Planning in the Know: Hierarchical belief-space task and motion planning

Leslie Pack Kaelbling and Tomás Lozano-Pérez[*]

1 Introduction

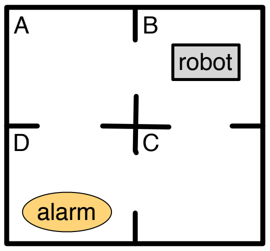
As robots become more physically robust and capable of sophisticated sensing, navigation, and manipulation, we want them to carry out increasingly complex tasks. A robot that helps in a household must plan over the scale of hours or days, considering abstract features such as the desires of the occupants of the house, as well as detailed models that support locating and getting 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.

Recent research [Erez and Smart, 2010; Platt *et al.*, 2010; Toit and Burdick, 2010] has shown the value of planning in belief space using simplified models and replanning. We have developed an approach to combining logic and geometry with online hierarchical planning that is effective in large deterministic domains with long planning horizons [Kaelbling and Lozano-Pérez, 2011]. The focus of this paper is to integrate these two ideas into a method for planning, acting, and estimating in large uncertain robotic domains.

Several other approaches to integrating task and motion planning exist, although none of them treats issues of uncertainty. In the work of Cambon et al. [Cambon *et al.*, 2009], a symbolic domain acts as a constraint and provides a heuristic function for a complete geometric planner. Plaku and Heger [Plaku and Hager, 2010] extend this approach to handle robots with differential constraints and provide a utility-driven search strategy. The work of Wolfe et al. [Marthi *et al.*, 2010] provides a hierarchical combined task and motion planner based on hierarchical transition networks (HTNs) and applies it to a manipulation-planning problem.

Hierarchical task and motion planning Our basic approach to integrating task planning and motion planning has two key properties: (1) It is aggressively hierarchical. It makes choices and commits to them, limiting the length of plans and exponentially decreasing the amount of search required. (2) It operates in the domain of continuous geometry, and does not require any *a priori* discretization of the state or action spaces.

Most work in hierarchical planning uses a hierarchical structure as a way to speed the construction of a complete low-level plan [Marthi *et al.*, 2007] . Instead, we construct a plan at an abstract level, commit to it, and then recursively plan and execute actions to achieve the first step in the abstract plan without constructing the rest of the plan in detail. The risk associated with this approach is that the abstract plan might not be executable: the particular way that the first step is carried out could make it impossible to carry out subsequent steps, at least without undoing the results of earlier steps. We attempt to avoid such failures by constraining the abstract plan steps so that they are *serializable*; that is, so that for any realization of the first plan step, there exist realizations for the subsequent ones. So, we simply execute the first abstract step, observe the resulting world state, and then plan in detail for the next one. This approach results in dramatic speed-ups from the hierarchical problem decomposition when serializability holds. If serializability fails, an interleaved plan is constructed that achieves the effects of both steps; as long as actions in the environment are ultimately reversible, then any goal can be achieved, at the expense of sub-optimality in the behavior.



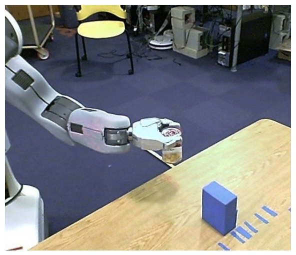


Figure 1: Simulated mobile robot searching for alarm; Willow Garage PR2 manipulating objects.

In complex, high-dimensional geometric spaces, it is crucial to avoid indiscriminate discretization. We handle the integration of continuous geometric planning with task planning by using geometric 'suggesters', which are fast, approximate geometric computations that construct appropriate choices for the parameters of an operator.

Handling uncertainty This paper describes an approach to hierarchical planning under uncertainty about the outcomes of actions as well as about the present state.

It manages uncertainty in the outcome of actions by planning in a determinized approximation of the original domain

[*]CSAIL, MIT, Cambridge, MA 02139 {lpk, tlp}@csail.mit.edu

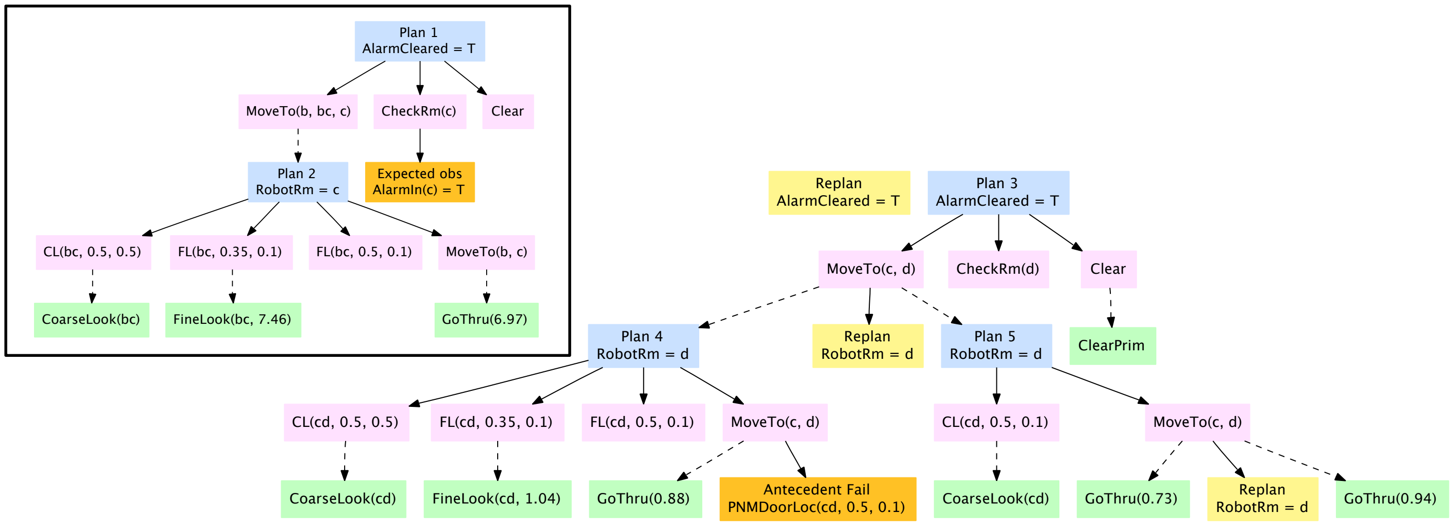


Figure 2: Process of planning and execution while searching for and silencing an alarm.

and replanning when execution does not have the expected results. The hierarchical structure of plans allows localized execution monitoring and replanning, in many cases handling an execution failure by replanning just the current subtask rather than the whole hierarchical plan.

It manages uncertainty about the current state of the world by planning in the space of beliefs. 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 fluents to specify limited properties of belief states, as [Kaelbling and Lozano-Pérez, 2011] does for properties of geometric configurations. Regression-based planning allows the construction of high-level plans to achieve goals articulated in terms of those fluents, without explicitly formalizing the complete dynamics on the underlying continuous space.

We demonstrate our approach in an illustrative simulated domain and provide some preliminary results in a real mobile-robot manipulation problem (figure 1).

2 Example

Consider a mobile robot whose task is to find an alarm that is going off and silence it. The robot is in a house and knows the connectivity of the rooms, and is able to stay well localized with respect to the objects in the house, but is uncertain, a priori, about the location of the doors within the walls of the house. The belief space involves continuous aspects (locations of doors) and discrete aspects (which room contains the alarm). The robot needs to explicitly sense in order to locate the doors. We assume it has two sensors: one with low accuracy and a wide field of view, and one with high accuracy but a narrower field of view. These (idealized) sensors each deliver an estimated location of the center of the door; the sensed location is a random variable with the true location as the mean and Gaussian noise.

Figure 2 shows the process of planning and execution in order to achieve the goal of silencing the alarm. Blue nodes represent planning problems, pink nodes represent subtasks, and green nodes represent primitive actions. Orange and yellow nodes indicate replanning in response to execution failures or violations of expected observations.

There are four rooms in the house (figure 1, left). The robot is initially in room B. It initially believes that the alarm is most likely to be in room C, so it plans (Plan 1) to move to room C, search for the alarm, and silence it. To move from room B to room C, it needs to locate the door precisely enough to go through it. So, it plans (Plan 2) to first look with the coarse (wide field-of-view) sensor, then refine the estimate with the more accurate sensor, and finally move through the door.

The robot executes the first step and obtains an observation which is used to update an estimate of the position of the door, and the next subtask, which is to aim the narrow FOV sensor at the most likely location of the door is considered. The sensor is positioned and an observation received. This observation decreases the uncertainty sufficiently that the next planned observation is not necessary and the robot moves through the door from room B to room C.

At this point, the robot expected to hear the alarm. It does not hear it (indicated by the orange node), so the remaining plan is invalidated, and a new plan (Plan 3) is made, taking into account the knowledge that the alarm is not in room C. It plans to move to room D: this is not the most likely location, but it is cheapest, because moving to A would take an extra step. The robot does not know the location of the door between C and D, so again it plans to gain information. This goes much as before, except that when the robot attempts to go through the door, it fails (because its estimate of the door's location was too far wrong), as indicated by the yellow node. Information gained from the failure is used to update its estimate of the position of the door, and it decides that it does not have enough information to attempt to go through again (indicated by the orange node), so it replans (Plan 5) to look before attempting to go through the door. The attempt to go through the door fails again, but it is repeated with success.

Once in room D, the robot hears the alarm and successfully clears it. This example illustrates: the use of the plan hierarchy to monitor for important changes, localized replanning, planning to gain information, and robust execution. We examine some of the details of this example in subsequent sections.

3 Probabilistic dynamics

The traditional 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 [Weld, 1999]. 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 [Erez and Smart, 2010; Platt *et al.*, 2010; Toit and Burdick, 2010] as well as symbolic planning domains [Yoon *et al.*, 2007].

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. An alternative method is to consider 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. We integrate 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 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, implemented as follows:

```
HPN($belief, goal, abs, world$):
    p = PLAN($belief$, $goal$, $abs$)
    for $(a_i, g_i)$ in p
        while HOLDS($g_{i-1}$, $belief$) and not HOLDS($g_i$, $belief$)
            if ISPRIM($a_i$)
                $obs$ = $world$.EXECUTE($a_i$)
                $belief$.UPDATE($a_i$, $obs$)
            else
                HPN($belief$, $g_i$, NEXTLEVEL($abs$, $a_i$), $world$)
        if not HOLDS($g_i$, $belief$) return
```

It is invoked by HPN($belief, goal, abs, world$), where *belief* is a description of the robot's belief about the current state of world; *goal* is a conjunction of fluents (symbolic predicates with time-varying values) describing a set of goal states; *abs* is a structure that specifies, for any fluent, 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. The PLAN procedure depends on a set of operator descriptions that describe the domain dynamics and returns a list $((-, g_0), (a_1, g_1), ..., (a_n, g_n))$ where the a_i are operator instances, $g_n = goal$, g_i is the weakest precondition of g_{i+1} under a_i, and $belief \in g_0$. PLAN works by goal regression; it computes, for each plan step, a_i, the weakest conjunctive subgoal for that step, g_{i-1}; these subgoals serve as the goals for the planning problems at the next level down in the hierarchy.

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 postcondition, 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.

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. Hierarchical planning and information gain fit together nicely: the system can make a high-level plan to gather information and then use it, and the interleaved hierarchical planning and execution architecture ensures that detailed planning for how to use the information naturally takes place after the information has been gathered.

4 Symbolic planning in belief space

Traditional belief-space planning approaches either attempt to find entire policies, mapping all possible belief states to actions [Smallwood and Sondik, 1973; Kaelbling *et al.*, 1998; Sanner and Kersting, 2010] 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 [Ross *et al.*, 2008]. In order to take advantage of the approach outlined above to hierarchical planning and execution, however, we will take a regression-based approach to planning in belief space.

Fluents and regression In 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 regression conditions. A symbolic world state 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.

In this paper, we use fluents to characterize aspects of the robot's *belief state*, for specifying goals and regression conditions. For example, the condition $\Pr(In(cup, cupboard)) > 0.95$, which describes a *set* of belief states, 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 regression of a set of belief states described by a conjunction of fluents through 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 [Boutilier, 1997; Fritz and McIlraith, 2009; Scherl *et al.*, 2009].

In the rest of this section, we provide examples of representations of beliefs using logical fluents, for both discrete and continuous domains, and illustrate them on the example from section 2.

Knowledge and lack of knowledge Consider a situation in which there is initial uncertainty about the state of the world, but where some actions can generate observations that will completely resolve uncertainty about a particular fluent. Aspects of many real-world domains have this character: a robot may not know the location of a particular object in the house, but when it looks inside a cupboard, it will either know that the object is in that cupboard or know that it is not. We use explicit logical representation of knowledge and lack of knowledge [Petrick and Bacchus, 2004] to model such situations.

The knowledge of an agent with respect to a property ϕ of the external world can be characterized in two different ways. It may be that the agent *knows* **the** *value* of ϕ, defined as $K(\phi = v) \equiv \Pr(\phi = v) > 1 - \epsilon$, for some fixed small ϵ. Alternatively, we might characterize an agent's future knowledge state by saying that it *knows* **a** *value* of ϕ: $KV(\phi) \equiv \exists v. K(\phi = v)$. This formula characterizes the set of belief states in which the agent is relatively sure about the value of ϕ, without committing to which value it will be. Knowing a value can be particularly useful as a precondition to a more concrete action that will make use of the fluent value: knowing the location of an object in order to pick it up, or knowing a phone number in order to call it. We use K and KV as shorthand for describing fluents that make these knowledge assertions about the belief state.

Consider an operator with the knowledge result $KV(\phi)$. Such an operator is not sufficient to establish a result $K(\phi = V)$ during planning: because it cannot know in advance what observation will result, it cannot promise what the resulting value of ϕ will be. We apply the determinization strategy of allowing the planner to choose any of the possible outcomes, with a cost inversely proportional to their probability. Thus, we can treat a single operator with a $KV(\phi)$ effect as multiple operators, each of which achieves the condition $K(\phi = V)$ with a cost $c/\Pr(\phi = V)$. We also add the precondition that the value of ϕ is not yet known.

Example Now we re-examine process shown at the top level of figure 2 in more detail. The operator descriptions are:

MOVETO(Q, R): $K(RobotRoom = R)$
pre: $K(RobotRoom = Q), K(adjacent(Q, R) = \mathrm{T})$

CHECKROOM(R): $K(AlarmIn(R) = \mathrm{T})$
pre: $K(RobotRoom = R), KV(AlarmIn(R)) = \mathrm{F}$
cost: $1/\Pr(AlarmIn(R) = \mathrm{T})$

CLEAR(R): $K(AlarmClear = \mathrm{T})$
pre: $K(RobotRoom = R), K(AlarmIn(R) = \mathrm{T})$

The first line of each one gives the name of the operator and its arguments, followed by the fluent that is the main effect of the operator. Following is a list of fluents describing the preconditions, and a cost (omitted if the cost is 1).

We consider a simpler case in which there is a prior probability of 0.2 that the alarm is in room A and a probability of 0.8 that it is in room C; the planner generates the regression search tree in figure 3. Each node contains a list of fluents specifying conditions under which that partial plan will achieve the goal condition; green nodes represent the solution path.

The final operator must be to clear the alarm: this can happen only if the robot and the alarm are in the same room, so there are three ways in which this operator can be executed. The green successor node requires that the robot know that the alarm is in room C. The only way for it to come to know that is to perform the CHECKROOM operator in room C; that operator has the precondition that the robot be in room C, and that is achieved by an initial step of moving from room B to room C. The search also considered solutions that involved finding the alarm in rooms A and B. Compare the costs on the paths that check those rooms to the path for checking room C. Each primitive action has a cost of 1; but because the outcome of checking a room is probabilistic, the cost for selecting the outcome in which we find the alarm is $1/\Pr(AlarmIn(R) = T)$. So, the cost of finding the alarm in room A is $1/0.2 = 5$, the cost of finding it in room B is effectively infinite, and the cost of finding it in room C is $1/0.8 = 1.25$. For this reason, it searches in room C. It will not always go to the most likely room, however: the costs of other action and uncertainties along the way will be combined to find the least cost path.

When this plan is executed, the robot moves to room C, checks, and finds that the alarm is not there. It does a belief-state update based on this information. This plan is terminated and a new plan is constructed, which causes the robot to move to room A, find the alarm, and clear it. This example illustrates the strength and simplicity of the approach of planning in belief space with a determinized dynamics model: it forces the robot to perform information-gathering actions and robustly handles cases where the information is 'surprising' by replanning.

Noisy observation of a discrete variable In a more general case, observations will not give certain knowledge of properties of the world, and state transitions will be stochastic. To plan in this more general belief space, we need to establish a representation for sets of belief states that attach different probability values to the underlying world property, and show how to perform regression of those belief sets through the robot's operators. The planning goal and the sub-

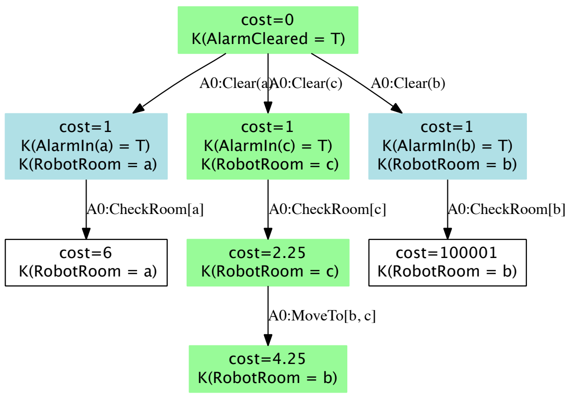


Figure 3: Part of the regression search to find the room with the alarm.

goals constructed during regression will be represented as conjunctions of fluents that are bounds on the probabilities of properties of states of the world.

Define the state estimator $SE(b_t, a_t, o_t)$ as the procedure that takes a belief state at time t, the action at time t and the observation at time $t+1$ and returns the belief state at time $t+1$. The regression of a set of beliefs B_{t+1} through an action a, under the assumption of observing o is the set

$$\mathit{regress}(B_{t+1}, a, o) = \{b_t \mid SE(b_t, a, o) \in B_{t+1}\} \ .$$

We assign this action cost $c/\Pr(o \mid b_t)$, where c is the cost to take a once; we have to aggregate this cost over all b_t.

Here is a very simple illustrative example: a domain with a single binary-valued property A, with the goal that $\Pr(A_{t+1} = \mathrm{T}) > \theta$. If we execute a sensing action a that generates an observation, but does not change the underlying state, then if observation o is made, the regression of the goal under a is:

$$\Pr(A_t = \mathrm{T}) > \frac{\theta \Pr(o \mid A = \mathrm{F})}{(1-\theta)\Pr(o \mid A = \mathrm{T}) + \theta \Pr(o \mid A = \mathrm{F})} \ .$$

For this action, contingent on getting observation o, we would pay a cost $1/\Pr(o \mid A = T)$. Concretely, if the observation is binary, $\Pr(o \mid A = T) = 0.9$ and $\Pr(o \mid A = F) = 0.3$, and our goal is $\Pr(A_{t+1} = T) > 0.95$, then the regression under the sensing action, assuming observation o, would be: $\Pr(A_t = T) > 0.86$. We might represent such a condition with a fluent $PrA(0.86) = T$.

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 prediction 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 $X \sim \mathcal{N}(\mu, \sigma^2)$,

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

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

$$\mathcal{N}\left(\frac{\mu_t\sigma_0^2 + o\sigma_t^2}{\sigma_o^2 + \sigma_t^2}, \frac{\sigma_o^2\sigma_t^2}{\sigma_o^2 + \sigma_t^2}\right) \ .$$

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

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

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

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

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.

Example Now we can understand the process shown at the lower level of figure 2 in more detail in terms of the operator descriptions:

MOVETO$(Q, D, R, \theta) : K(RobotRoom = R)$:
pre: $K(RobotRoom = Q), K(adjacent(Q, D, R) = \mathrm{T}),$
$PNMDoorLoc(D, \theta, doorMargin) = T$
prim: GOTHRU
cost: $1/\theta$

COARSELOOK$(D, \theta, \delta) : PNMDoorLoc(D, \theta, \delta) = \mathrm{T}$:
pre: $PNMDoorLoc(D, PNMRegress(\theta, \delta, \sigma^2_{coarse}), \delta) = \mathrm{T}$

FINELOOK$(D, \theta, \delta, \theta_{fov}) : PNMDoorLoc(D, \theta, \delta) = \mathrm{T}$:
pre: $PNMDoorLoc(D, PNMRegress(\theta, \delta, \sigma^2_{fine}), \delta) = \mathrm{T},$
$PNMDoorLoc(D, \theta_{fov}, fov/2) = \mathrm{T}$
cost: $1/\theta_{fov}$

The MOVETO operator is elaborated with argument D for door, and a precondition that the location of the door be known to within the margin between the robot and the door with probability θ. When the primitive is executed, it will drive the robot as if the door were located at the *mode* of the

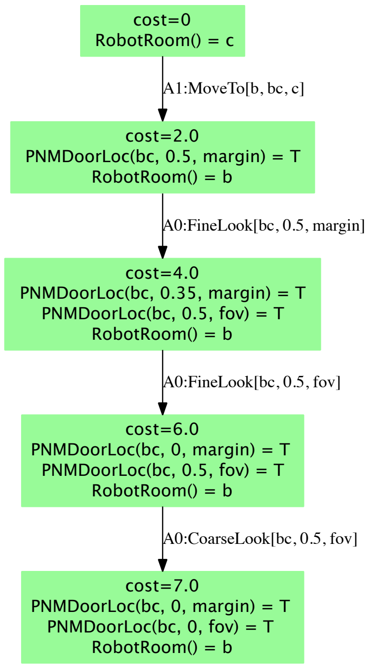

cost=0
RobotRoom() = c
A1:MoveTo[b, bc, c]
cost=2.0
PNMDoorLoc(bc, 0.5, margin) = T
RobotRoom() = b
A0:FineLook[bc, 0.5, margin]
cost=4.0
PNMDoorLoc(bc, 0.35, margin) = T
PNMDoorLoc(bc, 0.5, fov) = T
RobotRoom() = b
A0:FineLook[bc, 0.5, fov]
cost=6.0
PNMDoorLoc(bc, 0, margin) = T
PNMDoorLoc(bc, 0.5, fov) = T
RobotRoom() = b
A0:CoarseLook[bc, 0.5, fov]
cost=7.0
PNMDoorLoc(bc, 0, margin) = T
PNMDoorLoc(bc, 0, fov) = T
RobotRoom() = b

Figure 4: Planned action sequence for going through door

belief distribution. The θ value is a free parameter; the planner tries a small number of samples of that value. The higher θ, the lower the cost of the operator, because the more likely it is to succeed. We have two operators that can achieve a condition on *PNMDoorLoc*. The first uses the coarse sensor, and simply uses the *PNM* regression condition to determine the precondition on the knowledge about the door's location. The σ_{coarse} is fairly high. The FINELOOK operator requires aiming a sensor; it will aim it at the *mode* of the belief distribution, and will only see the door if its center is within the sensor's field of view. This sensor's observations have a much smaller standard deviation. This operator has a free parameter θ_{fov}, which governs both the cost of the operator based on its likelihood of success and the stringency of the condition on how well the door's location is already known.

If the robot attempts to move through the door and fails, or if it tries to look with the fine sensor and fails to see the door, information is gained, and the belief-state update will reflect that. Given this particular formalization of the domain, however, the planner cannot explicitly decide to use these operators to gain information. If it were important, those information-gain aspects could be formalized.

Figure 4 shows the regression plan for going through the door. The goal is for the robot to be in room C. It selects a version of the MOVETO operator that requires the door be localized to within an appropriate margin with probability 0.5 (which means that half of the attempts to go through the door will fail, on average). If attempts to go through the door had higher cost, it would select a version of this operator that was less likely to fail. Now, we have to find a way to gain information. The FINELOOK operator is used twice, until there is

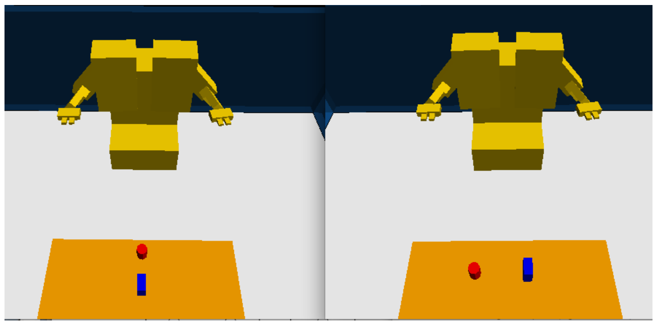


Figure 6: The situation on the left is the real state of the world; the one on the right represents the mode of the initial belief.

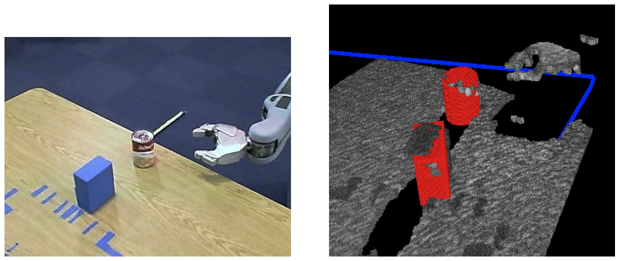


Figure 7: Point cloud (on right) for scene (on left); red points correspond to the models at their perceived poses.

no a priori requirement on the knowledge of the door's location, under the assumption of getting observations from the fine sensor. However, using the fine sensor requires enough information to aim it, which introduces the requirement of knowing the location to within its field of view with probability at least 0.5. The COARSELOOK operator is used to establish this condition.

Although the computation of the regression conditions on information gain from the door sensors assumes the belief state is Gaussian, the state estimator need not use that representation. Our implementation uses a particle filter, which is able to represent the multi-modality of the belief distribution that arises, e.g., when the fine sensor looks and fails to see the door. The process of searching for the location of the door can be shown to converge in a finite number of replanning steps if the state estimation is exact: on each step, the robot either looks at or tries to move through the location that is the mode of its belief distribution. In so doing, it either receives negative information, which will rule out this location and increase the probability assigned to other locations, or it receives positive information, increasing the likelihood of the mode. In both cases, it non-trivially increases the probability mass associated with the true location, so that eventually the distribution will be as concentrated as we require. This argument is a version of Wald's sequential analysis [Wald, 1945].

5 PR2 manipulation example

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 6 shows a planning problem in

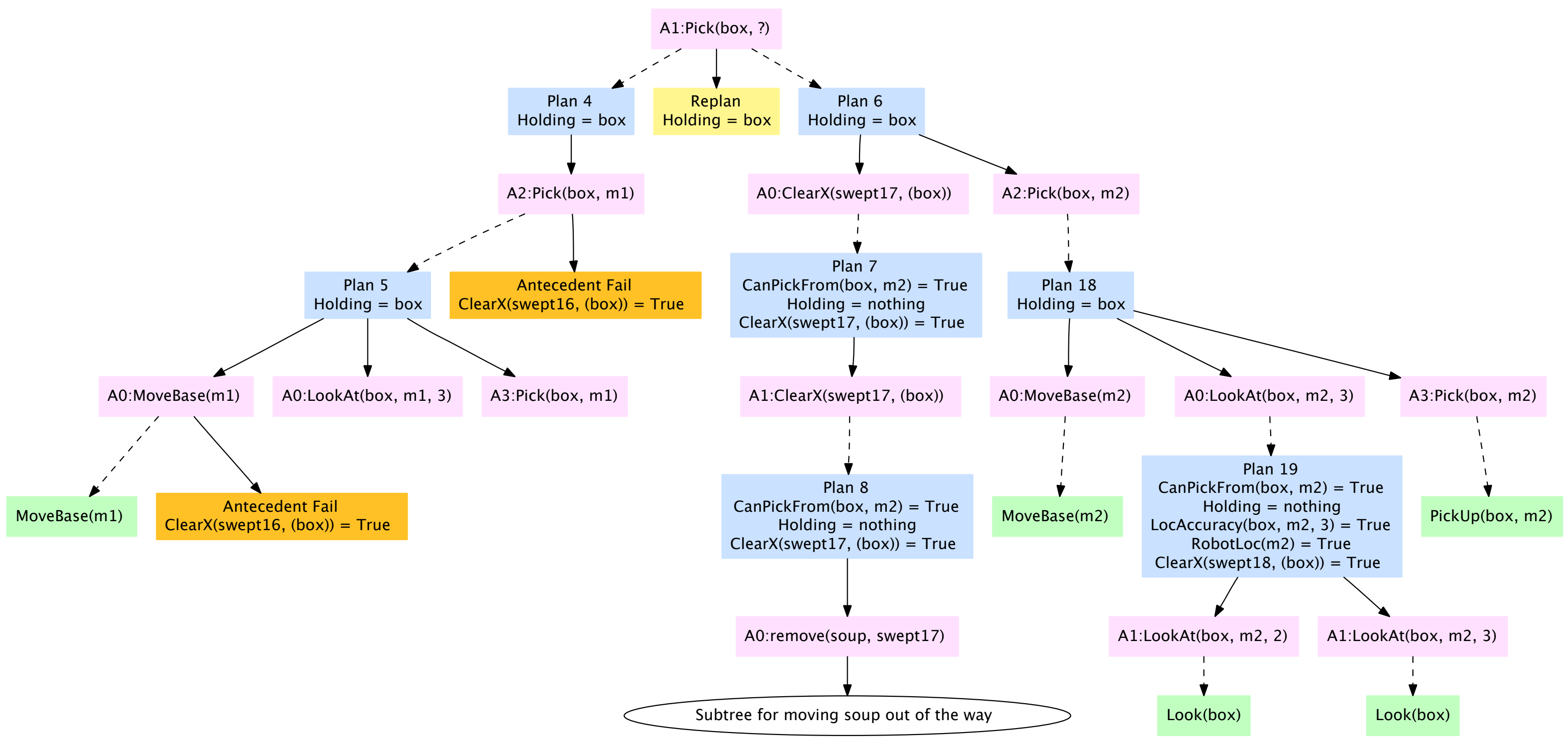


Figure 5: Partial planning and execution tree; the robot notices that the soup can is in the way and removes it.

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 7. 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 [Kaelbling and Lozano-Pérez, 2011]. Here, we emphasize uncertainty handling in the real robot. Here are three of the relevant operator descriptions:

$\textsc{Pick}(O, M) : K(\mathit{Holding} = O)$:
pre: $K(\mathit{ClearX}(\mathit{PickSwept}(M), O) = T)$,
$K(\mathit{CanPickFrom}(O, M) = T), K(\mathit{Holding} = \textsc{Nothing})$,
$K(\mathit{RobotLoc}(M) = T), \mathit{LocAccuracy}(O, M, 3) = T$

$\textsc{LookAt}(O, M, N) : \mathit{LocAccuracy}(O, M, N) = T$
pre: $K(\mathit{RobotLoc}(M) = T), \mathit{LocAccuracy}(O, M, N{-}1) = T$

$\textsc{MoveBase}(M) : K(\mathit{RobotLoc}(M) = T)$
pre: $K(\mathit{ClearX}(\mathit{MoveSwept}(M), ()) = T)$
sideEffect: $\mathit{LocAccuracy}(O, M', N) = F$

The PICK operator describes conditions under which the robot can pick up an 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) calling a modified version of the Willow-Garage reactive grasping procedure that uses the tactile sensing to robustly grasp 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; M is selected by a "suggester" 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 clear, with the exception of the object to be picked up; that the object O 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 not currently holding anything; that the robot base is at 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. 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 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 held 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 clear. Importantly, it also declares that it has the effect of invalidating the accuracy of the estimates of all object poses.

Figure 5 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 HPN) and returns to a higher level in the recursive planning and execution process. This is indicated in planning and execution tree by the orange boxes, showing that it fails up two levels, until the high-level PICK operation is planned for 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.

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

[Boutilier, 1997] C. Boutilier. Correlated action effects in decision theoretic regression. In *UAI*, 1997.

[Cambon *et al.*, 2009] Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research*, 28, 2009.

[Erez and Smart, 2010] T. Erez and W. Smart. A scalable method for solving high-dimensional continuous POMDPs using local approximation. In *UAI*, 2010.

[Fritz and McIlraith, 2009] C. Fritz and S. A. McIlraith. Generating optimal plans in highly-dynamic domains. In *UAI*, 2009.

[Kaelbling and Lozano-Pérez, 2011] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *ICRA*, 2011.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[Marthi *et al.*, 2007] Bhaskara Marthi, Stuart Russell, and Jason Wolfe. Angelic semantics for high-level actions. In *ICAPS*, 2007.

[Marthi *et al.*, 2010] Bhaskara Marthi, Stuart Russell, and Jason Wolfe. Combined task and motion planning for mobile manipulation. In *ICAPS*, 2010.

[Petrick and Bacchus, 2004] R. P. A. Petrick and F. Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *ICAPS*, 2004.

[Plaku and Hager, 2010] Erion Plaku and Gregory Hager. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *ICRA*, 2010.

[Platt *et al.*, 2010] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *RSS*, 2010.

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

[Sanner and Kersting, 2010] S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *AAAI*, 2010.

[Scherl *et al.*, 2009] R. B. Scherl, T. C. Son, and C. Baral. State-based regression with sensing and knowledge. *International Journal of Software and Informatics*, 3, 2009.

[Smallwood and Sondik, 1973] 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.

[Toit and Burdick, 2010] N. E. Du Toit and J. W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *ICRA*, 2010.

[Wald, 1945] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2), 1945.

[Weld, 1999] Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.

[Yoon *et al.*, 2007] S. W. Yoon, A. Fern, and R. Givan. FF-replan: A baseline for probabilistic planning. In *ICAPS*, 2007.