

Pre-image backchaining in belief space for mobile manipulation

Leslie Pack Kaelbling and Tomás Lozano-Pérez

1 Introduction

As robots become more physically robust and capable of sophisticated sensing, navigation, and manipulation, we want them to carry out increasingly complex tasks. A robot that helps in a household must plan over the scale of hours or days, considering abstract features such as the desires of the occupants of the house, as well as detailed geometric models that support locating and manipulating objects. The complexity of such tasks derives from very long time horizons, large numbers of objects to be considered and manipulated, and fundamental uncertainty about properties and locations of those objects.

We have developed an approach to integrated task and motion planning that integrates geometric and symbolic representations in an aggressively hierarchical planning architecture, called HPN [10]. The hierarchical decomposition allows efficient solution of problems with very long horizons and the symbolic representations support abstraction in complex domains with large numbers of objects and are integrated effectively with the detailed geometric models that support motion planning. In this paper, we extend the HPN approach to handle two types of uncertainty: *future-state* uncertainty about what the outcome of an action will be, and *current-state* uncertainty about what the current state actually is. Future-state uncertainty is handled by planning in approximate deterministic models, performing careful execution monitoring, and replanning when necessary. Current-state uncertainty is handled by planning in *belief space*: the space of sets or probability distributions over possible underlying world states.

There have been several recent approaches to integrated task and motion planning [3, 14, 13] but they do not address uncertainty. The use of belief space (and information spaces [11]) is central to decision-theoretic approaches to uncertain domains [9].

Leslie Pack Kaelbling · Tomás Lozano-Pérez
CSAIL, MIT, Cambridge, MA 02139 e-mail: {lpk, tlp}@csail.mit.edu

2 Hierarchical pre-image backchaining

Most planning problems require time on the order of the minimum of: $|S|$ and $|A|^h$, where S is the size of the state space, $|A|$ is the size of the action space and h is the horizon (the length of the solution). Our approach to making planning tractable is to construct a temporal hierarchy of short-horizon problems, thus reducing the complexity of the individual planning problems we have to solve. The hierarchical approach will not always produce optimal plans; it is, however, complete in domains for which the goal is reachable from every state.

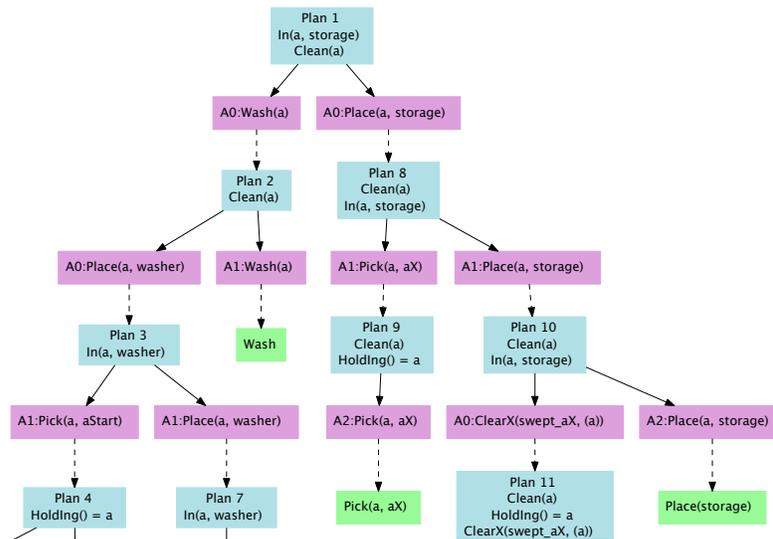


Fig. 1 Example of part of a hierarchical plan and execution tree.

We formalize the effects of the robot’s actions in a hierarchy of increasingly abstract operator descriptions; this hierarchy is constructed by postponing the consideration of preconditions of an action until more concrete levels of abstraction. Figure 1 shows part of a hierarchical plan for washing an object and putting it into storage. The operations are first considered abstractly, making a two-step, high-level plan. Then, the plan to wash the object is elaborated into two steps, of placing the object into the washer and then washing it. At the next level of abstraction down, we plan first to pick the object, and then to place it in its destination.

Even short-horizon plans are difficult to find when the branching factor is high. Searching forward from an initial, completely detailed, state description typically has a very high (or infinite) branching factor, and it can be difficult to prune actions heuristically and maintain the effectiveness of the planner. We address this problem by using *pre-image backchaining* [12], also known as *goal regression* [21], to search backward from the goal description. We presume that the goal has a simple abstract

description, for example, requiring *can A* to be in *cupboard B*. That description denotes an infinite set of possible world states (varying according to the exact position of *can A*, the robot’s pose, the locations and states of other objects in the world, the weather, etc.). Chaining backward from the goal, using symbolic operator descriptions together with the goal-driven construction of geometric preconditions, allows planning to proceed with a relatively low branching factor. Plan 10 in figure 1 shows a case in which, in order to place an object in storage, the planner determines that it must first clear out the region of space through which the object must be moved, and then place the object. A geometric description of that region of space was computed on demand to specify the pre-image of placing the object into storage.

In the next two sections, we describe how to extend the HPN approach to handle uncertainty. First, we consider the case in which the robot knows its current state, but in which the world dynamics are stochastic. Then, we extend the method to the case in which there is uncertainty about the current state of the world.

3 Future-state uncertainty

The decision-theoretic optimal approach to planning in domains with probabilistic dynamics is to make a *conditional plan*, in the form of a tree, supplying an action to take in response to any possible outcome of a preceding action [22]. For efficiency and robustness, our approach to stochastic dynamics is to construct a deterministic approximation of the dynamics, use the approximate dynamics to build a plan, execute the plan while perceptually monitoring the world for deviations from the expected outcomes of the actions, and replan when deviations occur. This method has worked well in control applications [4, 15, 20, 6] as well as symbolic planning domains [23].

Determinization There are several potential strategies for constructing a determinized model. A popular approach is to assume, for the purposes of planning, that the most likely outcome is the one that will actually occur. This method can never take advantage of a less-likely outcome of an action, even if it is the only way to achieve a goal. We pursue an alternative method, which considers all possible outcomes, but rather than modeling them as a randomized choice that is made by nature, instead modeling them as a choice that can be made by the agent [1]. This method integrates the desire to have a plan with a high success probability with the desire to have a plan with low action cost by adopting a model where, when an undesirable outcome happens, the state of the world is assumed to stay the same, allowing the robot to repeat that action until it has the desired result. If the desired outcome has probability p of occurring and the cost of taking the action is c , then in this model the expected cost to make the transition to the desired state is c/p . We will search for the plan that has the least cost under this model.

Interleaved planning and execution The planning and execution process can be thought of as a depth-first tree traversal. Figure 1 illustrates this: The blue nodes

represent planning problems at different levels of abstraction. The abstraction hierarchy is not rigid: it is constructed on the fly as the structure of the problem demands. Purple nodes are operations at different levels of abstraction, and green nodes are primitive actions. Note that, for instance, the primitive actions in the sub-tree for plan 2 are executed before plan 8 is constructed. This online, interleaved planning allows the details of the construction of plan 8 to depend on the concrete state of the world that came about as a result of the recursive planning and execution of plan 2.

Assume we have a PLAN procedure, which takes as arguments *state*, the current world state, *goal*, a description of the goal set and *abs* a description of the level of abstraction at which planning should take place; it additionally depends on a set of operator descriptions that describe the domain dynamics. The PLAN procedure returns a list $((-, g_0), (a_1, g_1), \dots, (a_n, g_n))$ where the a_i are operator instances, $g_n = \text{goal}$, g_i is the pre-image of g_{i+1} under a_i , and $\text{state} \in g_0$. The pre-image g_i is the set of world states such that if the world is in some state g_i and action a_{i+1} is executed, then the next world state will be in g_{i+1} ; these pre-images are subgoals that serve as the goals for the planning problems at the next level down in the hierarchy.

The HPN process is invoked by $\text{HPN}(\text{state}, \text{goal}, \text{abs}, \text{world})$, where *state* is a description of the the current state of world; *goal* is a set of world states; *abs* is a structure that specifies, for any goal condition, the number of times it has served as a plan step in the HPN call stack above it; and *world* is an actual robot or a simulator in which primitive actions can be executed. In the prototype system described in this paper, *world* is actually a geometric motion planner coupled with a simulated or physical robot.

```

HPN(state, goal, abs, world):
  p = PLAN(state, goal, abs)
  for (ai, gi) in p
    while state ∈ gi-1 and not state ∈ gi
      if ISPRIM(ai)
        state = world.EXECUTE(ai)
      else
        HPN(state, gi, NEXTLEVEL(abs, ai), world)
    if not state ∈ gi return

```

HPN starts by making a plan p to achieve the top-level goal. Then, it executes the plan steps, starting with action, a_1 . Each plan step is executed repeatedly, until either its desired post-condition, g_i , holds in the environment, which means that the execution has been successful, or until its pre-condition, g_{i-1} , ceases to hold in the environment, which means that the suffix of the plan starting with this step can no longer be expected to achieve the goal. If the pre-condition becomes false, then execution of the plan at this level is terminated and control is returned to the level of abstraction above.

This process will re-plan and re-execute actions until the goal is reached, as long as the goal is reachable from every state in the determinized model.

4 Pre-image backchaining in belief space

When there is uncertainty about the current state of the world, we plan in the space of beliefs about the world state, instead of the space of world states itself. In this work, we use probability distributions over world states as belief states. Planning in this space enables actions to be selected because they cause the robot to gain information that will enable appropriate physical actions to be taken later in the plan, for instance.

Planning in belief space is generally quite complex, because it seems to require representing and searching for trajectories in a very high-dimensional continuous space of probability distributions. This is analogous to the problem of finding plans in very high-dimensional continuous space of configurations of a robot and many objects. We take direct advantage of this analogy and use symbolic predicates to specify limited properties of belief states, as our previous approach [10] does for properties of geometric configurations. So, for instance, we might characterize a set of belief states by specifying that “the probability that the cup is in the cupboard is greater than 0.95.” Pre-image backchaining allows the construction of high-level plans to achieve goals articulated in terms of those predicates, without explicitly formalizing the complete dynamics on the underlying continuous space.

Traditional belief-space planning approaches either attempt to find entire policies, mapping all possible belief states to actions [19, 9, 17] or perform forward search from a current belief state, using the Bayesian belief-update equation to compute a new belief state from a previous one, an action and an observation [16]. In order to take advantage of the approach outlined above to hierarchical planning and execution, however, we will take a pre-image backchaining approach to planning in belief space.

HPN in belief space The basic execution strategy for HPN need not be changed for planning in belief space. The only amendment, shown below, is the need to perform an update of the belief state based on an observation resulting from executing the action in the world:

```

BHPN(belief, goal, abs, world):
  p = PLAN(belief, goal, abs)
  for (ai, gi) in p
    while belief ∈ gi-1 and not belief ∈ gi
      if ISPRIM(ai)
        obs = world.EXECUTE(ai)
        belief.UPDATE(ai, obs)
      else
        BHPN(belief, gi, NEXTLEVEL(abs, ai), world)
    if not belief ∈ gi return

```

After each primitive action is executed, an observation is made in the world and the belief state is updated to reflect both the predicted transition and the information contained in the observation *obs*. It is interesting to note that, given an action

and an observation, the belief state update is deterministic. However, the particular observation that will result from taking an action in a state is stochastic; that stochasticity is handled by the *BHPN* structure in the same way that stochasticity of action outcomes in the world was handled in the *HPN* structure.

Hierarchical planning and information gain fit together beautifully: the system makes a high-level plan to gather information and then uses it, and the interleaved hierarchical planning and execution architecture ensures that planning that depends on the information naturally takes place after the information has been gathered.

Symbolic representation of goals and subgoals

When planning in belief space, goals must be described in belief space. Example goals might be “With probability greater than 0.95, the cup is in the cupboard.” or “The probability that more than 1% of the floor is dirty is less than 0.01.” These goals describe *sets* of belief states. The process of planning with pre-image backchaining computes pre-images of goals, which are themselves sets of belief states. Our representational problem is to find a compact yet sufficiently accurate way of describing goals and their pre-images.

In traditional symbolic planning, *fluents* are logical assertions used to represent aspects of the state of the external physical world; conjunctions of fluents are used to describe sets of world states, to specify goals, and to represent pre-images. States in a completely symbolic domain can be represented in complete detail by an assignment of values to all possible fluents in a domain. Real world states in robotics problems, however, are highly complex geometric arrangements of objects and robot configurations which cannot be completely captured in terms of logical fluents. However, logical fluents can be used to characterize the domain at an abstract level for use in the upper levels of hierarchical planning.

We will take a step further and use fluents to characterize aspects of the robot’s *belief state*, for specifying goals and pre-images. For example, the condition “With probability greater than 0.95, the cup is in the cupboard,” can be written using a fluent such as $PrIn(cup, cupboard, 0.95)$, and might serve as a goal for planning. For any fluent, we need to be able to test whether or not it holds in the current belief state, and we must be able to compute the pre-image of a set of belief states described by a conjunction of fluents under each of the robot’s actions. Thus, our description of operators will not be in terms of their effect on the state of the external world but in terms of their effect on the fluents that characterize the robot’s belief. Our work is informed by related work in partially observed or probabilistic regression (back-chaining) planning [2, 5, 18]. In general, it will be very difficult to characterize the exact pre-image of an operation in belief space; we will strive to provide an approximation that supports the construction of reasonable plans and rely on execution monitoring and replanning to handle errors due to approximation.

In the rest of this paper, we provide examples of representations of belief sets using conjunctions of logical fluents, for both discrete and continuous domains, and illustrate them on example robotics problems.

5 Coarse object pose uncertainty

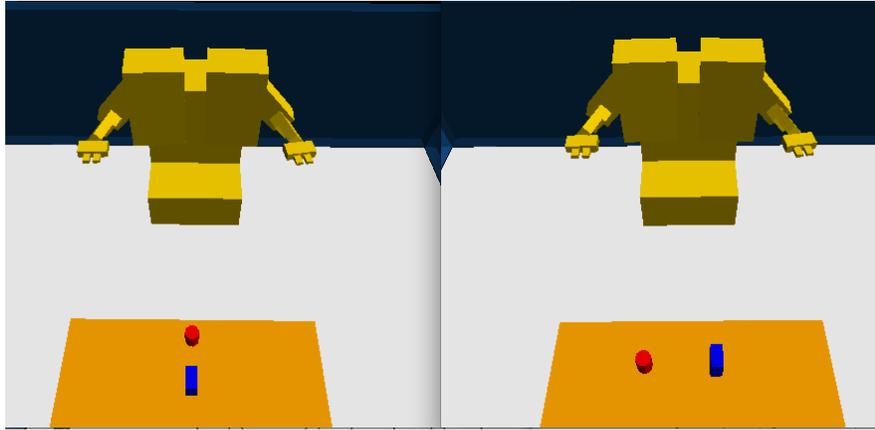


Fig. 2 The situation on the left is the real state of the world; the one on the right represents the mode of the initial belief.

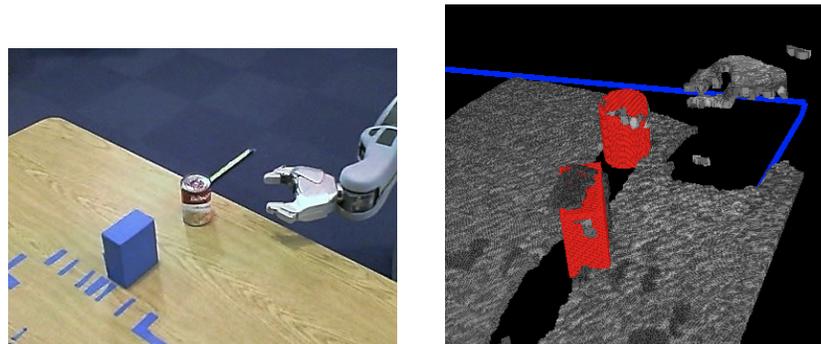


Fig. 3 Point cloud (on right) for scene (on left); red points correspond to the models at their perceived poses.

We have a pilot implementation of the HPN framework on a Willow Garage PR2 robot, demonstrating integration of low-level geometric planning, active sensing, and high-level task planning, including reasoning about knowledge and planning to gain information. Figure 2 shows a planning problem in which the robot must move the blue box to another part of the table. The actual state of the world is shown on the left, and the mode of its initial belief is shown on the right.

Objects in the world are detected by matching known object meshes to point clouds from the narrow stereo sensor on the robot; example detections are shown

in figure 3. As with any real perceptual system, there is noise in both the reported poses and identities of the objects. Furthermore, there is significant noise in the reported pose of the robot base, due to wheel slip and other odometric error. There is additional error in the calibration of the stereo sensor and robot base. The state estimation process for the positions of objects is currently very rudimentary: object detections that are significantly far from the current estimate are rejected; those that are accepted are averaged with the current estimate. A rough measure of the accuracy of the estimate is maintained by counting the number of detections made of the object since the last time the robot base moved. A detailed description of the geometric representation and integration of task and motion planning is available [10]. Here, we emphasize uncertainty handling in the real robot. Here are three of the relevant operator descriptions:

PICK(O, M) : $KHolding = O$
pre: $KClearX(PickSwept(M), O) = T$,
 $KCanPickFrom(O, M) = T, KHolding = NOTHING$,
 $KRobotNearLoc(M) = T, LocAccuracy(O, M, 3) = T$

LOOKAT(O, M, N) : $LocAccuracy(O, M, N) = T$
pre: $KRobotNearLoc(M) = T, LocAccuracy(O, M, N - 1) = T$

MOVEBASE(M) : $KRobotNearLoc(M) = T$
pre: $KClearX(MoveSwept(M), ()) = T$
sideEffect: $LocAccuracy(O, M', N) = F$

The **PICK** operator describes conditions under which the robot can pick up an object O , making the fluent $KHolding$ have the value O . The fluent name, $KHolding$, is meant to indicate that it is a condition on belief states, that is, that the robot *know* that it is holding object O . The primitive pick operation consists of (1) calling an RRT motion planner to find a path for the arm to a 'pregrasp' pose, (2) executing that trajectory on the robot, (3) grasping the object, then (4) lifting the object to a 'postgrasp' pose.

The operator description has a free variable M , which describes a trajectory for the robot base and arm, starting from a home pose through to a pose in which the object is grasped; one or more possible values of M are generated by a procedure that takes constraints on grasping and motion into account and uses an efficient approximate visibility-graph motion planner for a simple robot to find a path. This is not the exact path that the motion primitives will ultimately execute, but it serves during the high-level planning to determine which other objects need to be moved out of the way and where the base should be positioned while performing the pick operation. The preconditions to the primitive pick operation are that: the swept volume of the path for the arm be known to be clear, with the exception of the object to be picked up; that the object O be known to be in a pose that allows it be picked up when the robot base pose is the one specified by M ; that the robot is known not to be currently holding anything; that the robot base is known to be near the pose specified by M , and that the pose of the object O is known with respect to the robot's

base pose in M with accuracy level 3 (that is, it has to have had at least three separate good visual detections of the object since the last time the base was moved).

The LOOKAT operator looks at an object O from the base pose specified in motion M , and can make the fluent $LocAccuracy(O, M, N)$ true, which means that the accuracy of the estimated location of object O is at least level N . The primitive operation computes a head pose that will center the most likely pose of object O in its field of view when the robot base is in the specified pose and moves the head to that pose. The image-capture, stereo processing, and object-detection processes are running continuously and asynchronously, so the primitive does not need to explicitly call them. This operation achieves a location accuracy of N if the base is known to be in the appropriate pose and the object had been previously localized with accuracy $N - 1$.

The MOVEBASE operator moves to the base pose specified in motion M . The primitive operation (1) calls an RRT motion planner to find a path (in the full configuration space of the base and one arm—the other arm is fixed) to the configuration specified in M and (2) executes it. It requires, as a precondition, that the swept volume of the suggested path be known to be clear. Importantly, it also declares that it has the effect of invalidating the accuracy of the estimates of all object poses.

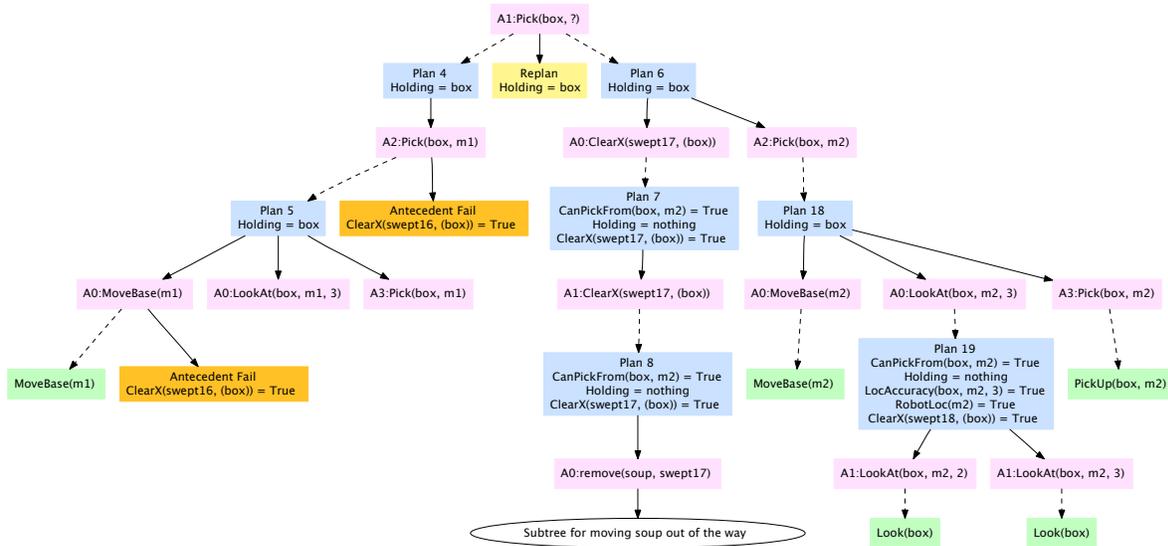


Fig. 4 Partial planning and execution tree for picking up the box, in which the robot notices that the soup can is in the way and removes it.

Figure 4 shows a fragment of the planning and execution tree resulting from an execution run of the system. At the highest level (not shown) a plan is formulated to

pick up the box and then place it in a desired location. This tree shows the picking of the box. First, the system makes an abstract plan (Plan 4) to pick the box, and then refines it (Plan 5) to three operators: moving the base, looking at the box until its location is known with sufficient accuracy, and then executing the pick primitive. The robot base then moves to the desired pose (green box); once it has moved, it observes the objects on the table and updates its estimate of the poses of all the objects. In so doing, it discovers that part of the precondition for executing this plan has been violated (this corresponds to the test in the last line of code for BHPN) and returns to a higher level in the recursive planning and execution process. This is indicated in the planning and execution tree by the orange boxes, showing that it fails up two levels, until the high-level PICK operation is planned again. Now, Plan 6 is constructed with two steps: making the swept volume for the box clear, and then picking up the box. The process of clearing the swept volume requires picking up the soup can and moving it out of the way; this process generates a large planning and execution tree which has been elided from the figure. During this process, the robot had to move the base. So Plan 18 consists of moving the base to an appropriate pose to pick up the box, looking at the box, and then picking it up. Because the robot was able to visually detect the box after moving the base, the LOOKAT operation only needs to gather two additional detections of the object, and then, finally, the robot picks up the box.

6 Fine object pose uncertainty

In the previous example, we had a very coarse representation of uncertainty: the robot's pose was either known sufficiently accurately or it wasn't; the degree of accuracy of a position estimate was described by the number of times it had been observed. For more fine-grained interaction with objects in the world, we will need to reason more quantitatively about the belief states. In this section, we outline a method for characterizing the information-gain effects of operations that observe continuous quantities in the environment. We then illustrate these methods in a robotic grasping example.

Characterizing belief of a continuous variable

We might wish to describe conditions on continuous belief distributions, by requiring, for instance, that the mean of the distribution be within some value of the target and the variance be below some threshold. Generally, we would like to derive requirements on beliefs from requirements for action in the physical world. So, in order for a robot to move through a door, the estimated position of the door needs to be within a tolerance equal to the difference between the width of the robot and the width of the door. The variance of the robot's estimate of the door position is not the best measure of how likely the robot is to succeed: instead we will use the concept of the *probability near mode* (PNM) of the distribution. It measures the amount of probability mass within some δ of the mode of the distribution. So, the robot's pre-

diction of its success in going through the door would be the PNM with δ equal to half of the robot width minus the door width.

For a planning goal of $PNM(X, \delta) > \theta$, we need to know expressions for the regression of that condition under the a and o in our domain. In the following, we determine such expressions for the case where the underlying belief distribution on state variable X is Gaussian, the dynamics of X are stationary, a is to make an observation, and the observation o is drawn from a Gaussian distribution with mean X and variance σ_o^2 .

For a one-dimensional random variable $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$PNM(X, \delta) = \Phi\left(\frac{\delta}{\sigma}\right) - \Phi\left(\frac{-\delta}{\sigma}\right) = \operatorname{erf}\left(\frac{\delta}{\sqrt{2}\sigma}\right),$$

where Φ is the Gaussian CDF. If, at time t the belief is $\mathcal{N}(\mu_t, \sigma_t^2)$, then after an observation o , the belief will be

$$\mathcal{N}\left(\frac{\mu_t \sigma_o^2 + o \sigma_t^2}{\sigma_o^2 + \sigma_t^2}, \frac{\sigma_o^2 \sigma_t^2}{\sigma_o^2 + \sigma_t^2}\right).$$

So, if $PNM(X_t, \delta) = \theta_t = \operatorname{erf}\left(\frac{\delta}{\sqrt{2}\sigma_t}\right)$ then

$$PNM(X_{t+1}, \delta) = \theta_{t+1} = \operatorname{erf}\left(\frac{\delta}{\sqrt{2}} \sqrt{\frac{\sigma_o^2 + \sigma_t^2}{\sigma_o^2 \sigma_t^2}}\right)$$

Substituting in the expression for σ_t^2 in terms of θ_t , and solving for θ_t , we have:

$$\theta_t = PNMregress(\theta_{t+1}, \delta, \sigma_o^2) = \operatorname{erf}\left(\sqrt{\operatorname{erf}^{-1}(\theta_{t+1})^2 - \frac{\delta^2}{2\sigma_o^2}}\right).$$

So, to guarantee that $PNM(X_{t+1}, \delta) > \theta_{t+1}$ holds after taking action a and observing o , we must guarantee that $PNM(X_t, \delta) > PNMregress(\theta_{t+1}, \delta, \sigma_o^2)$ holds on the previous time step.

Integrating visual and tactile sensing for grasping

In general, visual sensing alone is insufficient to localize an object sufficiently to guarantee that it be grasped in a desired pose with respect to the robot's hand. Attempts to grasp the object result in tactile feedback that significantly improve localization. The Willow Garage reactive grasping ROS pipeline [8] provides a lightweight approach to using the tactile information, assuming that the desired grasp orientation is fixed relative to the robot (rather than the object). Other work [7] takes a belief-space planning approach to this problem. It constructs a fixed-horizon search tree (typically of depth 2), with branches on possible observations. That approach is effective, but computationally challenging due to (1) replanning on every step; (2) a large branching factor and (3) the fact that a completely detailed belief-state update must be computed at each node in the forward search tree.

In this section, we outline an approach of intermediate complexity. It makes a complete plan using a deterministic and highly abstract characterization of pre-images in belief space, which is very efficient. It can handle objects of more general shapes and different grasping strategies, and automatically trades off the costs and benefits of different sensing modalities.

Operator descriptions We have reformulated the *pick* operator from section 5, with the accuracy requirement for the location of O specified with the fluent $PNMLoc(O, \theta, \delta_p)$, which is true if and only if the probability that object O 's location is within δ_p of its most likely location is greater than θ . The definition of near is now on three-dimensional object poses with threshold δ_p , selected empirically.

PICK(O, M) : $KHolding = O$:

pre: $KClearX(PickSwept(M), O) = T$,

$KCanPickFrom(O, M) = T, KHolding = \text{NOTHING}$,

$KRobotNearLoc(M) = T, PNMLoc(O, \theta, \delta_p) = T$

cost: $1/\theta$

The threshold θ is a free parameter here. If δ_p is the amount of pose error that can be tolerated and still result in a successful grasp, then this operator says that if we know the object's location to within δ_p with probability θ , then we will succeed in grasping with probability θ . This success probability is reflected in the cost model.

We have two ways of acquiring information. The first is by looking with a camera. The field of view is sufficiently wide that we can assume it always observes the object, but with a relatively large variance, σ_{vision}^2 , in the accuracy of its observed pose, and hence a relatively small increase in the PNM.

LOOK(O, θ, δ) : $PNMLoc(O, \theta, \delta) = T$:

pre: $PNMLoc(O, PNMRegress(\theta, \delta, \sigma_{vision}^2), \delta) = T$

cost: 1

The second method of gaining information is by attempting to grasp the object. If the probability of the object's pose being within the hand's size of the mode is very low, then an attempt to grasp is likely to miss the object entirely and generate very little information. On the other hand, if the probability of being within a hand's size of the mode is high, so that an attempted grasp is likely to contact the object, then observations from the touch sensors on the hand will provide very good information about the object's position.

TRYGRASP(O, θ, δ) : $PNMDoorLoc(D, \theta, \delta) = T$:

pre: $KClearX(PickSwept(M), O) = T$,

$KCanPickFrom(O, M) = T, KHolding = \text{NOTHING}$,

$KRobotNearLoc(M) = T$

$PNMLoc(O, PNMRegress(\theta, \delta, \sigma_{tactile}^2), \delta) = T$,

$PNMLoc(O, \theta_{handSize}, handSize/2) = T$

cost: $1/\theta_{handSize}$

In this operator description, we use the *PNMRegress* procedure as if the information gained through an attempted grasp were the result of an observation with Gaussian error being combined with a Gaussian prior. That is not the case, and so this

is a very loose approximation; in future work, we will estimate this non-Gaussian PNM pre-image function from data.

Both the GRASP and TRYGRASP operations are always executed with the mode of the belief distribution as the target pose of the object. This planning framework could support other choices of sensing actions, including other grasps or sweeping the hand through the space; it could also use a different set of fluents to describe the belief state, including making distinctions between uncertainty in position and in orientation of the object.

State estimation In order to support planning in belief space, we must implement a state estimator, which is used to update the belief state after execution of each primitive action in *BHPN*. The space of possible object poses is characterized by x, y, θ , assuming that the object is resting on a known stable face on a table of known height: x and y describe positions of the object on the table and θ describes its orientation in the plane of the table.

The grasping actions are specified by $[targetObjPose, grasp]$, where $targetObjPose$ is a pose of the object and $grasp$ specifies the desired placement of the hand with respect to the object. An action consists of moving back to a pregrasp configuration with the hand open, then performing a guarded move to a target hand pose and closing the fingers. The fingers are closed compliantly, in that, as soon as one finger contacts, the wrist moves away from the contacting finger at the same time the free finger closes. The resulting two-finger contact provides a great deal of information about the object’s position and orientation. During the guarded move, if any of the tactile sensors or the wrist accelerometer is triggered, the process is immediately halted. In the case of an accelerometer trigger, the fingers are compliantly closed.

The observations obtained from grasping are of the form

$$[handPose, gripperWidth, trigger, contacts] ,$$

where $handPose$ is the pose of the robot’s hand (reported based on its proprioception) at the end of the grasping process; $gripperWidth$ is the width of the gripper opening; $trigger$ is one of $\{rightTip, leftTip, accel, None\}$ indicating whether the motion was stopped due to a contact from one of the fingertip sensors, the accelerometer, or was not stopped; and $contacts$ is a list of four Boolean values indicating whether each of four sensors (left finger tip, right finger tip, left internal pad, and right internal pad) is in contact.

In order to perform state estimation, we need to provide an observation model $\Pr(o|s, a)$, which specifies the probability (or density) of making a particular observation o if the object is actually in pose s and the robot executes the action specified by a . There is stochasticity in the outcome of commanded action a due to both to error in the robot’s proprioception and in its controllers, causing it to follow a trajectory t that differs from the one specified by a . So, we can formulate the observation probabilities as

$$\Pr(o|s, a) = \int_t \Pr(o|s, t) \Pr(t|s, a) .$$

Decomposing o into c (contacts and trigger) and h (hand-pose and gripper width), we can write

$$\begin{aligned}\Pr(o|s, a) &= \int_t \Pr(c, h|s, t) \Pr(t|s, a) \\ &= \int_t \Pr(c|s, t) \Pr(h|t) \Pr(t|s, a)\end{aligned}$$

However, the integral is too difficult to evaluate, so we approximate it by considering the trajectory t that maximizes the right hand side

$$\Pr(o|s, a) \approx \max_t \Pr(c|s, t) \Pr(h|t) \Pr(t|s, a) .$$

This, too, is difficult to compute; we approximate again by assuming that there is little or no error in the actual contact sensors, and so we consider the set of trajectories that would generate the contacts c if the object were at pose s , $\tau(c, s)$, so

$$\Pr(o|s, a) \approx \max_{\tau(c, s)} \Pr(h|t) \Pr(t|s, a) .$$

We consider the class of trajectories that are rigid displacements of the commanded trajectory a that result in contacts c , and such that no earlier point in the trajectory would have encountered an observed contact. We compute the configuration space obstacle for the swept volume of the gripper relative to the object placed at s . Facets of this obstacle correspond to possible observed or unobserved contacts. We first compute a_h , which is the displacement of a that goes through the observed hand pose h , and then seek the smallest displacement of a_h that would move it to $\tau(c, s)$; this is the smallest displacement δ that produces a contact in the relevant facet of the configuration-space obstacle. We assume that the displacements between actual and observed trajectories are Gaussian, so we finally approximate $\Pr(o|s, a)$ as $\mathcal{G}(\delta, 0, \sigma^2)$ where \mathcal{G} is the Gaussian PDF for value δ in a distribution with mean 0 and variance σ^2 .

The relationship between observations and states is highly non-linear, and the resulting state estimates are typically multi-modal and not at all well approximated by a Gaussian. We represent the belief state with a set of samples, using an adaptive resolution strategy to ensure a high density of samples in regions of interest, but also to maintain the coverage necessary to recover from unlikely observations and to maintain a good estimate of the overall distribution (not just the mode).

Results Using *BHPN* with the operator descriptions and state estimator described in this section, we constructed a system that starts with a very large amount of uncertainty and has the goal of believing with high probability that it is holding an object. Figure 5 shows an example planning and execution tree resulting from this system. It plans to first look, then do an attempted grasp to gain information, and then to do a final grasp. It executes the *Look* and the *TryGrasp* steps with the desired results in belief space, and then it does another grasp action. At that point, the confidence that the robot is actually grasping the object correctly is not high

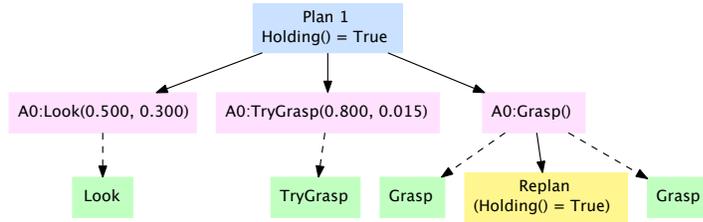


Fig. 5 Planning and execution tree for grasping, showing the initial plan, and a re-execution.

enough. However, the PNM precondition of the *Grasp* action still holds, so it replans locally and decides to try the *Grasp* action again, and it works successfully.

We compared three different strategies in a simulation of the PR2: using only the *TryGrasp* action to gain information, using only the *Look* action to gain information, and using a combination of the *Look* and *TryGrasp* actions, mediated by the planner and requirements on PNM. We found the following average number of actions required by each strategy:

- *TryGrasp* only: 6.80 steps
- *Look* only: 73 steps (estimated)
- *LookandTryGrasp*: 4.93 steps

The *Look* operation has such high variance that, although in the limit the PNM would be sufficiently reduced to actually grasp the object, it would take a very long time. The estimate of 73 is derived using the PNM regression formulas (and in fact the planner constructs that plan). The *TryGrasp*-only strategy works reasonably well because we limited the region of uncertainty to the workspace of the robot arm, and it can reasonably easily rule out large parts of the space. Were the space larger, it would take considerably more actions. Using the *Look* action first causes the PNM to be increased sufficiently so that subsequent *TryGrasp* actions are much more informative. It also sometimes happens that, if the noise in the *TryGrasp* observations is high, the PNM falls sufficiently so that replanning is triggered at the high level and a new *Look* action is performed, which refocuses the grasp attempts on the correct part of the space. These results are preliminary; our next steps are to integrate this planning process with the rest of the motion and manipulation framework.

7 Conclusion

This paper has described a tightly integrated approach, weaving together perception, estimation, geometric reasoning, symbolic task planning, and control to generate behavior in a real robot that robustly achieves tasks in complex, uncertain domains.

It is founded on these principles: (1) Planning explicitly in the space of the robot's *beliefs* about the state of the world is necessary for intelligent information-gathering behavior; (2) Planning with simplified domain models is efficient and can be made robust by detecting execution failures and replanning online; (3) Combining logical and geometric reasoning enables effective planning in large state spaces; and (4) Online hierarchical planning interleaved with execution enables effective planning over long time horizons.

References

1. Jennifer L. Barry, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. DetH*: Approximate hierarchical solution of large markov decision processes. In *IJCAI*, 2011.
2. C. Boutilier. Correlated action effects in decision theoretic regression. In *UAI*, 1997.
3. Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research*, 28, 2009.
4. T. Erez and W. Smart. A scalable method for solving high-dimensional continuous POMDPs using local approximation. In *UAI*, 2010.
5. C. Fritz and S. A. McIlraith. Generating optimal plans in highly-dynamic domains. In *UAI*, 2009.
6. Kris Hauser. Randomized belief-space replanning in partially-observable continuous spaces. In *WAFR*, 2010.
7. K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez. Task-driven tactile exploration. In *RSS*, 2010.
8. Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E. Gil Jones. Contact-reactive grasping of objects with partial shape information. In *IROS*, 2010.
9. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.
10. Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *ICRA*, 2011.
11. Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
12. 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.
13. Bhaskara Marthi, Stuart Russell, and Jason Wolfe. Combined task and motion planning for mobile manipulation. In *ICAPS*, 2010.
14. Erion Plaku and Gregory Hager. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *ICRA*, 2010.
15. R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *RSS*, 2010.
16. S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 2008.
17. S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *AAAI*, 2010.
18. R. B. Scherl, T. C. Son, and C. Baral. State-based regression with sensing and knowledge. *International Journal of Software and Informatics*, 3, 2009.
19. R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.
20. N. E. Du Toit and J. W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *ICRA*, 2010.
21. Richard Waldinger. Achieving several goals simultaneously. In *Machine Intelligence 8*. Ellis Horwood Limited, Chichester, 1977. Reprinted in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate, eds., Morgan Kaufmann, 1990.
22. Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.
23. S. W. Yoon, A. Fern, and R. Givan. FF-replan: A baseline for probabilistic planning. In *ICAPS*, 2007.