

# Domain and Plan Representation for Task and Motion Planning in Uncertain Domains

Leslie Pack Kaelbling and Tomás Lozano-Pérez

## I. 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, whether ingredients and cooking tools for preparing a meal or finding medicines for an elderly patient. 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.

This paper describes a tightly integrated approach, weaving together perception, estimation, geometric reasoning, symbolic task planning, and control to generate behavior in real robots that robustly achieves tasks in complex, uncertain domains. It is founded on four main principles:

- Planning explicitly in the space of the robot's *beliefs* about the state of the world is necessary for intelligent information-gathering behavior;
- Planning with highly simplified domain models is efficient and can be made robust by using perception to detect execution failures and replanning online;
- Combining logical and geometric reasoning enables effective planning in extremely large state spaces;
- Online hierarchical planning interleaved with execution enables effective planning over very long time horizons.

Recent research [1], [2], [3] has established the value of planning in belief space using simplified models and replanning. We have recently developed an approach to combining logical and geometric methods with online hierarchical planning that is effective in large deterministic domains with long planning horizons [4]. Our goal 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. [5], a symbolic domain acts as a constraint and provides a heuristic function for a complete geometric planner. Plaku and Heger [6] extend this approach to handle robots with differential constraints and provide a utility-driven search strategy. The work of Wolfe et al. [7] 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 [8]. 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.

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 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

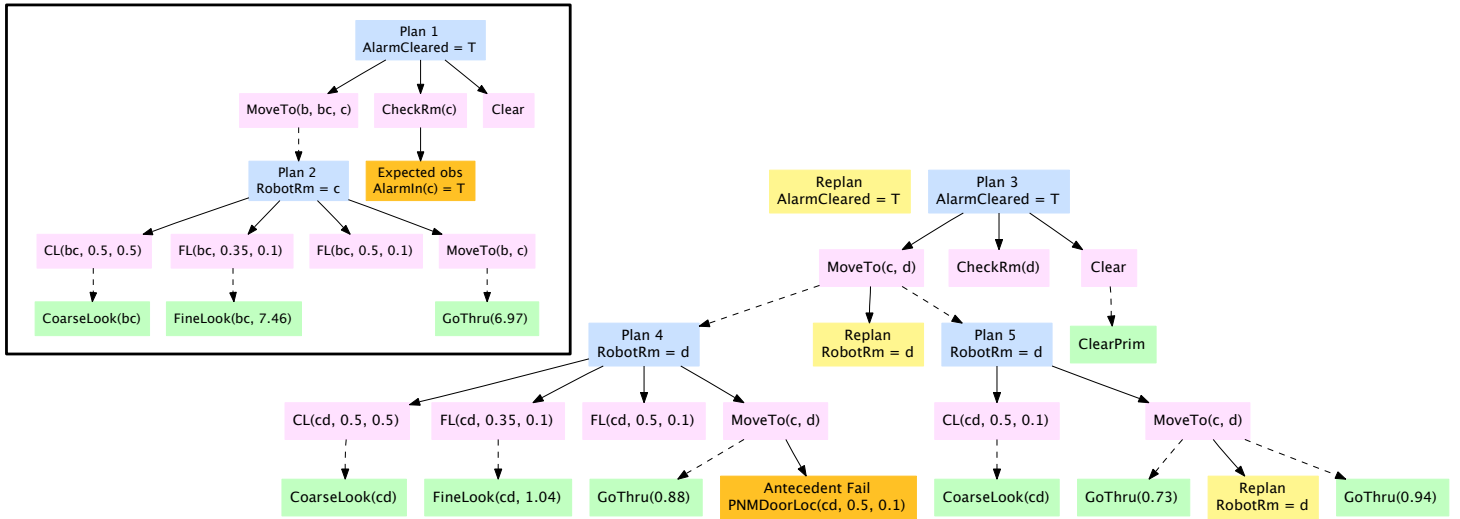


Fig. 1. Process of planning and execution while searching for and silencing an alarm.

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 our previous work [4] 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.

## II. 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 1 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. 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 isn't 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.

## III. PROBABILISTIC DYNAMICS

The traditional approach to planning in domains with probabilistic dynamics is to make a *conditional plan*, supplying

an action to take in response to any possible outcome of a preceding action [9]. 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 [1], [2], [3] as well as symbolic planning domains [10].

**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 (ai, gi) in p
    while HOLDS(gi-1, belief) and not HOLDS(gi, belief)
      if ISPRIM(ai)
        obs = world.EXECUTE(ai)
        belief.UPDATE(ai, obs)
      else
        HPN(belief, gi, NEXTLEVEL(abs, ai), world)
    if not HOLDS(gi, belief) return

```

It is invoked by  $\text{HPN}(\textit{belief}, \textit{goal}, \textit{abs}, \textit{world})$ , 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  $((-, g_0), (a_1, g_1), \dots, (a_n, g_n))$  where the  $a_i$  are operator instances,  $g_n = \textit{goal}$ ,  $g_i$  is the weakest precondition of  $g_{i+1}$  under  $a_i$ , and  $\textit{belief} \in g_0$ . 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 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.

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.

#### IV. SYMBOLIC PLANNING IN BELIEF SPACE

Traditional belief-space planning approaches either attempt to find entire policies, mapping all possible belief states to actions [11], [12], [13] 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 [14]. 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  $\text{Pr}(\textit{In}(\textit{cup}, \textit{cupboard})) > 0.95$ , which describes a *set* of belief states, can be written using a fluent such as  $\textit{PrIn}(\textit{cup}, \textit{cupboard}, 0.95)$ , 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 [15], [16], [17]. 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 II.

**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 [18] to model such situations.

The knowledge of an agent with respect to a property  $\phi$  of the external world can be characterized in two different ways. It may be that the agent *knows the value* of  $\phi$ , defined as  $K(\phi = v) \equiv \Pr(\phi = v) > 1 - \epsilon$ , for some fixed small  $\epsilon$ . Alternatively, we might characterize an agent's future knowledge state by saying that it *knows a value* of  $\phi$ :  $KV(\phi) \equiv \exists v.K(\phi = v)$ . This formula characterizes the set of belief states in which the agent is relatively sure about the value of  $\phi$ , 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  $\phi$  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  $\phi$  is not yet known.

**Example** Now we re-examine process shown at the top level of figure 1 in more detail. The operator descriptions are:

MOVETO(Q, R):  $K(RobotRoom = R)$   
**pre:**  $K(RobotRoom = Q), K(Adjacent(Q, R) = T)$

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

CLEAR(R):  $K(AlarmClear = T)$   
**pre:**  $K(RobotRoom = R), K(AlarmIn(R) = T)$

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).

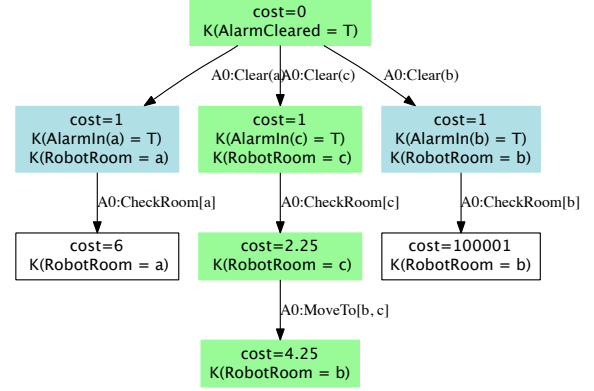


Fig. 2. Part of the regression search to find the room with the alarm.

We'll 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 2. 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.

**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  $\delta$  of the mode of the distribution. So, the robot’s prediction of its success in going through the door would be the PNM with  $\delta$  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  $\sigma_o^2$ .

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, where

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

**Example** Now we can understand the process shown at the lower level of figure 1 in more detail in terms of the operator descriptions:

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

**COARSELOOK**( $D, \theta, \delta$ ) :  $PNMDoorLoc(D, \theta, \delta) = T$ :  
**pre:**  $PNMDoorLoc(D, PNMregress(\theta, \delta, \sigma_{coarse}^2), \delta) = T$

**FINELOOK**( $D, \theta, \delta, \theta_{fov}$ ) :  $PNMDoorLoc(D, \theta, \delta) = T$ :  
**pre:**  $PNMDoorLoc(D, PNMregress(\theta, \delta, \sigma_{fine}^2), \delta) = T,$   
 $PNMDoorLoc(D, \theta_{fov}, fov/2) = T$   
**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  $\theta$ . When the primitive is executed, it will drive the robot as if the door were located at the *mode* of the belief distribution. The  $\theta$  value is a free parameter; the planner tries a small number of samples of that value. The higher  $\theta$ , 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

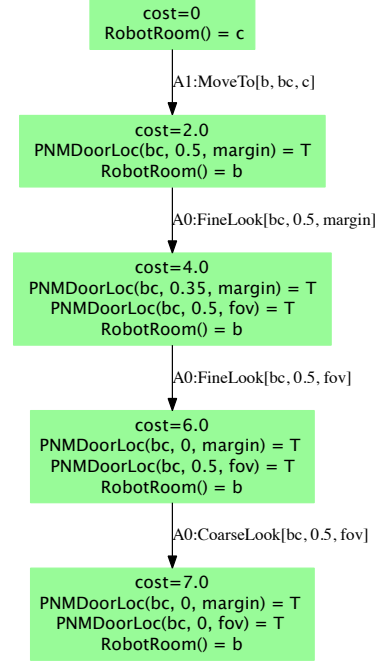


Fig. 3. Planned action sequence for going through door

the coarse sensor, and simply uses the *PNM* regression condition to determine the precondition on the knowledge about the door’s location. The  $\sigma_{coarse}$  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  $\theta_{fov}$ , 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 3 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 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 require-

ment 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 [19].

#### V. ONGOING WORK

To support general-purpose operations in complex domains, we need to represent distributions over geometric and non-geometric properties of and relations among objects in the environment. Exact belief-state estimation in these large models very difficult. We consider highly factored models, and assume that the basic perceptual mechanism, given models of individual objects or object classes and a prior on whether and where they might appear in images, can deliver detections of objects, including labels, 3D pose estimations, and degree of confidence. The state of the estimator consists of a set of objects and a distribution over their properties and relations. Poses of objects may be represented in a single global frame with correlated error (as in many simultaneous localization and mapping methods [20], [21]; or they may be represented as a collection of pairwise marginals.

The state estimator would have the following steps: (1) Apply world dynamics to previous belief; (2) Add newly perceived objects; (3) Reason about identity; merge new detections with existing objects; and (4) Perform belief propagation to determine the marginals on the fluents. One important technical problem is how to represent probability distributions on 6-DOF relative poses. We need to represent relative poses among objects, between object and robot, between robot base and its hand, etc. Most work in state estimation is concerned only with translations between point features, but for application involving manipulation of objects, rotations are crucially important.

A pose consists of a rotation (a unit quaternion) and a translation (a vector in 3 space). We are developing a novel distribution on poses, consisting of a joint Bingham distribution on the rotation [22] and a Gaussian on the translation. To use this distribution in the state estimator, we must be able to condition on observations, and to compute the distribution on the composition of two transforms from

the distributions on the individual transforms. Unfortunately, the distribution is not closed under these operations, but we believe that good approximations can be achieved through moment matching.

Crucial to efficient and effective state estimation is the control of attention and ‘priming’ by the planning and control module. At the physical level, the planning and control module must ensure that the robot takes actions to gather information, such as ensuring that the robot is in a position and its cameras are at an angle that puts the objects of interest in the field of view. The state estimator can also feed its predictions about what it expects to see into the perception module: estimating the poses of objects already detected or the types of objects likely to be encountered for the first time allows the perception to be more efficient.

#### REFERENCES

- [1] T. Erez and W. Smart, “A scalable method for solving high-dimensional continuous POMDPs using local approximation,” in *UAI*, 2010.
- [2] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, “Belief space planning assuming maximum likelihood observations,” in *RSS*, 2010.
- [3] N. E. D. Toit and J. W. Burdick, “Robotic motion planning in dynamic, cluttered, uncertain environments,” in *ICRA*, 2010.
- [4] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical task and motion planning in the now,” in *ICRA*, 2011.
- [5] S. Cambon, R. Alami, and F. Grivot, “A hybrid approach to intricate motion, manipulation and task planning,” *International Journal of Robotics Research*, vol. 28, 2009.
- [6] E. Plaku and G. Hager, “Sampling-based motion planning with symbolic, geometric, and differential constraints,” in *ICRA*, 2010.
- [7] B. Marthi, S. Russell, and J. Wolfe, “Combined task and motion planning for mobile manipulation,” in *ICAPS*, 2010.
- [8] —, “Angelic semantics for high-level actions,” in *ICAPS*, 2007.
- [9] D. S. Weld, “Recent advances in AI planning,” *AI Magazine*, vol. 20, no. 2, pp. 93–123, 1999.
- [10] S. W. Yoon, A. Fern, and R. Givan, “FF-replan: A baseline for probabilistic planning,” in *ICAPS*, 2007.
- [11] R. D. Smallwood and E. J. Sondik, “The optimal control of partially observable Markov processes over a finite horizon,” *Operations Research*, vol. 21, pp. 1071–1088, 1973.
- [12] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, 1998.
- [13] S. Sanner and K. Kersting, “Symbolic dynamic programming for first-order POMDPs,” in *AAAI*, 2010.
- [14] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, “Online planning algorithms for pomdps,” *Journal of Artificial Intelligence Research*, 2008.
- [15] C. Boutilier, “Correlated action effects in decision theoretic regression,” in *UAI*, 1997.
- [16] C. Fritz and S. A. McIlraith, “Generating optimal plans in highly-dynamic domains,” in *UAI*, 2009.
- [17] R. B. Scherl, T. C. Son, and C. Baral, “State-based regression with sensing and knowledge,” *International Journal of Software and Informatics*, vol. 3, 2009.
- [18] R. P. A. Petrick and F. Bacchus, “Extending the knowledge-based approach to planning with incomplete information and sensing,” in *ICAPS*, 2004.
- [19] A. Wald, “Sequential tests of statistical hypotheses,” *The Annals of Mathematical Statistics*, vol. 16, no. 2, 1945.
- [20] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Proceedings of the Uncertainty in Artificial Intelligence (UAI) Conference*, 1986, pp. 435–461.
- [21] H. Durrant-Whyte and T. Bailey, “Simultaneous localisation and mapping (SLAM): Part I the essential algorithms,” *Robotics and Automation Magazine*, 2006.
- [22] K. V. Mardia and P. E. Jupp, *Directional Statistics*. John Wiley and Sons, 2000.