# Grasping POMDPs

Kaijen Hsiao and Leslie Pack Kaelbling and Tomás Lozano-Pérez

***Abstract*— We provide a method for planning under uncertainty for robotic manipulation by partitioning the configuration space into a set of regions that are closed under compliant motions. These regions can be treated as states in a partially observable Markov decision process (POMDP), which can be solved to yield optimal control policies under uncertainty. We demonstrate the approach on simple grasping problems, showing that it can construct highly robust, efficiently executable solutions.**

## I. Introduction

A great deal of progress has been made on the problem of planning motions for robots with many degrees of freedom through free space [10], [9], [13]. These methods enable robots to move through complex environments, as long as they are not in contact with the objects in the world. However, as soon as the robot needs to contact the world, in order to manipulate objects, for example, these strategies do not apply. The fundamental problem with planning for motion in contact is that the configuration of the robot and the objects in the world is not exactly known at the outset of execution, and, given the resolution of sensors, it cannot be exactly known. In such cases, traditional open-loop plans (even extended with simple feedback) are not reliable.

It is useful to distinguish between modes of uncertainty that can be effectively modeled, and those that cannot. In situations with unmodelable uncertainty, such as insertion of keys into locks, very fine-grained details of the surfaces can have large effects on the necessary directions of applied forces, and the available sensors can gain little or no information about those surfaces. When the uncertainty is unmodelable, we must fall back to strategies such as "wiggling" the key, which are highly successful without ever building a model of the underlying situation.

Modelable uncertainty, on the other hand, typically occurs at a coarser scale. In attempting to pick up a mug, for example, a robot with vision or range sensing might have a good high-level model of the situation, but significant remaining uncertainty about the pose or shape of the mug. Based on sensor feedback, it can reason about whether the hand is currently in contact with the handle or the cup body, and choose actions that will both gather more information about the situation and make progress toward a desired grasp with a multi-fingered hand.

This research was supported in part by DARPA IPTO Contract FA8750-05-2-0249, "Effective Bayesian Transfer Learning", and in part by the Singapore-MIT Alliance agreement dated 11/6/98.

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139 {kjhsiao, lpk, tlp}@csail.mit.edu

An early approach to planning in the presence of modelable uncertainty was developed in [15]. They used a worst-case model of sensor and motion error, and developed a framework for computing conservative plans under these assumptions. This method was computationally complex, and prone to failure due to overconservatism: if there was no plan that would work for all possible configurations consistent with the initial knowledge state, then the entire system would fail.

In this paper, we build on those ideas, addressing the weaknesses in the approach via abstraction and probabilistic representation. By modeling the initial uncertainty using a probability distribution, rather than a set, and doing the same for uncertainties in dynamics and sensing, we are in a position to make trade-offs when it is not possible to succeed in every possible situation. We can choose plans that optimize a variety of different objective functions involving those probabilities, including, most simply, the plan most likely to achieve the goal. The probabilistic representation also affords an opportunity for enormous computational savings through a focus on the parts of the space that are most likely to be encountered.

By building an abstraction of the underlying continuous configuration and action spaces, we lose the possibility of acting optimally, but gain an enormous amount in computational simplification, making it feasible to compute solutions to real problems. Concretely, we will use methods of model minimization to create an abstract model of the underlying configuration space, and then model the problem of choosing actions under uncertainty as a *partially observable Markov decision process* [25].

## II. Background and Approach

The approach we outline here applies to any domain in which a robot is moving or interacting with other objects and there is non-trivial uncertainty in the configuration. In this paper, we concentrate on the illustrative problem of a robot arm and hand performing pick-and-place operations. We assume that the robot's position in the global frame is reasonably well known, but that there is some uncertainty about the relative pose and/or shape of the object to be manipulated. Additionally, we assume that there are tactile and/or force sensors on the robot that will enable it to perform compliant motions and to reasonably reliably detect when it makes or loses contacts. We frame this problem primarily as a planning problem. That is, we assume that a reasonably accurate model of the task dynamics and sensors is known, and that the principal uncertainty is in the configuration of the robot and the state of the objects in

the world. In future work, we will address the problem of learning underlying models from experience.

Excellent summaries of the robot planning literature are available [10], [13]. An early formulation of the problem of automatically planning robot motions under uncertainty was the preimage backchaining framework [15]. It was extended in [5], [6], [11]. More recently, [12] formulated both probabilistic and nondeterministic versions of the planning problem through information space. [2] describes a direct probabilistic extension of the preimage backchaining framework that shares much with our approach, including the idea of constructing an abstract state space by grouping underlying states with similar transition behavior. They describe how one might compute the necessary probabilistic pre-images, but do not have effective algorithms for doing so, nor do they provide a method for conditioning actions on observations. A recent extension of these ideas to dynamic tasks by sequential composition of feedback primitives is described in [3]. In addition, [8] describes a system that learns optimal control policies in an information space that is derived from the changes in the observable modes of interaction between the robot and the object it is manipulating.

There is a substantial body of previous work (for example [1], [14], [24], [16]) that specifies assembly strategies as a path through a sequence of contact states, typically a contact between particular surfaces of the assembled objects. The relevant sequences of states to achieve the goal are variously obtained from a human designer or from analysis of a human performing the assembly. The control strategy is then to identify the current state of the assembly from the history of perceived positions and forces [18], [20], [16], [24], [14], [1] and to choose the appropriate next action; this is a problem that can be quite challenging and which has been the subject of a great deal of work.

The work we describe here advances on this approach in several ways: (a) it provides an automated way of finding the states and the plans through the state space, (b) it does not require unambiguous identifications of the current state of the manipulation but only a characterization of a probability distribution over the states (the belief state) and (c) it provides an integrated way to choose actions that will provide information relevant to disambiguating the state while also achieving the goal.

There is a long history of applying POMDPs to mobile-robot navigation, beginning with simple heuristic solution methods [4], [23], then applying more sophisticated approximations [27], [21], [22], [26], [29].

### A. POMDPs

Partially observed Markov decision processes (POMDPs) [25] are the primary model for formalizing decision problems under uncertainty. A POMDP model consists of finite sets of states $S$, actions $A$, and observations $O$; a *reward* function $R(s, a)$ that maps each underlying state-action pair into an immediate reward; a *state-transition model* $P(s'|s, a)$ that specifies a probability distribution over the resulting state $s'$, given an initial state $s$ and action $a$; and an *observation model* $P(o|s)$ that specifies the probability of making an observation $o$ in a state $s$. Problems that are naturally described as having a goal state can be encoded in this framework by assigning the goal states a high reward and all the rest zero; but an advantage of a more general reward function is that it can easily also penalize other conditions along the way, or assert two goal regions, one of which is more desirable than the other, and so on.

Given the model of a POMDP, the problem of optimal control can be broken into two parts: state estimation, in which a probability distribution over the underlying state of the world, or *belief state*, is recursively estimated based on the actions and observations of the agent; and policy execution, in which the current belief state is mapped to the optimal control action.

Belief-state update is a straightforward instance of a Bayesian filter. The robot's current state estimate is an $n$-dimensional vector, $b_t$, representing $\Pr(s_t|o_1 \dots o_t, a_1 \dots a_{t-1})$, a probability distribution over current states given the history of actions and observations up until time $t$. Given a new action $a_t$ and an observation $o_{t+1}$, the new belief state $b_{t+1}$ is given by

$$\begin{aligned}
&\Pr(s_{t+1} = i|o_1 \dots o_{t+1}, a_1 \dots a_t) \\
&\quad \propto \sum_j \Pr(s_t|o_1 \dots o_t, a_1 \dots a_{t-1}) \\
&\qquad\qquad \cdot \Pr(s_{t+1} = i|s_t = j, a_t = a) \Pr(o_{t+1} = o|s_{t+1} = i) \\
&\quad = \sum_j b_j \Pr(s_{t+1} = i|s_t = j, a_t = a) \\
&\qquad\qquad \cdot \Pr(o_{t+1} = o|s_{t+1} = i)
\end{aligned}$$

Note that the first factor is an element of the state transition model and the second is an element of the observation model. The constant of proportionality is determined by the constraint that the elements of $b_{t+1}$ must sum to 1.

The problem of deriving an optimal policy is much more difficult. The policy for a POMDP with $n$ states is a mapping from the $n$-dimensional simplex (the space of all possible belief states) into the action set. Although a policy specifies only the next action to be taken, the actions are selected in virtue of their long-term effects on the agent's total reward. Generally, we seek policies that choose actions to optimize either the expected total reward over the next $k$ steps (finite-horizon) or the expected infinite discounted sum of reward, in which each successive reward after the first is devalued by a discount factor of $\gamma$.

These policies are quite complex because, unlike in a completely observable MDP, in which an action has to be specified for each state, in a POMDP, an action has to be specified for *every* probability distribution over states in the space. Thus, the policy will know what to do when the robot is completely uncertain about its state, or when it has two competing possibilities, or when it knows exactly what is happening.

Computing the exact optimal finite or infinite-horizon solution of a POMDP is generally extremely computationally

intractable. However, it is often possible to derive good approximate solutions by taking advantage of the fact that the set of states that are reachable under a reasonable control policy is typically dramatically smaller than the original space [21], [27], [26].

### B. State and action abstraction

Robot manipulation problems are typically framed as having high-dimensional continuous configuration spaces, multi-dimensional continuous action spaces (positions or torques), possibly continuous time, and deterministic dynamics. Our approach will be to construct discrete abstractions of the robot's state and action spaces, and to "make up" for the precision lost in so doing by modeling the effects of actions as stochastic.

It is possible to use a grid discretization of the continuous belief space, but the high dimensionality of that space makes it infeasible for most problems of interest. Instead, we pursue a discretization strategy that is more directly motivated by the uncertainty in the problem. When there is uncertainty with respect to the configuration of the robot or obstacles, we will generally want to execute actions that reduce uncertainty, while making progress toward a goal. There are two ways to reduce uncertainty through action: one is to act to obtain observations that contain information about the underlying state; the other is to take actions that are "funnels," mapping large sets of possible initial states to a smaller set of resulting states.

We start by considering the MDP, defined over complete configurations of the robot and object, that underlies our problem, and construct abstract state and action spaces and an abstract state transition model on those spaces. We will use the abstract MDP as the basis for an abstract POMDP. We construct the abstract space for the MDP by choosing a set of abstract actions [28] and using them to induce the state space. We will work with a set of "guarded" compliant motions as our action space. A guarded motion causes the robot to move along some vector until it makes or breaks a contact or reaches the limit of the workspace. Our action set includes guarded motions through free space, as well as compliant motions, in which the robot is constrained to maintain an existing contact while moving to acquire another one. Note that these actions serve as "funnels," producing configurations with multiple contacts between the robot and an object, and generating information about the underlying state. In the current work, we allow the robot to move only one degree of freedom at a time: there are motions in two directions for each DOF, which attempt to move in the commanded direction while maintaining the existing set of contacts, if possible.

Abstraction methods for MDPs [7], derived from abstraction methods for finite-state automata, take an underlying set of states, a set of actions and a reward function, and try to construct the minimal abstract state space. This new state space is a partition of the original state space, the regions of which correspond to the abstract states. The abstract space must have the properties that, for any sequence of actions, the distribution of sequences of rewards in the abstract model is the same as it would have been in the original model, and, furthermore, that any two underlying states that are in the same abstract state have the same expected future value under the optimal policy.

So, given a commitment to guarded motions as our action set, the known deterministic continuous dynamics of the robot, and a specification of a goal (or, more generally, a reward function), we can apply model-minimization methods to determine an abstract model that is effectively equivalent to the original. We obtain a large reduction through not having to represent the free space in detail, because after the first action, the robot will always be in contact, or in a very limited subspace of the whole free space. In addition, large regions of the state space will behave equivalently under funneling actions that move until contact.

We begin by assuming an idealized deterministic dynamics, derived from the geometry, both in free space and in contact, and construct an abstract state space using those dynamics. Given that abstract state space, we will go back and compute more realistic transition probabilities among the abstract states. Finally, we will feed this resulting model into an approximate POMDP solver to derive a policy.

### C. Example

To give an intuition for the approach, we present a very simple example. Consider a two-dimensional Cartesian robot with a single rectangular finger, and a rectangular block on a table. The robot has contact sensors on the tip and each side of the finger, and so it can make eight possible observations (though combinations involving contact on both sides of the finger are impossible in this scenario).

The action space is the set of compliant guarded moves *up, down, left*, and *right*. If the robot has a contact in one direction, say down, and tries to move to the left, it does so while attempting compliantly to maintain contact with the surface beneath. The robot moves until its observed contacts change. A motion can also be terminated when the robot reaches its limits (we'll assume a rectangular workspace).

The robot's "goal" is to have the tip of its finger on top of the block. It has an additional action, called "lift" which is intended to mean that it could engage a vacuum actuator, or simply declare success. If the robot lifts when it has its finger on top of the block, then it gets a reward of +10 and the trial is terminated. If it lifts when it is in any other configuration, then it gets a reward of -10, and the trial is terminated. Additionally, it gets a reward of -1 on each step to encourage shorter trajectories.

The configuration space for this robot is two dimensional, with reachable configurations everywhere except for a box that is "grown" by the dimensions of the finger. Figure 1A shows how the configuration space is initially segmented based on tactile observations (various one-dimensional loci of contact configurations are shown as thin rectangular boxes). The goal configurations are further differentiated into their own region in figure 1B. Finally, we can propagate those regions so that we arrive at a segmentation that is closed, in

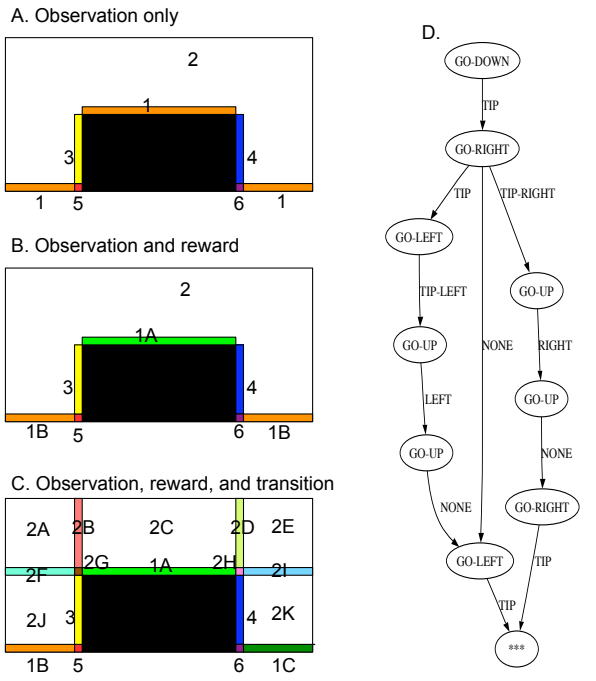


Fig. 1. A. Observation partition; B. Observation and reward partition; C. Closed partition; D. Partial policy graph for robot starting in an unknown state above the table, with a deterministic transition and observation model.

the sense that all of the states in a single region, given a particular action, transition to a particular other region, as shown in figure 1C.

Even with completely deterministic transition and observation models, the POMDP is a useful tool. If, for instance, we examine the optimal policy for the initial belief state in which the robot could be in state $2A$, $2C$, or $2E$, with equal probability, then we get the partial policy graph shown in figure 1.D. In a policy graph, the nodes are labeled with actions and the arcs with observations. This policy has the robot begin by moving down until contact. At this point, it could be in regions $1B$, $1A$, or $1C$. The policy specifies that it move to the right, putting it in one of regions $5$, $2H$, or $1C$. Each of these regions has a different observation (*tip-right*, *none*, or *tip*), and so the rest of the strategy is determined in each case.

If we add a significant amount of noise to the actions and observations, so that they have their "nominal" outcome about 0.8 of the time, and generate erroneous readings or results with the remaining probability, this problem becomes much more complicated, and much too difficult to solve by hand. We can solve this model for the optimal policy, which embodies a fairly different strategy. Even the initial move is different: it asks the robot to move all the way over to the left at the very beginning, which moderately reliably funnels all the initial states into one, removing some of the initial uncertainty at the cost of performing an additional action. If the actions were even noisier, the policy might ask that the robot move to the left multiple times, to further reduce uncertainty.

## III. Model construction

We construct the initial model by using a simple, nearly deterministic "simulation" of the compliant guarded moves, based on geometry. It is appropriate to think of this simulation somewhat analogously to the simulation that happens in a planner: it is a computational process, in the "head" of the robot, that will eventually lead to a plan. Note that the derived model can be re-used with different reward functions, to plan for different goals in the same environment.

In our implementation, the robot and the objects in the world are all modeled as polygons, and the simulator operates in a state space that includes the joint-space configuration of the robot and the contact state. The contact state specifies, for each vertex and surface of the robot, which, if any vertex or surface of the world it is in contact with. Both of the examples in this paper are done in a two-dimensional world space, with robots of 2 or 3 degrees of freedom. The approach extends naturally to three-dimensional world spaces, although the number of possible contacts grows quickly. The robot is assumed to have a set of contact sensors that can give some of the information in the contact state. Our typical model will be that the robot can tell what vertices or surfaces of the robot are in contact, but not with what vertices or surfaces in the world.

*Constructing the abstract action space:* The abstract actions consist of two guarded, compliant move commands for each degree of freedom, one in each direction, including the gripper. When there are no contacts in the starting state, the robot simply moves along the commanded degree of freedom, holding the others constant, until the observed contacts change.

When the robot is already in contact, it is controlled by a sliding model, which is related to a damper model used by [15], and there are three cases of interest: (1) The current contact is in the opposite direction of the commanded motion. In this case, the robot simply breaks the contact and moves away through free space until it makes a new contact or reaches the limits of its workspace. (2) The current contact is in the same direction as the commanded motion. In this case, the robot cannot, and does not, move. The action is terminated if there is no motion over a short period of time. (3) The current contact has a component that is orthogonal to the commanded motion. In this case, the robot seeks to move the degree of freedom in the commanded direction, while maintaining the existing contact. It does this by making small motions in the commanded direction, then moving back to regain the contact. This can result in sliding along a surface, closing a parallel-jaw gripper while maintaining contact of one finger with the object, or pivoting down to complete a grasp while maintaining contact of one finger on the object. The action is terminated when the observed contacts change or when the current contact no longer has a component orthogonal to the commanded motion. This set of abstract actions can now be used to induce a discrete abstract state space that can serve as a basis for our POMDP model.

*Creating an abstract model:* Following [7], we begin by

considering how to create an abstract model of a discrete system with deterministic transitions and observations. Let $T(s, a)$ be the deterministic state transition function, specifying the state resulting from taking action $a$ in state $s$; let $O(s)$ be the deterministic observation function specifying the observation resulting from being in state $s$; and let $R(s)$ be the deterministic reward function.

Given a low-level state space $S$, our goal will be to find a partition $\Phi$, which consists of a set of non-overlapping regions $R_i$, such that $\bigcup_i R_i = S$. We will write $\Phi(s)$ for the region to which primitive state $s$ is assigned. The partition $\Phi$ should also be the minimal partition that satisfies two requirements. First, that it is uniform with respect to rewards and observations, so that, for every $R_i$ in $\Phi$, there exists a reward $r$ and observation $o$, such that for all $s$ in $R_i$, $R(s) = r$ and $O(s) = o$. Second, that it is uniform with respect to *action sequences*, so that for every sequence of actions $a_1 \ldots a_n$ and every region $R_i$, there exists a resulting region $R_j$, such that for all $s$ in $R_i$, the result of taking that action sequence in that state is in the same partition:

$$\Phi(T(T(\ldots T(T(s, a_1), a_2), \ldots), a_n) = R_j \quad .$$

The algorithm for finding such a minimal partition is a relatively simple process of region-splitting. We'll say that a region $R_i$ is *deterministically uniform* with respect to action $a$ and partition $\Phi$ if there exists a region $R_j$, such that for all $s$ in $R_i$, $\Phi(T(s, a)) = R_j$.

- Let $\Phi$ be the partitioning of $S$ such that each region in the partition is uniform with respect to reward and observation.
- While there exists a region $R$ in $\Phi$ and an action $a$ such that $R$ is not deterministically uniform with respect to $a$ and $P$, split $R$ into a set of sub-regions $R_i$ that are deterministically uniform with respect to $a$ and $\Phi$, and replace $R$ with the $R_i$ in $\Phi$.
- Return $\Phi$.

The resulting abstract state space is the set of regions in $\Phi$. From this, it is straightforward to construct a POMDP with deterministic transitions and observations.

Unfortunately, even though our simple model of the robot's dynamics is deterministic, our situation is more complex. The state space we are working in, at the lowest level, is continuous, so we cannot enumerate the states. It is possible, in principle, to construct a version of the splitting algorithm that operates on analytically represented subregions of the configuration space; in fact, the "preimage backchaining" method [15] is related to this algorithm. However, using compliant motions in a complex configuration space makes it very challenging to compute these regions analytically.

Instead, we will take advantage of our ability to simulate the results of the abstract actions, to build a "model" of the transition dynamics of these actions via sampling. We are given a set of possible starting configurations (in the case of the examples in this paper, they were a discrete set of positions of the robot up at the top of the workspace). Based on these initial configurations, we gather a large sample of state-action-next-state ($\langle s, a, s' \rangle$) triples, by trying each possible action at each initial state, then at each of the states $s'$ reachable from an initial state, etc.

Because of the nature of the action set, if the simulation were truly deterministic, we would expect this process to "close" in the sense that eventually no new states would be found to be reachable. In practice, due to numerical sensitivities in the simulation of the compliant motions, exact closure doesn't happen. We handle this problem by clustering reachable states together whenever they have equal contact conditions and geometric distance between robot configurations less than a fixed threshold. We draw samples until we have experience of taking each action from each cluster.

Now, each of these clusters is treated as a primitive state, and the most frequent outcome under each action is defined to be its successor. This data is now used as input to the region-splitting algorithm described above.

*Adding stochasticity:* Once we have this partition of the state space, we need to determine the observation and transition models. We take a very simple approach to adding stochasticity. For observations, we assume that the contact sensors have some probability of failing to detect a contact (independent failures for each sensor), but no chance of mistakenly sensing a contact when there shouldn't be one.

For transitions, we add two forms of stochasticity. First, we reason that, in executing a particular action $a$ from a particular abstract state $R_i$, it might be possible to reach, in error, any of the other states that can be reached from $R_i$ using other actions. So we add some probability of making a transition from $R_i$ to $R_j$ under action a, when there exists any $a'$ such that $T(R_i, a') = R_j$. Second, we note that there are some states that are very "unstable", such as those involving single-point contacts, in the sense that the robot is very likely to overshoot them by not noticing the relevant contact changes. So, for any state $R_j$ that is unstable, and such that $T(R_i, a) = R_j$ there exists an action $a'$ for which $T(R_j, a') = R_k$, we add a non-zero probability of a transition from $R_i$ to $R_k$ under action $a$. In addition, if one of these resulting states $R_k$ is unstable, we add transitions to its successors as well.

This is a very simple model of the stochasticity in the system, which is certainly inaccurate. One advantage of using POMDPs for control is that they are generally quite robust with respect to variations in the transition and observation probabilities. It would also be possible to further tune the error model using a high-fidelity dynamics simulation or even data from a physical robot.

## IV. Solving the POMDP

Even for the simplest problems, as soon as we add noise, it is infeasible to solve the resulting POMDPs exactly. We have used HSVI [26], a form of *point-based value iteration*, which samples belief states that have a relatively high probability of being encountered, and concentrates its representational and computational power in those parts of the belief space.

HSVI returns policies in the form of a set of $\alpha$ *vectors* and associated actions. The expected discounted sum of values when executing this policy from some belief state $b$ is

$$V(b) = \max_{\alpha_i} b \cdot \alpha_i$$

and the best action is the action associated with the maximizing alpha vector. The $\alpha$ vectors define hyperplanes in the belief space, and the maximization over them yields a value function that is piecewise-linear and convex. By construction, each of the $\alpha$-vectors is maximal over some part of the belief space; and the space is partitioned according to which $\alpha$-vector is maximizing over that region.

So, to execute a policy, we apply a state estimator as described earlier. The state estimator starts in some initial belief state, and then consumes successive actions and observations, maintaining the Bayes optimal belief state. To generate an action, the current belief state is dotted with each of the $\alpha$-vectors, and the action associated with the winning $\alpha$-vector is executed. This process is quite efficient, even if the policies were slow[1] to derive off-line.

## V. Results

As a proof of concept, we have tested the approach described above in two planar problems: one involving placing one finger on a stepped block and one for two-finger grasping of a block. We derive stochastic policies from simulations on a simple planar model. We then run the policy for the stochastic model in a high-fidelity dynamics simulation and measure average total reward per episode. Note that the stochastic model and the high-fidelity model differ in some substantial details: the dimensions of the block and the geometry of the fingers are different and the actual sensor and detailed control behavior are different. Therefore, some of the trajectories that are most common in the high-fidelity simulation have relatively low probability in the stochastic model. These simulations gives us a measure of how much the mis-estimation of the probabilities in the stochastic model decreases performance. As a comparison, we report the results for a simple but reasonable fixed strategy, as well.

*Single finger/Stepped block:* This domain is similar to the one described in the example in section II-C except that the block is "stepped". The goal is to place the finger in the corner at the left step. Note that, since the robot is lacking position information, the goal is locally indistinguishable to the sensors from the corner where the block rests on the table. The rewards are similar to the earlier example (+15 for reaching the goal, -50 for lifting in the wrong state, -1 for each motion) except that we also penalize (-5) being in the states at the limits of the designated problem workspace. This discourages long motions that leave the vicinity of the block. The abstract state space for this problem has $40$ states.

A trajectory derived by following the stochastic policy for this problem is shown in Figure 2. This policy, found by solving the POMDP formulated as described above, succeeded in 466 out 506 trials ($92\%$) in the high-fidelity simulation. Its average reward is $-1.59$. We can compare this to a fixed policy that simply moves the hand in a fixed pattern of *left*, *down*, *right*, *right*, *up*, *right*, *right*, *right* (*LDRRURRR*). This fixed policy succeeded in 154 out of 190 trials ($81\%$), with an average reward of $-10.632$. The POMDP policy is considerably more robust than the fixed strategy.

To help gain intuition about the kinds of strategies being developed, we can examine the solutions for deterministic versions of these problems (solutions for the noisy versions are very difficult to understand intuitively; see figure 4 if you can). Figure 3 shows the deterministic policy for placing the finger at the left step of the stepped block. It first asks the robot to move down; then, depending on the observation that is made, it selects a strategy. If it feels a tip contact, then it moves to the left, and now there are three possible observations: *none*, which means it was on top of the left part or the top of the block, and has now lost contact, *tip(-x)*, which means that it is at the negative-x limit of the table, and *left&tip*, which means that it's feeling contact on the outer (left) part of the finger and the tip, so it is either on top of the right step of the block, or the right part of the table. The simplest situation is when it feels *none*: now it goes right and regains the tip contact, and moves right again. If this move results in *none* as an observation, then we know we're on top of the block (just off the left corner), and have to move down to the left-hand step. On the other hand, if we get *right&tip* as the observation, we know the robot is in the right place, and we can command a "lift" action, which in this case simply is a signal for success.

We stress that this policy is only illustrative, since it is derived from a deterministic model; the stochastic policy is qualitatively different. Because the actions and observations are much less reliable, it has to handle many more cases, including rational responses to observation sequences that would have been impossible in the deterministic model.

*Two-dimensional Grasping:* A more interesting domain is two-fingered grasping in two dimensions. In this case, the robot has three degrees of freedom: motion in the $x$ and $z$ planes, as well as opening and closing the parallel fingers. The abstract state space for this problem has $408$ states. Figure 5 shows a policy we automatically derived for grasping an object with two fingers, in the absence of noise. In this figure, the contacts for the left finger and right finger are shown on the arcs connected by a hyphen, e.g. TIP-NO indicates *tip* contact detected on the left finger and no contact detected on the right finger. The -G notation indicates that the fingers are at their wide open "limit stop". Each finger has tip, inside and outside sensors.

Once again we see that the policy first asks the robot to move down; then, depending on the observation that is made, it selects a strategy. If it feels two tip contacts, *tip-tip(-g)*, then there are three possible situations: the fingers straddling the block, completely to the left of the block or completely to the right. It moves to the right, and now there

[1]For large problems, the POMDP approximation methods may become slow, but for all the results reported here, the POMDP solutions ran in under 10 minutes.

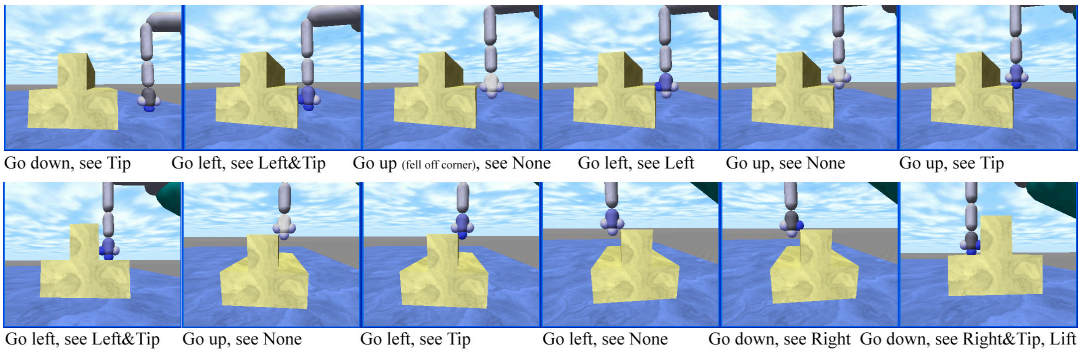


Fig. 2. Sample run of one-finger policy on stochastic stepped block model.

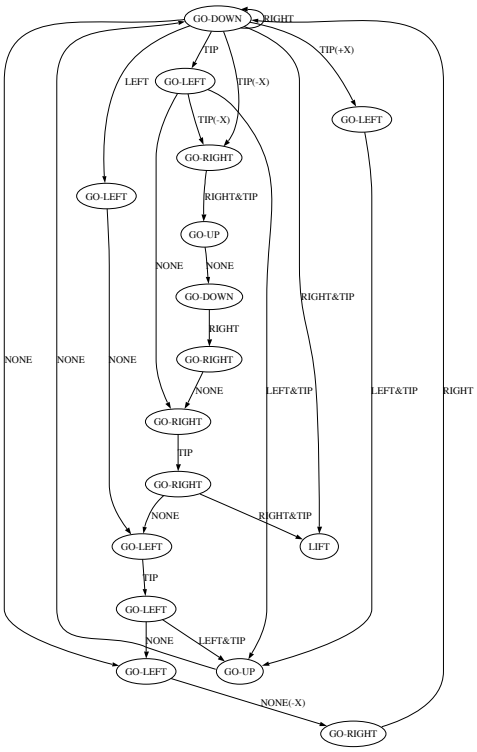


Fig. 3. One-finger policy for deterministic stepped block model.

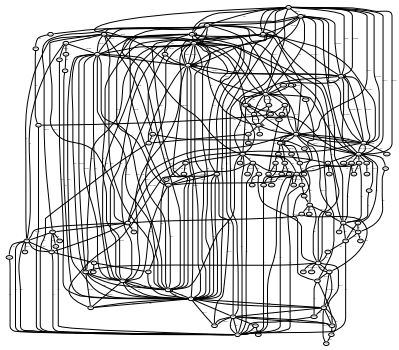

Fig. 4. One-finger policy for noisy stepped block model.

are three possible observations: *it-tip(-g)*, which means that the left finger contacted the block and so the fingers started straddling the block, *tip-tip(-g,+x)*, which means the robot got to the positive-x limit of the workspace and so must have started completely to the right of the block, and *tip-ot(-g)*, that is, the right finger touched the block on its outer sensor and so must have started completely to the left of the block. The rest of the policy proceeds in a similar fashion. This whole policy is represented internally by $100$ $\alpha$-vectors.

An example trajectory derived by following the stochastic policy for this problem is shown in Figure 6. This policy, found by solving the POMDP formulated as described above (and encoded in approximately $1000$ $\alpha$-vectors), succeeded in $115$ out of $115$ trials ($100\%$) in the high-fidelity simulation. Its average reward is $4.0$. We can compare this to a fixed policy that simply moves the hand in a fixed pattern of *LDRRURRDDDG*. The fixed policy succeeded in 86 out of $113$ trials ($76\%$), with an average reward of $-17.24$, which is significantly worse than the behavior of the POMDP policy.

This work is an initial attempt to apply POMDPs to the problem of robot manipulation. It demonstrates that considerable advantages in robustness can be gained through the POMDP formulation. Important next steps will be to address problems with shape uncertainty, to allow objects to slide or tip, and to handle interactions with other objects.

## REFERENCES


[1] L. Brignone and M. Howarth. A geometrically validated approach to autonomous robotic assembly. *IEEE/RSJ International Conference on Intelligent Robots and System*, 2:1626–1631, 2002.

[2] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A computational framework. *International Journal of Robotics Research*, 15(1):1–23, 1996.

[3] Robert R. Burridge, Alfred A. Rizzi, and Daniel E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *I. J. Robotic Res.*, 18(6):534–555, 1999.

[4] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[5] Bruce Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence*, 37:223–271, 1988.

[6] Michael Erdmann. Using backprojection for fine motion planning with uncertainty. *IJRR*, 5(1):240–271, 1994.

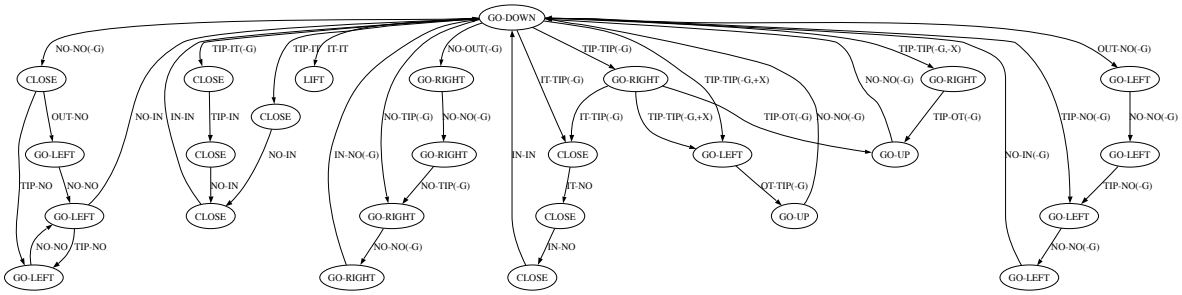


Fig. 5. Two-finger grasping policy for deterministic model.

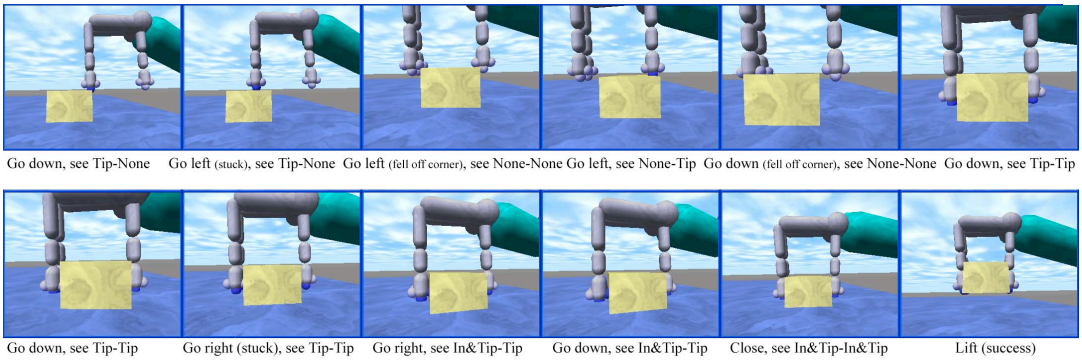


Fig. 6. Sample run of two-finger grasp policy in high-fidelity simulation.


[7] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artif. Intell.*, 147(1-2):163–223, 2003.

[8] R.A. Grupen and Jr. J.A. Coelho. Acquiring state from control dynamics to learn grasping policies for robot hands. *Advanced Robotics*, 16(5):427–443, 2002.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[10] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Mass, 1991.

[11] Jean-Claude Latombe, Anthony Lazanas, and Shashank Shekhar. Robot motion planning with uncertainty in control and sensing. *Artif. Intell.*, 52(1):1–47, 1991.

[12] S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems*, pages 1772–1779, September 1994.

[13] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006, to appear.

[14] T. Lefebvre, H. Bruyninckx, and J. DeSchutter. Polyhedral contact formation identification for autonomous compliant motion: Exact nonlinear bayesian filtering. *IEEE Transactions on Robotics*, 21(1):124–129, 2005.

[15] Tomás Lozano-Pérez, Matthew Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.

[16] Brenan J. McCarragher and Haruhiko Asada. A discrete event approach to the control of robotic assembly tasks. In *ICRA*, pages 331–336, 1993.

[17] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. In *UAI*, pages 427–436, 1999.

[18] Wyatt S. Newman, Michael Branicky, Yoh-Han Pao, Craig Birkhimer, Siddharth Chhatpar, Jing Wei, and Yonghong Zhao. Intelligent strategies for compliant robotic assembly. In *Proc. 11th Yale Workshop on Adaptive and Learning Systems*, pages 139–146, 2001.

[19] Andrew Y. Ng and Michael I. Jordan. Pegasus: A policy search method for large mdps and pomdps. In *UAI*, page 406, 2000.

[20] Anya Petrovskaya, Oussama Khatib, Sebastian Thrun, and Andrew Y. Ng. Bayesian estimation for autonomous object manipulation based on tactile sensors. In *ICRA*, 2006.

[21] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, 2003.

[22] N. Roy and S. Thrun. Coastal navigation with mobile robot. In *Proceedings of Conference on Neural Information Processing Systems (NIPS99)*, 2000.

[23] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[24] M. Skubic and R. A. Volz. Acquiring robust, force-based assembly skills from human demonstration. *IEEE Transactions on Robotics and Automation*, 16(6):772–781, 2000.

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

[26] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527, Arlington, Virginia, United States, 2004. AUAI Press.

[27] M.T.J. Spaan and N. Vlassis. Perseus: randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[28] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

[29] Georgios Theocharous and Leslie Pack Kaelbling. Approximate planning in POMDPs with macro-actions. In *Advances in Neural Information Processing Systems 16 (NIPS03)*, 2004.