Robust Belief-Based Execution of Manipulation Programs

Kaijen Hsiao, Tomás Lozano-Pérez, and Leslie Pack Kaelbling

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, {kjhsiao, tlp, lpk}@csail.mit.edu

Abstract: We describe a simple approach for executing manipulation programs in the presence of significant, but bounded, uncertainty. The key idea is to maintain a belief-state (a probability distribution over world states) and to execute fixed trajectories relative to the most-likely state of the world. These *world-relative* trajectories, as well as the transition and observation models needed for belief update, are all constructed off-line, so the approach does not require any on-line motion planning.

1 Introduction

We would like robots to be able to manipulate objects robustly and reliably, in relatively unstructured environments. We concentrate on situations where there is an initial estimate of an object's state, made by a passive 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. So, 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.

The foundation of our approach is "belief-based" programming, in which strategies for robot behavior are described relative to a probability distribution over the uncertain aspects of the world state, which summarizes information received by the system up until the moment a decision is being made. Belief-based programming suggests dividing our robot programs into two modules: a state estimator and a policy. It is relatively easy to construct a state estimator that recursively computes a belief state, as a function of the previous belief state, action and observation. The problem of finding a good policy is much more difficult. In some cases, human programmers can write policies directly; but as domains become more complex, we would like to be able to generate policies automatically, given a model of sensing and action dynamics.

The partially observed Markov decision process (POMDP) framework [13] allows optimal policies to be derived for domains with finite state, action, and observation spaces, but even in such simple cases, finding the optimal policy for a POMDP can be highly computationally intractable. There are new approximation methods that can solve discrete POMDPs with several thousand states effectively [24, 14]. However, the problem of grasping under uncertainty has continuous state, action, and observation spaces, making it particularly difficult to address in this framework.

An efficient and effective, though suboptimal, strategy for POMDPs is to explicitly switch between goal-achievement and information-gain modes during execution, depending on properties of the current belief state. In goal-achievement mode, actions are selected based on assuming that the world state that is believed to be most likely is, in fact, the true world state, and using traditional planning methods. In information-gain mode, actions are selected based on their short-term ability to reduce entropy in the belief state (optimal planning for long-term reduction of entropy in the belief state is as hard as the whole POMDP planning problem.) This mode-switching strategy was applied by Cassandra et al. [6] to mobile robot navigation in relatively small discretized domains. The details of those methods, both in goal-achievement and information-gain modes, depend crucially on the enumerability (and, ultimately, relatively small size) of the state, action, and observation spaces.

In this paper, we describe how to construct such a mode-switching controller for robot manipulation problems, with the significant added difficulty of working in high-dimensional continuous state, action and observation spaces. We use a uniform discretization of the uncertain state space, which is the pose of the object to be grasped. Crucially, we will not attempt to cover the entire space of actions and observations via uniform discretization. We instead use a robot motion planner, in an off-line phase, to generate a relatively small set of motion trajectories that will be executed on-line. These trajectories will be useful in a wide variety of belief situations, because they are expressed relative to an estimate of the configuration of the underlying world. These trajectories will be accompanied by a characterization of their expected observations as a function of the target object's state. Based on this observation model, we can evaluate the trajectories' goal-achievement and information-gain properties and thus allow an on-line controller to select among them efficiently and dynamically, based on the current belief state.

2 Related work

This paper addresses the problem of constructing a robust control strategy for a partially observable domain. There is a rich literature that addresses mobile robot planning under uncertainty within the POMDP framework, for the fully discrete case (e.g., [6, 23]). There were some early attempts to address problems of this kind in robot manipulation within a non-probabilistic

uncertainty framework (e.g., [18, 15]) as well as a probabilistic framework (e.g, [16, 4]). More recently there have been some direct applications of the POMDP framework to robot manipulation tasks (e.g., [12]), 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 et al.[1] 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 through the space that will maximize the probability of success or related other objectives (e.g., [22, 11, 7, 19]). Burns et al. [5] 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 [25]. Although it is much less common in the manipulation literature, several examples exist (e.g., [21, 10, 12]). Ideas of using entropy as a measure of the utility of information-seeking actions are quite old. They have been applied to the problem of deciding where to drive a mobile robot during a SLAM process [3].

Our work is related to the idea of "active localization" of Erickson et al. [9]. Their goal is a plan to localize a robot in a known map from the expected contacts that result from "move-until-contact" commands. They also maintain a belief state and use entropy as a heuristic for picking among actions. One key difference is that we employ world-relative trajectories and on-line belief updates to adjust to on-line outcomes, rather than planning off-line for a fixed action sequence.

The work in this paper can also be viewed as a methodology for robot programming in the presence of uncertainty, in which the programmer (or off-line planner) specifies a set of trajectories, a transition model and observation model and the execution environment (on-line system) chooses the actions to be performed at each time. The Bayesian Programming approach [17] has similar goals; it builds on the framework of Bayesian networks to relate observations to actions, but lacks an explicit model of utility.

3 Technical Approach

We know how to plan robot motions that carry out complex tasks when the state of the world is known almost exactly; this is the basis of most industrial automation. In less well-known or well-structured environments, our choices are to: (a) find motion strategies that achieve the goals in spite of the world uncertainty, (b) improve the sensing so that the world uncertainty is reduced sufficiently that a fixed motion can be used, or (c) combine motion and sensing in a sequential strategy that will ultimately achieve the goal. We will follow the

last of these approaches, taking advantage of the information gained during the course of execution.

One classic approach to combining motion and sensing to carry out tasks in the presence of uncertainty is the "most-likely state" approach. The idea is to maintain a belief (a probability distribution over world states), then to choose the most likely world state, plan a motion to achieve the goal in that state, execute the motion, use any sensory information obtained during execution to update the belief and repeat. There are a number of drawbacks to this approach: It will not plan actions to gain information, only to achieve the goal; it requires on-line motion planning to achieve the goal; and, importantly, it requires us to know what observations are likely to result from such motions in all of the possible world states. We pursue a variant of this "most-likely state" approach that addresses these drawbacks: it shifts, when necessary, to an explicit information-gathering mode, and does time-consuming motion planning and geometric simulation, including the construction of an observation model for belief-state update, off-line.

3.1 Problem formulation

We will assume that we know the robot's pose (in its base coordinate frame) perfectly, but have uncertainty about the object to be manipulated. 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. This could be generalized to contain information about the poses, shapes, or other properties of a set of objects. 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.

In this work, we use a discretized representation of $\mathcal{W}$, allowing representation of belief states as multinomial distributions, and making other aspects of the model simpler to represent. We will, therefore, allow w to range over *cells* in W rather than individual 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). 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 world state (region) 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.

3.2 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 space 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.

Our strategy will be to generate, off-line, a relatively small set of *world-relative trajectories* and to characterize their effectiveness in terms of achieving the goal and gaining information. Then, during the on-line execution phase, we will use this information, together with the continually updated belief state, to select and execute appropriate trajectories. One way to think of these trajectories is as temporally extended "macro actions." This approach has a relatively small branching factor, and results in effective goal-directed action with only one step lookahead.

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. The robot will begin each trajectory execution by moving back to the home pose ϕ_h.

Each τ is characterized by *feasibility*, *observation*, and *result* functions. The feasibility function $F_\tau(w)$ is true if trajectory $\tau(w)$ is kinematically feasible for the robot, and false, otherwise. Observation and result functions are 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 position (joint angles) of the robot when the contact occurs and c is the

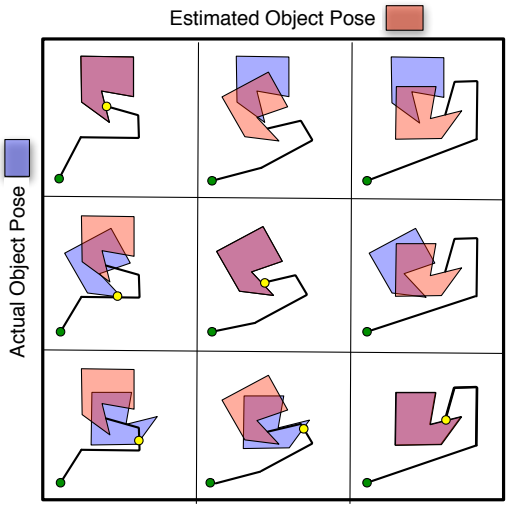

Fig. 1. The $\Omega_\tau(w, e)$ matrix for a WRT τ.

local sensor readings that can be expected when a sensed contact occurs. The result function, $R_\tau(w, e)$ has value $\langle \phi, w \rangle$ when the trajectory $\tau(e)$ will either run to completion or to a sensed contact and terminate in robot pose ϕ; if, instead, a collision will occur that cannot be sensed (e.g., with the back of the hand), then the object is likely to be knocked over or moved significantly, and we will consider it a failure, and $R_\tau(w, e)$ will have value *fail*.

Figure 3.4 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. Each estimated pose e induces a different actual trajectory $\tau(e)$ in robot coordinate space (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 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.

We can take the approach of pre-calculating the feasibility, observation, and result functions off-line. This may seem prohibitive, but it is important to note that if trajectory $\tau(e)$ is kinematically feasible and there are no other objects nearby, then the observation and result depend only on the relative transformation between w and e. Furthermore, the range of relative transformations is bounded by the initial measurement uncertainty. So, the number of simulations required to compute the relevant entries in the outcome function for a discretized (x, y, θ) state space and reasonable values of uncertainty is tractable. Alternatively, these functions can be computed on-line during belief update, only for the relevant states with non-zero probability.

3.3 Belief-state update

If we use a uniform discretization of W with n states, belief-state update is a straightforward instance of a Bayesian filter. The agent's current state estimate is an n-dimensional vector, b, 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 and an observation o, the new belief state $b' = UB(b, a, o)$ is

$$UB(b, a, o)(s') = \frac{\Pr(o|s', a) \sum_s \Pr(s'|s, a) b(s)}{\Pr(o|b, a)} \quad . \tag{1}$$

The first factor in the numerator of Eq. 1 is an element of the observation model, $P(o|s', a)$, that specifies the probability of making an observation o after arriving in state s' by using action a, and the second factor is an element of the 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.

In our case, the actions are particular trajectories, $\tau(e)$. The transition model captures the task dynamics: for example, it can model how the object might slide on the table in response to contact with the robot. In our experiments we have assumed that the transition model for world states is roughly diagonal, that is, that the robot has a "light touch" that will not substantially disturb the object's pose, if it contacts the object with a surface that has tactile sensing. Otherwise, the assumption is that the object is substantially disturbed, and the system enters a special *failed* state, from which it is not expected to recover. More general models, with significant stochasticity and dynamics can be incorporated readily. The transition model for the robot's pose is assumed to be deterministic: the robot has reliable control over its own position, subject to kinematic feasibility constraints.

The observation model describes the sensory conditions (e.g., finger contacts) that can result from a given action in a given state. In an off-line process, for each WRT τ, we construct a representation of the observation function,

$\Omega_\tau(w, e)$, and result function, $R_\tau(w, e)$ on the discretized w, e space. In the case of a single object with a canonical support surface on a table, the space of w and e are is characterized by the x, y, θ coordinates of the object. We are assuming, here, that the uncertainty is not huge: therefore, for any w, we only need to compute and store $\Omega_i(w, e)$ for a set of b that are not too distant from w. Thus, this table is effectively only three-dimensional. Furthermore, the entries in the observation function are dependent only on the relationship between w and e, and not their absolute values, so a number of the entries can be determined from a single simulation.

Computing an entry of these matrices 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 in that situation. This is a geometric computation that can be done entirely off-line, relieving the on-line system of performing simulations. The Ω function can then be used to define the observation model $P(o|s, a)$, as follows:

$$P(\phi, c|w, \tau(e)) = \mathcal{N}(\Omega_\tau(w, e), \Sigma) \quad .$$

The state of the world is w, the action is the trajectory $\tau_i(e)$, and the nominal observation $\Omega_\tau(w, e)$. We then assert that the probability of making some particular observation of local sensors c when the arm is at a pose with joint angles ϕ is a Gaussian distribution about that nominal observation with some fixed covariance. Having computed the nominal observations in advance means that the observation probabilities required for the on-line belief-state update can be calculated with little additional work.

The belief-state update operation is, in the worst case, quadratic in the size of the state space; but in our case the transition distribution is very sparse, so there are a bounded number of states s' such that $\Pr(s'|s, a)$ is non-zero, which makes the complexity ultimately linear in the size of the state space instead. Furthermore, we are assuming that the belief state is also quite sparse (due to initial sensor information), so the complexity is further reduced considerably, making this operation straightforward to compute online.

3.4 Robust world-relative trajectory execution

Consider the execution of a single WRT with the goal of grasping the object and let the system be in initial belief state b. Now, let $w^*(b)$ be the world state w for which $b(w)$ is maximized; it is the most likely state. Then, $\tau(w^*(b))$ is a sequence of poses in robot coordinates. 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 an empty grasp with no contact) results in an observation that can be used to update the belief state. Then we will execute $\tau(w^*(b))$ again, but this time with respect to a new $w^*(b)$.

More formally, we can describe the algorithm in pseudocode:

while not $G_\delta(\phi, b)$ **do**
  $traceBack(\phi_h)$
  $(\phi, c) = guardedExecute(\tau(w^*(b)))$
  $b = UB(b, \tau(w^*(b)), \langle \phi, c \rangle)$
end while

The *guardedExecute* command follows the trajectory by executing a sequence of guarded moves between the waypoints, interpolating linearly between joint coordinates, but stops if a contact is sensed before the waypoint is reached. When the guardedExecute command terminates, it returns two quantities: ϕ is the robot's current pose in joint angles and c is a description of the information from the local sensors, if a contact occurred (could be forces, torques, locations and/or normals of surface contacts, etc.). The values of ϕ and c are then used 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 model effectively), relativize τ to the new most likely world state, and re-execute.

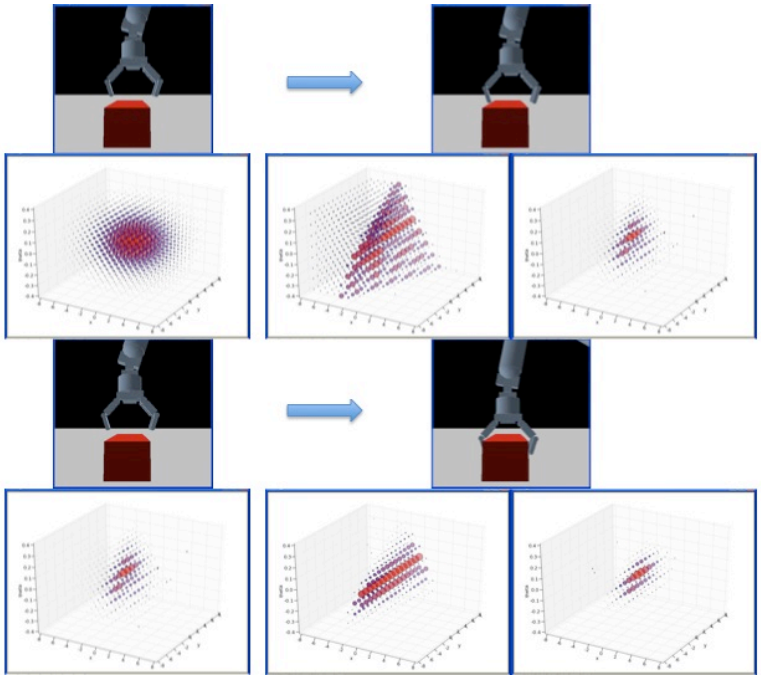

Fig. 2. Execution of WRT and (x, y, θ) belief state update.

Figure 2 shows the operation of the system while grasping a rectangular box using a single WRT. The belief state images depict probabilities via the

radius of the balls shown at grid points on the (x, y, θ) state space of the box. The first figure in the second row shows an initial belief state. The robot attempts to execute a grasping trajectory, relative to the most likely element of that belief state. The first figure in the top row shows the robot at the first via-point 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). Now, the hand executes a guarded move toward the next waypoint in the trajectory, and it is terminated by a fingertop contact, as shown in the second figure in the top row. The middle figure in the second row shows the probabilities 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, 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. This most basic strategy can be effective in many situations, as demonstrated in the experimental results. However, there are many situations in which it may fail. The two main problems are kinematic infeasibility and lack of information.

3.5 Multiple goal-seeking trajectories

Our premise is that there is, *a priori*, a large set $\mathcal{W}$ of possible world configurations and that we don't know which ones we will encounter in the online phase. For many tasks, there will be no single WRT τ that can effectively achieve the goal independent of where the object is in the workspace. Kinematic constraints arising from the boundaries of the workspace may render execution of $\tau(w)$ simply infeasible for some values of w. In addition, it may be that, even when it is executed in the appropriate world state (that is that $\tau(w)$ is executed in w) a collision will result, due to the fact that the robot is not, in fact, a disembodied hand. In such situations, it will be necessary to have a policy π made up of a set of WRTs that effectively "cover" the space of possible world configurations. Given several possible WRTs, and a current belief state b, how can we choose which one to execute? Intuitively, we would like to execute a trajectory that will result in the goal condition being satisfied. We define the *belief pre-image* of a goal condition G_δ with respect to the execution of trajectory $\tau(e)$ as

$$BPre(G_\delta, \tau(e)) = \{b \mid P_b(G(R(\tau(e), w))) > 1 - \delta\} \ .$$

That is, the set of belief states b such that, according to the probability measure $b(w)$, the likelihood that G will be true in the state resulting from executing $\tau(e)$ is greater than $1 - \delta$.

So, if there is a trajectory $\tau(e)$ such that $b \in BPre(G_\delta, \tau(e))$, then we would expect that executing $\tau(e)$ would result in a belief state that satisfies the goal condition. Generally speaking, the space of e may be too large to search exhaustively, and the WRTs are designed to work most effectively when executed relative to the correct world state, so we will restrict our attention to $\tau(w^*(b))$ for all $\tau \in \pi$. Since we have computed R off-line and we assume we have access to G, testing whether the current b is in the belief pre-image of the goal under $\tau(w^*(b))$ is relatively easy.

An additional consideration is the possibility that executing a particular trajectory will generate an undetectable contact. This is a highly undesirable situation that will probably move or knock over the object, requiring a major intervention. We can define the failure probability of trajectory $\tau(e)$ in b as:

$$FP(\tau(e), b) = P_b(R(\tau(e), w) = fail) \ ,$$

and use it to break ties among multiple WRTs for which b is in the belief pre-image of the goal.

3.6 Explicit information gathering

The execution process described in the previous section will succeed if the trajectories result in local sensory observations that provide enough information to update the belief state so that it is eventually concentrated around the correct world state. However, this will not necessarily happen. For example, consider the final belief state shown in figure 2. 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, and 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 trajectories that are designed explicitly to reduce uncertainty in the belief state, but not to achieve the ultimate desired world configuration.

If we are to choose a trajectory for its information-gathering properties, we need to be able to evaluate its expected effect on the belief state. It is very difficult to predict the effects of repeated execution of a trajectory, so we will model only the information gained as a result of the first contact that can be expected to result from executing the trajectory.

Concretely, assume we are currently in belief state b, and are considering executing some WRT τ. In order to make that trajectory concrete, we have to execute it relative to some world configuration. We might select $w^*(b)$ as the conjectured world, as we do for goal-oriented trajectories. Doing so determines a row in the observation function, $\Omega_\tau(\cdot, w^*(b))$, which specifies the expected

observation as a function of the true world state in which $\tau(w^*(b))$ is executed. Now, we can consider the effects that such observations would have on the belief state. Intuitively, we would like to select a trajectory that will result in observations that disambiguate among world configurations that are likely in the current belief state.

So, we define the *information gain* of a WRT τ in belief state b as follows:

$$IG(\tau, b) = H(b) - \sum_j b(w_j) H(UB(b, \tau(w^*(b)), \Omega_\tau(w_j, w^*(b)))) \ ,$$

where H is the entropy of a probability distribution and UB is the belief-state update function, that computes a new belief state based on an old belief state, an action, and a new observation. Intuitively, we are comparing the entropy of the current belief state with the expected entropy of the belief state that will result from incorporating the observation made due to executing the trajectory. Here is pseudo-code for the control algorithm that takes information-gathering actions into account:

while not $G_\delta(\phi, b)$ **do**
  $traceBack(\phi_h)$
  if there is a τ^* such that $b \in BPre(G_\delta, \tau^*(w^*(b)))$ **then**
    $(\phi, c) = guardedExecute(\tau^*(w^*(b)))$
  else
    $\tau^* = \arg\max_\tau (IG(\tau, b) - cFP(\tau, b))$
    $(\phi, c) = guardedExecute(\tau^*(w^*(b))$
  end if
  $b = UB(b, \tau(w^*(b)), \langle \phi, c \rangle)$
end while

This algorithm has the same basic structure as before, but it checks to see whether one of the WRTs, relativized to the current most likely state, is in the belief pre-image of the goal. If so, then it executes that WRT. If not, then it executes the WRT that optimizes some linear combination of information gain and failure probability (the weighting constant, c, embodies a measure of the risk tolerance of the system).

3.7 Off-line generation of world-relative trajectories

The previous section provides an on-line strategy for selecting and executing WRTs from a set that has been provided for on-line execution. How can we generate a good set of WRTs? We divide the problem into generating WRTs for goal achievement and for information gain.

Generating reasonable goal-oriented trajectories is a task-dependent problem. It could be done by explicit teaching or using a general grasp planner [2] to generate target grasps and calling a robot motion planner to generate trajectories that achieve the grasps. Then, re-expressing the trajectories in

world-relative coordinates will allow them to be used relative to other world configurations in which they are kinematically feasible.

Generating trajectories this way has two potential weaknesses, however: they may not generalize well to world configurations other than the ones for which they were planned; and they may have very small domains, due to having tight clearances with respect to objects in the world or being near workspace limits. It will be part of future work to formulate strategies for putting aspects of these criteria into a planner used to search for trajectories. Clearances can be increased by, for example, selectively increasing the size of the robot. In the experiments reported below, we use a single goal-oriented trajectory that was hand-constructed.

Information-gain trajectories can be constructed automatically by considering different major surfaces of the objects in the world and planning trajectories to touch those surfaces; or, to find face pairs that are graspable and to construct trajectories to grasp them. Each of these trajectories can be expected to reduce uncertainty in some dimensions, and we can use the information-gain metric to select among appropriate ones on-line. In our implementation, information-seeking trajectories were limited to grasping face pairs; it will also be useful to have trajectories that contact individual surfaces but we have not yet implemented this. We used the following procedure: (1) Process an object model to find pairs of nearly parallel surfaces on the object whose mean distance does not exceed the hand opening. (2) Generate a set of evenly-spaced target grasp points on the surfaces. (3) Define target hand frames that (a) place the finger tips at these grasp points, (b) have the fingers roughly parallel to the grasp surfaces and (c) have one of 8 evenly spaced approach angles. (4) Attempt to find a collision-free trajectory to the specified hand frames, for a single placement of the target object, using the default planners in the OpenRave motion planning system [8]. (Each trajectory took at most a few tens of seconds to plan.) The resulting trajectories become candidate WRTs, which were then evaluated on the basis of their kinematic robustness as well as their potential to gain information as well as to fail by knocking the object, by computing their associated feasibility, observation, and result functions. The two leftmost frames in Figure 3 shows two different automatically generated grasps on an elongated box; both have quite high information gain scores starting from a uniform belief (no prior information): 4.95 for the one on the left and 4.44 for the one on the right. The key difference between them can be seen when we consider other belief states. The grasp on the left, since it's at one end of the box, gives excellent information about x displacements (narrow dimension of the box) and some information about y displacements (long dimension of the box). The grasp on the left gives the same information about x displacements but it gives almost no information about y displacements in the relevant range. The information gain scores for a belief in which y is known but x and θ are uncertain are nearly the same: 4.05 and 4.03. In a belief state where x is known but y and θ are uncertain, the scores are: 4.61 and 3.97.

4 Implementation and experiments

We have implemented an initial version of this system, in simulation. The robot is a 7-DOF Barrett Arm with a Barrett Hand. It is picking objects of known shape from a table of known height. Each finger has contact sensors and an individual force-torque sensor at the finger tips.

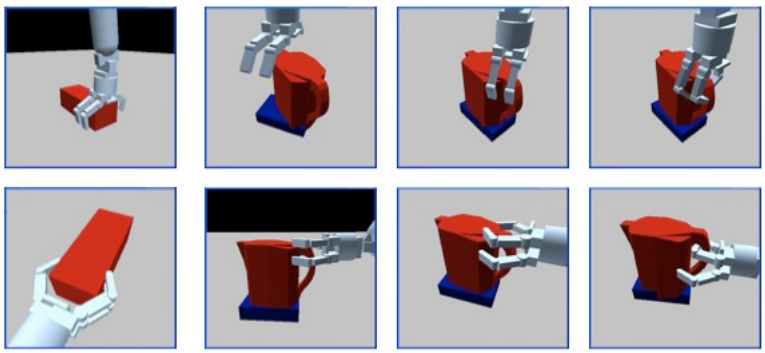

Fig. 3. Top and bottom left: information-gathering grasps for a box; Rest of frames are grasps for the pitcher; dark blue box shows most likely state. Top left: Initial state. Top middle: first execution of information gathering grasp. Top right: second execution of same grasp. Bottom left: a second information gathering grasp. Bottom middle: a second execution of same grasp. Bottom right: final grasp

The 6 rightmost frames in Figure 3 show three different grasps on a Brita pitcher. The first two grasps are information-gathering grasps, aimed at reducing the spread of the belief state; these grasp trajectories were planned automatically as described earlier. The last grasp is the intended grasp, which was chosen by hand; two of the fingers are being used to avoid rotation about the handle.

To test the effectiveness of our robust-execution approach, we did a large number (at least 100 for each setting) of simulation runs using several variants of the approach for four different levels of uncertainty: 1 cm standard deviation in x and y and 3 degrees in θ, 3 cm and 9 degrees, 5 cm and 15 degrees, and 5 cm and 30 degrees. The results are shown in Figure 4.

We tried the following 5 control algorithms:

- *Best WRT, open loop, with perfect info*: robot knows exactly where the object is when it carries out the nominal WRT, for the true state. The only failures are due to positions of the pitcher that lead to hand positions outside of the robot workspace. This is an upper bound on performance.
- *Best WRT, open loop, with imperfect info*: robot executes nominal WRT, for the initially most likely state, open-loop. This is a lower bound on performance.

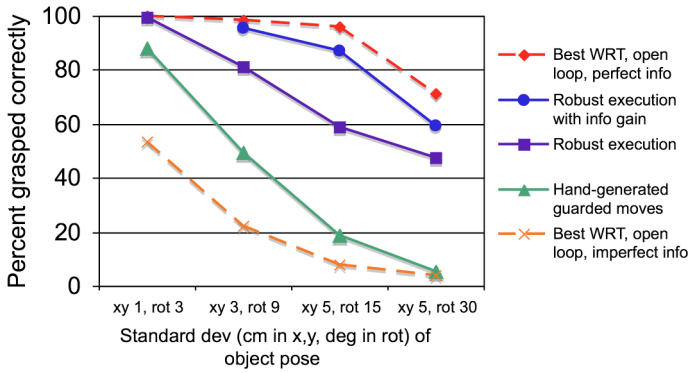

Fig. 4. Results for the Brita pitcher.

- *Hand-generated guarded moves*: robot executes a hand-generated, fixed, guarded-move trajectory (move -x 20 cm or until touch, back off 1 cm, move +y 7 cm or until touch, back off .65 cm, move right 2 cm or until touch, close fingers).
- *Robust execution*:, repeated robust execution of the nominal WRT relative to the most likely state (algorithm in section 3.4).
- *Robust execution with info gain*: Algorithm from section 3.6 with two possible information-gathering grasps (the second one shown in Figure 3 and another grasp).

We can see that the performance of the control system based on WRTs with information gathering is very near the optimal possible performance for a stationary arm on this problem.

The approach presented in this paper enables robust execution of manipulation programs with no on-line planning. There are a number of extensions to the basic approach that should be explored. The offline planning strategy should be extended to be more cognizant of the needs of on-line execution: kinematic robustness, avoiding undetectable contacts and information gain, e.g, as in [20]. Ultimately, we should consider some mix of on-line and off-line planning. In addition, better results may be obtained by doing a few steps of lookahead to select information-gain actions, particularly to enable gaining information that is relevant to the specific goal of the system.

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

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

2. D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner. Grasp planning in complex scenes. In *Humanoids07*, December 2007.
3. F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte. Information based adaptive robotic exploration. In *IROS*, 2002.
4. R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A computational framework. *IJRR*, 15(1):1–23, 1996.
5. B. Burns and O. Brock. Sampling-based motion planning with sensing uncertainty. *ICRA*, 2007.
6. A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *IROS*, 1996.
7. A. Censi, D. Calisi, A. D. Luca, and G. Oriolo. A bayesian framework for optimal motion planning with uncertainty. *ICRA*, pages 1798–1805, 2008.
8. R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, CMU, 2008.
9. L. H. Erickson, J. Knuth, J. M. O'Kane, and S. M. Lavalle. Probabilistic localization with a blind robot. *ICRA*, 2008.
10. K. Gadeyne, T. Lefebvre, and H. Bruyninckx. Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation. *IJRR*, 24:615, 2005.
11. J. P. Gonzalez and A. Stentz. Planning with uncertainty in position an optimal and efficient planner. *IROS*, pages 2435–2442, 2005.
12. K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez. Grasping pomdps. *ICRA*, 2007.
13. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.
14. H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *RSS*, 2008.
15. J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Mass, 1991.
16. S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *IROS*, pages 1772–1779, September 1994.
17. O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robot programming. *Auton. Robots*, 16(1):49–79, 2004.
18. T. Lozano-Pérez, M. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *IJRR*, 3(1), 1984.
19. N. A. Melchior and R. Simmons. Particle rrt for path planning with uncertainty. *ICRA*, 2007.
20. P. Michel, C. Scheurer, J. Kuffner, N. Vahrenkamp, and R. Dillmann. Planning for robust execution of humanoid motions using future perceptive capability. In *IROS)*, pages 3223–3228, October 2007.
21. A. Petrovskaya and A. Y. Ng. Probabilistic mobile manipulation in dynamic environments, with application to opening doors. *IJCAI*, 2007.
22. S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. *ISRR*, 2007.
23. R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, pages 1080–1087, 1995.
24. T. Smith and R. G. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *UAI*, pages 542–547, 2005.
25. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.