \in G} \mathrm{dist}(s, s')}{d_{\max}} \right\rfloor$$

using the straight-line distance $\mathrm{dist}(s, s')$ and the maximum length $d_{\max}$ of an action. Note that we could not directly use the distance $\mathrm{dist}_{s' \in G}(s, s')$ in $\bar{V}^{\mathrm{MDP}}$ because it omits the discount factor γ.

We combine all three relaxations to form the heuristic function

$$h(v) = \gamma^t p_{\mathrm{nc}} \int_S \bar{V}^{\mathrm{MDP}}(s) b(s) ds,$$

which is admissible. This heuristic assumes full observability, allows the robot to pass through the movable object, and under-estimates the time required to reach the goal region. However, it still guides the graph search towards vertices that are closer to the goal region.

6.4 *Simulation Experiments*

This section evaluates our planner in simulation experiments. We use a QMDP policy, which does not plan to take multi-step information-gathering actions, as a baseline for comparison [Littman et al., 1995].

We begin by evaluating the performance of the post-contact policy in isolation. Since information-gathering is important for contact sensing (challenge 2), we hypothesize:

H1. *The POMDP post-contact policy will achieve higher value than the baseline QMDP policy.*

H2. *The POMDP post-contact policy will achieve success with higher probability than the baseline QMDP policy.*

The key difference between these two hypotheses is the model used for evaluation: H1 uses the discrete POMDP model used for planning and H2 uses the continuous model.

Next, we evaluate the impact of decomposing planning into pre- and post-contact phases. Theorem 6.1 suggests that this decomposition will not significantly harm performance, so we hypothesize:

Ideally, we would compare against a policy computed by a point-based POMDP solver over the entire state space. Unfortunately, we were not able to get implementation of SARSOP provided by the APPL toolkit [Kurniawati et al., 2008] to successfully load the full discrete POMDP model due to its size.

H3. *The pre-contact trajectory will not significantly affect the success rate of a post-contact policy.*

H4. *The full POMDP policy will outperform the full QMDP policy.*

Finally, we evaluate the effect of varying sensor coverage on these policies. Increasing sensor coverage should reduce the burden on the planner to take information-gathering actions, so we hypothesize:

H5. *Both policies will improve with better sensor coverage.*

Since the QMDP policy is optimal when state is fully observed, we expect it to benefit more than the POMDP policy:

H6. *The QMDP policy will improve more than the POMDP policy.*

Our results suggest that all of these hypotheses are satisfied.

6.4.1 Experimental Design

We evaluated both policies on a quasistatic simulation of a Barrett-Hand [Townsend, 2000] pushing a 3.5 cm × 5 cm box in the plane. The goal was to push the center of the box into a 4 cm × 8 cm goal region in front of the palm. The robot received feedback from binary contact sensors on its two fingertips. The contact sensors were perfectly discriminative, but had a 10% error rate of generating an incorrect observation during contact.

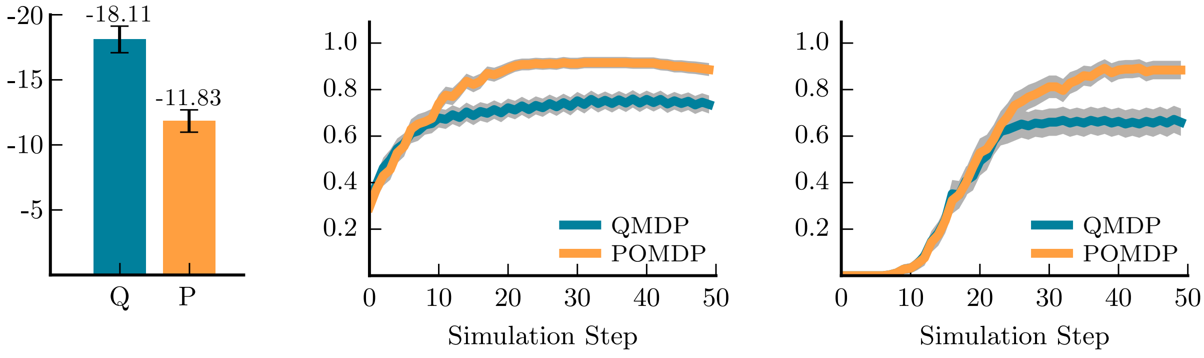
We chose a trust region S_{trust} of size 15 cm × 50 cm. We discretized the no-contact portion of S_{trust} at a 2 cm × 2 cm × 12° resolution and the contact portion at a 1 cm × 12° resolution. The discrete state space consisted of 2625 no-contact states, 1613 contact states, and one sentinel state to represent $S \setminus S_{\text{trust}}$.

Actions that are not aligned with the discretization can lead to discretization artifacts that affect the performance of the post-contact policy.

The robot could take five purely translational actions: forward, left, right, forward-left, and forward-right. We assigned the axis-aligned actions a length of 2 cm and the diagonal actions a length of $2\sqrt{2}$ cm align all five actions with our discretization of the state space.

We used a particle filter [Gordon et al., 1993] to implement the belief update during execution of ζ. Once we observed contact, we switched to using a discrete Bayes filter to implement the belief update [Thrun et al., 2005]. This ensured that the belief update used during execution matched that used during planning.

Our use of the discrete Bayes filter to track the belief state during execution of the post-contact policy differs from the experiments presented in earlier versions of this work [Koval et al., 2014]. In those experiments, we used the manifold particle filter to track state during execution of the full policy. As a result, the results that we present in this thesis slightly differ from those presented in prior work.



(a) Discrete Value

(b) Post-Contact Success Rate

(c) Full Policy Success Rate

Figure 6.5: Performance of the post-contact policy. (a) Expected value, at depth 50, of the QMDP (Q) and POMDP (P) post-contact policies evaluated on the discrete system dynamics. Reward is always negative, so less-negative values are better. Success rate of the (b) post-contact and (c) full policy evaluated on the continuous system dynamics. The error bars and gray shaded region denote a 95% confidence interval.

We evaluated a policy π on two metrics. First, we estimated its value V^π by performing rollouts of π and recording the reward that it achieved. This is the quantity that was directly maximized by our planner. Second, we measured the *success rate* $\Pr(s_t \in G)$ of the policy's ability to push the movable object into the goal region.

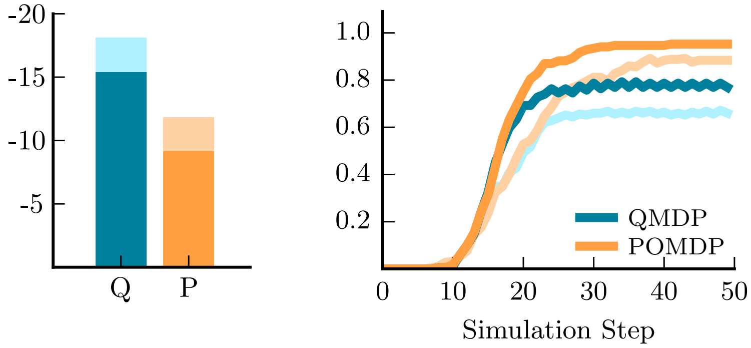
6.4.2 *Post-Contact Policy (H1 and H2)*

We solved for a post-contact QMDP policy by running MDP value iteration on the discrete POMDP model until it converged. This process required 1729 iterations and took 8.36 s (4.84 ms per backup). The resulting policy consisted of five α-vectors and took 7.64 ms to evaluate on a discrete belief state.

We repeated this procedure to generate the post-contact POMDP policy by running a point-based solver on $|B_c| = 15$ initial belief points, each assigned equal weight. Each belief point $b_i \in B_c$ was a Gaussian distribution $b_i = \mathcal{N}(\mu_i, \Sigma_i)$ with mean $\mu_i = [x_i, y_i, \theta_i]^T$ and covariance matrix $\Sigma_i^{1/2} = \text{diag}[5 \text{ cm}, 5 \text{ cm}, 15°]$. The mean was a fixed distance $x_i = 20$ cm in front of the hand with lateral offset $y_i \sim \text{uniform}[-10 \text{ cm}, 10 \text{ cm}]$ and orientation $\theta_i \sim \text{uniform}[0°, 360°]$.

We solved the resulting discrete POMDP using the SARSOP implementation provided by APPL Tookit [Kurniawati et al., 2008]. We ran SARSOP for 5 minutes, during which it produced a policy that consisted of several thousand α-vectors and took 7.62 ms to evaluate. We intentionally ran SARSOP until convergence, as we did for QMDP, to eliminate this as a variable in our analysis: it may be possible to terminate planning much earlier with negligible impact on the quality of the policy.

We evaluated the QMDP and POMDP post-contact policies on the discrete model by performing 1000 trials on initial belief selected from the set described above. Figure 6.5a shows that the POMDP policy significantly outperformed the QMDP policy. This result confirms H1: it is advantageous to plan a policy that is capable of



(a) Discrete Value

(b) Full Policy Success Rate

Figure 6.6: Effect on (a) discrete value and (b) success rate of increasing sensor coverage from only the fingertips (faded colors) to the entire surface of the hand (solid colors). Again, note that less-negative values are better. Error bars are omitted to reduce visual clutter.

taking information-gathering actions to reduce uncertainty.

Next, we ran 500 trials of the same experiment on the continuous model. Figure 6.5b shows that the higher-quality discrete POMDP policy translated to a high-quality policy on the underlying continuous model. The POMDP policy successfully achieves the goal with higher probability than the QMDP policy, supporting H2.

6.4.3 *Pre-Contact Trajectory (H3 and H4)*

Next, we evaluated the impact of decomposing a policy into pre- and post-contact stages. We planned a pre-contact trajectory ζ using a weighted A* search with $\epsilon_w = 2$. The search terminated after 20 s or once a vertex was expanded with $p_{nc} = 1$ or 85% of b in S_{trust}.

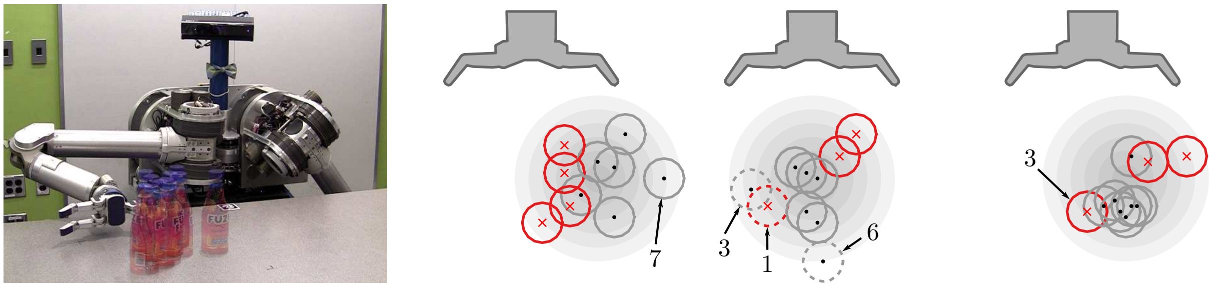
These termination conditions differs from the standard A* termination condition stated in section 6.3.2 to return a solution quickly when when a full search to depth H is not necessary.

We simulated each policy on initial belief states of the form described above, except with the distance from the mean to the palm increased to $x_i = 50$ cm. All of these belief states laid significantly outside of S_{trust} and, thus, could not be directly solved by π^c.

Figure 6.5c shows the success rate of the robot executing the combined policy. Both the QMDP and POMDP policies achieved success rates comparable to those in section 6.4.2. These result supports H3: the pre-contact trajectory faithfully extends a post-contact policy to a longer horizon. They also support H4: the POMDP policy outperforms QMDP even when executed after a pre-contact trajectory.

6.4.4 *Sensor Coverage (H5 and H6)*

Finally, we analyzed the impact of varying sensor coverage on the performance of the policies. To do so, we consider an alternate observation model where where each of the hand's seven links served as a binary contact sensor. We analyzed analytic representation of the contact manifold to compute that 10 of the $2^7 = 128$ possible contact observations are geometrically feasible. This is a large increase over the three observations that were feasible with fingertip sensors.



(a) Experimental Setup (b) Belief 1 (c) Belief 2 (d) Belief 3

Figure 6.7: (a) HERB in the initial configuration used to begin each trial. (b)–(d) Samples (circles) drawn from the three initial belief states (light gray iso-contours) used for evaluation. Samples are drawn with a black dot in the center if the QMDP policy succeeded and a red × if it failed. In (c) belief 2, the three samples on which SARSOP failed are drawn with a dotted line. Labels correspond to the trial numbers from table 6.1.

Figure 6.6a shows that increasing sensor coverage improved the value achieved by both post-contact policies. Figure 6.6b shows the improvement in the post-contact policies directly translated into improvement of the full policies. These results support H5.

Our experiments also show that improving sensor coverage yielded larger improvements in the QMDP policy than in the POMDP policy. This is most visible in the final success probability achieved by each policy, near simulation step 50 in fig. 6.6b. This supports H6: taking information-gathering actions becomes less important as sensor coverage improves.

6.5 *Real-Robot Experiments*

We evaluated the QMDP and SARSOP policies on HERB [Srinivasa et al., 2012] manipulating a bottle into a 4 cm × 8 cm goal region in front of the palm. HERB used a Jacobian pseudo-inverse controller to execute the five translational actions described in section 6.4.1 and used the BarrettHand's strain gauges (see section 5.5) to detect contact with the distal link of each finger.

We generated three Gaussian initial belief states with means $\mu_1 = [35\text{ cm}, 12\text{ cm}]$, $\mu_2 = [35\text{ cm}, 0\text{ cm}]$, and $\mu_3 = [35\text{ cm}, 6\text{ cm}]$ relative to the hand and variance $\Sigma^{1/2} = \text{diag}[5\text{ cm}, 5\text{ cm}]$. We drew ten samples from each initial belief state and used the resulting thirty samples, shown in figs. 6.7b to 6.7d, to evaluate both policies.

Table 6.1 shows the outcome of each trials. The position of the bottle used in each trial is shown in fig. 6.7. A trial terminated when the selected action was infeasible or the bottle reached the edge of the table. A trial was considered successful if the bottle was in the goal region at the moment of termination. The POMDP policy achieved success on 27/30 trials (90%). In contrast, the QMDP policy only achieved success on only 20/30 trials (67%).

The pre-contact trajectories for both policies aligned the mean of the initial belief state with the center of the palm. Once the trajectory was exhausted, the QMDP post-contact policy resembled an open-

		1	2	3	4	5	6	7	8	9	10
Belief 1	QMDP	$\cdot$	$\times$	$\cdot$	$\cdot$	$\cdot$	$\times$	$\cdot^*$	$\cdot$	$\times$	$\times$
	POMDP	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
Belief 2	QMDP	$\times$	$\cdot$	$\cdot$	$\times$	$\cdot$	$\cdot$	$\cdot$	$\times$	$\cdot$	$\cdot$
	POMDP	$\times^\dagger$	$\cdot$	$\times^\dagger$	$\cdot$	$\cdot$	$\times^\ddagger$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
Belief 3	QMDP	$\cdot$	$\cdot$	$\times$	$\cdot$	$\times$	$\cdot$	$\times$	$\cdot$	$\cdot$	$\cdot$
	POMDP	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$

Table 6.1: Outcome of executing the QMDP and POMDP policies on HERB. Each policy was executed on ten samples from three initial belief states. The symbol $\cdot$ denotes success and $\times$ failure to push the object into the goal region. The trial marked with $*$ succeeded because the object contacted a sensor during execution of the pre-contact trajectory, trials marked with † failed due to kinematics, and the trial marked with ‡ failed due to un-modelled errors in the observation model (see text).

loop push-grasp [Dogar and Srinivasa, 2010] by moving straight. The POMDP policy moved straight for several time steps, then executed a sideways information-gathering to force the bottle into a contact sensor. Once the bottle was localized, the policy moved to achieve the goal. Figure 6.8-Bottom shows an example this behavior.

All failures of the QMDP policy occurred because the bottle came to rest in a configuration that was: (1) outside of the goal region and (2) not in contact with a sensor. The policy chose the "forward" action because moving to either side would reduce the probability of the bottle being in the goal region at the next time step. This highlights the key limitation of a QMDP policy: it does not plan multi-step information gathering actions [Littman et al., 1995].

However, the QMDP policy does incorporate feedback from contact sensors during execution. This was critical in belief 1 trial 7 (marked with a $*$ in table 6.1 and section 6.5-Top). In this trial, the QMDP policy achieved success despite the fact that the initial pose of the bottle began outside of the capture region of the open-loop push grasp because the bottle naturally came into contact with a sensor.

The POMDP policy used information-gathering actions to localize the object regardless of its initial configuration. As a result, the SARSOP policy succeeded on 20/20 trials on beliefs 1 and 3. Surprisingly, the SARSOP policy failed on three trials from belief 2. Two trials (marked with † in table 6.1, section 6.5-Bottom shows one of these) occurred because HERB's elbow collided with the table. In both cases, HERB could have avoided collision by moving left (instead of right) to localize the bottle. We address this limitation in the next chapter.

6.6 Discussion

This chapter introduced a planning algorithm that exploited the natural decomposition between pre- and post-contact (section 6.1, insight 2) to gain efficiency. First, in an offline pre-computation step, we used a point-based method to plan a post-contact policy over post-contact belief space (section 6.2). Then, when presented with an initial belief state, we used a deterministic shortest path

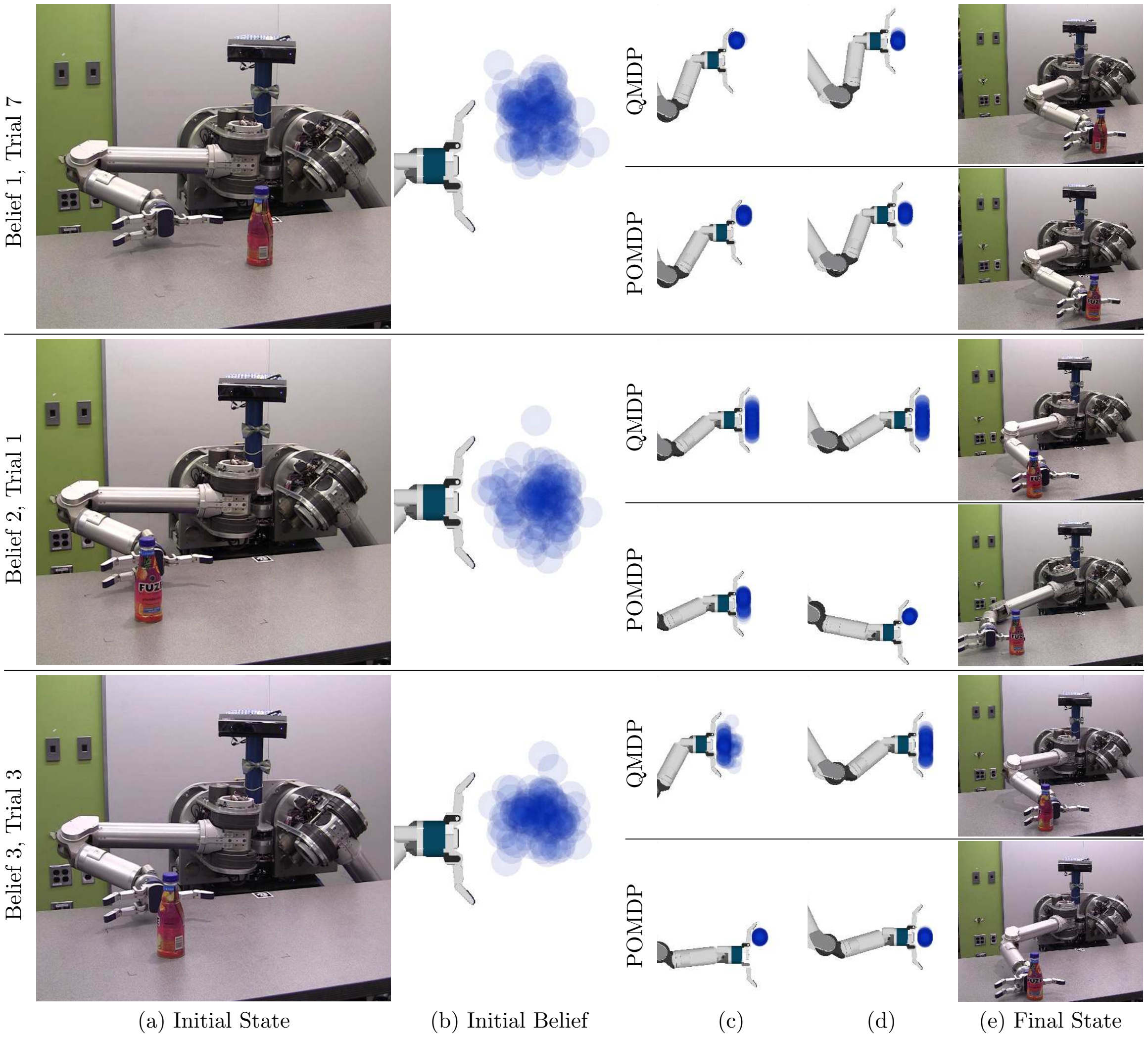


Figure 6.8: Three trials executed on HERB with the QMDP and POMDP policies. Columns (a) and (e) show, respectively, the initial and final pose of the object relative to the hand. Column (b) shows several samples drawn from the initial belief state. Columns (c) and (d) show the belief state over object pose at two points during execution. The trial numbers referenced in this figure correspond to those used in fig. 6.7 and table 6.1.

algorithm to plan a pre-contact trajectory that terminates once contact is observed (section 6.3). Our experimental results, both in simulation (section 6.4) and on a real robot (section 6.5), show that the policies produced by our approach outperform a baseline policy that does not take multi-step information-gathering actions.

Our real-robot experiments identified two important failure modes in the policies planned with our algorithm. First, one trial failed because of an un-modelled error in the observation model. We discuss how to recover from such a failure in section 8.3.3. Second, the robot collided with the environment in two trials because our algorithm does not consider kinematic constraints during planning. The next chapter addresses this limitation by planning under model 3.

7
Configuration Space Planning

This chapter is adapted from Koval et al. [2016a,b].

Planning for a hand-relative model (model 1 from chapter 3), as we did in chapter 6, often produces policies that are infeasible to execute in clutter. *Kinematic constraints*—like reachability, joint limits, and collision—are constraints on robot configuration. As a result, they fundamentally cannot be incorporated into this type of planner.

This chapter introduces a planner that considers both object pose uncertainty and kinematic constraints. Doing so is challenging because the robot's configuration space is typically continuous and high-dimensional. Additionally, any policy that we compute offline, e.g. the post-contact policy in chapter 6, could be rendered invalid by a change to the obstacles present in the environment.

Instead, we use DESPOT [Somani et al., 2013], an online POMDP solver, to plan for a model that includes both object pose uncertainty and kinematic constraints (model 3 from chapter 3). We leverage two key insights for computational efficiency. First, we lazily discretize the configuration space into a lattice of configurations connected by constrained trajectories. Second, we leverage heuristics computed from the hand-relative model to guide the search (insight 3).

Similar to the pre-contact search described in the previous chapter, this algorithm is an online search guided by heuristics derived from a hand-relative QMDP policy. However, since it plans in the joint space of robot configurations and object poses, we cannot pre-compute a post-contact policy to share between problem instances. We describe how to unify these two approaches in section 7.4.

7.1 Configuration Lattice

Planning in this space requires choosing a suitable representation of robot configuration. We choose to represent the robot's configuration as a point in a configuration space lattice. By exploiting the lower-dimensional structure of planar manipulation (insight 3), we constrain the end-effector to have a fixed transformation ${}^{\text{sup}}T_{\text{ee}} \in \text{SE}(3)$

relative to the support surface $T_{\text{sup}} \in \text{SE}(3)$. A configuration $q \in Q$ satisfies this constraint iff

$$T_{\text{ee}}(q) = T_{\text{sup}}{}^{\text{sup}}T_{\text{ee}} \, \text{Trans}([x_{\text{r}}, y_{\text{r}}, 0]) \, \text{Rot}(\theta_{\text{r}}, \hat{e}_z) \tag{7.1}$$

where $(x_{\text{r}}, y_{\text{r}}, \theta_{\text{r}}) \in X_{\text{r}} \subseteq \text{SE}(2)$ is the pose of the end-effector in the plane, $\text{Rot}(\theta, \hat{v})$ is a rotation about $\hat{v}$ by angle θ, $\text{Trans}(v)$ is a translation by v, and $\hat{e}_z = [0, 0, 1]^T$.

Just as with end-effector pose, we also constrain the movable object to have a fixed transformation ${}^{\text{sup}}T_{\text{o}} \in \text{SE}(3)$ relative to the support surface. We parameterize its pose as

$$x = T_{\text{sup}}{}^{\text{sup}}T_{\text{o}} \, \text{Trans}([x_{\text{o}}, y_{\text{o}}, 0]) \, \text{Rot}(\theta_{\text{o}}, \hat{e}_z)$$

where $(x_{\text{o}}, y_{\text{o}}, \theta_{\text{o}}) \in X_{\text{o}} \subseteq \text{SE}(2)$ is the pose of the object in the plane.

We discretize the space of the end-effector poses X_{r} by constructing a *state lattice* $X_{\text{r,lat}} \subseteq X_{\text{r}}$ with a translational resolution of $\Delta x_{\text{r}}, \Delta y_{\text{r}} \in \mathbb{R}^+$ and an angular resolution of $\Delta\theta_{\text{r}} = 2\pi/n_\theta$ for some integer value of $n_\theta \in \mathbb{N}$ [Pivtoraiko and Kelly, 2005]. The lattice consists of the discrete set of points

$$X_{\text{r,lat}} = \{(i_x \Delta x_{\text{r}}, i_y \Delta y_{\text{r}}, i_\theta \Delta\theta_{\text{r}}) : i_x, i_y, i_\theta \in \mathbb{Z}\}.$$

Each *point* $x_{\text{r}} \in X_{\text{r,lat}}$ may be reachable from multiple configurations. We assume that we have access to an *inverse kinematics function* $q_{\text{lat}}(x_{\text{r}})$ that returns a single solution $\{q\}$ that satisfies $T_{\text{ee}}(q) = x_{\text{r}}$ or $\emptyset$ if no such solution exists. A solution may not exist if x_{r} is not reachable, the end-effector is in collision, or the robot collides with the environment in all possible inverse kinematic solutions.

The inverse kinematics function does not necessarily depend on the structure of the lattice. We use q_{lat} to denote inverse kinematic solutions to remain consistent with the other notation used in this section.

Defining this configuration lattice allows us to plan in the state space $S_{\text{lat}} = Q_{\text{lat}} \times X_{\text{o}}$ where $Q_{\text{lat}} = \bigcup_{x_{\text{r}} \in X_{\text{r,lat}}} q_{\text{lat}}(x_{\text{r}})$ is the set of configurations returned by q_{lat} on all lattice points.

7.1.1 Action Templates

Most actions do not transition between states in the lattice S_{lat}, so we restrict ourselves to actions that are instantiated from one of a finite set A_{lat} of action templates. An *action template* $a_{\text{lat}} = (\xi_{\text{lat}}, \Delta t) \in A_{\text{lat}}$ is a Cartesian trajectory $\xi_{\text{lat}} : [0, \Delta t] \to \text{SE}(3)$ that specifies the relative motion of the end-effector. The template starts at the origin $\xi_{\text{lat}}(0) = I$ and ends at some lattice point $\xi_{\text{lat}}(\Delta t) \in X_{\text{r,lat}}$. More than one action template may end at the same lattice point.

An action $a = (\xi, \Delta t)$ instantiates action template a_{lat} at lattice point $x_{\text{r}} \in X_{\text{r,lat}}$ if it satisfies three conditions: (1) starts at configuration $\xi_{\text{lat}}(0) = q_{\text{lat}}(x_{\text{r}})$, (2) ends at configuration $\xi_{\text{lat}}(\Delta t) = q_{\text{lat}}(x_{\text{r}} \xi_{\text{lat}}(\Delta t))$, and (3) satisfies $T_{\text{ee}}(\xi(\tau)) = x_{\text{r}} \xi_{\text{lat}}(\tau)$ for all $0 \leq \tau \leq$

Δt. These conditions are satisfied iff ξ moves between two configurations in Q_{lat} while producing the same motion as ξ_{lat}.

We define the function $\text{Proj}(x_{\text{r}}, a) \mapsto a_{\text{lat}}$ to map an action a to the action template a_{lat} that it instantiates. The pre-image $\text{Proj}^{-1}(x_{\text{r}}, a_{\text{lat}})$ defines set of all possible instantiations of a_{lat} at x_{r}. We assume that we have access to a *local planner* $\psi(q, a_{\text{lat}})$ that returns a singleton action $\{a\} \subseteq \text{Proj}^{-1}(q, a_{\text{lat}})$ from this set or $\emptyset$ to indicate failure. The local planner may fail due to kinematic constraints, end-effector collision, or robot collision.

7.1.2 *Configuration Lattice POMDP*

We use the lattice to define the *configuration lattice POMDP*, Lat-POMDP, with the state space S_{lat}, actions A_{lat}, observations O, transition model T_{lat}, observation model Ω_{lat}, and reward function R_{lat}. The structure of the lattice guarantees that that all instantiations of the action template a_{lat} execute the same motion ξ_{lat} of the end-effector, regardless of the configuration of the robot.

If the movable object only contacts the end-effector—not other parts of the robot or the environment—then the motion of the object is also independent of the robot configuration. We refer to a violation of this assumption as *un-modelled contact*. The lattice transition model $T_{\text{lat}}(s_{\text{lat}}, a_{\text{lat}}, s'_{\text{lat}})$ is identical to $T(s, a, s')$ when a_{lat} is feasible and no un-modelled contact occur. If either condition is violated, the robot transitions to the absorbing state s_{bad}. Similarly, the lattice observation model $\Omega_{\text{lat}}(o, s_{\text{lat}}, a_{\text{lat}})$ is identical to $\Omega(o, s, a)$ for valid states and is uniform over O for s_{bad}.

We penalize s_{bad}, infeasible actions $\psi(q_{\text{lat}}(x_{\text{r}}), a_{\text{lat}}) = \emptyset$, and un-modelled contact in the reward function by assigning them -1 reward. Otherwise, we define $R_{\text{lat}}(s, a) = R(s, a)$. This choice guarantees that an optimal policy of Lat-POMDP will never take an infeasible action:

Theorem 7.1. *An optimal policy π^*_{lat} of Lat-POMDP will not execute an infeasible action in belief b if $V^*_{lat}[b] > \frac{-1}{1-\gamma}$.*

Proof. Suppose $V^*_{\text{lat}}[b] > \frac{-1}{1-\gamma}$ and an optimal policy π^*_{lat} executes infeasible action a_{lat} in belief state b. The robot immediately receives a reward of -1 and transitions to s_{bad}. For all time after that, regardless of the actions that π^*_{lat} takes, the robot receives a reward of $R_{\text{lat}}(s_{\text{bad}}, \cdot) = -1$ at each time step. This yields a value of $V^*_{\text{lat}}[b] = \frac{-1}{1-\gamma}$, which is the minimum reward possible to achieve.

The optimal value function satisfies the Bellman equation $V^*_{\text{lat}}[b] = \max_{a_{\text{lat}} \in A_{\text{lat}}} Q^*_{\text{lat}}[b, a_{\text{lat}}]$, where $Q^*_{\text{lat}}[b, a_{\text{lat}}]$ denotes the value of taking action a_{lat} in belief state b, then following the optimal policy for all time. We know that $V^*_{\text{lat}}[b] > \frac{-1}{1-\gamma}$, so there must exist at least one feasible action $a^*_{\text{lat}} \in A_{\text{lat}}$ for which $Q^*_{\text{lat}}[b, a^*_{\text{lat}}] > \frac{-1}{1-\gamma}$.

The Bellman equation also asserts that an optimal policy chooses $\pi^*_{\text{lat}}[b] = \arg\max_{a_{\text{lat}} \in A_{\text{lat}}} Q^*_{\text{lat}}[b, a^*_{\text{lat}}]$. This is a contradiction: the policy π^* chose a_{lat} in belief state b, but the fact that $Q^*_{\text{lat}}[b, a_{\text{lat}}] < Q^*_{\text{lat}}[b, a^*_{\text{lat}}]$ implies that π^* is not optimal. Therefore, an optimal policy will not execute an infeasible action if $V^*_{\text{lat}}[b] > \frac{-1}{1-\gamma}$.

□ □

We can strengthen our claim if we guarantee that every reachable lattice point from has at least one feasible action and it is possible to achieve the goal with non-zero probability. Under those assumptions we know that $V^*_{\text{lat}}[b] > \frac{-1}{1-\gamma}$ and theorem 7.1 guarantees that π^*_{lat} will never take an infeasible action. One simple way to satisfy the first condition is to require that all actions be reversible.

7.2 Online POMDP Planner

Lat-POMDP has a large state space that changes whenever obstacles are added to, removed from, or moved within the environment. This makes it difficult to apply an offline method, like we used in chapter 6, to this problem. Instead, we use DESPOT [Somani et al., 2013], an online POMDP solver, to efficiently plan in this space. DESPOT incrementally explores the action-observation tree rooted at $b(s_0)$ by performing a series of trials. Each *trial* starts at the root node, descends the tree, and terminates by adding a new leaf node to the tree.

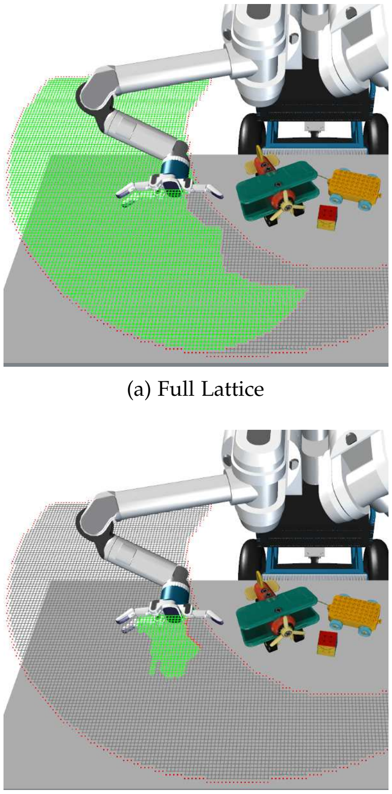
In each step, DESPOT chooses the action that maximizes the upper bound $\bar{V}[b]$ and the observation that maximizes *weighted excess uncertainty*, a regularized version of the gap $\bar{V}[b] - \underline{V}[b]$ between the bounds. This strategy heuristically focuses exploration on the optimally reachable belief space [Kurniawati et al., 2008]. Finally, DESPOT backs up the upper and lower bounds of all nodes visited by the trial.

We leverage two key ideas for computational efficiency. First, we interleave lattice construction with planning to evaluate only the parts of the lattice that are visited by DESPOT (section 7.2.1). Second, we guide DESPOT with upper (section 7.2.2) and lower (section 7.2.3) bounds derived from model 1: a relaxation of the problem that considers only the pose of the movable object relative to the hand.

7.2.1 Configuration Lattice Construction

DESPOT uses upper and lower bounds to focus its search on belief states that are likely to be visited by the optimal policy. We exploit this fact to avoid constructing the entire lattice. Instead, we interleave lattice construction with planning and only instantiate the lattice edges visited by the search, similar to the concept of *lazy evaluation* used in motion planning [Bohlin and Kavraki, 2000, Hauser, 2015].

We begin with no pre-computation and run DESPOT until it queries the transition model T_{lat}, observation model Ω_{lat}, or reward function R_{lat} for a state-action pair $(x_{\text{r}}, a_{\text{lat}})$ that has not yet been evaluated. When this occurs, we pause the search and check the feasibility of the action by running the local planner $\psi(x_{\text{r}}, a_{\text{lat}})$. We use the outcome of the local planner to update the Lat-POMDP model and resume the search. Figure 7.1 shows the (a) full lattice and (b) subset evaluated by DESPOT, only a small fraction of the full lattice.


(a) Full Lattice

(b) Evaluated Lattice

Figure 7.1: The (a) full lattice and the (b) subset evaluated by DESPOT during planning.

It is also possible to use a hybrid approach by evaluating some parts of the lattice offline and deferring others to be computed online. For example, we may compute inverse kinematics solutions, kinematic feasibility checks, and self-collision checks in an offline pre-computation step. These values are fixed for a given support surface and, thus, can be used across multiple problem instances.

7.2.2 *Hand-Relative Upper Bound*

Recall from section 7.1.2 that the motion of the movable object relative to the hand is independent of the robot's configuration. Inspired by model 1, we define a *hand-relative POMDP* (Rel-POMDP) with the state space S_{rel}, actions A_{lat}, observations O, transition model T_{rel}, observation model Ω_{rel}, and reward function R_{rel}. The state space S_{rel} includes only the pose of the movable object relative to hand. The hand-relative transition model, observation model, and reward function are identical to the original model when no un-modelled contact occurs.

Rel-POMDP is equivalent to assuming that environment is empty and the robot is a lone end-effector actuated by an incorporeal planar joint. As a result, Rel-POMDP is a relaxation of Lat-POMDP:

Theorem 7.2. *The optimal value function V^*_{rel} of Rel-POMDP is an upper bound on the optimal value function V^*_{lat} of Lat-POMDP.*

Proof. Define a *scenario* $\psi = (s_0, \psi_1, \psi_2, \dots)$ to be an abstract simulation trajectory that captures all uncertainty in our POMDP model [Ng and Jordan, 2000, Somani et al., 2013]. A scenario is generated by drawing the initial state $s_0 \sim b(s_0)$ from the initial belief state and each random number $\psi_i \sim \text{uniform}[0, 1]$ from the unit interval. Given a scenario ψ, we assume that the outcome of executing a sequence of actions is deterministic; i.e. all stochasticity in the transition and observation models is captured in the random numbers.

Suppose an arbitrary Rel-POMDP policy π executes the sequence of actions $a_{\text{lat},1}, a_{\text{lat},2}, \ldots$ in scenario ψ. The policy visits the sequence of states $s_{\text{rel},0}, s_{\text{rel},1}, \ldots$ and receives the sequence of rewards $R_1, R_2, \ldots$ given by the reward function $R_i = R_{\text{rel}}(s_{\text{rel},i}, a_{\text{lat},i})$.

Consider executing the same π in the same scenario ψ on Lat-POMDP. Let H be the time step at which π first takes an infeasible action or makes un-modelled contact. The policy receives the sequence of rewards $R_1, R_2, \ldots, R_2, \ldots, R_{H-1}, -1, -1, \ldots$ as it did on Rel-POMDP until time step H. Then, it receives -1 reward, transitions to s_{bad}, and continues receive -1 reward for all time.

The policy π achieves value $V^{\pi}_{\text{rel},\psi} = \sum_{t=0}^{\infty} \gamma^t R_t$ on Rel-POMDP and $V^{\pi}_{\text{lat},\psi} = \sum_{t=0}^{H-1} \gamma^t R_t - \frac{\gamma^H}{1-\gamma}$ on Lat-POMDP in scenario ψ. Since $R_t \geq -1$, we know that $V^{\pi}_{\text{rel},\psi} \geq V^{\pi}_{\text{lat},\psi}$. The value function is the expectation $V^{\pi} = E_{\psi}[V^{\pi}_{\psi}]$ over scenarios under the distribution described above, so this relationship proves that $V^{\pi}_{\text{rel}} \geq V^{\pi}_{\text{lat}}$.

Consider the optimal policy π^*_{lat} of Lat-POMDP. There exists some Rel-POMDP policy π_{mimic} that executes the same sequence of action as π^*_{lat} in all scenarios. From the reasoning above, we know that $V^{\pi_{\text{mimic}}}_{\text{rel}} \geq V^*_{\text{lat}}$. We also know that $V^*_{\text{rel}} \geq V^{\pi_{\text{mimic}}}_{\text{rel}}$ because the value of any policy is a lower bound on the optimal value function. Therefore, by the transitive property, we have shown that $V^*_{\text{rel}} \geq V^*_{\text{lat}}$.

□ □

This result implies that any upper bound $\bar{V}_{\text{rel}} \geq V^*_{\text{rel}}$ for Rel-POMDP is also an upper bound on the optimal value function of Lat-POMDP. The key advantage of doing so is that $\bar{V}_{\text{rel}}$ may be pre-computed once per hand-object pair. In contrast, an upper bound specifically constructed for Lat-POMDP must be re-computed for each problem instance.

7.2.3 *Hand-Relative Lower Bound*

We exploit the fact that the value function of any policy is a lower bound on the optimal value function to define $\underline{V}_{\text{lat}}$. We use offline pre-computation to compute a *rollout policy* on π_{rollout} for Rel-POMDP once per object pair, e.g. using MDP value iteration [Littman et al., 1995] or a point-based method [Pineau et al., 2003].

Given π_{rollout}, we construct an approximate lower bound $\underline{V}_{\text{lat}}$ by estimating the value $V^{\pi_{\text{rollout}}}_{\text{lat}}$ of executing π_{rollout} on Lat-POMDP with Monte Carlo rollouts. Approximating a lower bound with a rollout policy is commonly used in POMCP [Silver and Veness, 2010], DESPOT [Somani et al., 2013], and other online POMDP solvers.

Each *rollout* begins by drawing a state $s_0 \sim b(s_0)$ from the initial belief state. Then, for each time step $t = 1, \ldots, H$, we take the action

$a_t = \pi_{\text{rollout}}[b(s_{t-1})]$, receive reward $R_t = R(s_{t-1}, a_t)$, transition to $s_t \sim T(s_{t-1}, a_t, s_t)$, and simulate $o_t \sim \Omega(o_t, s_t, a_t)$. Once a rollout is complete, we compute its value as $\sum_{t=1}^{H} \gamma^{t-1} R_t$. We take the mean of this value across many rollouts to approximate $\underline{V}_{\text{lat}} = V^{\pi_{\text{rollout}}}$.

7.3 *Simulation Experiments*

This section evaluates our planner in simulation experiments of HERB [Srinivasa et al., 2012], the same robot used for the real-robot experiments in chapters 5 and 6, manipulating a movable object on a cluttered table top.

We begin by verifying that DESPOT [Somani et al., 2013], the online method used in this chapter, performs similarly to SARSOP, the offline method we used in chapter 6. We compare the two algorithms on Rel-POMDP and expect:

H1. *Rel-DESPOT performs similarly to Rel-SARSOP on Rel-POMDP.*

Next, we evaluate the importance of considering kinematic constraints and information-gathering during planning. We compare two policies on Lat-POMDP. The first, Rel-SARSOP takes information-gathering actions, but does not consider kinematic constraints. The second, Lift-QMDP considers kinematic constraints, but does not take information-gathering actions.

Our results in chapter 6 indicate that both aspects of manipulation are important. Therefore, we expect to find problem instances for which each algorithm performs better:

H2. *Rel-SARSOP can outperform Lift-QMDP on Lat-POMDP.*

H3. *Lift-QMDP can outperform Rel-SARSOP on Lat-POMDP.*

Our algorithm (Lat-DESPOT) considers both information-gathering and kinematic constraints during planning, so we hypothesize:

H4. *Lat-DESPOT performs the best.*

Our results suggest that all four of these hypotheses are satisfied.

7.3.1 *Experimental Design*

Our simulation experiments required HERB to manipulate a bottle on a cluttered table using a Barrett WAM arm [Salisbury et al., 1988] equipped with a BarrettHand [Townsend, 2000] end-effector. The configuration space $Q = \mathbb{R}^7$ contained the seven joint angles of the WAM arm. In each trial, the robot began at a known configuration and the pose of the movable object was drawn from a Gaussian distribution centered in front of the hand with variance $\Sigma^{1/2} = \text{diag}[5 \text{ mm}, 10 \text{ cm}]$.

Configuration Lattice. First, we constructed a lattice of resolution $\Delta x_\text{r} = \Delta y_\text{r} = 1$ cm at a constant height above the table top. Next, we selected an inverse kinematic solution $q_\text{lat}(x_\text{r})$ for each lattice point $x_\text{r} \in X_\text{r,lat}$ using an iterative inverse kinematics solver initialized with the configuration of an adjacent lattice point. Finally, we used a Cartesian motion planner to find a trajectory that connects those configurations. We deferred collision checking of a trajectory until its feasibility was queried by a planner.

In most configurations, the 1 cm resolution of the lattice is sufficiently dense to simply connect adjacent lattice points with a straight-line trajectory in configuration space.

Transitions and Observation Models. At each time step, the robot executed one of four 1 cm translational actions templates: forward, back, left, or right. We simulated the motion of the movable object using Box2D [Catto, 2010] and introduced noise into its parameters at each time step [Duff et al., 2010]. After taking an action, the robot received binary observations from two contact sensors on its fingertips that perfectly discriminated between contact and no-contact.

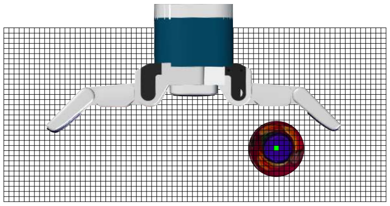


Figure 7.2: Discrete state space used to represent the pose of the object relative to the hand used during planning.

Hand-Relative Discretization. We discretized S_rel to speed up evaluation of the model and to enable calculation of QMDP [Littman et al., 1995] and SARSOP [Kurniawati et al., 2008] policies. We discretized a region of size 20 cm × 44 cm at a 1 cm resolution (fig. 7.2). We did not separate contact and no-contact states because the simple geometry and high resolution of the discretization made doing so unnecessary.

Dependent Measure. We evaluated a policy π on the same two metrics as we used in chapter 6. First, we estimated the value V^π achieved by the policy on the discrete POMDP model. This is the quantity that was directly maximized by our planner. Second, we measured the *success rate* $\Pr(s_t \in G)$ of the policy's ability to push the movable object into the goal region.

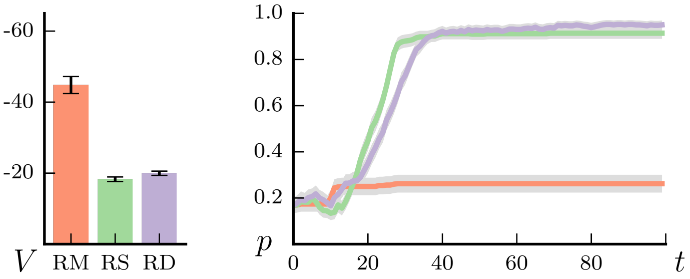
7.3.2 Policies under Evaluation

We evaluate the performance of several policies:

Rel-QMDP applied the QMDP approximation [Littman et al., 1995] to Rel-POMDP. This policy neither takes information-gathering actions nor considers kinematic constraints during planning. This is equivalent to the QMDP post-contact policy from chapter 6.

Rel-SARSOP uses SARSOP [Kurniawati et al., 2008], a point-based method, to compute an offline policy for Rel-POMDP that is capable of taking information-gathering actions. This is equivalent to the POMDP post-contact policy from chapter 6.

Rel-DESPOT uses DESPOT [Somani et al., 2013] to plan for Rel-POMDP using Rel-QMDP as $\bar{V}$ and Rel-QMDP as π_rollout. DESPOT is an online solver that requires minimal pre-computation.



(a) Value, V_{rel}

(b) Success Probability, $\Pr(s_t \in G)$

Figure 7.3: Performance of Rel-QMDP (RM 🟧 —), Rel-SARSOP (RS 🟩 —), and Rel-DESPOT (RD 🟪 —) on Rel-POMDP. (a) Value V_{rel} achieved by each policy. Note that the y-axis is inverted; lower (less negative) is better. (b) Success probability on the continuous Rel-POMDP problem. Results are averaged over averaged over 500 trials and error bars denote a 95% confidence interval. Best viewed in color.

Lift-QMDP and Lift-SARSOP use the state lattice to evaluate the feasibility of the action selected by Rel-QMDP or Rel-SARSOP before executing it. If that action is infeasible, the policy instead executes the feasible action with the next highest value estimate. This is a heuristic solution to modify a policy to obey kinematic constraints.

Lat-DESPOT, the proposed algorithm, uses DESPOT [Somani et al., 2013] to plan for Lat-POMDP using Rel-QMDP as $\bar{V}$ and Lift-QMDP π_{rollout}. This is the only algorithm to both plan information-gathering actions and consider kinematic constraints during planning.

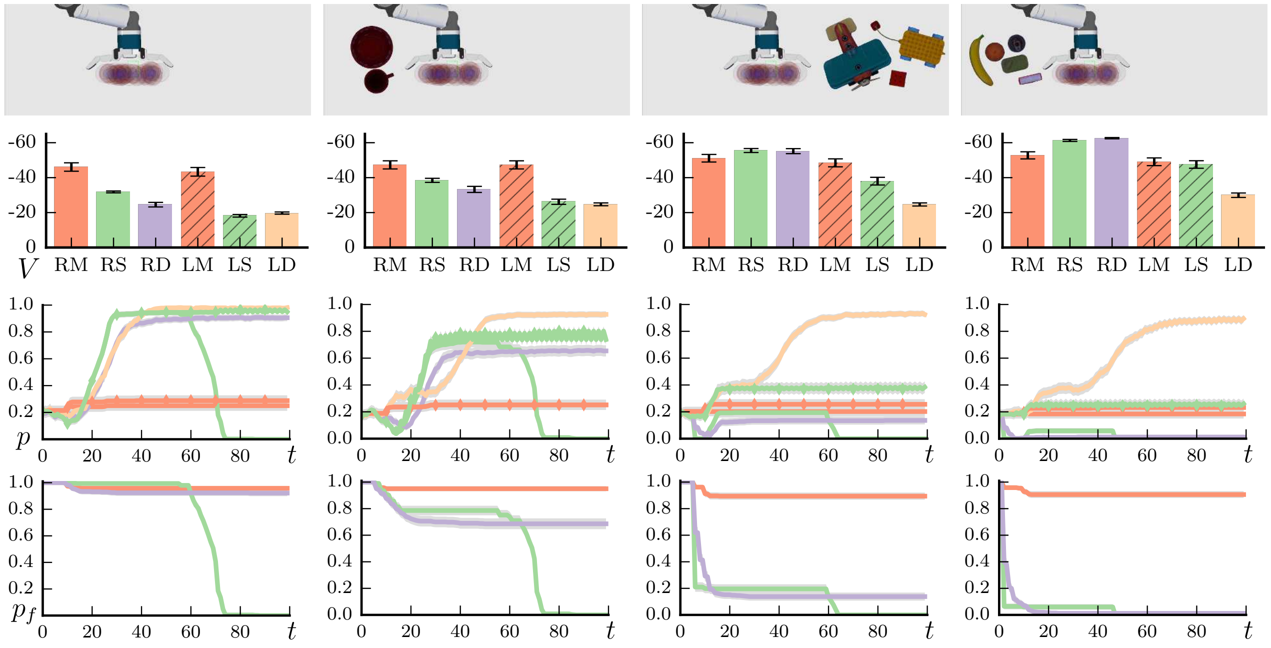
We used the implementations of SARSOP [Kurniawati et al., 2008] and DESPOT [Somani et al., 2013] provided by the APPL toolkit. We tuned DESPOT's parameters for Rel-DESPOT and Lat-DESPOT on a set of trials distinct from the results presented in this thesis.

7.3.3 *Rel-POMDP Experiments*

We begin by considering Rel-POMDP to isolate the effect of uncertainty from that of kinematic constraints. Figure 7.3a shows the value achieved by three policies over 100 time steps on the discretized Rel-POMDP model. Figure 7.3b shows the success probability of each policy when simulated using the continuous model. This confirms that higher value on the discrete model translates into a higher success rate on the continuous model.

Consistent with our findings in chapter 6, these results show that Rel-SARSOP (🟩 —) outperforms Rel-QMDP (🟧 —). This is expected because SARSOP plans multi-step information-gathering actions, whereas QMDP cannot.

Figure 7.3 also supports H1: Rel-DESPOT (🟪 —) performs comparably to Rel-SARSOP. This is an encouraging result because DESPOT, unlike SARSOP, is an online planner that requires minimal offline pre-computation.



(a) No Obstacle

(b) Right Obstacles

(c) Left Obstacles

(d) Complex Obs.

Figure 7.4: Performance of Rel-QMDP (RM 🟧 —), Rel-SARSOP (RS 🟩 —), Rel-DESPOT (RD 🟪 —), Lift-QMDP (LM ▨ ◆), Lift-SARSOP (LS ▨ ◆), and Lat-DESPOT (LD 🟧 —), the proposed algorithm, on four Lat-POMDP environments. (Top) Value achieved by the discretized problem. (Middle) Success probability on the continuous Lat-POMDP problem. (Bottom) Probability that execution is feasible; Lift-QMDP, Lift-SARSOP, and Lat-DESPOT are omitted because they do not take infeasible actions. Results are averaged over 500 trials and error bars denote a 95% confidence interval. Best viewed in color.

7.3.4 *Lat-POMDP Experiments*

Next, we evaluate planning on the Lat-POMDP model in in four different environments: (a) an empty table, (b) obstacles on the right, (c) obstacles on the left, and (d) different obstacles on the right. These scenes are constructed out of objects selected from the YCB dataset [Calli et al., 2015]. Kinematic constraints are present in all of these environments in the form of reachability limits, self-collision, and collision between the arm and the environment.

Figure 7.4 shows results for each scene. Figure 7.4-Top shows the value achieved on the discretized model and Figure 7.4-Middle shows the success rate of each policy. Trials that took an infeasible action were terminated and assigned zero success probability. Figure 7.4-Bottom shows the proportion p_f trials that failed in this way.

Rel-QMDP (RM 🟧 —) and Lift-QMDP (LM ▨ ◆) performed poorly across all four environments, achieving $< 30\%$ success probability, because they did not take multi-step information-gathering actions. This result supports H2: it is important to gather information even when kinematic constraints are present.

Rel-SARSOP (RS 🟩 —) and Rel-DESPOT (RD 🟪 —) performed well on environments (a) and (b) because they hit obstacles late in execution. The converse was true on environments (c) and (d): the policies hit obstacles so quickly that they performed worse than Rel-QMDP! This result supports H3: it is important to consider kinematic constraints even when uncertainty is present.

Lift-SARSOP (LS ▨ ─◆─) performed near-optimally on environments (a) and (b) because it did not take infeasible actions and gathers information. However, it performed no better than Rel-QMDP on problem (d). This occured because Lift-SARSOP myopically considered obstacles in a one-step lookahead and often oscillated when blocked. Small changes in the environment were sufficient to induce this behavior: the key difference between environments (b) and (d) was the introduction of a purple box that creates a cul-de-sac.

Our approach, Lat-DESPOT (🟧 —), avoided myopic behavior by considering action feasibility during planning. Lat-DESPOT performed no worse than Lift-SARSOP on environments (a) and (b) and outperformed it on environments (c) and (d). Unlike Rel-SARSOP, Lat-DESPOT identified the cul-de-sac in (d) during planning and avoided becoming trapped in it. In summary, this result supports H4: Lat-DESPOT is the only policy that performs near-optimally on all four environments because it considers both uncertainty and kinematic constraints during planning.

Our unoptimized implementation of Lat-DESPOT took between 200 μs and 2.4 s to select an action on a single core of a 4 GHz Intel Core i7 CPU. The policy was slowest to evaluate early in execution, when information-gathering was necessary, and fastest once the movable object was localized because the upper and lower bounds become tighter. The QMDP and SARSOP policies, which were computed offline, took an average of 1.6 μs and 218 μs to evaluate respectively.

7.3.5 *Upper Bound Validation*

Finally, we combine the data in Figure 7.3-Left and Figure 7.4-Top to empirically verify the bound we proved in Theorem 7.2. The value of Rel-SARSOP (🟩 —) and Rel-DESPOT (🟪 —) on Rel-POMDP (Figure 7.3) was greater (i.e. less negative) than the value of all policies evaluated on Lat-POMDP (Figure 7.4-Top). The data supports theorem 7.2: the optimal value achieved on Rel-POMDP is no worse than the highest value achieved on Lat-POMDP in environment (a) and greater than the highest value achieved in environments (b), (c), and (d).

7.4 *Discussion*

This chapter introduced a planning algorithm that considers both object pose uncertainty and kinematic constraints (model 3) during planning. Our key insight was to represent robot configuration as a point in a configuration space lattice (section 7.1). We gain efficiency by lazily constructing the lattice as it is searched by an online planning algorithm guided by heuristics derived from a hand-relative relaxation of the problem (section 7.2). Our experimental results in simulation (section 7.3) show that our approach outperforms policies that consider either uncertainty or kinematic constraints in isolation.

Instead of planning with a hand-relative model, like we did in chapter 6, we used the hand-relative model as a heuristic to guide a planner in the joint space of object poses and robot configurations. This allowed our planner to avoid taking infeasible actions by reasoning about kinematic constraints while planning.

However, our planner does not fully leverage the pre- and post-contact policy decomposition introduced by chapter 6. We derived the upper and lower bounds used to guide DESPOT in our experiments from a QMDP policy that does not take multi-step information-gathering actions. As a result, DESPOT must plan information-gathering actions without aid from its heuristic.

There are two ways of potentially unifying the approaches. First, we could use a point-based method to derive tighter upper and lower bounds on the optimal value function that do take information-gathering actions. Second, we could plan using a graph-based method that re-uses the same sub-policy across multiple belief states, e.g. using Monte Carlo value iteration [Bai et al., 2011] or by merging similar belief states under a suitable distance metric [Littlefield et al., 2015].

Finally, our approach does not address the challenge of unmodelled error. This limitation could be addressed by estimating static parameters (section 8.2.4) and learning more accurate physics models from data (section 8.3.3). Since DESPOT is an online method, we are optimistic that our planning approach can scale to the continuous, frequently-changing models introduced by these extensions.

8
Conclusion

This thesis proposed a framework for using feedback from contact sensors to reliably manipulate objects under uncertainty. We formulated manipulation as a stochastic discrete time dynamic system (chapter 3) and used this formulation to develop several algorithms.

First, we used our insight that contact constrains state to a manifold (insight 1) to develop the manifold particle filter for object pose (chapter 4) and robot configuration (chapter 5) estimation. Our results, both in simulation and on real robots, showed that the manifold particle filter outperforms a conventional baseline.

Next, we leveraged our insight that contact decouples the optimal policy (insight 2) to combine online and offline planning for manipulating an object relative to the end-effector (chapter 6). Our results in simulation and on HERB showed that our policy achieves a higher success rate than a baseline.

Finally, we exploited our insight that manipulation occurs in a lower-dimensional space (insight 3) to derive a heuristic for an online planner that considers both uncertainty and kinematic constraints (chapter 7). Our simulation results showed that our policy outperforms baseline policies that consider either aspect in isolation.

This chapter concludes by discussing themes shared throughout this thesis (section 8.1), outlining directions for future work (section 8.2), and identifying open research questions (section 8.3).

8.1 Discussion

This thesis outlined three key insights about manipulation and contact sensing: (1) contact constrains state to a manifold, (2) contact decouples the optimal policy, and (3) manipulation occurs in a lower-dimensional space. This section describes how we translated this insight into efficient algorithms for state estimation and planning.

8.1.1 Formulating Manipulation as a POMDP

We formulated contact manipulation as a partially observable Markov decision process (POMDP, Smallwood and Sondik [1973], Kaelbling et al. [1998]) in the joint space of robot configurations and object poses. Our formulation included all major aspects of manipulation: physics, reachability, collision with obstacles, self-collision, object pose uncertainty, sensor noise, and proprioceptive error.

It was daunting to address all of these at once, so we designed algorithms using the simplified models listed in section 3.5. We omitted robot configuration from chapters 4 and 6, object pose uncertainty from chapter 5, and proprioceptive error from chapter 7.

However, we still leveraged the common formulation to share vocabulary, notation, and insight between problems. For example, we used the fact that contact constrains state to a manifold (insight 1) to estimate object pose in chapter 4, estimate robot configuration in chapter 5, and constrain the size of the post-contact belief space in chapter 6. We are optimistic that our insights will be equally applicable to future problems.

Additionally, our formulation provided us with several concrete benefits.

First, we were able to leverage the rich history of work on Bayesian state estimation and POMDP planning. We adapted the dual proposal distribution from a previous application of Bayesian estimation to mobile robot localization [Thrun et al., 2000a], used a point-based POMDP solver to plan the post-contact policy [Kurniawati et al., 2008], and used an online POMDP solver to cope with kinematic constraints [Somani et al., 2013].

Second, we were able to make rigorous statements about performance. We proved that the manifold particle filter implements a Bayes update, decomposing the policy into pre- and post-contact stages is near-optimal (theorem 6.1), and our online search does not take infeasible actions (theorem 7.1). Theory does not—nor can it—guarantee that our algorithms perform well in practice. It does, however, suggest that our positive results will generalize to other domains by isolating flaws in the model from those in the algorithm.

Third, the value function imposed a natural trade-off between quickly completing a task with low probability and slowly completing it with high probability. Optimizing either quantity in isolation tends to produce poor results. Maximizing success probability results in a circuitous policy that lacks goal-direction. Minimizing time requires committing to a desired success probability, which may not be achievable for a given problem instance. The policies produced by our planning algorithms suffer from neither drawback.

Finally, our formulation generalized to several physics models. We switched between a planar quasi-static physics model (Lynch et al. [1992], chapter 4), the Box2D physics engine (Catto [2010], chapters 6 and 7), and a frictionless model that projects states out of penetration (chapter 5). In future work, these models could be replaced by a three-dimensional physics engine or a model learned from data. Doing so will make the algorithms in this thesis perform better by providing them with a more accurate model of the environment.

8.1.2 Respecting the Contact Constraint

Manipulation differs from many planning and control problems because contact is discontinuous (challenge 1). As a result, applying methods developed for continuous problems to contact sensing often yields poor results. Chapters 4 and 5 demonstrated this: the conventional particle suffered from particle deprivation because its proposal distribution is ill-suited for contact sensing.

The manifold particle filter avoids particle deprivation by sampling from the contact manifold when receiving contact observations. Chapter 6 incorporates the same insight into planning by differentiating between states that are in contact and those that are not while discretizing the state space. In both cases, we successfully tailored a well-known approach to manipulation by respecting the sharp discontinuity between “contact” and “no-contact.”

8.1.3 Combining Online and Offline Planning

Modern belief space planners are broadly split into two categories. Offline solvers plan a policy before starting execution. Once planning is complete, the agent does not deviate from that policy. Online planners interleave planning with execution. The agent plans one action, executes it, and re-plans after receiving an observation.

Offline solvers produce policies that are efficient to evaluate by performing most costly computation, e.g. evaluating the transition and observation models, offline. Unfortunately, doing so requires foreknowledge of the initial belief state that is often not available in manipulation. Additionally, it requires a policy representation that is amenable to optimization. Such representations, e.g. α-vectors [Smallwood and Sondik, 1973], scale poorly to large state spaces.

Online planners avoid these challenges by deferring all computation until the policy is queried during execution [Ross et al., 2008]. Doing so allows online planners to scale to large, continuous state spaces that do not emit a compact policy representation. However, online planners tend to produce policies that are expensive to evaluate online. Chapter 7 showed that an offline policy produced by

SARSOP [Kurniawati et al., 2008] takes only 218 μs to evaluate, compared to an online policy produced by DESPOT [Somani et al., 2013] that takes up to 2.4 s.

This thesis combined online and offline planning to achieve a level of performance that is possible by neither alone. Chapter 6 used an offline post-contact policy to truncate an online search when contact is observed. Chapter 7 used an offline solver to compute a heuristic that guides an online search. In both cases, we carefully constructed the problem such that the offline solver does not require foreknowledge of the initial belief state. Then, we used an online solver to adapt the offline solution to a particular problem instance. This is a powerful technique that could apply to other domains.

8.1.4 Focusing Effort with Lazy Evaluation

Manipulation often occurs in a high-dimensional state space. This makes it intractable for to build explicit structures, like the contact manifold in chapter 4 or the configuration lattice in chapter 7, over state space. We leveraged two techniques to gain tractability.

First, we used domain knowledge to construct the subset of the structure relevant to the task. Chapter 4 showed that the trajectory rollout representation of the contact manifold outperformed all others by focusing on the reachable subset of state space. Chapter 6 applied the same insight to planning by computing the post-contact policy only over a small trust region near the hand. This mirrors how a point-based POMDP solver samples the reachable [Pineau et al., 2003] or optimally reachable [Kurniawati et al., 2008] belief space.

Second, we used lazy evaluation to defer construction until necessary. Chapter 5 avoided explicitly representing the contact manifold by representing it implicitly as the iso-contour of a function. Doing so allowed us to scale the manifold particle filter from object pose estimation to robot configuration estimation.

Chapter 7 combined both techniques. DESPOT [Somani et al., 2013] uses upper and lower bounds on the optimal value function to focus tree construction on the optimally reachable belief space [Kurniawati et al., 2008]. Our algorithm gained further efficiency by deferring evaluation of the local planner—and, thus, collision checking of lattice edges—until evaluated by DESPOT. This mirrors the lazy collision checking strategy employed by lazy motion planning algorithms [Bohlin and Kavraki, 2000, Hauser, 2015].

8.2 Limitations and Future Work

Our state estimation and planning algorithms produced promising results both in simulation and on real-robots. However, they suffer from several limitations that we outline in this section. Addressing these limitations would enable our algorithms to handle more complex tasks, produce higher-quality policies, and scale up from the simplified models in section 3.5 to the full manipulation problem.

8.2.1 Planning in a Continuous State Space

Chapters 6 and 7 discretized the space of object poses relative to the hand for planning. We did so for two important reasons.

First, the discretized transition and observation models were faster to evaluate than the corresponding continuous models. Table 8.1 shows that the difference in speed ranges from a factor of 54× for Box2D to more than 400× for MuJoCo, the fastest three-dimensional physics engine we evaluated.

Second, a discrete state space allowed us to represent policies and value functions with α-vectors. It is possible to generalize α-vectors to α-functions over continuous space [Porta et al., 2006], but doing so requires making restrictive assumptions about the structure of the value function. Evaluating a policy graph [Bai et al., 2011], the most common policy representation used for continuous state spaces, requires many computationally-expensive evaluations of the transition and observation models.

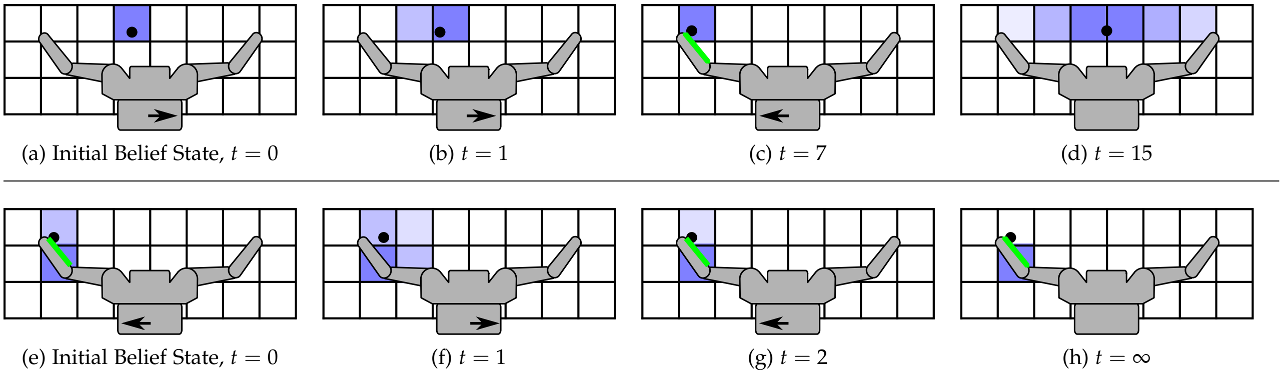
Method	×RT
DART [Liu et al., 2016]	194.6
MuJoCo [Todorov, 2016]	245.8
Box2D [Catto, 2010]	1852.7
Discrete	101841.0

Table 8.1: Speed, as a multiple of real-time (×RT), for a 1 cm planar push. The discretized model is several orders of magnitude faster than all physics models we evaluated.

Unfortunately, discretization also introduces artifacts into the transition and observation models. Figure 8.1 shows two examples of artifacts in a simple problem domain. The robot has contact sensors on its fingertips and can move left, right, or forward in increments of 3 mm. The object is a point, drawn as a small black circle. The state space is discretized into a 1 cm grid and the discrete belief state is drawn by shading each cell in proportion to its probability.

In fig. 8.1 (a)–(d) the robot moves right to perform an information-gathering action, then moves left to grasp the object. After (b) the first timestep there is 2/3 probability that the object remains in its initial state and 1/3 probability that it transitions to the adjacent state. This uncertainty is eliminated when (c) the object contacts the sensor. Unfortunately, by (d) the time the information-gathering action is complete, enough uncertainty has accumulated to render the action useless. This demonstrates that discretization can *introduce additional uncertainty* that is not present in the continuous model.

Discretization can also cause the robot to become overly confident about its state. In fig. 8.1 (e)–(h), the robot's contact sensor cannot



(a) Initial Belief State, $t = 0$ (b) $t = 1$ (c) $t = 7$ (d) $t = 15$

(e) Initial Belief State, $t = 0$ (f) $t = 1$ (g) $t = 2$ (h) $t = \infty$

Figure 8.1: Examples of discretization (a)–(d) introducing uncertainty and (e)–(g) causing overconfidence in belief. See the text for a detailed description.

differentiate between two discrete states. We assign 2/3 and 1/3 probability, respectively, to the lower and upper states that overlap with the sensor. In this situation, the discrete model implies that the robot can localize the object in the lower state by (e) moving left, (f) moving right, and (g)–(h) repeating. In reality, the robot has not localized the object. This shows that discretization can *lead to overconfidence* compared to the continuous model.

Scaling up to longer horizon plans and more intricate geometry exacerbates this problem, making discretization infeasible. Discretization can be thought of as a learned model that encodes discrete state as a one-hot binary feature [Sherstov and Stone, 2005]. Combining a richer set of features with a learning algorithm tailored for multi-step prediction [Venkatraman et al., 2015] could yield the similar computational benefits as naïve discretization with fewer artifacts.

Additionally, we could leverage the loose coupling between model evaluations performed during planning to parallelize computation.

8.2.2 Planning in a Continuous Action Space

Chapters 6 and 7 also discretized the action space. This was necessary because most POMDP solvers, with one notable exception [Seiler et al., 2015], require a finite set of actions. Unfortunately, restricting our planner to that set artificially limits the quality of the policy that it can find. Other policies could achieve higher value by executing actions outside of that set.

Additionally, discretization ignores valuable structure in the action space. Nearby actions are likely to transition to similar belief states and, thus, have similar value. A planner could exploit this structure to share lower bound, upper bound, and value estimates between nearby actions. Formalizing this concept, e.g. through Lipschitz continuity of the transition model, could provide optimality guarantees with respect to the continuous action space.

Macro actions could allow us to leverage continuous structure

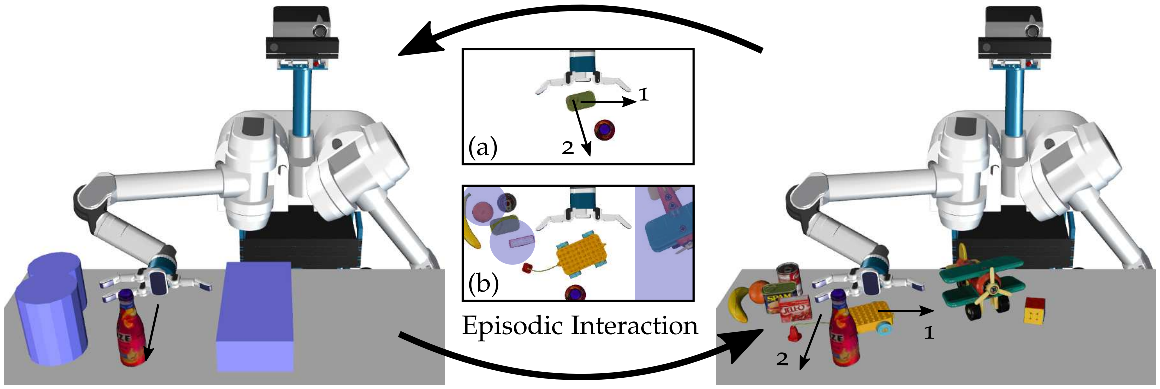


Figure 8.2: HERB uses (a) lower bounds derived from harder and (b) upper bounds derived from easier problems to guide its search. As HERB gains more experience, he can transition from solving easy (left) to hard (right) problem instances.

without abandoning discrete search. Consider a macro action that executes a primitive action k times in sequence. Combining macro actions of different lengths could allow a planner to quickly explore with long actions, then switch to short actions to achieve the goal. This strategy is similar to an experience graph, which biases an A^* search over a large graph to a small sub-graph that may represent the connectivity of the space [Phillips et al., 2012].

While macro actions increase the effective planning horizon, they do not expand the set of available primitives. We could use a planner like GPS-ABT to incrementally refine a discrete set of actions [Seiler et al., 2015]. Alternatively, we could use an optimizer to plan directly in the infinite-dimensional space of trajectories [Zucker et al., 2013, Schulman et al., 2014]. Incorporating trajectory optimization into a macro action [Vien and Toussaint, 2015] could allow a POMDP solver to break free from a finite set of action primitives.

8.2.3 Learning through Episodic Interaction

Section 8.1.3 showed how offline computation can be used to speed up online evaluation. This was possible because manipulation is often episodic: it consists of similar tasks being repeated many times with only subtle variation between episodes. As a result, or same reason, we expect that experience gained in one episode could prove valuable in future episodes.

Consider the example shown in fig. 8.2. Suppose the robot has an existing policy for grasping a cylindrical object under uncertainty and in the presence of clutter (left). Now, for the first time, the robot experiences an object in front of its goal (Episode (a)). If we were to simply execute the policy, it would fail by hitting an obstacle.

However, similar to chapter 7, we could use the past experience as a heuristic to guide an online planner. Easier episodes, like fig. 8.2 (a), could be used as an upper bound and harder episodes, like fig. 8.2 (a), could be used as a lower bound on the optimal value function. The planner would detect the problem, search through the space of possible solutions, and find a policy that completes the task. The robot would then update its heuristic accordingly.

Leveraging past experience requires balancing the value provided by a heuristic against the cost of evaluating it. The planner must identify a relevant subset of past episodes to use as a heuristic. If the set is too small, then the planner will lack the focus necessary to find a high-quality solution. If the set is too large, then the cost of evaluating the heuristic will dominate planning. We are optimistic that domain knowledge, such as the relaxation used in chapter 7, and a distance metric could be used to quickly identify relevant episodes from a large corpus of experience.

8.2.4 Static Parameter Estimation

Our model assumes that its transition and observations models are corrupted by noise that is independent between time steps when conditioned on state. This assumption, required by the Markov assumption, and is valid for difficult-to-model transient error. Unfortunately, it is violated by systematic error in static parameters of the model (e.g. geometry, friction coefficients).

Incorporating this uncertainty into our model requires adding all unknown parameters to state. Unfortunately, estimating static parameters with a particle filter quickly leads to particle deprivation due to their deterministic dynamics. The Liu and West filter [Liu and West, 2001] avoids degeneracy by kernel smoothing its belief over static parameters. This smoothing operation is similar to—and could be combined with—the kernel density estimate used to compute dual importance weights (section 4.2.3) in the manifold particle filter.

Estimating static parameters provides another mechanism for a robot to learn from experience. When a robot encounters an object for the first time, it has little information about its physical properties and plans a conservative policy to achieve the goal. As it completes the task, the robot learns about the object and refines its belief state over those properties. Next time the robot encounters that object, it could transfer that knowledge to the initial belief state and complete the task more quickly.

8.2.5 Multiple Sensing Modalities

Robotic manipulators typically have a variety of sensors, including cameras, depth sensors, joint encoders, and contact sensors. This thesis focused on proprioception and contact sensing at the exclusion of other modalities. By doing so, we developed algorithms that cope with the unique challenges associated with those modalities.

We could further improve the performance of the system by incorporating observations from all of the robot's sensors. This is theoretically straightforward: our formulation can naturally incorporate any sensor for which a probabilistic model is available. The Bayes update optimally combines all available observations into a unified belief state. Different sensing modalities typically have uncorrelated error, so the conditional independence assumption we made in chapter 3 still allows us to implement the update efficiently.

There are two key challenges to integrating vision and depth sensing into our framework. First, it is challenging to build an probabilistic model of these sensing modalities. Error is often spatio-temporally correlated due to occlusion, lighting, specularity, and shadows. Second, the space of observations is high-dimensional. A single image captured by a camera contains millions of pixels, each of which has one (depth image) or three (color image) channels. Most of this information is irrelevant to the task, so we must plan using algorithms that improve—instead of degrade due to excess branching—when observations are numerous.

8.3 Open Questions

We identified several open research questions in preparation of this thesis. Answering these questions may be important to better understand the theory of manipulation and advance the start-of-the-art in robustness.

8.3.1 Measure-Theoretic Formulation

This thesis exploits the fact that contact constrains state to a manifold (insight 1) to improve state estimation (chapters 4 and 5) and adaptively discretize the state space (chapter 6). These algorithms rely on our ability to factor the belief state $b(s_t) = b(s_t \in S_\text{c})b(s_t|S_\text{c}) + b(s_t \in S_\text{nc})b(s_t|S_\text{nc})$ into a distribution $b(s_t|S_\text{c})$ over the contact manifold and a distribution $b(s_t|S_\text{nc})$ over free space. However, this definition introduces an apparent inconsistency: How can $b(s_t \in S_\text{c})$ be non-zero when S_c is a lower-dimensional manifold?

This section may appear to be mathematical pedantry. However, rigorous definitions are required to make any meaningful statements about probability in the presence of contact.

We formalize this question using measure theory. Our probability space (S, F, μ) consists of the sample space S, the σ-algebra of events $F \subseteq 2^S$, and a probability measure $\mu : F \to \mathbb{R}$. Most applications assume that μ is isomorphic to the Lebesgue measure $\lambda : F \to \mathbb{R}$ over the unit interval [Rokhlin, 1962]. Since the n-dimensional Lebesgue measure assigns zero measure to any set with dimension less than n, there does not exist a measurable map between μ and λ because any such map would assign $\mu(S_c) = 0$.

Instead, we express the probability measure $\mu = \mu_c + \mu_{nc}$ as the sum of two measures over, where $\mu_c : 2^{S_c} \to \mathbb{R}$ is a measure over S_c and μ_{nc} is a measure over S_{nc}. If μ_c and μ_{nc} are *partial probability measures* and satisfy $\mu_c(S_c) + \mu_{nc}(S_{nc})$, then μ is a probability measure over S. Any such probability distribution μ' over S can be defined in terms of a *probability density function* $p(s)$ with respect to μ. In this case, the probability density function is the Radon–Nikodym derivative of μ' and $\int_A p(s) d\mu(s)$ is the Lebesgue integral of $p(s)$ over $A \subseteq S$, both taken with respect to measure μ [Resnick, 1999].

A measure μ is a partial probability measure if $\mu(\emptyset) = 0$, $\mu(S) \leq 1$, and μ_j is σ-additive.

We intuitively arrived at the same understanding by factoring the belief state as a weighted sum. The marginal $b(s_t \in S_c) = \int_{S_c} b(s_t) d\mu(s_t)$ is the total probability contributed by the partial probability measure μ_c, the probability of s_t residing on S_c. The conditional distribution $b(s_t|S_c)$ is simply the measure of the corresponding partial probability measure normalized such that it sums to one.

Technically, the conditional belief $p(A|B)$ is undefined if $p(B) = 0$. This is why we defined μ as the sum of partial probability measures instead of as the convex combination of full probability measures. This is not a concern for the MPF because it only samples from A when it has positive marginal probability.

8.3.2 Value of Information-Gathering

The planning algorithms introduced in thesis take information-gathering actions when doing so is beneficial. Our results confirm that this behavior is valuable: the policies produced by our algorithms consistently outperform QMDP [Littman et al., 1995], a baseline that does not take multi-step information-gathering actions. However, planning these policies is more computationally expensive than evaluating a policy that does not do information-gathering.

In some cases, information-gathering may not be necessary. Contact is used both to reduce uncertainty and to complete the task. As a result, the robot may naturally gain information while completing the task. Section 6.5 showed an example of this: one instance of the QMDP policy achieved success after the object naturally came into contact with one of HERB's fingertip contact sensors.

Chapter 6 also showed that increasing sensor coverage improves the success rate of QMDP policy more than that of our policy. Further increasing sensor resolution could continue to close the gap between the two policies. Integrating other sensing modalities (section 8.2.5), like vision or depth sensing, could have the same effect. Designing an effective manipulation system requires balancing hardware complexity (sensor coverage, resolution) with software complexity (information-gathering, multiple sensing modalities).

Note that formulating manipulation as a POMDP (chapter 3) is valuable even if information-gathering is not required. Modeling uncertainty is required to plan robust policies, like QMDP, that do not take information-gathering actions. Modern online POMDP solvers, like SARSOP [Kurniawati et al., 2008] and DESPOT [Somani et al., 2013], use these policies as heuristics to guide their search. When those heuristics are informative, i.e. the QMDP policy performs well, DESPOT is negligibly slower than the heuristics that guide it.

8.3.3 Un-Modelled Error

Our model incorporates uncertainty in the initial belief state, transition model, and observation model. If these stochastic models are accurate, then a Bayes filter optimally estimates state and a POMDP solver optimally trade off between information-gathering and goal-directed behavior. The same is not true if the model is inaccurate.

We intentionally introduced one source of error by assuming that unknown properties, e.g. friction coefficient or object geometry, are corrupted by independent noise at each time step. In reality, these parameters are often static or highly correlated over time. This may result in error accumulating, rather than cancelling. We found this to be an acceptable compromise to avoid addressing the challenges of estimating static parameters outlined in section 8.2.4.

Other sources of error are more insidious. A small error in object geometry can cause a seemingly-infeasible observation to occur. If this occurs during execution, the Bayes update would produce a degenerate belief state. Sensor hysteresis, a common issue with force sensors, can lead to the same outcome.

Both of these sources of errors are difficult to model and, even if accurate models were available, other sources of un-modelled error would arise. It is difficult to even formulate this challenge as a research question: any attempt to model un-modelled error is inherently flawed. Discriminative learning provides a potential solution: we can use any inconsistencies we detect during execution to improve our models for future use. This is an exciting opportunity to combine recent work in machine learning with the model-based estimation and planning algorithms introduced by this thesis.

There is a stark distinction between "known unknowns" that are included in our stochastic model and "unknown unknowns" that are not. It is fundamentally impossible to optimize a policy over an unknown distribution of error.

8.4 Final Thoughts

Our long-term vision is for a robot to maintain a probabilistic representation of the world that is updated as its interacts the environment. This representation should include uncertainty in the configuration of the robot, the pose of objects in the environment, the geometry of obstacles, and inaccuracies in the robot's model of physics. Using this representation, the robot may reason about the propagation of uncertainty through multi-step manipulation tasks and take sensing actions when necessary to achieve success.

This thesis made a first step towards this vision by developing state estimation (chapters 4 and 5) and planning (chapters 6 and 7) algorithms tailored for manipulation with contact sensors. Consistent with our vision, these algorithms maintain a principled representation of uncertainty based on Bayesian probability. Our proposed future work (section 8.2) outlines several concrete extensions of these algorithms to improve their generality and performance.

We are optimistic that the insights (section 1.2) and techniques (section 8.1) developed by this thesis will help to realize this vision.

Bibliography

A.-A. Agha-mohammadi, S. Chakravorty, and N.M. Amato. FIRM: Feedback controller-based information-state roadmap–a framework for motion planning under uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011. DOI: 10.1109/IROS.2011.6095010.

M. Athans. The role and use of the stochastic linear-quadratic-Gaussian problem in control system design. *IEEE Transactions on Automatic Control*, 16(6):529–552, 1971. DOI: 10.1109/TAC.1971.1099818.

J.A. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T.Y. Liu, N. Pollard, et al. An integrated system for autonomous robotics manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. DOI: 10.1109/IROS.2012.6385888.

H. Bai, D. Hsu, W.S. Lee, and V.A. Ngo. Monte Carlo value iteration for continuous-state POMDPs. In *Workshop on the Algorithmic Foundations of Robotics*, 2011. DOI: 10.1007/978-3-642-17452-0_11.

P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14: 239–256, 1992. DOI: 10.1109/34.121791.

J. Bimbo, L. D. Seneviratne, K. Althoefer, and H. Liu. Combining touch and vision for the estimation of an object's pose during manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013. DOI: 10.1109/IROS.2013.6696931.

R. Bohlin and L.E. Kavraki. Path planning using lazy PRM. In *IEEE International Conference on Robotics and Automation*, pages 521–528, 2000. DOI: 10.1109/ROBOT.2000.844107.

R. Bolles and R. Paul. The use of sensory feedback in a programmable assembly system. Technical Report STAN-CS-396, Computer Science Department, Stanford University, Stanford, CA, 1973.

B. Boots, A. Byravan, and D. Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *IEEE International Conference on Robotics and Automation*, 2014. DOI: 10.1109/ICRA.2014.6907443.

M. Brokowski, M. Peshkin, and K. Goldberg. Curved fences for part alignment. In *IEEE International Conference on Robotics and Automation*, 1993. DOI: 10.1109/ROBOT.1993.292216.

E. Brunskill, L.P. Kaelbling, T. Lozano-Pérez, and N. Roy. Continuous-state POMDPs with hybrid dynamics. In *International Symposium on Artificial Intelligence and Mathematics*, 2008.

B. Calli, A. Singh, A. Walsman, S.S. Srinivasa, P. Abbeel, and A.M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *IEEE International Conference on Advanced Robotics*, 2015. DOI: 10.1109/ICAR.2015.7251504.

E. Catto. Box2D. `http://box2d.org`, 2010.

H. Dang, J. Weisz, and P.K. Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *IEEE International Conference on Robotics and Automation*, pages 5917–5922, 2011. DOI: 10.1109/ICRA.2011.5979679.

M.A. Diftler, J.S. Mehling, M.E. Abdallah, N.A. Radford, L.B. Bridgwater, A.M. Sanders, R.S. Askew, D.M. Linn, J.D. Yamokoski, F.A. Permenter, et al. Robonaut 2–the first humanoid robot in space. In *IEEE International Conference on Robotics and Automation*, 2011. DOI: 10.1109/ICRA.2011.5979830.

M.R. Dogar. *Physics-Based Manipulation Planning in Cluttered Human Environments*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2013.

M.R. Dogar and S.S. Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010. DOI: 10.1109/IROS.2010.5652970.

M.R. Dogar and S.S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3):217–236, 2012. DOI: 10.1007/s10514-012-9306-z.

M.R. Dogar, V. Hemrajani, D. Leeds, B. Kane, and S. Srinivasa. Proprioceptive localization for mobile manipulators. Technical Report CMU-RI-TR-10-05, The Robotics Institute, Carnegie Mellon University, 2010.

D.J. Duff. *Visual motion estimation and tracking of rigid bodies by physical simulation*. PhD thesis, University of Birmingham, 2011.

D.J. Duff, J. Wyatt, and R. Stolkin. Motion estimation using physical simulation. In *IEEE International Conference on Robotics and Automation*, 2010. DOI: 10.1109/ROBOT.2010.5509590.

M.A. Erdmann and M.T. Mason. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, 1988. DOI: 10.1109/56.800.

P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(19):415–428, 2012. DOI: 10.4086/toc.2012.v008a019.

K. Gadeyne, T. Lefebvre, and H. Bruyninckx. Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation. *International Journal of Robotics Research*, 24(8):615–630, 2005. DOI: 10.1177/0278364905056196.

J.D. Gammell, S.S. Srinivasa, and T.D. Barfoot. BIT*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs. In *IEEE International Conference on Robotics and Automation*, 2015. DOI: 10.1109/ICRA.2015.7139620.

E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. In *IEEE International Conference on Robotics and Automation*, 1988. DOI: 10.1109/56.2083.

N.J. Gordon, Salmond D.J., and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F*, 1993. DOI: 10.1049/ip-f-2.1993.0015.

S. Haidacher. *Contact Point and Object Position from Force/Torque and Position Sensors for Grasps with a Dextrous Robotic Hand*. PhD thesis, Technical University of Munich, 2004.

K. Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *IEEE International Conference on Robotics and Automation*, pages 2951–2957, 2015. DOI: 10.1109/ICRA.2015.7139603.

P. Hebert, T. Howard, N. Hudson, J. Ma, and J.W. Burdick. The next best touch for model-based localization. In *IEEE International Conference on Robotics and Automation*, 2013. DOI: 10.1109/ICRA.2013.6630562.

M. Horowitz and J. Burdick. Interactive non-prehensile manipulation for grasping via POMDPs. In *IEEE International Conference on Robotics and Automation*, 2013. DOI: 10.1109/ICRA.2013.6631031.

K. Hsiao. *Relatively robust grasping*. PhD thesis, Massachusetts Institute of Technology, 2009.

K. Hsiao, L.P. Kaelbling, and T. Lozano-Pérez. Grasping POMDPs. In *IEEE International Conference on Robotics and Automation*, 2007. DOI: 10.1109/ROBOT.2007.364201.

K. Hsiao, T. Lozano-Pérez, and L.P. Kaelbling. Robust belief-based execution of manipulation programs. In *Workshop on the Algorithmic Foundations of Robotics*, 2008.

Y.K. Hwang and N. Ahuja. Gross motion planning–a survey. *ACM Computing Surveys*, 1992. DOI: 10.1145/136035.136037.

N. Hyafil and F. Bacchus. Conformant probabilistic planning via CSPs. In *International Conference on Automated Planning and Scheduling*, 2003.

S. Javdani, M. Klingensmith, J.A. Bagnell, N.S. Pollard, and S.S. Srinivasa. Efficient touch based localization through submodularity. In *IEEE International Conference on Robotics and Automation*, 2013. DOI: 10.1109/ICRA.2013.6630818.

E.T. Jaynes. Information theory and statistical mechanics. *The Physical Review*, 106(4):620–630, 1957. DOI: 10.1103/PhysRev.106.620.

S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 1997.

L.P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *International Journal of Robotics Research*, 2013. DOI: 10.1177/0278364913484072.

L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998. DOI: 10.1016/S0004-3702(98)00023-X.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960. DOI: 10.1115/1.3662552.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7): 846–894, 2011. DOI: 10.1177/0278364911406761.

L.E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4): 556–580, 1996. DOI: 10.1109/70.508439.

M. Klingensmith, T. Galluzzo, C. Dellin, M. Kazemi, J.A. Bagnell, and N. Pollard. Closed-loop servoing using real-time markerless arm tracking. In *IEEE International Conference on Robotics and Automation Humanoids Workshop*, 2013.

M. Klingensmith, M.C. Koval, S.S. Srinivasa, N.S. Pollard, and M. Kaess. The manifold particle filter for state estimation on high-dimensional implicit manifolds. In *arXiv*, 2016. `arXiv:1604.07224 [cs.RO]`.

M.C. Koval, M.R. Dogar, N.S. Pollard, and S.S. Srinivasa. Pose estimation for contact manipulation with manifold particle filters. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013a. DOI: 10.1109/IROS.2013.6697009.

M.C. Koval, M.R. Dogar, N.S. Pollard, and S.S. Srinivasa. Pose estimation for contact manipulation with manifold particle filters. Technical Report CMU-RI-TR-13-11, The Robotics Institute, Carnegie Mellon University, 2013b.

M.C. Koval, N.S. Pollard, and S.S. Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In *Robotics: Science and Systems*, 2014. DOI: 10.15607/RSS.2014.X.034.

M.C. Koval, J.E. King, N.S. Pollard, and S.S. Srinivasa. Robust trajectory selection for rearrangement planning as a multi-armed bandit problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015a. DOI: 10.1109/IROS.2015.7353743.

M.C. Koval, N.S. Pollard, and S.S. Srinivasa. Pose estimation for planar contact manipulation with manifold particle filters. *International Journal of Robotics Research*, 34(7), June 2015b. DOI: 10.1177/0278364915571007.

M.C. Koval, D. Hsu, N.S. Pollard, and S.S. Srinivasa. Configuration lattices for planar contact manipulation under uncertainty. In *arXiv*, 2016a. arXiv:1605.00169 [cs.RO].

M.C. Koval, D. Hsu, N.S. Pollard, and S.S. Srinivasa. Configuration lattices for planar contact manipulation under uncertainty. In *Workshop on the Algorithmic Foundations of Robotics*, 2016b.

M.C. Koval, N.S. Pollard, and S.S. Srinivasa. Manifold representations for state estimation in contact manipulation. In *International Symposium of Robotics Research*, 2016c. DOI: 10.1007/978-3-319-28872-7_22.

M.C. Koval, N.S. Pollard, and S.S. Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. *International Journal of Robotics Research*, 35(1–3):244–264, JanuaryâĂŞ-March 2016d. DOI: 10.1177/0278364915594474.

J.J. Kuffner and S.M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, 2000. DOI: 10.1109/ROBOT.2000.844730.

H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008. DOI: 10.15607/RSS.2008.IV.009.

S.M. LaValle and S.A. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *International Journal of Robotics Research*, 1998. DOI: 10.1177/027836499801700104.

W.S. Lee, N. Rong, and D.J. Hsu. What makes some POMDP problems easy to approximate? In *Advances in Neural Information Processing Systems*, 2007.

Q. Li, C. Schürmann, R. Haschke, and H. Ritter. A control framework for tactile servoing. In *Robotics: Science and Systems*, 2013. DOI: 10.15607/RSS.2013.IX.045.

Z. Littlefield, D. Klimenko, H. Kurniawati, and K.E. Bekris. The importance of a suitable distance function in belief-space planning. In *International Symposium of Robotics Research*, 2015.

M.L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: Scaling up. *International Conference on Machine Learning*, 1995. DOI: 10.1016/B978-1-55860-377-6.50052-9.

C.K. Liu, M.X. Grey, and J. Lee. Dynamic Animation and Robotics Toolkit. `http://dartsim.github.io`, 2016.

J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice*. Springer, 2001. DOI: 10.1007/978-1-4757-3437-9_10.

T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 1983. DOI: 10.1109/TC.1983.1676196.

T. Lozano-Pérez, M. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984. DOI: 10.1177/027836498400300101.

K.M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992. DOI: 10.1109/IROS.1992.587370.

L. Manuelli and R. Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. Under Review, 2016.

M.T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986. DOI: 10.1177/027836498600500303.

W. Meeussen, J. Rutgeerts, K. Gadeyne, H. Bruyninckx, and J. De Schutter. Contact-state segmentation using particle filters for programming by human demonstration in compliant-motion tasks. *IEEE Transactions on Robotics*, 23(2):218–231, 2007. DOI: 10.1109/TRO.2007.892227.

A.Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence*, 2000.

E. Nikandrova, J. Laaksonen, and V. Kyrki. Towards informative sensor-based grasp planning. *Robotics and Autonomous Systems*, 62(3): 340–354, 2014. DOI: 10.1016/j.robot.2013.09.009.

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. *International Journal of Robotics Research*, 33(5), 2014. DOI: 10.1177/0278364913514466.

P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011. DOI: 10.1109/IROS.2011.6095059.

J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley, 1984. ISBN 978-0201055948.

A. Petrovskaya and O. Khatib. Global localization of objects via touch. *IEEE Transactions on Robotics*, 27(3):569–585, 2011. DOI: 10.1109/TRO.2011.2138450.

M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev. E-Graphs: Bootstrapping planning with experience graphs. In *Robotics: Science and Systems*, 2012. doi: 10.15607/RSS.2012.VIII.043.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005.

R. Platt, A.H. Fagg, and R.A. Grupen. Nullspace grasp control: theory and experiments. *IEEE Transactions on Robotics*, 26(2):282–295, 2010a. doi: 10.1109/TRO.2010.2042754.

R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*, 2010b. doi: 10.15607/RSS.2010.VI.037.

R. Platt, L. Kaelbling, T. Lozano-Pérez, and R. Tedrake. Simultaneous localization and grasping as a belief space control problem. In *International Symposium of Robotics Research*, 2011.

R. Platt., L.P. Kaelbling, T. Lozano-Pérez, and R. Tedrake. Non-Gaussian belief space planning: Correctness and complexity. In *IEEE International Conference on Robotics and Automation*, 2012. doi: 10.1109/ICRA.2012.6225223.

I. Pohl. *Practical and theoretical considerations in heuristic search algorithms*. University of California, Santa Cruz, 1977.

J.M. Porta, N. Vlassis, M.T.J. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.

S. Resnick. *A Probability Path*. Birkhäuser, Boston, 1999. ISBN 978-0817684082. doi: 10.1007/978-0-8176-8409-9.

ReFlex Hand. RightHand Robotics, 2016.

V.A. Rokhlin. *On the fundamental ideas of measure theory*. American Mathematical Society, 1962.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 2008. doi: 10.1613/jair.2567.

K. Salisbury, W. Townsend, B. Eberman, and D. DiPietro. Preliminary design of a whole-arm manipulation system (WAMS). In *IEEE International Conference on Robotics and Automation*, 1988. DOI: 10.1109/ROBOT.1988.12057.

S. Sanan, M.H. Ornstein, and C.G. Atkeson. Physical human interaction for an inflatable manipulator. In *International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 7401–7404, 2011. DOI: 10.1109/IEMBS.2011.6091723.

T. Schmidt, R. Newcombe, and D. Fox. DART: Dense articulated real-time tracking. In *Robotics: Science and Systems*, 2014. DOI: 10.15607/RSS.2014.X.030.

T. Schmidt, K. Hertkorn, R. Newcombe, Z. Marton, M. Suppa, and D. Fox. Depth-based tracking with physical constraints for robot manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015. DOI: 10.1109/ICRA.2015.7138989.

A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. DOI: 10.1109/IROS.2009.5354648.

J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, 33(9):1251–1270, August 2014. DOI: 10.1177/0278364914528132.

K. Seiler, H. Kurniawati, and S.P.N. Singh. GPS-ABT: An online and approximate solver for POMDPs with continuous action space. In *IEEE International Conference on Robotics and Automation*, 2015. DOI: 10.1109/ICRA.2015.7139503.

A.A. Sherstov and P. Stone. Function approximation via tile coding: Automating parameter choice. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 194–205. Springer, 2005. DOI: 10.1007/3-540-44914-0.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

B.W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 97–99, 1981.

R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973. DOI: 10.1287/opre.21.5.1071.

D.E. Smith and D.S. Weld. Conformant graphplan. In *National Conference on Artificial Intelligence*, 1998.

A. Somani, N. Ye, D. Hsu, and W.S. Lee. DESPOT: Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems*, 2013.

S.S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M.R. Dogar, A.D. Dragan, R.A. 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*, 100(8): 1–19, 2012. DOI: 10.1109/JPROC.2012.2200561.

F. Stulp, E. Theodorou, J. Buchli, and S. Schaal. Learning to grasp under uncertainty. In *IEEE International Conference on Robotics and Automation*, pages 5703–5708, 2011. DOI: 10.1109/ICRA.2011.5979644.

Y. Tenzer, L.P. Jentoft, and R.D. Howe. The feel of MEMS barometers: Inexpensive and easily customized tactile array sensors. 21(3):89–95, September 2014. DOI: 10.1109/MRA.2014.2310152.

S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distribution. In *National Conference on Artificial Intelligence*, 2000a.

S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. Technical Report CMU-CS-00-125, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 2000b.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005. DOI: 10.1145/504729.504754.

E. Todorov. *MuJoCo: Modeling, Simulation and Visualization of Multi-Joint Dynamics with Contact*. Roboti Publishing, Seattle, 2016.

W. Townsend. The BarrettHand grasper–programmably flexible part handling and assembly. *Industrial Robot: An International Journal*, 27 (3):181–188, 2000. DOI: 10.1108/01439910010371597.

J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Robotics: Science and Systems*, 2010. DOI: 10.15607/RSS.2010.VI.017.

J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research*, 31, 2011. DOI: 10.1177/0278364911406562.

G. van den Bergen. *Proximity Queries and Penetration Depth Computation on 3D Game Objects*. 2001.

A. Venkatraman, M. Hebert, and J.A. Bagnell. Improving multi-step prediction of learned time series models. In *National Conference on Artificial Intelligence*, pages 3024–3030, 2015.

N.A. Vien and M. Toussaint. POMDP manipulation via trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 242–249, 2015. DOI: 10.1109/IROS.2015.7353381.

S.N. Šimunović. *An Information Approach to Parts Mating*. PhD thesis, Massachusetts Institute of Technology, 1979.

R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng, and B. Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9-11):612–621, 2006. DOI: 10.1007/s00371-006-0052-0.

P.M. Will and D.D. Grossman. An experimental system for computer controlled mechanical assembly. *IEEE Transactions on Computers*, 100 (9):879–888, 1975. DOI: 10.1109/T-C.1975.224333.

J. Xiao. Automatic determination of topological contacts in the presence of sensing uncertainties. In *IEEE International Conference on Robotics and Automation*, 1993. DOI: 10.1109/ROBOT.1993.291962.

D. Xu, G.E. Loeb, and J.A. Fishel. Tactile identification of objects using Bayesian exploration. In *IEEE International Conference on Robotics and Automation*, 2013. DOI: 10.1109/ICRA.2013.6631001.

H. Zhang and N.N. Chen. Control of contact via tactile sensing. *IEEE Transactions on Robotics and Automation*, 16(5):482–495, 2000. DOI: 10.1109/70.880799.

L. Zhang and J.C. Trinkle. The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing. In *IEEE International Conference on Robotics and Automation*, 2012. DOI: 10.1109/ICRA.2012.6225125.

L. Zhang, S. Lyu, and J.C. Trinkle. A dynamic Bayesian approach to simultaneous estimation and filtering in grasp acquisition. In *IEEE International Conference on Robotics and Automation*, 2013. DOI: 10.1109/ICRA.2013.6630560.

M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C.M. Dellin, J.A. Bagnell, and S.S. Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research*, 32(9–10):1164–1193, August/September 2013. DOI: 10.1177/0278364913488805.