Unifying Perception, Estimation and Action for Mobile Manipulation via Belief Space Planning

Leslie Pack Kaelbling and Tomás Lozano-Pérez

Abstract— **In this paper, we describe an integrated strategy for planning, perception, state-estimation and action in complex mobile manipulation domains. The strategy is based on planning in the *belief space* of probability distribution over states. Our planning approach is based on hierarchical symbolic regression (pre-image back-chaining). We develop a vocabulary of fluents that describe sets of belief states, which are goals and subgoals in the planning process. We show that a relatively small set of symbolic operators lead to task-oriented perception in support of the manipulation goals.**

I. INTRODUCTION

A mobile robot in a complex environment can never be completely certain about the state of the environment. It will have to take sensing actions, including pointing its cameras or moving to new poses in order to get a better view, in support of doing high-level tasks, such as putting objects away in a kitchen. In this paper, we describe an integrated strategy for planning, perception, state-estimation and action in complex mobile manipulation domains.

We have developed an approach to combined task and motion planning that integrates geometric and symbolic representations in an aggressively hierarchical planning architecture, called HPN [1]. 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. We extended this approach to handle uncertainty by changing the space in which we plan from the space of underlying configurations of the robot and objects to the *belief space* of probability distributions over configurations of the robot and objects [2]. In this paper, we develop much richer representations of belief space that support both state estimation using a very high-fidelity model and planning using an abstract symbolic representation.

What should the robot represent about its environment? In order to manipulate objects, it needs to know their poses, and the poses of other nearby objects fairly accurately. In order to move through space (both with base and arms), it needs to know whether that space is free. Early work in mapping explicitly represented knowledge about free space in occupancy grids, and scan-based SLAM methods implicitly contain information about where the robot has looked. In mapping methods based on object pose estimation, however, the focus has been on explicit representation of object poses and implicitly assumes that all space not explicitly marked as occupied is in fact free.

We propose a two-part state representation, with explicit joint pose estimation of the robot and objects in the world, together with a variable-resolution grid representation of the parts of space that have been recently observed. Thus, if a region of space has been recently observed and it doesn't overlap any of the known objects, the the robot believes with high probability that the space is free and can be traversed.

Our planning approach is based on hierarchical symbolic planning, using a technique that is called *regression* in the AI planning literature and *pre-image back-chaining* in the robotics literature. We develop a vocabulary of logical fluents that describe sets of belief states; these sets serve as goals and subgoals in the planning process. These fluents can never jointly represent a belief state in complete detail, but are grounded in procedures that test their truth in the current detailed belief state. They are just sufficiently detailed to support planning while maintaining a tractably small representation of complex subgoals.

We briefly touch on related work, then describe the underlying representation for belief states. We go on to describe the formalism used by the planner and give example planning operator descriptions that illustrate interactions between sensing and acting. Finally, we demonstrate the proposed methods on a task with partial information and noisy sensing using a simulated Willow Garage PR2 robot.

II. RELATED WORK

Advances in 3D perception, navigation and motion planning have enabled sophisticated manipulation systems that interleave perception, planning and acting in realistic domains (e.g., [3], [4]). In most such systems, perception tends to be loosely coupled to the task, usually by assigning the perception subsystem the job of constructing some sort of map of the environment, in which planning will be done. There is also a body of work in which perception is actively controlled to achieve some goal. For example, in active vision (e.g., [5]), cameras are controlled to detect objects more effectively. In robot exploration (e.g., [6], [7], [8]), the target locations of the robots are chosen to so as to perceive the most uncertain locations. However, in this existing work, the planning for perception is not driven by or tightly integrated into planning for some larger task, such as manipulation; in most cases, perception is the task.

We use planning in belief space to achieve a tight integration of perception and action; perception is used to achieve desired belief states while in the process of accomplishing

This work was supported in part by AFOSR and in part by NSF.

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

a manipulation goal. Belief space, either in the form of logical representation of sets of states [9], [10] or in the form of probability distributions over an underlying state space [11] provides the key representation for integrated planning of perception and action. Recent research [12], [13], [14] has established the value of control in belief space using simplified models and replanning. Our approach to belief space planning builds directly on this work.

III. Belief state representation

The belief state for mobile manipulation under uncertainty needs to contain enough information to support queries both about the nominal (mean or mode) state of the world and about the system's confidence in those estimates. The confidence estimates must be accurate enough to support decision-theoretic trade-offs between, for example, going through the door or taking another look to localize it more accurately. It also needs to be a sufficient statistic for the history of observations so that recursive updates may be made effectively. We do not believe there is a good uniform representational strategy for all aspects of information about the domain, so we have separate representations for poses of known objects, for poses of objects that have not yet been observed (but are believed to exist) and for free space.

A. Object poses

Object detection and recognition systems based on visual (intensity or depth) perception usually have fairly good spatial accuracy, though they are susceptible to misclassification, resulting in data-association problems. We are operating under the assumption that the misclassification issues can be handled using robust-statistics strategies and outlier rejection (e.g., [15]) and that unimodal spatial distributions will suffice for such objects.

In our experimental work, we use an extended Kalman filter, but an unscented Kalman filter with outlier rejection might result in a more robust system. The state of the system that is being filtered consists of the poses of all known objects as well as the robot base. The object and robot poses are represented with four degrees of freedom: x, y, z, and θ, which is rotation in the horizontal plane. It is assumed that the robot and all objects are resting on a known stable face.

We establish the following notation for characterizing aspects of the belief state maintained by the filter:

- Objects: integer indices $i, j, \ldots$; index 0 is robot base.
- Pose of object i is a random variable: $\Phi_i = \langle X_i, Y_i, Z_i, \Theta_i \rangle$ taking on values $\phi_i = \langle x_i, y_i, z_i, \theta_i \rangle$. Poses are all expressed in an arbitrary global coordinate frame.
- Mean of pose estimate for object i is μ_i.
- Covariance of pose estimate for object i is Σ_i: this is a 4 by 4 matrix.
- Mean of pose estimate for object i given that the pose of object j is at its mean is is $\mu_{i|j}$.
- Covariance of pose estimate of object i given that the pose of object j is at its mean is $\Sigma_{i|j}$: this is a 4 by 4 matrix.

Observation updates of the filter are performed with an observation vector that consists of the poses of objects that were perceptually detected, in robot-relative coordinates. Transition updates of the filter are performed with a control input computed as the difference between the uncorrected raw odometry of the robot now and at the time of the previous update.

Care must be taken to estimate poses correctly when there are angular dimensions. We handle this by using a *wrapped Gaussian* representation for the angular dimension, essentially constructing a real space tangent to the unit circle at the mean of the angular distribution. This approach is convenient because the product can be taken of multiple tangent spaces together with regular real spaces for the other dimensions, and a single multivariate Gaussian used to represent the entire joint distribution over the product space[16]. An additional concern is object interpenetration: any joint pose of the objects that would cause a collision is invalid. The work described in this paper does not address that problem, but we hope to incorporate a solution based on truncated distributions [17].

The first row of figure 1 shows four states of the EKF during the course of a planning and execution run. The robot is drawn in its mean pose. The other objects are drawn using their conditional distribution given the mean robot pose: $\mu_{i|0}, \Sigma_{i|0}$. The mean pose of the object is drawn in its regular color, and then a "shadow" consisting of the object drawn at poses in which each individual dimension is extended to its positive or negative 95% confidence limit is drawn in gray. In the leftmost frame, there are three known objects: a table, a cupboard, and a cup, and there is substantial uncertainty about their poses. The next frame shows the situation after the robot has made an observation and detected the table and cupboard. They still have moderate uncertainty, but it is dificult to see in this figure; the cup has not yet been observed. In the third frame, a new object is discovered and added to the state of the filter; but it occluded the view of the cup, so that object remains significantly uncertain. In the last frame the cup has been observed and its distribution and its uncertainty reduced.

Unobserved objects There may be objects, or object types, about which the system has prior knowledge, but which it has not yet observed. For instance, the system might believe that, with high probability, there is salt somewhere in the kitchen, but not know which cupboard it is in. In such situations, the spatial details of the distribution are not important, and we do not include such objects in the state of the Kalman filter until they are actually observed. Rather, we maintain a mixture distribution over the possible abstract locations that might contain the object, such as rooms or cupboards, and update that distribution based on negative information, until the object is actually observed.

B. Observed space

Another important query the robot needs to make of the belief state is whether a region of space is known to be clear of obstacles and therefore safe to traverse. To answer such

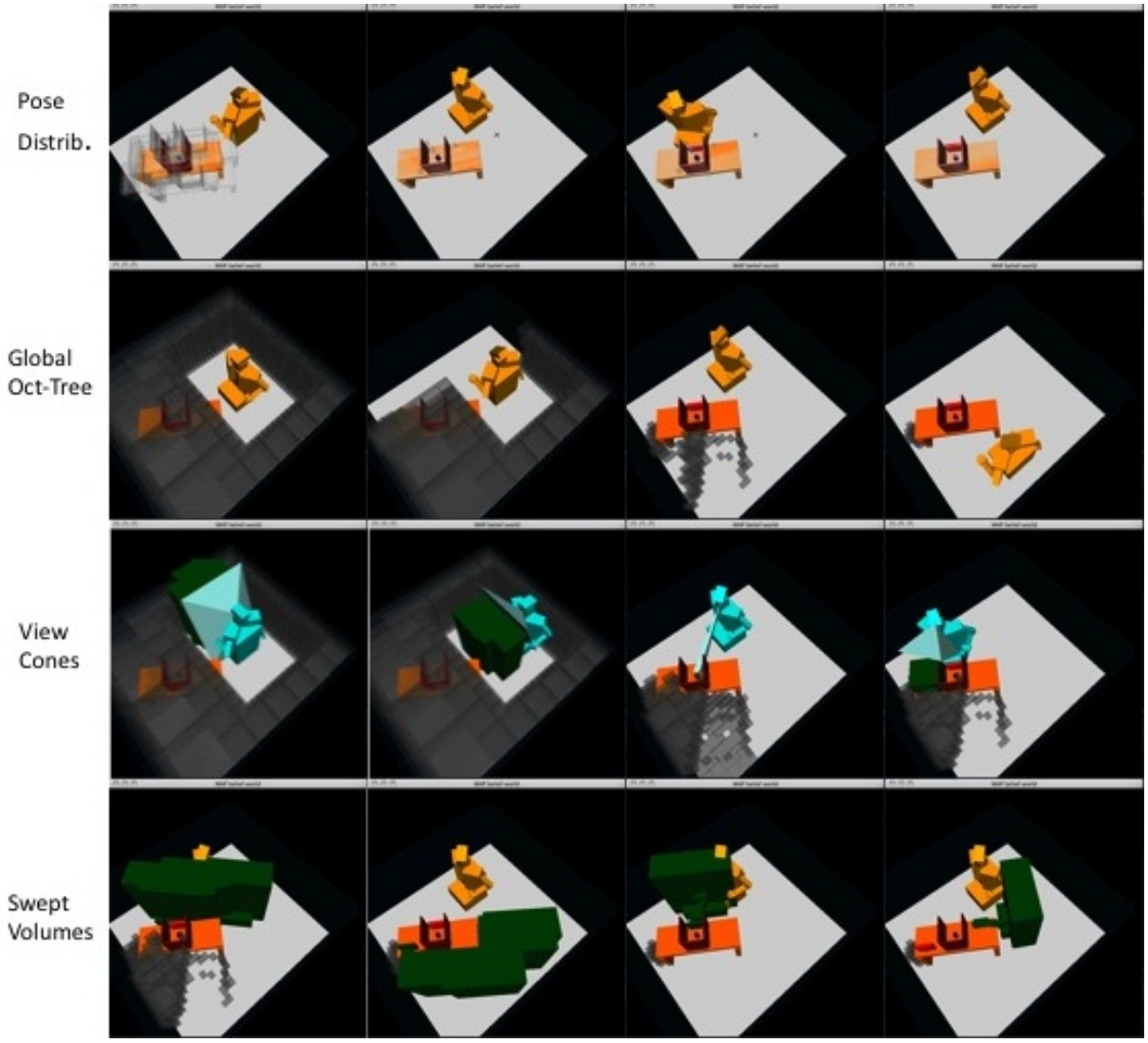

Fig. 1. The first row shows the conditional distribution of the objects given the mean robot pose. The second row shows the unobserved space (as gray boxes). The third row shows suggested robot poses (cyan), view cones (light blue) and target regions (green) for looking operations. The fourth row shows swept volumes for suggested robot motions.

queries, we represent the parts of the space that the robot has recently observed with its depth sensors.

Keeping an accurate probabilistic model of known-clear space is quite difficult: the typical approach in two-dimensional problems is an occupancy grid [18]. It requires a detailed decomposition of the space into grid cells and although there are some attempts to handle odometry error in the robot [19], they are not particularly effective. A more principled strategy would be to maintain the history of robot poses in the EKF, rather than just the current one, and combine the depth maps sensed at each of those poses into a global map of observed space.

We take a much simpler approach, operating under two assumptions: first, that the observed-space maps we construct will only be used in the short term; second, that the mechanisms for detecting objects and tracking their poses will provide a high-accuracy estimate of the poses of material objects. Looking is not too expensive, and objects may be dynamic, so we expect, for instance, when the robot re-enters a room, that it will need to reconstruct the observed-space map. Thus, handling long-distance relative odometry errors is not crucial. For this reason, we simply attach each depth scan to the most likely pose estimate for the robot in the Kalman filter (this is much more accurate than the raw odometry). We integrate the observed-space information from the sequence of scans into an oct-tree representation the space that has been observed by the robot. This representation of known space can be much coarser than an occupancy grid, which must also represent the boundaries between free and occupied regions of the environment.

In the following sections, we will denote space that has been observed as $\mathcal{S}_{obs}$. The second row of figure 1 shows the observed-space oct-tree at four different points during execution; space that is filled with dark-grey cells has not yet been observed by the robot. At initialization time, the robot knows the contents of the region of space right around it.

As it moves and scans (in this case, using both the scanning laser on the torso as well as the narrow-field stereo cameras on the head), it clears out more of the space, until in the final frame, it has observed most of the observable space in the room. One very important role that the observed-space map plays is to constrain the RRT planner when it is determining a trajectory for final execution of a motion primitive; any part of the space that has not yet been observed is marked as an obstacle and cannot be traversed.

IV. Belief set estimation for planning

When planning in belief space, goals must be described in belief space. Example goals are "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 [20], [21], [22]. 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 relies on execution monitoring and replanning to handle errors due to approximation.

We will represent *belief sets* as conjunctions of fluents. Each fluent is a test on an actual belief state: the belief state is in the belief set if all of the fluents test to true.

A. Fluents for mobile manipulation

We demonstrate the use of logical fluents for describing belief sets in the mobile manipulation domain. In this section, we describe the most important fluents in our formulation, and their definitions in terms of tests on belief states.

We specify 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.

The first set of fluents characterize belief about the pose of an object. We start by asserting that the mean of the distribution on the pose of object i is within δ of a desired pose ϕ:

$$PoseMeanNear(i, \phi, \delta) \equiv \|\mu_i - \phi\| < \delta \ .$$

Any appropriate norm can be used here.[1]

To characterize certainty about a pose, we must specify a frame of reference. Although all poses are expressed in a global coordinate frame, we are typically interested in the variance in the estimate of the pose of one object relative to another object. Thus, we are interested in the *conditional distribution* of object i, conditioned on object j being at its mean pose. (This represents a "slice" through the distribution along the i dimensions, rather than a projection of the distribution onto the i axis, and will in general be lower variance.) For example, it may frequently be the case that two objects have very uncertain pose relative to the global coordinate frame, but if they are observed in the same image, they have very certain poses relative to one another.

$$KVCondPose(i, j, \epsilon, \delta) \equiv \Pr(\|\Phi_i - \mu_{i|j}\| < \delta \mid \Phi_j = \mu_j) > 1-\epsilon \ .$$

Because of the symmetry of covariance and the fact that we are slicing through the mean, this fluent will have the same value with arguments i and j reversed. The name of the fluent is intended to indicate belief states in which we "know the value" of the pose, without specifying what the value will be.

[1] In our implementation of this domain, the test is actually computed componentwise: δ is a vector of four values (three metric distances and an angle) and $\|\langle x_1, y_1, z_1, \theta_1\rangle - \langle x_2, y_2, z_2, \theta_2\rangle\| = \langle |x_1 - y_1|, |x_2 - y_2|, |z_1 - z_2|, |\theta_1 - \theta_2|_{\pm\pi}\rangle$, where $|\theta|_{\pm\pi}$ denotes an angle equivalent to θ, in the interval $[-\pi, +\pi]$. Each of the four components of the difference has to be less than the associated component of δ, in order for the condition to be satisfied.

Combining these two concepts, we have a fluent that asserts that there is a high degree of belief that object i is at pose ϕ conditioned on object j being at its mean pose:

$$KCondPose(i, \phi, j, \epsilon, \delta) \equiv \Pr(\|\Phi_i - \phi\| < \delta \mid \Phi_j = \mu_j) > 1 - \epsilon$$

In order to specify conditions on the configuration of the robot, including the 4-dof base pose, as well as the arm (in this work, we only use one arm of the robot), we use the fluents *ConfMeanNear* and *KCondRobotConf*, which are analogous to *PoseMeanNear* and *KCondPose*, but which specify a particular configuration of robot arm. We are assuming that the robot arm configuration is known exactly, so the distributional conditions are not applied to those degrees of freedom.

The next set of fluents characterize beliefs about the contents of regions in space. The regions under consideration are not pre-discretized, but are computed dynamically during planning as part of the pre-image backchaining process.

The $KContents(r)$ fluent simply asserts that the region r has been completely observed.

$$KContents(r) \equiv r \subseteq \mathcal{S}_{obs} \ .$$

The $KClearX(r, x)$ fluent asserts that the region r is known to be clear with the exception of objects in the set x.

$$KClearX(r, x) \equiv KContents(r) \ \& \ \neg\exists i \notin x.overlaps(i, r) \ .$$

In formalizing the domain, it is also useful to be able to say "we don't know of anything in the way." For this, we use the fluent $NotKNotClear(r)$, defined as

$$NotKNotClear(r) \equiv KClear(r) \ || \ \neg KContents(r) \ .$$

An interesting additional fluent is

$$KIn(i, r, p) \equiv \Pr(contains(r, i)) > p \ ,$$

where i is an object, r is a region, and p is a probability. To test this in practice, we construct a union of the volumes obtained by placing the object at the p-percentile pose in each direction of each dimension, then test to see if that "shadow" region is contained in r.

B. Operator descriptions

Operator descriptions for planning characterize the belief pre-image of an action: the conditions that must be true of a belief state, so that the resulting belief state will satisfy the result condition of the operator. Because our domain dynamics are stochastic, even in the belief space, we cannot guarantee a result; these operation descriptions characterize the most likely outcome, and we will re-plan whenever that outcome fails to occur.

The following operator descriptions constitute part of the planning domain description that is used to generate the examples in section V. They illustrate two important ideas: (1) That motion and perception actions need to be tightly integrated in order to achieve goals in complex environments; and (2) That that the general-purpose planning mechanism of logical regression-based planning, applied in belief space, can be used to achieve this integration.

Each operator description has a name, then several components. The tag `pre` indicates a precondition fluent, `let` indicates a quantity that is being computed and named, `exists` indicates a call to a heuristic *suggester* procedure that may generate one or more bindings of variables that have large or infinite domains, and `result` indicates a result fluent. These operators are applied backward during planning: the goal is a conjunction of fluents. A step in the planning search is to match a fluent in the current goal with the result fluent of an operator, to remove that result from the goal, then to add the preconditions to the goal; if the new goal is not a contradiction, then it becomes a node in the search tree. Whenever a goal is satisfied in the current belief state (this happens when all of the fluents in the goal are true) then a legal plan has been found.

Following is a description of the `Pick` operator, which results in the robot knowing that it is holding the objct `O`, with high probability. One thing to note is that the preconditions are divided into two sets. The first precondition is that the pose of the object `O` be known, with large tolerances, with respect to the robot. Given that, the planner considers places from which the object might be picked up: typically, from the current mode of the distribution of the pose of that object in the `b0`, which is the belief state that holds at planning time, and given that pose, it suggests one or more paths `P` that the robot might need to move through in order to pick up the object at that pose. Given these suggestions, we establish another set of preconditions: that the swept volume of that path be known to be clear, that the robot not be olding anything, that the object be known to be close to the pose that we expected it to be in when we computed the path, and that the robot's configuration, conditioned on the object's pose, be known highly accurately. If all of these conditions hold, then the primitive procedure that picks up the object will succeed, resulting in the object being held. The domain description also contains a similar operator description for placing the object. The fact that there are some knowledge preconditions that must be satisfied before even computing the rest of the preconditions fits naturally into the hierarchical planning framework.

```
Pick(O, ObjPose):
  pre: KVCondPose(O, robot, bigEps, planDelta)
  exists: ObjPose in {modeObjPose(b0)} U generateParking(O)
          P in generatePickPaths(ObjPose)
  pre: KClearX(sweptVol(P), O)
       KHolding(None)
       KCondPose(O, ObjLoc, robot, eps, graspDelta)
       KCondRobotConf(O, baseConf(P), smallEps,grspDelta)
  result: KHolding(O)
```

In our planning domain, the primitive operation of looking at an object (by pointing the robot's head so that the object will be centered in the stereo camera frame) can be used to achieve several knowledge conditions. The operator description below characterizes how looking can be used to increase certainty that the pose of object `O` is within `Delta` of pose `OPose`. The most basic precondition is that the mean of the pose distribution for `O` be near the desired pose. If that is true, then we can suggest a configuration of the robot, `RobotConf`, which includes the configuration of the head,

that has the property that if the robot is in that pose, then it is possible to view object O. Additionally, Viewcone is a volume of space between the robot's camera and the object that must be free in order for the view not to be occluded. The next precondition is interesting: it is that the view cone be not known to be occluded. This means that if we know something is in the way, we will be required to move it out; but if we simply don't know the contents of the view cone, then that is sufficient for the most likely result of this operation to be a successful view. We rely on the replanning mechnisms of HPN: if, upon looking, it is reveled that there is an object occluding the view, then a new plan will be made that achieves the NotKNotClear condition by removing the occluding object(s).

Next, we require that the robot's configuration be near the one suggested for viewing the object. The requirement is currently only on the mean of the robot's pose distribution, so that it is in roughly the right place.

Finally, in order to have as a likely outcome KCondPose with probability 1 - Eps, we determine the pre-image of that knowledge fluent by finding a value PNRegress(eps, obsVar, Delta), which is larger than Eps, such that if an observation is made with variance obsVar and that starting degree of uncertainty, everything will come out fine. This operation can chain backwards, determining a sequence of several Look operations in order to guarantee the desired resulting confidence.

In order to achieve KCondPose when the PoseMeanNear condition does not hold, the planner will satisfy that condition using an instance of the Place operator, which will itself generate a condition that requires the Pick operator.

```
Look(O):
  pre: PoseMeanNear(O, OPose, Delta)
  exists: (RobotConf, ViewCone) in generateViewPose(O, OPose)
  pre: NotKNotClear(ViewCone)
       ConfMeanNear(RobotConf, lookDelta)
       KCondPose(O, OPose, RefObj,
                 PNRegress(eps, obsVar, Delta), Delta)
  result: KCondPose(O, OPose, RefObj, Eps, Delta)
```

The next two operators don't have actual primitives associated with them: they are essentially definitional, and compute a set of conditions under which the resulting condition will be true; applying the operator during planning simply replaces the target condition with the preconditions at the appropriate level of abstraction.

In order to achieve the condition that region R is known to be clear with the exception of a list of objects, we must first know the contents of R. Then, we require that each of the objects that is overlapping R in the current belief state be moved, with high probability to a part of the domain called the warehouse; in addition, we establish the requirement that the region be clear of all other objects, as well. This condition will, cause the region to be cleared out again if some exogenous process puts objects into it and will prevent other object placements from being made into that region.

```
KClearX:
    pre: KContents(R)
    let: occluders = objectsOverlapping(R, b0) - Exceptions
    pre: KClearX(R, Exceptions + occluders)
         for o in occluders: KIn(o, 'warehouse', placeEpsilon)
    result: KClearX(R, Exceptions):
```

Finally, to achieve the condition that the contents of a region R are known, we depend on a recursive decomposition of the region. If the region can be viewed in one look operation, then we suggest a configuration for the robot and associated view cone for viewing the region and require the view cone not to be known to be occluded and require the robot configuration to be near the suggested one; if the region is too big for a single view, then we split it into subregions, driven partly by the desire to aggregate space that has already been viewed into the same subregion and space that has not been viewed into different subregions. For each of the subregions, we assert that the contents must be known.

```
KContents:
if not viewable(R):
    let: subRegions = split(R, b0)
    pre: for s in subRegions: KContents(s)
else:
    exists: (RobotConf, ViewCone) in generateViewPose(R)
    pre: NotKNotClear(ViewCone)
         ConfMeanNear(RobotConf, lookDelta)
```

C. Regression of fluents

We defined fluents characterizing aspects of continuous probability distributions, and we use them in operator descriptions. It is necessary to be able to compute the pre-image of a fluent in belief space. We will begin with a simple one-dimensional case, and the describe how it is done for the robot and object pose fluents described in section IV-B.

For a planning goal of $KV(X, \epsilon, \delta)$, that is, knowing the amount of probability mass of random variable X that is within δ of the mode is greater than $1 - \epsilon$, we need to know expressions for the regression of that condition under the actions and observations 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, the action 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 $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$P(|X - \mu| < \delta) = \Phi\left(\frac{\delta}{\sigma}\right) - \Phi\left(-\frac{\delta}{\sigma}\right) = \text{erf}\left(\frac{\delta}{\sqrt{2}\sigma}\right) \quad ,$$

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

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

So, if $P(|X_t - \mu_t| < \delta) = \theta_t = \text{erf}\left(\frac{\delta}{\sqrt{2}\sigma_t}\right)$ then

$$P(|X_{t+1} - \mu_{t+1}| < \delta) = \theta_{t+1} = \text{erf}\left(\frac{\delta}{\sqrt{2}}\sqrt{\frac{\sigma_o^2 + \sigma_t^2}{\sigma_o^2 \sigma_t^2}}\right)$$

Now, we can solve for θ_t, yielding

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

So, to guarantee that $KV(X, \epsilon, \delta) = P(|X - \mu| < \delta) > 1 - \epsilon$ holds after taking action a and observing o, we must guarantee that $KV(X, 1-PNMregress(1-\epsilon, \delta, \sigma_o^2), \delta)$ holds on the previous step.

For the fluent $K(X, x, \epsilon, \delta) \equiv P(|X_t - x| < \delta) > 1 - \epsilon$, which requires $1 - \epsilon$ of the probability mass of X to be within δ of a specific value x, we do a similar regression calculation, though the result cannot be expressed as cleanly. It allows us to define the function `PNregress`, such that, to guarantee that $K(X, x, \epsilon, \delta)$ holds after taking action a and observing o, we must guarantee that $K(X, x, 1 - PNregress(1 - \epsilon, \delta, \sigma_o^2), \delta)$ holds on the previous step.

Now, to compute regression conditions for *KCondPose* for example, we might need to compute the preimage of a set of distributions on an entire pose. We are using four-dimensional vectors of ϵ and δ values, however, so we simply regress the epsilons individually for each dimension.

D. Suggesters

Because the symbolic planner is working in what is essentially an infinite domain (the space of poses, paths, volumes, etc.) it is impossible to create an *a priori* discretization or enumeration of the state space. Instead, we employ heuristic *suggester* procedures to generate candidate values of variables, such as paths and poses, that have infinite domains. We describe two example suggesters in some more detail in this section.

Because we are planning to gain visual information, one important question is where to place the robot in order to get a good view of an object or a region of space that is of interest to the system. The suggester works by constructing an approximate visibility graph of the configuration space of a robot model with simplified kinematics, and testing free configurations in this space to see if they satisfy the visibility requirements for the object or region to be viewed.

The third row of figure 1 shows example results of the view suggester. The robot is shown in cyan at the suggested pose. The cone emenating from the eye is the view cone, which must be unoccluded in order for this pose to result in a good view of the object. The suggester prefers to find poses with view cones that are unoccluded in the belief state at the time of planning; however, if necessary, it will plan to look 'through' another movable object. In that case, the planning operators will construct additional steps in the plan to move the occluding object out of the way before looking. The third frame shows the pose and view cone for looking at a small object in the back of the cupboard; the other frames show view cones for looking at regions of space (shown in green) that had not yet been entirely observed.

Similarly, when the robot needs to move to a new base pose or to pick up or place an object, the high level planner needs to guarantee that space will be available for that operation. The actual primitive robot motion operations, when they are executed, are planned using an RRT on a high-fidelity model. However, while we are considering multiple different high-level plans, it is not efficient to plan robot motions completely accurately. So, the path suggester also uses a visibility graph planner for an approximate robot, and it only tries to guarantee a path to a 'home' region. If, every time the robot moves, there is guaranteed to be a free path to the 'home' region, then it can never block itself in. The fourth row of figure 1 shows, in green, highly approximate, but conservative swept volumes of the robot moving through the suggested paths. It is these volumes that must be determined to be clear before the robot can execute the associated motion or manipulation actions.

V. Example planning and execution

Figure 2 shows a sequence of images depicting the planning and execution process, which is shown in much more detail in the accompanying video. The initial goal is for the small blue object to placed on the table to the robot's left.

Initially the robot has a small known area around it, the rest of the room is unknown—as shown in the first oct-tree in Figure1. To determine the contents of the swept regions of suggested motions (the large green regions in Figure1), a series of look motions and view cones are suggested (the cyan robot placement and light blue cones). When these scans are executed (steps 2–7), new areas of the oct-tree become known as illustrated in subsequent oct-trees in the Figure1.

After the first two scans (steps 2 and 3), the table has not been observed and so its pose distribution is very diffuse – as shown in the first pose distribution in Figure1. Also, the big red object has not been seen; note that it is not part of the initial model. After the table is scanned in step 3, its pose distribution becomes tight but the blue cup is still not visible (it is occluded by the big red object), so its pose distribution is still diffuse—as shown in the second pose distribution. Later, the red object is seen and added to the model (as seen in the third pose distribution). Finally, when going to look at the warehouse region, where the red object is to be moved, the robot serendipitously "sees" the blue cup and narrows its distribution, as seen in the fourth pose distribution.

After the required regions are known, the planning and execution proceeds as normal, resulting in a sequence of operations to move the red block to the warehouse, pick up the blue block and take it to its goal location.

Conclusion: This paper has provided methods for extending belief-space planning techniques to mobile manipulation problems, seamlessly integrating perception and manipulation actions, performing goal-directed perception in service of manipulation and goal-directed manipulation in service of perception. We believe that this work forms a basis for extending the application to very large, cluttered domains.

References

[1] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, 2011, pp. 1470–1477.

[2] ——, "Pre-image backchaining in the belief space for mobile manipulation," in *ISRR*, 2011.

[3] R. B. Rusu, I. A. Sucan, B. P. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Intelligent Robots and Systems (IROS)*, St. Louis, MO, 2009, pp. 4245 – 4252.

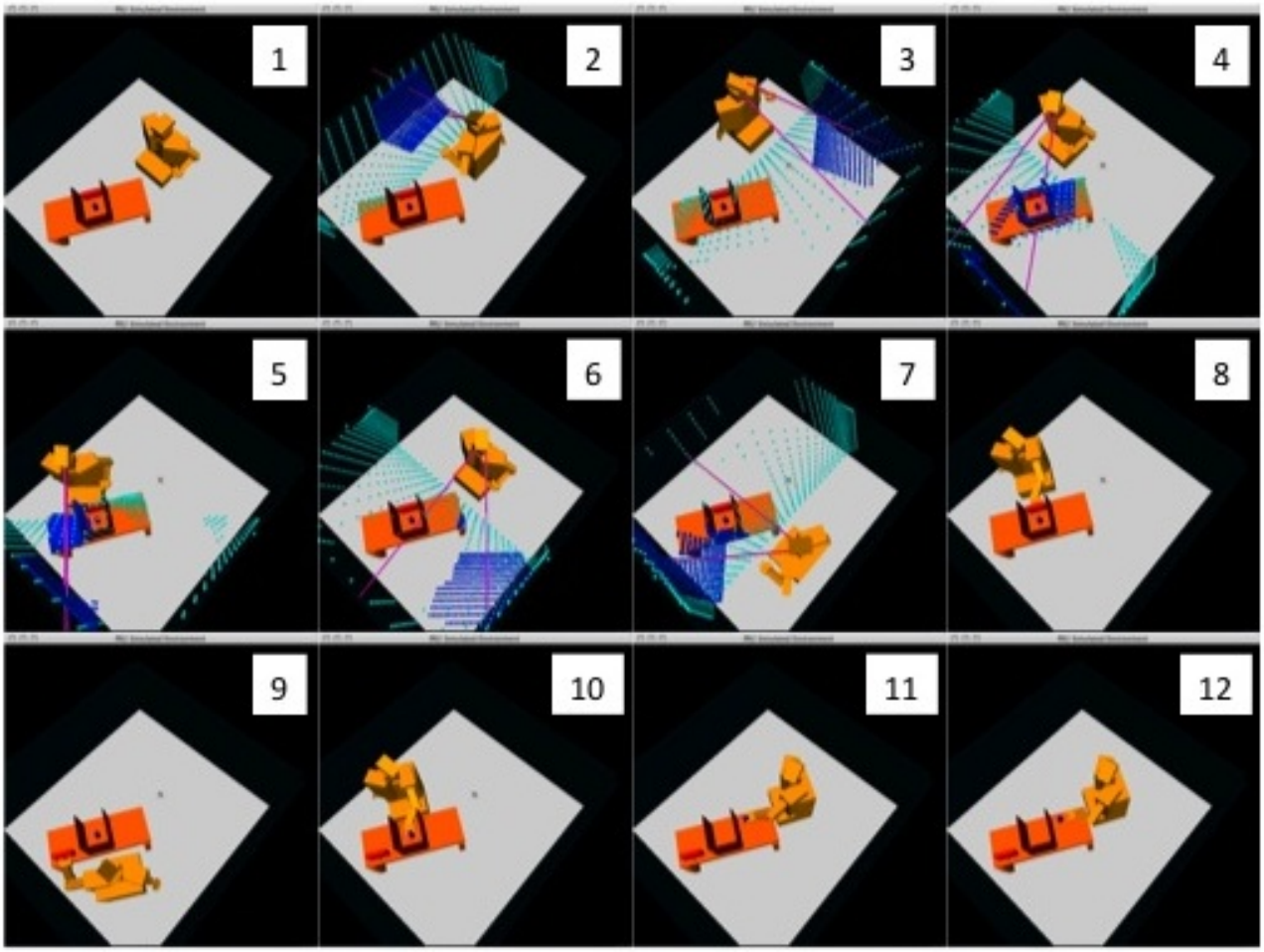

Fig. 2. The key steps in the execution of a plan to place the small blue cup in a target region at one end of the table. The red object is initially not in the object's model of the world. Scans with the head-mounted sensor are shown as dark blue points. Scans with the scanning laser are shown in cyan.

[4] D. Pangercic, M. Tenorth, D. Jain, and M. Beetz, "Combining perception and knowledge processing for everyday manipulation," in *IROS*, 2010, pp. 1065–1071.

[5] E. Sommerlade and I. Reid, "Probabilistic surveillance with multiple active cameras," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 440–445.

[6] C. Stachniss, *Robotic Mapping and Exploration*, ser. Springer Tracts in Advanced Robotics. Springer, 2009, vol. 55.

[7] P. Wang and K. Gupta, "View planning for exploration via maximal c-space entropy reduction for robot mounted range sensors," *Advanced Robotics*, vol. 21, no. 7, pp. 771–792, 2007.

[8] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Intell. Res. (JAIR)*, vol. 34, pp. 707–755, 2009.

[9] R. P. A. Petrick and F. Bacchus, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2004, pp. 2–11.

[10] D. Bryce, S. Kambhampati, and D. E. Smith, "Planning graph heuristics for belief space search," *Journal of Artificial Intelligence Research*, vol. 26, pp. 35–99, 2006.

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

[12] T. Erez and W. Smart, "A scalable method for solving high-dimensional continuous POMDPs using local approximation," in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[13] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proceedings of the Robotics Science and Systems Conference*, 2010.

[14] N. E. D. Toit and J. W. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 966–973.

[15] J.-A. Ting, E. Theodorou, and S. Schaal, "Learning an outlier-robust kalman filter," in *ECML*, 2007, pp. 748–756.

[16] P. T. Fletcher, S. Joshi, C. Lu, and S. Pizer, "Gaussian distributions on lie groups and their application to statistical shape analysis," in *Proceedings of Information Processing in Medical Imaging*, 2003, pp. 450–462.

[17] L. L. S. Wong, L. P. Kaelbling, and T. Lozano-Perez, "Collision-free state estimation," in *Submitted*, 2009.

[18] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[19] A. A. S. Souza, A. M. Santana, R. S. Britto, L. M. G. Gonalves, and A. A. D. Medeiros, "Representation of odometry errors on occupancy grids." in *ICINCO-RA (2)*, 2008, pp. 202–206.

[20] C. Boutilier, "Correlated action effects in decision theoretic regression," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997, pp. 30–37.

[21] C. Fritz and S. A. McIlraith, "Generating optimal plans in highly-dynamic domains," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI) Conference*, 2009.

[22] R. B. Scherl, T. C. Son, and C. Baral, "State-based regression with sensing and knowledge," *International Journal of Software and Informatics*, vol. 3, no. 1, pp. 3–30, 2009.