Relatively Robust Grasping

Kaijen Hsiao and **Tomás Lozano-Pérez** and **Leslie Pack Kaelbling**
Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology, {kjhsiao, tlp, lpk}@csail.mit.edu

Abstract

In this paper, we present an approach for robustly grasping objects under positional uncertainty. We maintain a belief state (a probability distribution over world states), model the problem as a partially observable Markov decision process (POMDP), and select actions with a receding horizon using forward search through the belief space. Our actions are *world-relative trajectories*, or fixed trajectories expressed relative to the most-likely state of the world. We localize the object, ensure its reachability, and robustly grasp it at a goal position by using information-gathering, reorientation, and goal actions. We choose among candidate actions in a tractable way online by computing and storing the observation models needed for belief update offline. This framework is used to successfully grasp objects (including a powerdrill and a Brita pitcher) despite significant uncertainty, both in simulation and with an actual robot arm.

Introduction

We would like robots to be able to manipulate objects robustly and reliably, in unstructured environments. We concentrate on situations where there is an initial estimate of an object's state, made by a non-contact sensing modality such as vision or range scanning, that has some residual uncertainty. This uncertainty may be too great to guarantee that an open-loop grasping strategy will succeed. We develop strategies that use local sensing (such as force or tactile sensing, or hand-mounted cameras or range sensors) in combination with attempts to grasp the object to refine the state estimate and eventually achieve the desired grasp.

We model the problem of grasping 3D objects as a POMDP. The partially observed Markov decision process (POMDP) framework (Smallwood and Sondik 1973) allows optimal policies to be derived for domains with finite state, action, and observation spaces, but even in simple cases, finding the optimal policy for a POMDP can be computationally intractable. There are new approximation methods that can solve discrete POMDPs with several thousand states effectively (Smith and Simmons 2005), (Kurniawati, Hsu, and Lee 2008). However, the problem of grasping under uncertainty has continuous state, action, and observation spaces, making it particularly difficult to address in this framework.

Instead of solving for an optimal policy offline, we use a receding horizon policy that selects actions online by using forward search through the belief space (the space of probability distributions over the underlying state space). We use *world-relative trajectories* as our actions, which are entire trajectories expressed relative to the current most likely object location. We can calculate the observation models for these world-relative trajectories offline, and then select them online using POMDP forward search. In this paper, we will demonstrate how this framework allows us to intelligently select actions that gather information about the location of the object, reorient it when unreachable, and achieve desired goal-seeking grasps, both in simulation and on a real robot.

Related Work

There were some early attempts to address grasping under uncertainty in robot manipulation within a non-probabilistic uncertainty framework (e.g., (Lozano-Pérez, Mason, and Taylor 1984), (Latombe 1991)) as well as a probabilistic framework (LaValle and Hutchinson 1994), (Brost and Christiansen 1996). More recently there have been some direct applications of the POMDP framework to robot manipulation tasks (Hsiao, Kaelbling, and Lozano-Perez 2007), made tractable by very aggressive aggregation of world states.

An intermediate position is to assume that there is uncertainty in the outcomes of actions, but that the uncertainty will immediately be resolved through observations. (Alterovitz, Simeon, and Goldberg 2007) construct and solve such an MDP model for guiding non-holonomic needles. There has been a great deal of recent work on generalizations of the motion planning problem that take positional uncertainty and the potential for reducing it via observations into account and plan trajectories that will maximize the probability of success or related other objectives (Prentice and Roy 2007), (Gonzalez and Stentz 2005), (Censi et al. 2008), (Melchior and Simmons 2007). (Burns and Brock 2007) have a different model, in which the system selectively makes observations to reduce uncertainty during planning, but the resulting plan is executed open-loop.

Belief-state estimation is a central part of most probabilistic approaches to control and has become standard in mobile-robot control (Thrun, Burgard, and Fox 2005). Although it is much less common in the manipulation literature, several examples exist (Petrovskaya and Ng 2007), (Gadeyne, Lefebvre, and Bruyninckx 2005), (Hsiao, Kaelbling, and Lozano-Perez 2007).

This paper builds on the work of (Hsiao, Kaelbling, and Lozano-Perez 2008), in which the problem of grasping objects in 3D is also modeled as a POMDP using world-relative trajectories as actions, but information-gathering actions are chosen only based on their short-term ability to reduce entropy in the belief state.

POMDP forward search

When using POMDPs, actions are chosen based on the current belief state, which is a probability distribution over the underlying state space. We need to be able to select an action for any belief state we might find ourselves in. One typical way to do this is to solve for an optimal policy offline that tries to map all possible belief states to actions. Approximate solvers generally search intelligently through the region of belief space that is likely to be encountered from a start belief state, and the resulting policy allows one to both interpolate and extrapolate to all possible belief states. However, using such a solver requires us to enumerate small, discrete sets of all the possible actions and observations beforehand, and the enumerated actions and observations cannot change depending on the current belief state.

Instead, we use online POMDP forward search to select actions. This allows us to use actions parameterized by the current belief state, such as "move to this position relative to the most likely state," which are not possible when using the type of offline policy solver described above. Parameterizing actions in this way allows us to access a continuous range of potential robot motions while only having to consider, at each time step, a small, discrete set of motions that are likely to be useful given what we currently know about the object location. We can also sample and consider only observations that are likely to result from those motions, from an observation space that is also high-dimensional and continuous.

To track the current belief state at each time step during execution, we use a state estimator (which is simply a discrete Bayesian filter), which estimates a new belief state given the previous belief state, the action taken, and the observation received. Based on the updated belief state, we would then like to pick an action that has the highest expected future reward.

In order to estimate the expected future rewards for each action, we construct a forward search tree like the one shown in Figure 1. At the top of the tree is our current belief state, b. From there, we branch on all of the possible actions. For each action, we expect that we might see a number of possible observation outcomes, so we enumerate and branch on possible observations. Each observation branch is associated with a new belief state, which is the result of doing a belief update on the parent belief using that branch's action and observation. Each new resulting belief state is akin to the root belief state; we can again branch on all possible actions, to consider what would happen if we took a second action, and branch further on the observations we might obtain after those second actions, updating the belief state again.

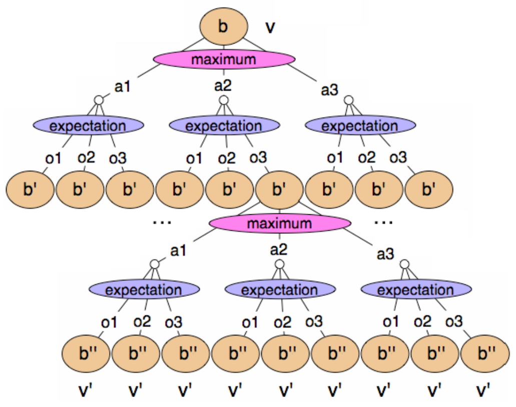


Figure 1: Forward search on the POMDP tree.

Repeated branching on actions and then observations continues until we reach a desired horizon H. In Figure 1, H is only 2, with all but one of the second-level node expansions omitted due to space constraints. When we reach the desired horizon, the leaf nodes of the tree are evaluated using a static evaluation function, which assigns a value to the resulting belief state. The static evaluation function can be based on, for instance, the entropy of the belief state, or the probability of succeeding.

We can then use the tree to calculate the value of taking each action at the root node. For each action node at level H, we take an expectation over the values assigned to its observation nodes and then subtract a fixed cost for taking an action. Once the action nodes at level H have values, we can select the action with the maximum value. We continue taking maximums over actions and expectations over observations all the way up the tree to the root, which can now choose the next action to execute.

Receding horizon planning

We could use POMDP forward search to explore large portions of the reachable space, and then use our extensive search tree to determine a policy that hopes to cover all the belief states that we might reach. However, most of those belief states will never arise, which means that we would be doing much more computation and using much more storage space than necessary (Ross et al. 2008). We instead plan using a receding horizon, which means that at each time step, we make an optimal plan that takes into account only the next H steps. We then take a single action, make an observation, and plan again using what we now know about the world. This approach requires a reasonably good static evaluation function for belief states at the leaves, so that even with limited lookahead we can choose actions intelligently. At the same time, it allows the POMDP search tree to be less accurate and exhaustive in its search, since we only have to pick a reasonable next action, knowing we will have a chance to plan again after the next observation.

Even when using POMDP forward search with a receding horizon, there are still a number of challenges involved in making action selection tractable. The belief state must be compact to represent and reasonbly efficient to update; the action and observation branching factors must both be quite low; the horizon must be short; and we need to be able to use the observation and transition models for belief update quickly. We will now discuss how we fill in the choices of state space, actions, and observations, and how we compute the observation and transition models.

The state space and belief space

To make the state space manageable, we partition it into observable and uncertain components. We assume that the robot's position is approximately observable based on proprioception, and thus do not need to include it in the uncertain part of our belief state. A POMDP with fully and partially observable components separated in this fashion is called a MOMDP, which stands for mixed observability MDP (Ong et al. 2005).

Let Φ be the set of possible robot poses, in absolute joint coordinates, and let $\mathcal{W}$ be the space of possible configurations of the world. In the simplest case, $w \in \mathcal{W}$ will be the pose of a single object of known shape, supported by a table, and specified by (x, y, θ) coordinates. A *belief state* of the system is a probability distribution over $\mathcal{W}$ representing the system's state of information about the world it is interacting with, together with a single element ϕ of Φ, representing the known robot pose. We represent belief states using a set of sampled points in $\mathcal{W}$ (spaced regularly on a grid) together with weights that are proportional to the probability density at those grid points.

Having described the state space of the system, we need to be able to articulate a goal condition. Most straightforwardly, we might imagine the goal to be some predicate $G(\phi, w)$, specifying a desired relation between the robot and the objects in the world (such as a particular range of grasp locations, or desired contacts between parts of the hand and parts of the object, or a combination of the two). The range of grasp locations can even include non-contiguous regions, if there are multiple locations on the object we might wish to grasp.

Having a goal condition on states of the world is not directly useful, however: the system will be unable to determine, with certainty, whether it actually holds in the world. So, we must formulate goal conditions on belief states, instead. We can construct a goal condition, $G_\delta(\phi, b)$ on belief states by requiring that the system believe, with confidence δ, that the goal condition holds; that is, that

$$\sum_w b(w) I[G(\phi, w)] > 1 - \delta \ ,$$

where b is a belief state, $b(w)$ is the probability that b assigns to the discrete world state w, and I is an indicator function with value 1 if its argument is true and 0 otherwise. For compactness, in future, we will write statements such as this as $P_b(G(\phi, w)) > 1 - \delta$, where P_b means probability, using belief state b as the measure on w.

Actions: World-relative trajectories

Our goal is to select among possible robot motions online, based on sensory information incorporated into the belief state. It is typical, in lower-dimensional control problems such as mobile-robot navigation, to use a uniform discretization of the primitive action space. Such a fine-grained discretization of the primitive action space for a robot with many joints presents two problems: first, there is a large branching factor in the choice of actions; second, the horizon (number of "steps" that must be made before the goal is reached) is quite long, requiring significant lookahead in planning to select an appropriate action.

Instead, our strategy will be to generate, offline, a relatively small set of *world-relative trajectories*. A *world-relative trajectory* (WRT) is a function that maps a world configuration $w \in \mathcal{W}$ into a sequence of Cartesian poses for the robot's end effector. In the simple case in which w is the pose of an object, then a world-relative trajectory can just be a sequence of end-effector poses in the object's frame. Given a WRT τ and a world configuration w, the sequence of hand poses $\tau(w)$ can be converted via inverse kinematics (including redundancy resolution) into a sequence of via-points for the arm in joint-angle space. So, if we knew w exactly and had a valid WRT for it, we could move the robot through the hand poses in $\tau(w)$ and reach the desired terminal configuration of the arm with respect to the object. The first point on every trajectory will be the same "home" pose, in a fixed robot-relative frame, and the robot will begin each trajectory execution by moving back to the home pose ϕ_h.

In general, we won't know w exactly, but we will have to choose a single w to use in calculating $\tau(w)$. Let $w^*(b)$ be the world state w for which $b(w)$ is maximized; it is the most likely state. We can execute $\tau(w^*(b))$, and have the highest probability of reaching the desired terminal configuration according to our current belief state. We command the robot to follow the trajectory by executing *guarded move* commands to each waypoint in the sequence, terminating early if a contact is sensed. An early contact (or reaching the end of the trajectory with no contact) results in an observation that can be used to update the belief state. In addition to the collision point, we obtain further contact observations by carefully closing the fingers when a collision is sensed.

One way to think of WRTs is as temporally extended "macro actions." This choice of actions allows our forward search to have a relatively small branching factor, and results in effective goal-directed action even with limited lookahead.

The observation and transition models

In order to do forward search in the belief space, we need to be able to calculate the effect on the belief state of executing each τ and receiving the resulting observation. Much of the relevant observation model can be pre-computed for each τ and stored. During the online execution phase, we will use the stored observation information, together with the continually updated belief state, to select and execute appropriate trajectories.

Each τ is characterized by *feasibility* and *contact obser-*

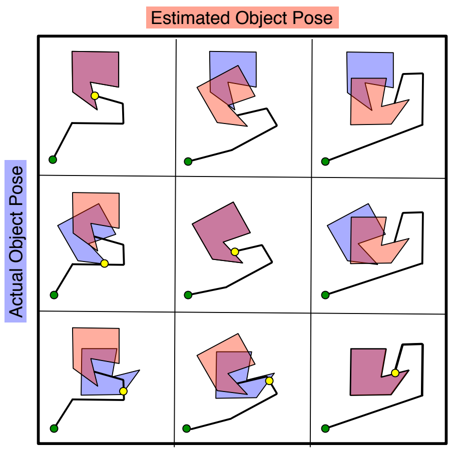


Figure 2: The $\Omega_\tau(w, e)$ matrix for a WRT τ.

vation functions. Each estimated pose e induces a different actual trajectory $\tau(e)$ in robot coordinate space. The feasibility function $F_\tau(e)$ is true if trajectory $\tau(e)$ is kinematically feasible for the robot, and false, otherwise. The observation function is indexed by an actual world configuration w and an estimated world configuration e, specifying what would happen if $\tau(e)$ were executed in world w; that is, if the robot acted as if the world were in configuration e, when in fact it was in configuration w. The observation function $\Omega_\tau(w, e) = \langle \phi, c \rangle$ specifies what contacts, if any, the robot will sense during execution of $\tau(e)$ in w, where ϕ is the Cartesian position of the robot hand relative to the object when the contact occurs (or reaches the end of the trajectory), and c is the local sensor readings that can be expected when a sensed contact occurs (none, if no sensor readings would be seen). The swept path of the robot is all of $\tau(e)$ in the event that the robot reaches the end of the trajectory without contact, and $\tau(e) - \phi$ (the trajectory up to the point of contact) in the event that the robot makes contact.

Figure 2 shows the $\Omega_\tau(w, e)$ function for a WRT τ and a space of 3 world configurations, and how it is determined. Each row corresponds to a different true pose (x, y, θ) of the object in the world, which is drawn in blue. Each column corresponds to a different estimated pose of the object, which is drawn in red. On the diagonals, the true and estimated poses are the same, so the figures lie on top of one another. The estimated pose e determines the trajectory $\tau(e)$ that the robot will follow (in this case, our robot is a point robot in x, y). The trajectories are shown in black. Each one starts from the same home pose, shown in green, and then moves to a sequence of waypoints that are defined relative to the estimated pose of the object. Yellow circles indicate situations in which the robot will make contact with the object. It happens on each of the diagonal elements, because the nominal trajectory makes contact with the object. In the elements in the bottom-left part of the figure, there is a contact between the robot and the actual object during the execution of the trajectory, before it would have been expected if the estimated pose had been the true one. In the elements in the

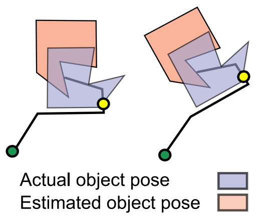


Figure 3: Predicted observations depend only on the relative transformation between the actual and estimated object pose.

upper right part of the figure, the trajectory terminates with no contact. In all of the off-diagonal cases, the observation gives information about the object's true location, which is used to update the estimated pose.

Computing the observation matrix

Computing an entry of the observation matrix requires simulating a trajectory forward from a starting robot pose, and calculating if and when it contacts objects in the world, and, if it does, what the nominal sensory readings will be. This simulation includes closing the fingers when a collision is detected, to gather additional contact information. This is a geometric computation that can be done entirely offline, relieving the online system of performing simulations.

This computation may seem prohibitive, since for a x, y, θ grid of just 31x31x25 = 24,025 points, having to simulate all combinations of w and e in pairs would require $24,025^2 = 207,792,225$ simulations. However, the crucial insight here is that if trajectory $\tau(e)$ is kinematically feasible and there are no other objects nearby, then the observation depends only on the relative transformation between w and e, as shown in Figure 3. For two sets of w and e with the same relative transformation, as with the examples in the figure, w and $\tau(e)$ may differ, but $\Omega_\tau(w, e)$, which is expressed relative to w, is the same. Thus, when calculating the full $\Omega_\tau(w, e)$ matrix for a WRT, we can pick a single e (for instance, the initial $w^*(b)$), compute $\tau(e)$, and simulate just that sequence of robot poses while varying w. The number of simulations required to compute $\Omega_\tau(w, e)$ is therefore merely the number of points in the belief grid that have nontrivial probability, and takes just a few seconds. Once the simulations are completed, the results can be stored for fast re-use when selecting actions online.

Based on these canonical observations, as well as the volume of free space swept out by the arm during the execution of the trajectory, we construct an observation model $P(o|w, \tau(e))$, which is used for belief state update. For a detailed discussion of the model, see (Hsiao 2009).

Reducing the observation branching factor

The space of possible observations given a single known world state is continuous due to noise, but the distribution has one or two fairly peaked modes, which we can treat as discrete clusters of observations. When the world state is not known, then we might have any of an enormous number of these observation clusters, arising from all possible underlying world states. We will aggressively cluster this space of observations, represent each cluster with a canonical observation, and branch on these canonical observations in the forward search.

The observations are clustered and pruned first in an offline phase and again in an online phase that depends on the current belief state. In the offline phase we group observations with similar contacts independent of the absolute pose in which the contacts take place, then prune clusters whose aggregate probability is less than .01. In our experiments, this results in a set of 68 observations, which is reduced from the initial 24,025 modes, but still too large to branch on in the forward search. So, online, given a belief node to expand, with trajectory $\tau(e)$, we group observations for which the resulting belief states after state estimation, $SE(b, \tau(e), o)$, have similar entropy and covariance. Finally, we take the set of highest-probability clusters that account for 50% of the total probability mass.

Computing the transition model

The transition model captures the dynamics of how the object moves in response to contact with the robot during execution of a trajectory, $\tau(e)$. If no contact is made with the object, it is assumed that the object does not move. If contact is made with the object, the transition model is a mixture of Gaussians around two outcomes: 1) the object remains in place, and 2) the object gets bumped through the contacts that were made when the trajectory terminated. Details are in (Hsiao 2009).

RRG-Goal: Robust WRT execution

In many situations, merely executing a single goal-seeking WRT over and over again until our belief state goal condition is met is sufficient to robustly grasp an object. Consider the execution of a single WRT with the goal of grasping the object and let the system be in initial belief state b. We first compute the most likely state, $w^*(b)$, and generate $\tau(w^*(b))$, a sequence of waypoints in robot coordinates. We then command the robot to move to each waypoint in turn using guarded moves, collect the resulting observation (either early contact or reaching the end of the WRT without touching anything), and use that observation to update the belief state. If the goal condition on the belief state is not satisfied, we retrace the previous trajectory back to the home pose (to avoid any further contacts, which we will not be able to predict effectively), relativize τ to the new most likely world state, and re-execute. Repeated execution of the goal-seeking WRT relative to the current most likely state is continued until the belief state goal condition is met, at which point the robot declares success and attempts to pick up the object. We refer to this simple algorithm for robust execution of a single, goal-seeking WRT (with the addition of allowing an optional reorient action to be used if the goal-seeking WRT is infeasible, which will be described shortly) as the *RRG-Goal* algorithm, short for Relatively Robust Grasping with Goal-seeking WRT.

Figure 4 shows the operation of the system while grasping a rectangular box using RRT-Goal. The robot attempts

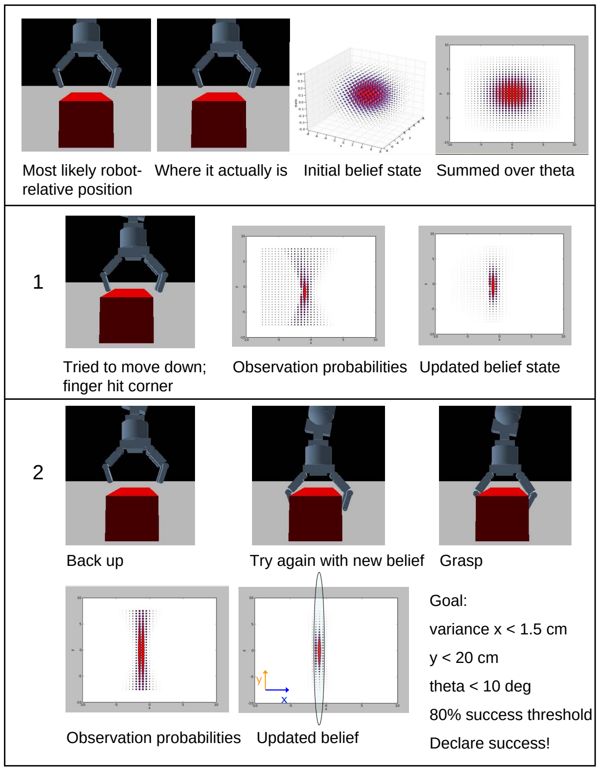


Figure 4: Execution of WRT and (x, y, θ) belief state update.

to execute a grasping trajectory, relative to the most likely element of that belief state. The first image in the top row shows where the robot thinks the most likely state of the box is relative to its hand. The second image shows the location of the robot at the first waypoint in that trajectory: we can see that the object is actually not in its most likely position (if it were, the robot's hand would be centered above it). The third image in the first row shows the initial belief state, with probabilities depicted via the radius of the balls shown at grid points on the (x, y, θ) state space of the box. Subsequent belief state images show the belief state summed over θ for easier visualization, as in the fourth image in the first row. In action 1 (row 2), the hand executes a guarded move toward the next waypoint in the trajectory, and is terminated by a fingertip contact on the corner of the box, as shown in the first figure in the second row. The middle figure in the second row shows the probability of observing that fingertip contact in each world configuration. Combining this information with the initial belief state, we obtain the updated belief state shown in the third figure in the second row. It is clear that the information obtained by the finger contact has considerably refined our estimate of the object's position. The third and fourth rows of figures show a similar process. The same WRT is executed a second time, now with respect to the most likely state in the updated belief state. This time, the hand is able to move all the way down, and the fingers close on the box, with the resulting belief state shown in the final figure. Now, given a goal condition such as having the box centered between the fingers within 1.5 cm and oriented within 10 degrees of being straight, but not being concerned where along the box the fingers are grasping, we can evaluate the probability that it holds in the current belief state. In this case, it holds with probability $\geq .8$, so if δ were .2, we would terminate.

Additional WRT actions

The most basic strategy of executing a single goal-seeking WRT can be effective in many situations, as demonstrated in the experimental results. However, there are many situations in which it may fail.

Goal-seeking trajectories

In our experiments, we use only a single, hand-generated goal-seeking WRT for each object. However, for some tasks, we may need more than one goal-seeking WRT to effectively achieve the goal for all possible object locations; there is no reason we need to limit ourselves to a single goal-seeking WRT. Kinematic constraints arising from the boundaries of the workspace may render execution of $\tau(w)$ simply infeasible for some values of w. It may also be that, even when it is executed in the appropriate world state ($\tau(w)$ is executed in w) a collision will result, due to obstacles. In addition, if our goal condition contains noncontiguous regions corresponding to different possible grasps of an object, we would need more than one WRT to be able to achieve the different goal grasps.

Explicit information gathering

The execution process described in the previous section will succeed if the goal-seeking trajectories result in local sensory observations that provide enough information to update the belief state so that it is eventually concentrated enough to meet the goal criterion. However, this will not necessarily happen. Consider the final belief state shown in Figure 4. It is clear that the object is well localized in the x and θ dimensions, but there is still considerable uncertainty in the y dimension (the grasp that the robot executed knows only that the fingers are on the box in the y dimension, but not where). If the goal had required that the box be grasped very near the center in the y dimension, for example, then this result would not have satisfied the goal condition, we would have no real way to improve the situation through further execution of our goal-seeking WRT, and the control loop would run forever. For this reason, we will sometimes need to execute WRTs that are designed explicitly to reduce uncertainty in the belief state, not to achieve the ultimate desired world configuration.

Reorienting and other actions that move the object

We can also execute trajectories that try to change the actual state of the world, rather than merely trying to reduce uncertainty in the belief state. If all of our goal-seeking trajectories are kinematically infeasible in $w^*(b)$, for instance, we may wish to reorient the object so that at least one of them becomes feasible. To do so, we can add a WRT that attempts to grasp the object (using a grasp that does not necessarily satisfy the goal conditions) and then rotates the object after successfully running to completion.

In our experiments, we have reorientation grasps for objects for which the goal-seeking trajectory can be kinematically infeasible at high orientation deviations; these reorientation grasps are all grasps from above that attempt to rotate the object about its center of mass. The observation model is the same for reorientation grasps as for other WRTs; however, the transition model additionally models the object rotation, taking into account the possibility of the rotation failing or leaving the object at an intermediate rotation.

Generating WRTs

Most of the WRTs used in this paper were generated through demonstration (by moving the robot and recording object-relative waypoints), although we have experimented with generating WRTs automatically, and a few of the WRTs that we use were generated in this fashion. WRTs can be constructed automatically simply by finding desired places on the object to grasp, finding collision-free trajectories to those hand positions, and then expressing those trajectories in object-relative, Cartesian coordinates. In our implementation, we constructed information-gain trajectories by finding hand positions that place the fingers on nearly parallel pairs of object surfaces. We then found collision-free trajectories to those hand positions, for the nominal object position, using the default planners in the OpenRAVE motion planning system (Diankov and Kuffner 2008). We also added an information-gathering WRT for most of the objects that just sweeps horizontally across the entire workspace looking for the object, which is particularly useful for roughly locating the object under high uncertainty.

Intuitively, for each object that we wish to grasp, we need to provide a set of information-gathering and goal-seeking WRTs that constrain the possible object states in such a way that the constrained object position is not easily confusable with any object position outside of the goal condition, within the start uncertainty. Furthermore, if we expect the workspace to be cluttered, and have geometric models of the obstacles (such as could be provided by a laser rangefinder), we can provide additional WRTs that would allow us to collect sufficient information even when some of our WRTs are infeasible due to possible collisions.

RRG-Info: Using POMDP forward search

Given a set of available WRTs, including information-gathering, goal-seeking, and reorientation WRTs, we would like to select a WRT that will allow us to reach our belief state goal condition quickly. To do so, we use POMDP forward search, as described above.

In our forward search, the static evaluation function for a belief state is simply the probability of succeeding, if we were to execute one of our goal-seeking WRTs in that state, and terminate. More specifically,

$$v_b(b) = \max_{\tau_g}[P_b(G(\Omega(\tau_g(w^*(b)), w)))]$$

which is, according to our belief state b, the maximum probability that G will be true in the state resulting from exe-

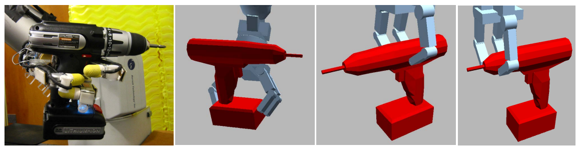


Figure 5: Goal-seeking and info-gathering grasps for the powerdrill.

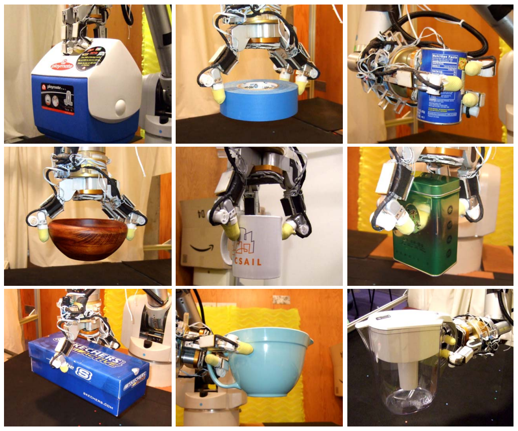


Figure 6: Goal-seeking grasps for all objects except the powerdrill. Top row: cooler, taperoll, can. Middle row: wooden bowl, cup, teabox. Bottom row: box, giant teacup, Brita pitcher.

cuting any of our goal-seeking actions using the most likely state, $\tau_g(w^*(b))$.

We refer to the use of POMDP forward search to select among information-gathering, goal-seeking, and reorientation WRTs as the *RRG-Info* algorithm, short for Relatively Robust Grasping with Information-Gathering.

Experiments

We conducted experiments using this framework with a 7-DOF Barrett Arm and Hand, both in simulation (using Open Dynamics Engine to simulate the physics of the world) and on an actual robot. The hand is outfitted with ATI Nano17 6-axis force/torque sensors at the fingertips, and simple binary contact pads covering the inside (and some outside) surfaces of the fingers and the palm.

Our experiments used 10 different objects, shown with their goal-seeking grasps in Figures 5 and 6. Goal regions and required contact locations for each object were hand-chosen to guarantee that being within the goal region ensures a stable grasp of the object. These regions are much larger for some objects than for others (for instance, the goal region for the can is large, since the hand only has to envelop it), and the goal regions for certain objects (the cup, wooden bowl, can, and taperoll) ignore the orientation of the object, looking only at the object position relative to the hand.

In all experiments, the maximum number of actions allowed was 10; after the 9th action, if the planner had not

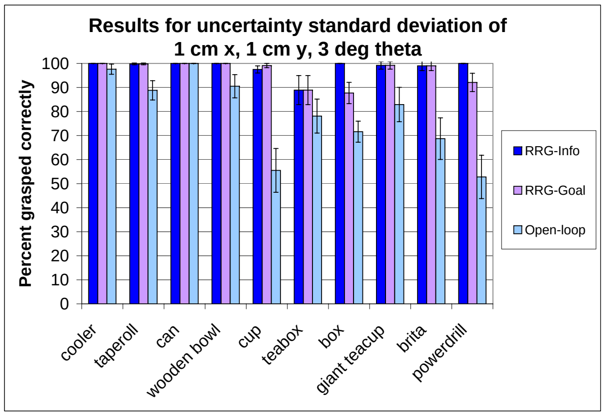


Figure 7: Sim results for all objects at low uncertainty.

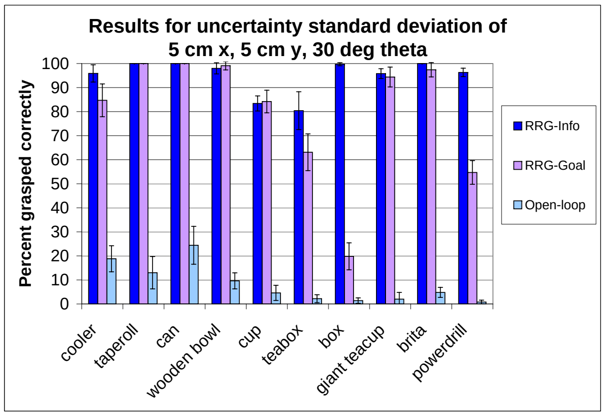


Figure 8: Sim results for all objects at high uncertainty.

yet declared success and terminated, the goal-seeking grasp was executed open-loop before evaluating whether the grasp had succeeded. All simulation experiments were carried out with at least 100 trials each. In our experiments, we compare three different strategies: 1) Open-loop: executing the goal-seeking WRT once, open-loop, 2) RRG-Goal, and 3) RRG-Info with a horizon of 2 (except where otherwise specified).

Figure 7 shows the results for experiments carried out in simulation with initial (Gaussian) uncertainty standard deviations of 1 cm in x, 1 cm in y, and 3 degrees in θ. The planner was asked to succeed at least 90% of the time ($\delta = .1$), and thus terminates when the probability of success rises above 90%. The graph shows the percentage of grasps that were executed successfully for each object placed at random positions drawn from the same distribution as the initial uncertainty, for the three algorithms. Even at these low uncertainty levels, executing the goal-seeking grasp open-loop fails with fairly high probability for many of the objects. Using just RRG-Goal allows us to succeed nearly all of the time, and using RRG-Info brings our success rate above 97% for all objects except the teabox, for which the planner only selects the goal-seeking WRT, because it recognizes that it will succeed with a probability of approximately 90% (it succeeds 89% of the time) just by selecting the one action. Raising the target success rate to 95% causes it to select other actions, raising the success rate to 98.4%.

Figure 8 shows the results for grasping all 10 objects in simulation with higher levels of initial uncertainty (standard deviations of 5 cm in x, 5 cm in y, and 30 degrees in θ). Note that at this level of uncertainty, it is possible for the object to be 15 cm and 45 degrees away from the initial estimated position; the object is somewhere on the table in front of the robot, but the robot has only a rough idea of where, and so it is essentially groping around blindly. With this much uncertainty, executing the goal-seeking grasp open-loop seldom succeeds. Using just RRG-Goal is sufficient for many of the objects, with the notable exception of the box (this is the example we gave when motivating information-gathering grasps) and the powerdrill, for which the goal-seeking grasp is grasping a nearly-cylindrical handle that gives it little information about the orientation of the drill. Using RRG-Info brings our success rate above 95% for all objects except the cup and teabox. The small sizes of both the cup and the teabox cause the fairly coarse grid sampling (the grid spacing that we use is 1 cm and .1 radians for this level of uncertainty) to be too poor an approximation for the actual continuous observation probability distribution. At a resolution this coarse, the majority of the probability mass sometimes lies between the grid points, and so the grid point with the highest sampled probability may not actually be the one closest to the actual most likely state. Thus, localizing the object within a 1 cm goal region is not always possible. However, for both objects, the planner is always able to localize the object to within a small area, and moreover, knows that it has done so. If we used a variable-resolution grid that switched to the same grid used at the low uncertainty levels when the planner became sure that the belief state is contained within the smaller grid, we would most likely obtain results similar to those in the low-uncertainty graph.

Figure 9 shows parametric results for just the powerdrill in simulation at the 5 cm/30 degree level of uncertainty, where δ was varied to show the trade-off between the number of actions executed and the probability of success. The four strategies used here are RRG-Goal and RRG-Info with a horizon of 3, 2, and 1. Each point on the graph represents the average number of actions taken before termination and the % successful for more than 100 simulated runs. Just executing the goal-seeking grasp repeatedly does not work well at all for this object, whereas just searching with a horizon of 1 works reasonably well. Increasing the horizon to 2 causes the planner to choose actions that result in a lower probability of success after just 2 actions, but that pay off in terms of a higher probability of success for fewer actions later on. This is largely due to the fact that, although infograsp 1 provides information about all three dimensions at once and infograsp 2 only provides information about two dimensions, infograsp 2 for the powerdrill (shown in Figure 5) acts as a 'funnel' for infograsp 1, enabing it to succeed more often. Increasing the horizon to 3 adds no additional benefit.

Although even searching with a horizon of 1 manages to

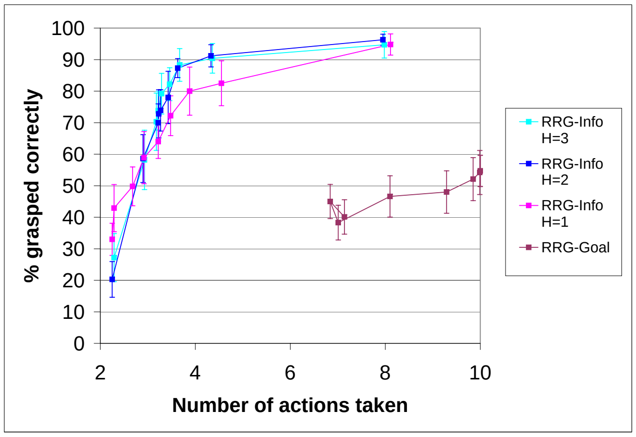


Figure 9: Parametric comparisions of various algorithms with the powerdrill in simulation.

succeed with essentially the same probability as using a horizon of 2 or 3 after 10 actions, searching 2 or 3 deep reduces the number of actions needed to succeed more than 90% of the time from an average of 7 actions to an average of 4 actions, which is a dramatic speedup. However, informally, we have observed that using a horizon of 1 is as good as using a horizon of 2 for most objects. Because even a horizon of 1 is usually sufficient to pick reasonable actions, if we have a large number of WRTs to select from (as we might need to have in the presence of a great deal of clutter), we can still select reasonable actions quickly. For comparison, selecting the first action from among the 5 available powerdrill actions takes us 3 seconds for a horizon of 1 (using a non-optimized Python implementation); using a horizon of 2 takes 10 times longer, and using a horizon of 3 takes 60 times longer.

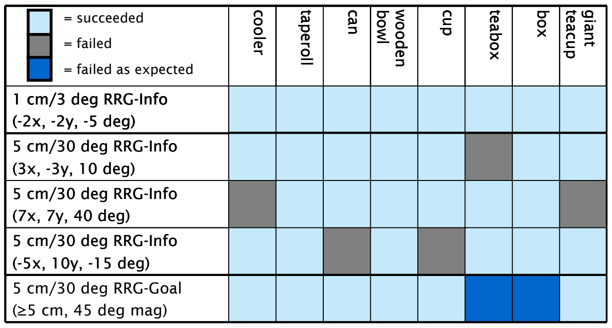
On the real robot, we ran 10 experiments each for both the Brita pitcher and the powerdrill at initial uncertainty levels of 5 cm in x, 5 cm in y, and 30 degrees in θ, again with random positions drawn from the same distribution, using RRG-Info. Both objects were grasped stably and lifted successfully 10 out of 10 times, with the trigger being pressed successfully on the powerdrill and the Brita pitcher being grasped properly by the handle. For the other objects, we ran five experiments each: 1 at uncertainty levels of 1 cm/3 deg with RRG-Info, and four at uncertainty levels of 5 cm/30 deg (three with RRG-Info, and one with RRG-Goal). The object placements and results are shown in the chart in Figure 10.

Most of the grasps succeeded. Two grasps (the box and teabox with RRG-Goal) failed as expected, due to the goal-seeking grasp's inability to collect information in a relevant goal dimension. The cooler grasp that failed was due to executing an information-gathering grasp that swept horizontally across the workspace but hit the corner of the cooler on a part of the hand with no sensors, thus shoving the object out of the workspace. The can was knocked over by a jerky hand movement. The cup failure was due to the same failure mode discussed in the simulation results; the teabox in real life adds the additional complication of being too light to sense without pushing the object significantly. The giant teacup grasp failed because the mesh (especially at the handle) is very thin, and our approximation of the robot's swept path can miss the fact that the fingers go through the handle in its most likely location. This failure mode is also seen in simulation. Videos of the real robot experiments can be seen at *http://people.csail.mit.edu/kjhsiao/wrtpomdps.*

Several of the failures were due to limitations in the current implementation, such as using a fixed-resolution as opposed to a variable-resolution grid, having jerky robot control, using an imprecise swept path approximation, and not computing and taking into account the probabilities of WRTs failing due to contacts with sensorless parts of the hand. Nonetheless, there are several general principles that we can observe from our experiments. The first is that a small search horizon is effective in our framework; a horizon of 1 is usually sufficient, and a horizon greater than 2 is generally not useful, which means that planning does not have to be very expensive. The second is that we are still able to choose effective actions despite the aggressive observation clustering that we do to limit the observation branching factor. The third is that one of the limitations of our framework is the quality of the transition model. Our system currently works well for objects that either do not move much when bumped into during a guarded move, or else that are large enough, or have sufficiently large goal regions, that information-gathering grasps (often on surfaces far away from the center) can provide enough information to overcome the fact that the object moves significantly every time we touch it. If we had a more predictive transition model that could more accurately estimate how objects move when we bump into them, we could further improve our results. A more accurate transition model could even enable us to add actions that purposely push objects in order to gain information, by, for instance, shoving them against walls or other immovable objects. Finally, some objects are too light or fragile to bump into even with the most sensitive touch sensors, and this method is unlikely to work well for those objects; in many of those cases, it may be better to use "pre-touch" sensors, such as IR or electric field sensors, to localize objects without having to touch them (Mayton et al. 2009), (Hsiao et al. 2009).

Conclusion

We have shown that our framework of using POMDP forward search with world-relative trajectories increases the robustness of grasping objects with positional uncertainty by a great deal. The fact that repeatedly executing the goal-seeking grasp does so well on its own for most objects means that much of the robustness comes just from tracking the belief state, executing actions relative to the most likely state, and reasoning about whether the grasp is likely to be within the goal region. However, information gathering and reorientation grasps are necessary to add robustness in some cases, and in those cases, using POMDP forward search is a good way to select actions intelligently.



▢ = succeeded ▨ = failed 🟦 = failed as expected	cooler	taperoll	can	wooden bowl	cup	teabox	box	giant teacup
1 cm/3 deg RRG-Info (-2x, -2y, -5 deg)	▢	▢	▢	▢	▢	▢	▢	▢
5 cm/30 deg RRG-Info (3x, -3y, 10 deg)	▢	▢	▢	▢	▢	▨	▢	▢
5 cm/30 deg RRG-Info (7x, 7y, 40 deg)	▨	▢	▢	▢	▢	▢	▢	▨
5 cm/30 deg RRG-Info (-5x, 10y, -15 deg)	▢	▢	▨	▢	▨	▢	▢	▢
5 cm/30 deg RRG-Goal (≥5 cm, 45 deg mag)	▢	▢	▢	▢	▢	🟦	🟦	▢

Figure 10: Results for grasping the other eight objects with the actual robot. Object placements as deviations from the expected position are shown in parentheses.

Acknowledgement: This research was supported in part by the National Science Foundation under Grant No. 0712012 and in part by DARPA IPTO Contract FA8750-05-2-0249, "Effective Bayesian Transfer Learning".

References

Alterovitz, R.; Simeon, T.; and Goldberg, K. 2007. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. *RSS*.

Brost, R. C., and Christiansen, A. D. 1996. Probabilistic analysis of manipulation tasks: A computational framework. *IJRR* 15(1):1–23.

Burns, B., and Brock, O. 2007. Sampling-based motion planning with sensing uncertainty. *ICRA*.

Censi, A.; Calisi, D.; Luca, A. D.; and Oriolo, G. 2008. A bayesian framework for optimal motion planning with uncertainty. *ICRA* 1798–1805.

Diankov, R., and Kuffner, J. 2008. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, CMU.

Gadeyne, K.; Lefebvre, T.; and Bruyninckx, H. 2005. Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition andgeometrical parameter estimation. *IJRR* 24:615.

Gonzalez, J. P., and Stentz, A. 2005. Planning with uncertainty in position an optimal and efficient planner. *IROS* 2435–2442.

Hsiao, K.; Nangeroni, P.; Huber, M.; Saxena, A.; and Ng, A. 2009. Reactive grasping using optical proximity sensors. In *ICRA*.

Hsiao, K.; Kaelbling, L. P.; and Lozano-Perez, T. 2007. Grasping pomdps. *ICRA*.

Hsiao, K.; Kaelbling, L. P.; and Lozano-Perez, T. 2008. Robust belief-based execution of manipulation programs. In *WAFR*.

Hsiao, K. 2009. *Relatively Robust Grasping*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *RSS*.

Latombe, J. 1991. *Robot Motion Planning*. Norwell, Mass: Kluwer Academic Publishers.

LaValle, S. M., and Hutchinson, S. A. 1994. An objective-based stochastic framework for manipulation planning. In *IROS*, 1772–1779.

Lozano-Pérez, T.; Mason, M.; and Taylor, R. H. 1984. Automatic synthesis of fine-motion strategies for robots. *IJRR* 3(1).

Mayton, B.; Garcia, E.; LeGrand, L.; and Smith, J. R. 2009. Electric field pretouch: Towards mobile manipulation. In *RSS Workshop on Mobile Manipulation in Human Environments*.

Melchior, N. A., and Simmons, R. 2007. Particle rrt for path planning with uncertainty. *ICRA*.

Ong, S. C. W.; Png, S. W.; Hsu, D.; and Lee, W. S. 2005. Pomdps for robotic tasks with mixed observability. In *RSS*.

Petrovskaya, A., and Ng, A. Y. 2007. Probabilistic mobile manipulation in dynamic environments, with application to opening doors. *IJCAI*.

Prentice, S., and Roy, N. 2007. The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. *ISRR*.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21:1071–1088.

Smith, T., and Simmons, R. G. 2005. Point-based pomdp algorithms: Improved analysis and implementation. In *UAI*, 542–547.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. Cambridge, MA: MIT Press.