Configuration Lattices for Planar Contact Manipulation Under Uncertainty

Michael Koval[1], David Hsu[2], Nancy Pollard[1], and Siddhartha Srinivasa[1]

[1] The Robotics Institute, Carnegie Mellon University
[2] Department of Computer Science, National University of Singapore

Abstract. This work addresses the challenge of a robot using real-time feedback from contact sensors to reliably manipulate a movable object on a cluttered tabletop. We formulate this task as a partially observable Markov decision process (POMDP) in the joint space of robot configurations and object poses. This formulation enables the robot to explicitly reason about uncertainty and all major types of kinematic constraints: reachability, joint limits, and collision. We solve the POMDP using DESPOT, a state-of-the-art online POMDP solver, by leveraging two key ideas for computational efficiency. First, we lazily construct a discrete lattice in the robot's configuration space. Second, we guide the search with heuristics derived from an unconstrained relaxation of the problem. We empirically show that our approach outperforms several baselines on a simulated seven degree-of-freedom manipulator.

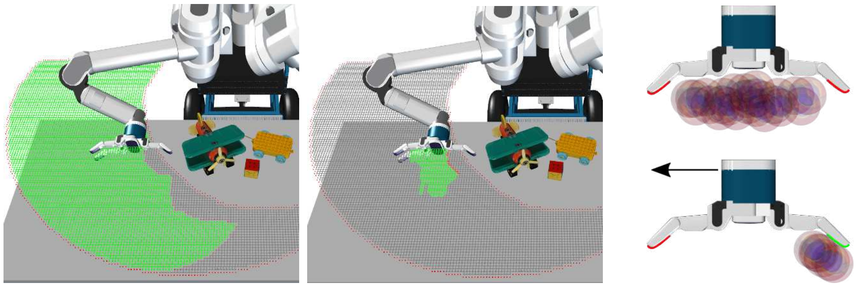
1 Introduction

Our goal is to enable robots to use real-time contact sensing to reliably manipulate their environments. We focus on contact sensing because it is an ideal source of feedback for manipulation: the sense of touch directly observes the forces that the robot imparts on its environment and is unaffected by occlusion.

Consider a robot trying to push a bottle into its palm (Figure 1). The robot is uncertain about the initial pose of the bottle, has only an approximate model of physics, and receives feedback from noisy contact sensors on its fingertips. To localize the object and complete the task, the robot must take *information-gathering actions* to force the bottle into contact with one of its sensors. Simultaneously, the robot must avoid *kinematic constraints* such as colliding with an obstacle or pushing the bottle outside of its workspace.

Prior work focuses on planning under kinematic constraints and under uncertainty in isolation (Section 2). Work that considers kinematic constraints, but ignores uncertainty, often produces brittle policies that fail when executed. Work that considers uncertainty, but ignores kinematic constraints, often produces policies that are infeasible to execute in clutter.

In this paper, we formulate planar manipulation as a *partially observable Markov decision process* (POMDP) [42] in the joint space of robot configurations and object poses (Section 3). This formulation enables the robot to explicitly reason about uncertainty and, when necessary, plan information-gathering



(a) Problem Definition (b) Online Search (c) Relaxed Problem

Fig. 1: HERB [44] uses real-time feedback from contact sensors to manipulate a bottle on a cluttered table. (a) We represent the robot's configuration as a point in a lattice that models the reachable (—) and unreachable (—) parts of the robot's configuration space. (b) We use an online POMDP solver guided by powerful heuristics to construct only the portions of the lattice relevant to completing the task. These heuristics are derived from (c) a relaxation of the problem that considers only the motion of the object relative to the end-effector.

actions to reduce it. Unlike in our prior work [25], this model also includes all major types of kinematic constraints: reachability, joint limits, and collision.

While exactly solving a POMDP is intractable in the worst case, recent advances in approximate online [41, 43] and point-based [26] solvers have enabled successful POMDP planning for manipulation tasks [17, 18, 25]. We use DESPOT [43], a state-of-the-art online POMDP solver, and leverage two key ideas for computational efficiency. First, we lazily discretize the robot's configuration space into a lattice of configurations connected by constrained trajectories (Section 4). Second, we leverage heuristics computed from an unconstrained relaxation of the problem to guide the online search (Section 5). We prove that the optimal policy will not take infeasible actions and our heuristics do not compromise the optimality of the solution.

We validate the algorithm on a simulation of HERB [44], a robot equipped with a 7-DOF Barrett WAM arm [39], manipulating an object on a cluttered tabletop (Figure 1, Section 6). Our results show that the proposed algorithm succeeds more often than approaches that consider either uncertainty or kinematic constraints in isolation.

2 Related Work

Our work builds on a long history of research on manipulating objects under uncertainty. Early work focused on planning open-loop trajectories that successfully reconfigure an object despite non-deterministic uncertainty [27] in its pose [5, 14]. Recently, the same type of worst-case analysis has been used to plan robust open-loop trajectories for grasping [9, 10] and rearrangement planning [11, 23]. Our approach makes the quasistatic assumption [32], similar to this prior work, but differs in two key ways: it (1) considers probabilistic uncertainty [27] and (2) produces a closed-loop policy that uses real-time feedback.

Another line of research aims to incorporate feedback from contact sensors into manipulator control policies, e.g. to optimize the quality of a grasp [37], servo towards a desired contact sensor observation [28, 47], or learn a feedback policy to achieve a grasp [34, 45]. These approaches achieve impressive real-time performance, but require a higher-level planner to specify the goal.

One common strategy is to plan a sequence of move-until-touch actions that localize an object, then execute an open-loop trajectory to complete the task [16, 21, 35]. Other approaches formulate the problem as a POMDP [42] and find a policy that takes information-gathering actions only when they are necessary to achieve the goal [17, 19]. Unfortunately, most of this work assumes that the end-effector can move freely in the workspace and that objects do not significantly move when touched.

Recent work, including our own [25], has relaxed the latter assumption by incorporating a stochastic physics model into the POMDP [17] and using SARSOP [26], an offline point-based POMDP solver, to find a policy for manipulating an object relative to the hand. Unfortunately, hand-relative policies often fail when executed on a manipulator due to kinematic reachability or collision with obstacles. We use a hand-relative policy to guide DESPOT [43], an online POMDP solver [38], in a search that explicitly models these constraints.

Our approach represents the robot's configuration space as a state lattice [36], a concept that we borrow from mobile robot navigation [29] and search-based planning [8]. Similar to many randomized motion planners [3, 15], we use lazy evaluation to defer collision checking until an action is queried by the planner.

3 Problem Formulation

We formulate planar contact manipulation as a partially observable Markov decision process (POMDP) [42]. A POMDP is a tuple (S, A, O, T, Ω, R) where S is the set of states, A is the set of actions, O is the set of observations, $T(s, a, s') = p(s'|s, a)$ is the transition model, $\Omega(s, a, o) = p(o|s, a)$ is the observation model, and $R(s, a) : S \times A \to \mathbb{R}$ is the reward function (Figure 2).

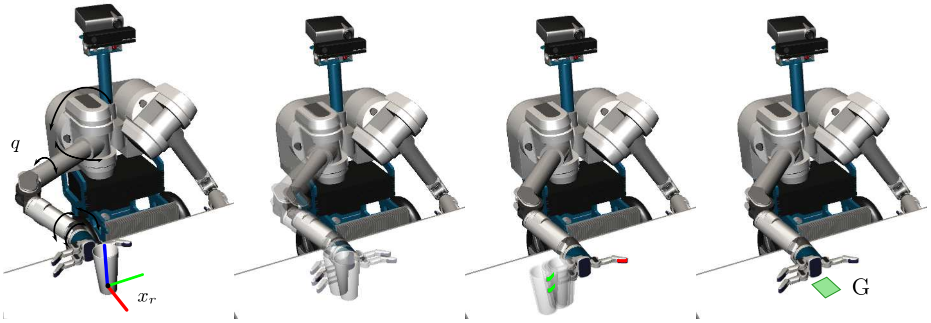
The robot does not know the true state s_t. Instead, the robot tracks the *belief state* $b(s_t) = p(s_t|a_{1:t}, o_{1:t})$, a probability distribution over the current state s_t conditioned on the history of actions $a_{1:t} = \{a_1, \ldots, a_t\}$ and observations $o_{1:t} = \{o_1, \ldots, o_t\}$. The set of all belief states is known as *belief space* Δ.

Our goal is to find a policy $\pi : \Delta \to A$ that optimizes the *value function*

$$V^{\pi}[b] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$$

where the expectation $E[\cdot]$ is taken over the sequence of states visited by π. The *discount factor* $\gamma \in [0, 1)$ adjusts the value of present versus future reward.

In our problem, *state* $s = (q, x_o) \in S$ is the configuration of the robot $q \in Q$ and the pose of the movable object $x_o \in X_o = \mathrm{SE}(3)$ (Figure 2a). An *action*



(a) State, s (b) Action, a (c) Observation, o (d) Goal Region, G

Fig. 2: We formulate planar contact manipulation as a POMDP with a state space (a) that contains the robot configuration q and the pose of the movable object x_r. The environment contains one movable object (white glass) and static obstacles. (b) action is a short joint-space trajectory ξ of duration T. After executing an action, the robot receives feedback from (c) binary contact sensors on its end-effector. The goal of pushing the object into (d) the hand-relative goal region G is encoded in the reward function $R(s, a)$.

$a = (\xi, T) \in A$ is a trajectory $\xi : [0, T] \to Q$ that starts in configuration $\xi(0)$ and ends configuration $\xi(T)$ at time T (Figure 2b). We assume that uncertainty over object pose dominates controller and proprioceptive error. Therefore, we treat q as fully-observable and assume that the manipulator is perfectly position controlled.

The robot executes a quasistatic push if it comes in contact with the movable object. The *quasistatic assumption* states that friction is high enough to neglect acceleration of the object, i.e. the object stops moving as soon as it leaves contact [32], and is accurate for many tabletop manipulation tasks [10, 11]. We define the stochastic *transition model* $p(s'|s, a) = T(s, a, s')$ in terms of a deterministic quasistatic physics model [32] by introducing noise into its parameters [12]. We do not attempt to refine our estimate of these parameters during execution.

After executing an action, the robot receives an *observation* $o \in \{0, 1\}^{n_\mathrm{o}} = O$ from its n_o binary contact sensors (Figure 2c). We assume that an *observation model* $p(o|s', a) = \Omega(s', a, o)$ is available, but make no assumptions about its form.

The robot's goal is to push the movable object into a hand-relative *goal region* $X_\mathrm{goal} \subseteq X$ (Figure 2d). We encode this in the *reward function* that assigns $R(s, a) = 0$ for states $[T_\mathrm{ee}(q)]^{-1} x_\mathrm{o} \in X_\mathrm{goal}$ in the goal region and $R(s, a) = -1$ otherwise. In this expression, $T_\mathrm{ee} : Q \to \mathrm{SE}(3)$ is the forward kinematics of the end effector. Note that the choice of -1 reward is arbitrary: any negative reward would suffice.

4 Configuration Lattice POMDP

Solving this POMDP is challenging. We simplify the problem by constraining the end effector to a fixed transformation relative to the support surface and build a lattice in configuration space (Section 4.1). Configurations in the lattice are connected by action templates that start and end on lattice points (Section 4.2) and are penalized if rendered infeasible by kinematic constraints (Section 4.3).

4.1 Configuration Lattice

We assume that the robot's end effector is constrained 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 satisfies this constraint iff

$$T_{\text{ee}}(q) = T_{\text{sup}} {}^{\text{sup}}T_{\text{ee}} \,\text{Trans}([x_{\text{r}}, y_{\text{r}}, 0])\, \text{Rot}(\theta_{\text{r}}, \hat{e}_z) \tag{1}$$

where $(x_{\text{r}}, y_{\text{r}}, \theta_{\text{r}}) \in X_{\text{r}} = SE(2)$ 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.

We also assume that the movable object also has a fixed transformation ${}^{\text{sup}}T_{\text{o}} \in SE(3)$ relative to the support surface. We parameterize its pose as

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

where $(x_{\text{o}}, y_{\text{o}}, \theta_{\text{o}}) \in X_{\text{o}} = \text{SE}(2)$ 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 $n_\theta \in \mathbb{N}$ [36]. 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.

4.2 Action Templates

Most actions do not transition between states in the lattice S_{lat}. Therefore, we restrict ourselves to actions that are instantiated from one of a finite set A_{lat} of action templates. An *action template* $a_{\text{lat}} = (\xi_{\text{lat}}, T) \in A_{\text{lat}}$ is a Cartesian trajectory $\xi_{\text{lat}} : [0, T] \to SE(3)$ 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 $\xi_{\text{lat}}(T) \in X_{\text{r,lat}}$. It is acceptable for multiple actions templates in A_{lat} to end at the same lattice point or have different durations.

An action $a = (\xi, T) \in A$ instantiates template a_{lat} at lattice point $x_{\mathrm{r}} \in X_{\mathrm{r,lat}}$ if it satisfies three conditions: (1) starts in configuration $\xi(0) = q_{\mathrm{lat}}(x_{\mathrm{r}})$, (2) ends in configuration $\xi(T) = q_{\mathrm{lat}}(x_{\mathrm{r}} \xi_{\mathrm{lat}}(T))$, and (3) satisfies $T_{\mathrm{ee}}(\xi(\tau)) = x_{\mathrm{r}} \xi_{\mathrm{lat}}(\tau)$ for all $0 \leq \tau \leq T$. These conditions are satisfied if ξ moves between two configurations in Q_{lat} and produces the same end effector motion as ξ_{lat}.

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

4.3 Configuration Lattice POMDP

We use the lattice to define a *configuration lattice POMDP*, Lat-POMDP, $(S_{\mathrm{lat}}, A_{\mathrm{lat}}, O, T_{\mathrm{lat}}, \Omega_{\mathrm{lat}}, R_{\mathrm{lat}})$ in state space $S_{\mathrm{lat}} = X_{\mathrm{r,lat}} \times X_{\mathrm{o}}$. The structure of the lattice guarantees that that all instantiations $\mathrm{Proj}^{-1}(q, a_{\mathrm{lat}})$ of the action template a_{lat} execute the same motion ξ_{lat} of the end effector. This motion is independent of the starting pose of the end effector x_{r} and configuration $q_{\mathrm{lat}}(x_{\mathrm{r}})$ 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 these variables. We refer to a violation of this assumption as *un-modelled contact.* The lattice transition model $T_{\mathrm{lat}}(s_{\mathrm{lat}}, a_{\mathrm{lat}}, s'_{\mathrm{lat}})$ is identical to $T(s, a, s')$ when a_{lat} is feasible and no un-modelled contact occurs. If either condition is violated, the robot transitions to the absorbing state s_{invalid}. Similarly, the lattice observation model $\Omega_{\mathrm{lat}}(s_{\mathrm{lat}}, a_{\mathrm{lat}}, o)$ is identical to $\Omega(s, a, o)$ for valid states and is uniform over O for $s_{\mathrm{lat}} = s_{\mathrm{invalid}}$.

We penalize invalid states $s_{\mathrm{lat}} = s_{\mathrm{invalid}}$, infeasible actions $\phi(q_{\mathrm{lat}}(x_{\mathrm{r}}), a_{\mathrm{lat}}) = \emptyset$, and un-modelled contact in the reward function R_{lat} by assigning them $\min_{s \in S, a \in A} R(s, a) = -1$ reward. Otherwise, we define $R_{\mathrm{lat}}(s, a) = R(s, a)$. This choice guarantees that an optimal policy π^*_{lat} will never take an infeasible action:

Theorem 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 $V^*_{\mathrm{lat}}[b] > \frac{-1}{1-\gamma}$ and an optimal policy π_{lat} executes the invalid action in belief state b. The robot receives a reward of -1 and transitions to s_{invalid}. For all time after that, regardless of the actions that π_{lat} takes, the robot receives a reward of $R_{\mathrm{lat}}(s_{\mathrm{invalid}}, \cdot) = -1$ at each timestep. This yields a total reward of $V^{\pi_{\mathrm{lat}}}_{\mathrm{lat}} = \frac{-1}{1-\gamma}$, which is the minimum reward possible to achieve.

The value function of the optimal policy satisfies the Bellman equation $V^*_{\mathrm{lat}}[b] = \arg\max_{a_{\mathrm{lat}} \in A_{\mathrm{lat}}} Q^*[b, a_{\mathrm{lat}}]$, where $Q^*[b, a]$ denotes the value of taking action a in belief state b, then following the optimal policy for all time. This contradicts the fact that $V^*_{\mathrm{lat}}[b] > \frac{-1}{1-\gamma}$ and $V^{\pi_{\mathrm{lat}}}_{\mathrm{lat}}[b] = \frac{-1}{1-\gamma}$. Therefore, π_{lat} must not be the optimal policy. □

We can strengthen our claim if we guarantee that every lattice point reachable from q_0 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 1 guarantees that π^*_{lat} will never take an infeasible action. One simple way to satisfy this condition is to require that all actions be reversible.

5 Online POMDP Planner

Lat-POMDP has a large state space that changes whenever obstacles are added to, removed from, or moved within the environment. We use DESPOT [43], 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 [26]. 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 5.1). Second, we guide DESPOT with upper (Section 5.2) and lower (Section 5.3) bounds derived from a relaxation of the problem that considers only the pose of the movable object relative to the hand.

5.1 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 [3, 15].

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 $\phi(x_{\text{r}}, a_{\text{lat}})$. We use the outcome of the local planner to update the Lat-POMDP model and resume the search. Figure 1 shows the (a) full lattice and (b) subset evaluated by DESPOT, only a small fraction of the full lattice.

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.

5.2 Hand-Relative Upper Bound

Recall from Section 4 that the motion of the end effector and the object is independent of the pose of the end effector x_{r} or the robot configuration q. We use this insight to define a *hand-relative POMDP*, Rel-POMDP, $(S_{\mathrm{rel}}, A_{\mathrm{lat}}, O, T_{\mathrm{rel}}, \Omega_{\mathrm{rel}}, R_{\mathrm{rel}})$ with a state space that only includes the pose $x_{\mathrm{o,rel}} = x_{\mathrm{r}}^{-1} x_{\mathrm{o}} \in S_{\mathrm{rel}}$ of the movable object relative to the hand. The hand-relative transition model T_{rel}, observation model Ω_{rel}, and reward function R_{rel} are identical to the original model when no un-modelled contact occurs.

Rel-POMDP is identical to the hand-relative POMDP models used in prior work [17, 24, 25] and 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 2. *The optimal value function V^*_{rel} of Rel-POMDP is an upper bound on the optimal value function V^*_{lat} of Lat-POMDP: $V^*_{rel}[b] \geq V^*_{lat}[b]$ for all $b \in \Delta$.*

Proof. We define a *scenario* $\psi = (s_0, \psi_1, \psi_2, \dots)$ as an abstract simulation trajectory that captures all uncertainty in our POMDP model [33, 43]. A scenario is generated by drawing the initial state $s_o \sim b(s_0)$ from the initial belief state and each random number $\psi_1 \sim \text{uniform}[0, 1]$ uniformly 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 is captured in the initial state s_0 and the sequence of random numbers $\psi_1, \psi_2, \dots$.

Suppose we have a policy π for Rel-POMDP that executes the sequence of actions $a_{\mathrm{lat},1}, a_{\mathrm{lat},2}, \dots$ in scenario ψ. The policy visits the sequence of states $s_{\mathrm{rel},1}, s_{\mathrm{rel},2}, \dots$ and receives the sequence of rewards $R_1, R_2, \dots$.

Now consider executing π in the same scenario ψ on Lat-POMDP. Without loss of generality, assume that π first takes an infeasible action or makes un-modelled contact with the environment at timestep H. The policy receives the same sequence of rewards $R_1, R_2, \dots, R_2, \dots, R_{H-1}, -1, -1, \dots$ as it did on Rel-POMDP until timestep H. Then, it receives -1 reward for taking an infeasible action, transitions to absorbing state s_{invalid}, and receives -1 reward for all time.

Policy π achieves value $V^{\pi}_{\mathrm{rel},\psi} = \sum_{t=0}^{\infty} \gamma^t R_t$ on Rel-POMDP and $V^{\pi}_{\mathrm{lat},\psi} = \sum_{t=0}^{H-1} \gamma^t R_t - \frac{\gamma^H}{1-\gamma}$ on Lat-POMDP in scenario ψ. Since $R_t \geq -1$, we know that $V^{\pi}_{\mathrm{rel},\psi} \geq V^{\pi}_{\mathrm{lat},\psi}$. The value of a policy $V^{\pi} = E_{\psi}[V^{\pi}_{\psi}]$ is the expected value of π over all scenarios.

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

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

5.3 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 hand-object pair, e.g. using MDP [31] or point-based [17, 25] value iteration.

Given π_{rollout}, we construct an approximate lower bound $\underline{V}_{\text{lat}}$ for Lat-POMDP by estimating the value $V_{\text{lat}}^{\pi_{\text{rollout}}}$ of executing π_{rollout} on Lat-POMDP via Monte Carlo rollouts. Approximating a lower bound with a *rollout policy* is commonly used in POMCP [41], DESPOT [43], and other online POMDP solvers.

6 Experimental Results

We validated the efficacy of the proposed algorithm by running simulation experiments on HERB [44], a robot equipped with a 7-DOF Barrett WAM arm [39] and the BarrettHand [46] end-effector. The robot attempts to push a bottle into the center of its palm on a table littered with obstacles.

6.1 Problem Definition

The state space of the problem consists of the configuration space $Q = \mathbb{R}^7$ of the robot and the pose of the object X_{o} relative to the end effector. The robot begins in known configuration q_0 and x_{o} is drawn from a Gaussian distribution centered in front of the palm with a covariance matrix of $\Sigma^{1/2} = \text{diag}[5 \text{ mm}, 10 \text{ cm}]$.

Transitions Model. At each timestep, the simulated robot chooses an action a_{lat} that moves 1 cm at a constant Cartesian velocity in the xy-plane. The motion of the object is simulated using the Box2D physics simulator [7]. We simulate uncertainty in the model by sampling the hand-object friction coefficient and center of the object-table pressure distribution at each timestep [12, 24].

Configuration Lattice. These actions define a lattice centered at $T_{\text{ee}}(q_0)$ with a resolution of $\Delta x_{\text{r}} = \Delta y_{\text{r}} = 1$ cm. To construct this lattice, we select a configuration $q_{\text{lat}}(x_{\text{r}})$ using an iterative inverse kinematics solver initialized with the solution of an adjacent lattice point. Then, we use a Cartesian motion planner to find a trajectory that connects adjacent points while satisfying the action template. As described in Section 5.1, the kinematic structure of the lattice is computed offline, but all collision checking is deferred until runtime. Forward kinematics, inverse kinematics, and collision detection is provided by the Dynamic Animation and Robotics Toolkit (DART) [1].

Observation Model. The simulated robot receives binary observations from contact sensors on its fingertips. We assume that the sensors perfectly discriminate between contact and no-contact [24, 25], but provide no information about where contact occurred on the sensor. The robot must take information-gathering actions, by moving side-to-side, to localize the object.

Discretization. We discretize S_{rel}, as in prior work [17, 25], to speed up evaluation of the model and to enable calculation of the QMDP [31] and SARSOP [26] policies. We discretize a region of size 20 cm × 44 cm centered around the palm at a 1 cm resolution (Figure 3a). To do so, we compute a discrete transition model, observation model, and reward function by taking an expectation over the underlying continuous state. States outside of this region are considered to be s_{invalid}.

6.2 Baseline Policies

We compare Lat-DESPOT against several baseline policies:

Rel-QMDP chooses the action at each timestep that greedily optimizes the Q-value of the MDP value function defined by Rel-POMDP [31]. QMDP does not perform multi-step information-gathering actions, but has been shown to perform well in domains where information is easily gathered [13, 20].

Rel-SARSOP uses SARSOP [26], a point-based method, to compute an offline policy for Rel-POMDP that is capable of taking information-gathering actions. SARSOP has been shown to perform well on Rel-POMDP in prior work [17, 25]. As in that work, we used the implementation of SARSOP provided by the APPL toolkit and allowed it to run for 10 minutes offline.

Rel-DESPOT uses DESPOT [43] to plan for Rel-POMDP using Rel-QMDP as an upper bound and rollouts of Rel-QMDP as a lower bound. We use the implementation of DESPOT provided by the APPL toolkit and tuned its parameters on a set of training problem instances distinct from these results.

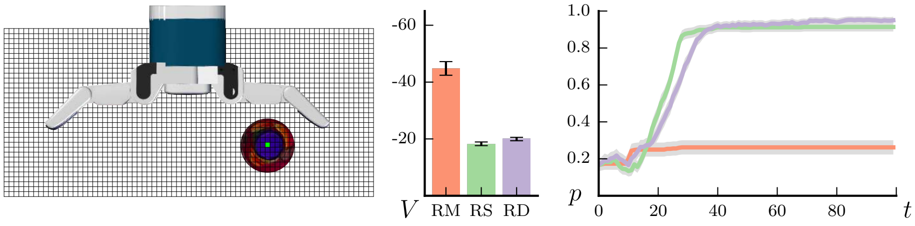
Lift-QMDP and Lift-SARSOP use the state lattice to evaluate the feasibility of the action returned by Rel-QMDP and Rel-SARSOP, respectively, before executing it. If the desired action is infeasible, instead execute the feasible action with the next highest Q-value. This represents a heuristic solution for modifying a Rel-POMDP policy to avoid taking infeasible actions.

Lat-DESPOT, the proposed algorithm, uses DESPOT [43] to plan for Lat-POMDP using Rel-QMDP as the upper bound and rollouts of Lift-QMDP as the lower bound. This algorithm considers both kinematic constraints and uncertainty during planning.

6.3 Rel-POMDP Experiments

We begin by considering Rel-POMDP to isolate the effect of uncertainty from that of kinematic constraints. First, we confirm that our POMDP formulation faithfully encodes our goal. Next, we demonstrate information-gathering is necessary to achieve good performance. Finally, we verify that DESPOT—an online method—does not sacrifice the solution quality achieved by offline methods.

Figure 3b shows the value achieved by each policy in a 100 timestep simulation of the discretized Rel-POMDP problem. Figure 3c shows the probability



(a) Hand-Relative Discretization (b) Discrete V (c) Continuous Success Prob.

Fig. 3: Performance of Rel-QMDP (RM 🟧 —), Rel-SARSOP (RS 🟩 —), and Rel-DESPOT (RD 🟪 —) on Rel-POMDP. (a) Discretization of Rel-POMDP used during planning. (b) Value V_{rel} achieved by each policy after 100 timesteps on the discretized Rel-POMDP problem. Note that the y-axis is inverted; lower (less negative) is better. (c) Probability $p = \Pr(s_t \in X_{\text{goal}})$ that the movable object is in the goal region at each timestep on the continuous Rel-POMDP problem. Results are averaged over 500 trials and error bars denote a 95% confidence interval. Best viewed in color.

that the movable object is in X_{goal} at each timestep when simulated using the continuous model. As expected, the higher value achieved by Rel-SARSOP (🟩 —) and Rel-DESPOT (🟪 —) on the discretized problem translates to those algorithms achieving a higher success rate than Rel-QMDP (🟧 —) on the continuous problem. This result suggests that *discretizing the state space does not harm a policy's performance on the continuous problem.*

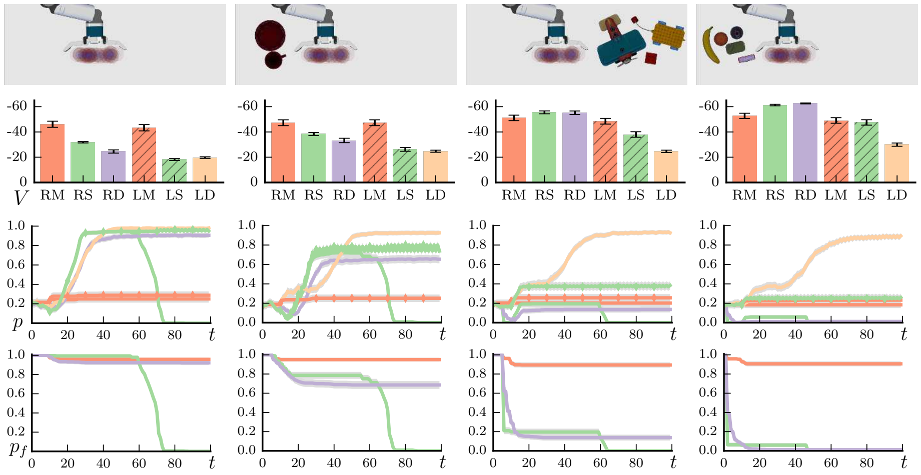
Rel-QMDP (🟧 —) performs poorly on this problem, achieving $< 30\%$ success probability, because QMDP does not take multi-step information-gathering actions [31]: the robot pushes straight without localizing the object.

Rel-SARSOP (🟩 —) and Rel-DESPOT (🟪 —) execute information-gathering actions by moving the hand laterally to drive the movable object into one of the fingertip contact sensors, then push the object into the goal region. These results replicate those in prior work [17, 25] by confirming that *information-gathering is necessary to perform well on this problem.* Our POMDP formulation provides a principled method of automatically constructing policies that gather information when necessary to complete the task.

Our intuition is that it is more difficult to solve Lat-POMDP than Rel-POMDP. Therefore, it is important that we verify that DESPOT solves Rel-POMDP before applying it to Lat-POMDP. Our results confirm *Rel-DESPOT (🟪 —) achieves comparable value and success probability to Rel-SARSOP (🟩 —).*

6.4 Lat-POMDP Experiments

We evaluate the proposed approach (Lat-DESPOT) on Lat-POMDP in four different environments: (a) an empty table, (b) obstacles on the right, (c) obstacles on the left, and (d) more complex obstacles on the right. Unlike in the



(a) No Obstacles (b) Right Obstacles (c) Left Obstacles (d) Complex Obs.

Fig. 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 each policy after 100 timesteps on the discretized problem (less negative is better). (Middle) Probability $p = \Pr(s_t \in X_{\text{goal}})$ that the movable object is in the goal region at each timestep on the continuous 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 and axis labels are shared across the plots in each row. Best viewed in color.

Rel-POMDP experiments, kinematic constraints are present in the form of reachability limits, self-collision, and collision between the arm and the table. Scenes (b), (c), and (d) are constructed out of objects selected from the YCB dataset [6].

Figure 4 shows results for each scene. Figure 4-Top shows the value V_{lat} achieved by each policy on the discretized Lat-POMDP. Figure 4-Middle shows the probability that the movable object is in X_{goal} at each timestep, treating instances that have terminated as zero probability. Figure 4-Bottom shows the proportion of Rel-QMDP, Rel-SARSOP, and Rel-DESPOT policies that are active at each timestep; i.e. have not yet terminated by taking an infeasible action.

Rel-QMDP (RM 🟧 —) and Lift-QMDP (LM ▨ -◆-) perform poorly across all environments, achieving $< 30\%$ success probability, because they do not take multi-step information-gathering actions. Figure 4 confirms this: both QMDP policies perform poorly on all four environments. This result demonstrates that *it is important to gather information even when kinematic constraints are present.*

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

Lift-SARSOP (LS ▨ ◆) performs near-optimally on environments (a) and (b) because it does not take infeasible actions and gathers information. However, it performs no better than Rel-QMDP on problem (d). This occurs because Lift-SARSOP myopically considers obstacles in a one-step lookahead and may oscillate when blocked. Small changes in the environment are sufficient to induce this behavior: the key difference between environments (b) and (d) is the introduction of a red box that creates a cul-de-sac in the lattice.

Our approach, Lat-DESPOT (🟧 —), avoids myopic behavior by considering action feasibility during planning. Lat-DESPOT performs no worse than Lift-SARSOP on environments (a) and (b) and outperforms it on environments (c) and (d). Unlike Rel-SARSOP, Lat-DESPOT identifies the cul-de-sac in (d) during planning and avoids becoming trapped in it. In summary, *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 is necessary, and fastest once the movable object is localized because the upper and lower bounds become tighter. The QMDP and SARSOP policies, which are computed offline, took an average of 1.6 μs and 218 μs to evaluate respectively.

We are optimistic about achieving real-time performance from Lat-DESPOT by optimizing our implementation of the algorithm in future work. Since DESPOT is an anytime algorithm, speeding up the search will both improve the quality of a solution given a fixed time budget and reduce the time required to find a solution of a desired quality.

6.5 Upper Bound Validation

Finally, we combine the data in Figure 3-Left and Figure 4-Top to empirically verify the bound we proved in Theorem 2. The value of Rel-SARSOP (🟩 —) and Rel-DESPOT (🟪 —) on Rel-POMDP (Figure 3-Left) are greater (i.e. less negative) than the value of all policies we evaluated on Lat-POMDP (Figure 4-Top). The data supports our theory: 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 Discussion

In this paper, we formulated the problem of planar contact manipulation under uncertainty as a POMDP in the joint space of robot configurations and poses of the movable object (Section 3). For computational efficiency, we simplify the problem by constructing a lattice in the robot's configuration space and prove that, under mild assumptions, the optimal policy of Lat-POMDP will never take an infeasible action (Section 4). We find a near-optimal policy for Lat-POMDP using DESPOT [43] guided by upper and lower bounds derived from Rel-POMDP (Section 5).

Our simulation results show that Lat-DESPOT outperforms five baseline algorithms on cluttered environments: it achieves a $> 90\%$ success rate on all environments, compared to the best baseline (Lift-SARSOP) that achieves only a $\sim 20\%$ success rate on difficult problems. This highlights the importance of reasoning about both object pose uncertainty and kinematic constraints during planning. However, Lat-DESPOT has several limitations that we plan to address in future work.

First, our approach assumes that the robot has perfect proprioception and operates in an environment with known obstacles. In practice, robots often have imperfect proprioception [4, 22] and uncertainty about the pose of *all* objects in the environment. We hope to relax both of these assumptions by replacing the deterministic transition model for robot configuration with a stochastic model that considers the probability of hitting an obstacle. This extension should not significantly affect computational complexity because DESPOT—as with most online solvers—does not scale directly with the size of the state space.

Second, we are excited to scale our approach up a larger repertoire of action templates (including non-planar motion), solving more complex tasks, and planning in environments that contain multiple movable objects. Solving these more complex problems will require more informative heuristics. We are optimistic that more sophisticated Rel-POMDP policies, e.g. computed by Monte Carlo Value Iteration [2], could be used to guide the search. Additionally, we are interested in using macro actions [30] consisting of the repeated execution of an action template to reduce the effective horizon of the search and methods that operate on a continuous action space to incrementally densify the lattice [40].

Third, our approach commits to a single inverse kinematics solution $q_{\text{lat}}(x_{\text{r}})$ for each lattice point. This prevents robots from using redundancy to avoid kinematic constraints. We plan to relax this assumption in future work by generating multiple inverse kinematic solutions for each lattice point and instantiating an action template for each. Our intuition is that many solutions share the same connectivity and, thus, may be treated identically during planning.

Finally, we plan to implement Lat-DESPOT on a real robotic manipulator and evaluate the performance of our approach on real-world manipulation tasks.

Acknowledgements

This work was supported by a NASA Space Technology Research Fellowship (award NNX13AL62H), the National Science Foundation (awards IIS-1218182 and IIS-1409003), the U.S. Office of Naval Research, and the Toyota Motor Corporation. We would like to thank Rachel Holladay, Shervin Javdani, Jennifer King, Stefanos Nikolaidis, and the members of the Personal Robotics Lab for their input. We would also like to thank Nan Ye for assistance with APPL.

Bibliography

[1] Dynamic Animation and Robotics Toolkit. http://dartsim.github.io (2013)

[2] Bai, H., Hsu, D., Lee, W., Ngo, V.: Monte Carlo value iteration for continuous-state POMDPs. In: WAFR (2011)

[3] Bohlin, R., Kavraki, L.: Path planning using lazy PRM. In: IEEE ICRA. pp. 521–528 (2000)

[4] Boots, B., Byravan, A., Fox, D.: Learning predictive models of a depth camera & manipulator from raw execution traces. In: IEEE ICRA (2014)

[5] Brokowski, M., Peshkin, M., Goldberg, K.: Curved fences for part alignment. In: IEEE ICRA (1993)

[6] Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., Dollar, A.: The YCB object and model set: Towards common benchmarks for manipulation research. In: ICAR (2015)

[7] Catto, E.: Box2D. http://box2d.org (2010)

[8] Cohen, B., Chitta, S., Likhachev, M.: Single-and dual-arm motion planning with heuristic search. IJRR (2013)

[9] Dogar, M., Hsiao, K., Ciocarlie, M., Srinivasa, S.: Physics-based grasp planning through clutter. In: R:SS (2012)

[10] Dogar, M., Srinivasa, S.: Push-grasping with dexterous hands: Mechanics and a method. In: IEEE/RSJ IROS (2010)

[11] Dogar, M., Srinivasa, S.: A planning framework for non-prehensile manipulation under clutter and uncertainty. AuRo 33(3), 217–236 (2012)

[12] Duff, D., Wyatt, J., Stolkin, R.: Motion estimation using physical simulation. In: IEEE ICRA (2010)

[13] Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: Approximate solutions for partially observable stochastic games with common payoffs. In: AAMAS (2004)

[14] Erdmann, M., Mason, M.: An exploration of sensorless manipulation. IEEE T-RA (1988)

[15] Hauser, K.: Lazy collision checking in asymptotically-optimal motion planning. In: IEEE ICRA (2015)

[16] Hebert, P., Howard, T., Hudson, N., Ma, J., Burdick, J.: The next best touch for model-based localization. In: IEEE ICRA (2013)

[17] Horowitz, M., Burdick, J.: Interactive non-prehensile manipulation for grasping via POMDPs. In: IEEE ICRA (2013)

[18] Hsiao, K.: Relatively robust grasping. Ph.D. thesis, MIT (2009)

[19] Hsiao, K., Lozano-Pérez, T., Kaelbling, L.: Robust belief-based execution of manipulation programs. In: WAFR (2008)

[20] Javdani, S., Bagnell, J., Srinivasa, S.: Shared autonomy via hindsight optimization. In: R:SS (2015)

[21] Javdani, S., Klingensmith, M., Bagnell, J., Pollard, N., Srinivasa, S.: Efficient touch based localization through submodularity. In: IEEE ICRA (2013)

[22] Klingensmith, M., Galluzzo, T., Dellin, C., Kazemi, M., Bagnell, J., Pollard, N.: Closed-loop servoing using real-time markerless arm tracking. In: IEEE ICRA Humanoids Workshop (2013)

[23] Koval, M., King, J., Pollard, N., Srinivasa, S.: Robust trajectory selection for rearrangement planning as a multi-armed bandit problem. In: IEEE/RSJ IROS (2015)

[24] Koval, M., Pollard, N., Srinivasa, S.: Pose estimation for planar contact manipulation with manifold particle filters. IJRR 34(7), 922–945 (2015)

[25] Koval, M., Pollard, N., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. IJRR (2015), in press

[26] Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: R:SS (2008)

[27] LaValle, S., Hutchinson, S.: An objective-based framework for motion planning under sensing and control uncertainties. IJRR (1998)

[28] Li, Q., Schürmann, C., Haschke, R., Ritter, H.: A control framework for tactile servoing. In: R:SS (2013)

[29] Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. IJRR 28(8), 933–945 (2009)

[30] Lim, Z., Hsu, D., Sun, L.: Monte Carlo value iteration with macro-actions. In: NIPS (2011)

[31] Littman, M., Cassandra, A., Kaelbling, L.: Learning policies for partially observable environments: Scaling up. ICML (1995)

[32] Lynch, K., Maekawa, H., Tanie, K.: Manipulation and active sensing by pushing using tactile feedback. In: IEEE/RSJ IROS (1992)

[33] Ng, A., Jordan, M.: PEGASUS: A policy search method for large MDPs and POMDPs. In: UAI (2000)

[34] Pastor, P., Righetti, L., Kalakrishnan, M., Schaal, S.: Online movement adaptation based on previous sensor experiences. In: IEEE/RSJ IROS (2011)

[35] Petrovskaya, A., Khatib, O.: Global localization of objects via touch. IEEE T-RO 27(3), 569–585 (2011)

[36] Pivtoraiko, M., Kelly, A.: Efficient constrained path planning via search in state lattices. In: i-SAIRAS (2005)

[37] Platt, R., Fagg, A., Grupen, R.: Nullspace grasp control: theory and experiments. IEEE T-RO 26(2), 282–295 (2010)

[38] Ross, S., Pineau, J., Paquet, S., Chaib-Draa, B.: Online planning algorithms for POMDPs. JAIR (2008)

[39] Salisbury, K., Townsend, W., Eberman, B., DiPietro, D.: Preliminary design of a whole-arm manipulation system (WAMS). In: IEEE ICRA (1988)

[40] Seiler, K., Kurniawati, H., Singh, S.: GPS-ABT: An online and approximate solver for POMDPs with continuous action space. In: IEEE ICRA (2015)

[41] Silver, D., Veness, J.: Monte-Carlo planning in large POMDPs. In: NIPS (2010)

[42] Smallwood, R., Sondik, E.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research 21(5), 1071–1088 (1973)

[43] Somani, A., Ye, N., Hsu, D., Lee, W.: DESPOT: Online POMDP planning with regularization. In: NIPS (2013)

[44] Srinivasa, S., Berenson, D., Cakmak, M., Collet, A., Dogar, M., Dragan, A., Knepper, R., Niemueller, T., Strabala, K., Vande Weghe, M.: HERB 2.0: Lessons learned from developing a mobile manipulator for the home. Proc. IEEE 100(8), 1–19 (2012)

[45] Stulp, F., Theodorou, E., Buchli, J., Schaal, S.: Learning to grasp under uncertainty. In: IEEE ICRA. pp. 5703–5708 (2011)

[46] Townsend, W.: The BarrettHand grasper–programmably flexible part handling and assembly. Industrial Robot: An International Journal 27(3), 181–188 (2000)

[47] Zhang, H., Chen, N.: Control of contact via tactile sensing. IEEE T-RA 16(5), 482–495 (2000)