Manipulation Planning using Model-Based Belief Dynamics

Shiraj Sen[1] and Roderic Grupen[1]

Abstract— **Planning in partially-observable domains require an agent to fuse prior knowledge with observations to update belief and to search for optimal plans that reduce uncertainty with respect to the task. This requires a knowledge organization that captures the underlying dynamics of the belief space and its probabilistic dependency on actions. In this paper, we present a functional representation for organizing knowledge about the environment in terms of interaction statistics. The representation utilizes a uniform, domain-general description of state that applies to a wide variety of tasks. We show how a planning algorithm can exploit this knowledge representation to build plans directly in the space of control actions. Given incomplete state information, the planner interacts with the task to acquire the information required to solve it. We illustrate the approach in multiple demonstrations of an object recognition task.**

I. Introduction

Traditionally, planning in robotics involves designing a state representation and a model (implicit or explicit) for the state transition dynamics, both of which are specific to a particular task. The use of a task-specific representation, however, makes it impossible to reuse the model for other tasks. In order for robots to act autonomously, control dynamics must be modeled in a manner that is applicable to a wide variety of tasks and thus, support planning at run-time to manage uncertainty. This paper addresses these dual problems of modeling and planning by using a belief-space representation. We use the empirical "state-action-next state" representation that captures the probabilistic relationship between belief and action in manipulation tasks organized in terms of models of "objects".

Real-world tasks require planners to search for solutions in a partially observable state space. This often involves modeling the dynamics of the task as a Partially Observable Markov Decision Process (POMDP). POMDPs support reasoning about uncertain observations and stochastic actions by modeling the state of the world as a hidden variable. The policies learned from POMDP descriptions of a task specify actions that maximize the expected reward given a distribution over states. Unfortunately, the generic solution of a POMDP has been shown to be PSPACE-complete [1]. Although the complexity bounds sound disheartening, the worst case hardness does not mean that computing plans is impossible. This is because many domains offer additional structure that can ease planning difficulties. The Belief Space Planner (BSP) that we introduce, exploits models of belief space dynamics that describe objects. Our planner is very useful in the case where complete prior object-action models exist. Under these conditions, models provide the necessary structure to plan efficiently in partially-observable and initially unknown environments.

Hierarchical approaches to planning have been proposed to speed up the search for plans. Since the work of Sacerdoti [2] on the ABSTRIPS method that generated a plan in a hierarchy of abstraction spaces, many researchers have suggested a hierarchical approach to interleaving planning and execution [3]. Wolfe *et al.* [4] provided a task and motion planner based on hierarchical transition networks (HTNs) [5]. Platt [6] showed that one can compute reasonable policies in the belief space based on a local linearization of the belief space dynamics. The controller then selects actions based not only on the current most-likely state of the robot, but on the information available to the robot. This approach is promising, however in general belief space dynamics need not be linear, and hence resulting policies are applicable only in the vicinity of a locally stabilized region. Kaelbling and Lozano-Pérez [7], [8] proposed a hierarchical planner that sacrifices optimality quite aggressively, for efficiency, by having a planner that makes choices and commits to them in a top-down fashion in an attempt to limit the length of plans that need to be constructed, and thereby exponentially decreasing the amount of search required. Our approach is similar to the above, in which the robot selects the best possible action based on the current belief. However, the actions the robot selects can both be informative (that manipulates the mass of belief over states/actions) and functional (creating mechanical artifacts that address the task).

II. Models

Our computational representation of knowledge is based on a framework called the *control basis* [9] that makes use of low-level controllers and their dynamics to learn robot specific knowledge structures. The control basis is a discrete, combinatorial basis for continuous multi-objective control that is derived directly from the sensory, motor, and computational embodiment of the robot. Primitive actions in the control basis are combinations of potential functions ($\phi \in \Phi$), sensory ($\sigma \subseteq \Sigma$), and motor resources ($\tau \subseteq \mathcal{T}$) defined by three finite sets:

- Φ is a set of navigation functions whose gradients lead asymptotically to fixed points [10].
- Σ is a set of feedback entities that can be computed by applying operators to sensory signals.
- $\mathcal{T}$ is a set of motor units that actuate independent degrees of freedom in the robot. A motor unit consists of

[1]Shiraj Sen (shiraj@cs.umass.edu) and Roderic Grupen (grupen@cs.umass.edu) are with the Laboratory for Perceptual Robotics at the Department of Computer Science, University of Massachusetts Amherst, MA 01003, USA

an equilibrium setpoint controller on a single degree-of-freedom that accepts a reference value $\mathbf{u}_\tau$. Higher-level controllers submit patterns of real-valued references $\mathbf{u}_\tau$ to synergies of motor units, $\tau \subseteq \mathcal{T}$.

In this work, we use the shorthand notations c or ϕ_τ^σ to describe closed loop controllers.

We use a three-valued classifier for discretizing the dynamics of a continuous time control action [11]. The state γ^t of a controller $c(\phi, \sigma, \tau)$ at time t, is given by

$$\gamma^t(c) = \begin{cases} - & : \quad \phi \text{ has undefined reference} \\ 0 & : \quad |\dot{\phi}| > \epsilon \\ 1 & : \quad |\dot{\phi}| \leq \epsilon \end{cases} \tag{1}$$

where ϵ is a small positive constant [11].

In this state representation, '$-$' indicates that the controller is currently not applicable since the reference input signal σ required to compute the gradient, is not present in the feedback, '0' indicates that the controller is making progress but hasn't yet reached its target fixed point, and '1' denotes convergence/quiescence evaluated relative to a small positive threshold, ϵ. The undefined reference state '$-$' is an absorbing state since the potential function has no gradient in this state and hence can't make progress towards the goal. A collection of n distinct primitive control actions forms a discrete state space $\mathbf{s}^k = [\gamma_1^k \cdots \gamma_n^k] \in \mathcal{S}$ at time k.

In our framework, there are two distinct types of actions that share potential functions and effector resources, but are distinguished by the source of their input signals : TRACK and SEARCH. TRACK actions, ϕ_τ^σ preserve a reference value in the feedback signal that originate in the external environment e.g., the position of a color feature on the image plane. TRACK actions are guaranteed to achieve its objective provided the potential function has a defined gradient ($\gamma \neq$ '$-$'). However, in the absence of an external stimuli, the potential function provides no means for the controller to make progress. For example, a closed loop controller that tracks a visual reference in its environment, using a camera on a pan-tilt head, cannot achieve its objective if the sensory reference is not directly present in the field of view of the camera. In such cases, a robot needs to execute a sequence of actions that can *orient* its sensors to regions where a sensory reference for the control action is present.

SEARCH actions are of the form $\phi_\tau^{\tilde{\sigma}}$—their input, $\tilde{\sigma}$, is derived from probabilistic models describing distributions over effector reference inputs ($\mathbf{u}_\tau$) where TRACK-ing actions have converged in the past ($\gamma(\phi_\tau^\sigma) = 1$). Initially the distribution $\Pr(\mathbf{u}_\tau | \gamma(\phi_\tau^\sigma) = 1)$ is uniform; however, as it is updated over the course of many learning episodes, this distribution will reflect the long term statistics of the run-time environment. For example, such a controller can be used to direct the field of view of a robotic system to look at places where a color feature has been found in the past.

A. Control program

In this paper, we use hierarchical control programs (visual tracking, touching, grasping, picking up, placing, orbiting, and rotating objects) that were developed in previous work using hierarchical reinforcement learning [12], [13].

A control program is defined by the tuple $p = (\pi, \mathcal{M})$, where $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy indicating the probability of taking an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$, and $\mathcal{M}$ is a set of probabilistic models of the form $\Pr(\mathbf{u}_\tau | \gamma(a) = 0 \vee 1)$, where $\mathbf{u}_\tau$ is a vector of effector references inputs and $\gamma(a)$ is the action state. $\mathcal{A}$ defines the set of all available control actions—primitive controllers c and control programs p. The state of a control program is evaluated in a similar fashion as that of primitive controllers by monitoring the dynamics of the value function.

The control status observations made by the robot at any time t is given by:

$$z_i^t = \{\gamma^t(a_i) | a_i \in \mathcal{A}, \gamma \in \{-, 0, 1\}\} \tag{2}$$

The status of each control program captures a small part of the dynamics of the environment that is conditionally dependent on the system state. The status of several executing control programs is partial evidence of the expected state of the environment. We use a vector z_i ($i = 1, \ldots, N$) over N such control status values as a surrogate for the underlyting state. Associating a return status with every control program allows a planner to construct sequential controllers at a higher level of abstraction where the continuous state of a temporally extended control program is classified into a ternary representation ($\gamma \in \{-, 0, 1\}$).

B. Objects

Objects are modeled as spatial and temporal distributions over the converged status of control programs. Figure 1 shows a graphical model that encodes the probabilistic dependencies between variables comprising the object model. An object $o \in O$ induces a distribution over a set of M mutually exclusive aspects. An aspect $x \in X$ is a latent variable that models patterns over the status of several control programs. It implicitly captures the kinematic constraints and the sensor geometry for a constellation of control programs—sets of programs that can or cannot track stimuli simultaneously in the environment. An aspect can include feedback from a set of visually-referenced actions, together with feedback from a set of force-referenced actions. The features comprising a visual aspect are mutually consistent with line of sight constraints (e.g., set of visual features on an object that can be tracked simultaneously). Elements of the haptic aspect are mutually consistent with kinematic reachability constraints.

Each aspect $x \in X$ induces a distribution over the state of N-TRACKing programs. There can be multiple instances of each aspect within an object. Each control program is represented by a Bernoulli random variable γ_j describing the state of each associated action. ($\gamma_j = 0 \vee 1$, if the action has a gradient, $\gamma =$'-', if the gradient is undefined).

The dependencies between the aspects (x^t and x^{t+1}) over two time steps $(t, t+1)$ and the control program being executed (a^t) is encoded by the two time slices of the Dynamic Bayesian Network (DBN). This part of the model

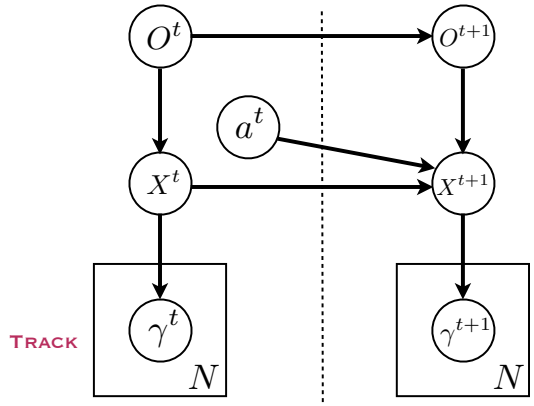


Fig. 1. Figure shows a Bayesian network model representing objects O as a temporal distribution over aspects X. An aspect induces a distribution over the state of N tracking programs (γ_j) as shown by the plate model. The two time slices in the model show the logical dependencies between aspects and an orienting action a. O, X and a are modeled as multinomial random variables. γ_j is modeled as a Bernoulli random variable.

describes how executing an action on an object influences the set of accessible control programs. For example, a hammer's handle affords the action of grasping, however if the handle is out of reach, the robot might have to use a non-prehensile tactile controller that pulls the hammer closer before it can succeed in grasping it. In this case, "pulling" changes the aspect of the object in a manner that supports the goal of grasping. Modeling objects in the world in terms of the properties derived from controllable actions and the spatial relationships between them allows an agent to use the same model for all interactions (and sequences thereof).

In [14], we presented results on learning a visual model of the objects autonomously and using it for the task of visual object recognition. Model learning will not be the focus of this paper.

III. Planning

Since the true state[1] s^{t+1} of the system cannot be observed, it must be inferred from observations z^{t+1} made after taking action a^t. The observation is related to the state by the likelihood function $\Pr(z^{t+1}|s^{t+1}, a^t)$, which is proportional to the probability that observation z^{t+1} is the result of taking an action a^t to reach a particular state s^{t+1}. The probability density function $\Pr(z^{t+1}|a^t)$ of the observation is obtained by marginalizing over all states

$$\Pr(z^{t+1}|a^t) = \sum_{s^{t+1}} \Pr(z^{t+1}|s^{t+1}, a^t) \Pr(s^{t+1}) \quad (3)$$

The important quantity in this formalism is the action a^t. Since the likelihood function $\Pr(z^{t+1}|s^{t+1}, a^t)$ is conditioned on the action, it is clear that actions influence observations. The goal is to update belief in the true state s^{t+1}, given the observation z^{t+1}.

Our planner selects actions that reduce the uncertainty of the state estimate optimally with respect to the task. Entropy measures the amount of uncertainty in the value of a random variable s^t.

$$H(s^t) = -\sum_{s^t} \Pr(s^t) \log(\Pr(s^t)) \quad (4)$$

The entropy is zero if the state is uniquely determined; it reaches its maximum if all states are equally likely.

In information theory, mutual information (MI) defines how much uncertainty is reduced in a random variable (s^t) provided an observation (z^{t+1}) is made. Since the information flow depends on the action a^t, we need to define conditional MI:

$$I(s^t; z^{t+1}|a^t) = H(s^t) - H(s^t|z^{t+1}, a^t) \quad (5)$$

The optimal next action $\hat{a}^t$, given a belief over states $\Pr(s^t)$ and observation model $\Pr(z^{t+1}|s^t, a^t)$ is

$$\hat{a}^t = \arg\max_{a^t} I(s^t; z^{t+1}|a^t) \quad (6)$$

Initially with no experience to draw on and before any observations are made, $\Pr(s^0)$ is initialized uniformly. However, as actions are executed by the planner to optimize mutual information, new observations are used to update the *a posteriori* probability of each state s^t,

$$\Pr(s^t|z, a) = \frac{\Pr(z|s^t, a) \Pr(s^t)}{\Pr(z|a)} \quad (7)$$

In the next time step, the planner uses the set of *a posteriori* probabilities as *a priori* probability for s^t. This allows the planner to utilize its current belief to act optimally.

All tasks in our framework can be posed as the manipulation of belief—condensing belief over objects (recognition), aspects (pose recovery), or status of control programs (tracking an external stimulus). In the next few subsections, we describe in detail how to use the same planner for building plans for each of the tasks.

A. Belief over Objects

Object recognition is still an open problem. From the choice of features to the actual classification problem, we are still far from possessing a global recipe that would allow for a complete discriminative approach to recognition. The large majority of work on object recognition has been focused on offline, database driven tasks. Probably the biggest challenge that arises from using such databases is the inability to exploit discriminative views of objects. Utilizing the belief-space representation, the task of object recognition is described as the process of selecting actions that maximally reduce the uncertainty over objects—condensing belief over objects given the aspect x. Since the aspect of an object is a latent variable that needs to be inferred from the observations, we make use of the maximum likelihood estimate over aspects. Using Equation 6, the optimal action to execute is given by

$$\hat{a}^t = \arg\max_{a^t} I(o^t; x^{t+1}|a^t) \quad (8)$$

[1]The state, s^{t+1} described in this section should not be confused with state s described in Section II.

Using the definitions of entropies and our object model, the conditional MI is given by

$$I(o^t; x^{t+1}|a^t) = \sum_{o^t} \sum_{x^{t+1}} \Pr(o^t) \Pr(x^{t+1}|o^t, a^t) \times \log \left(\frac{\Pr(x^{t+1}|o^t, a^t)}{\Pr(x^{t+1}|a^t)} \right) \quad (9)$$

where, the maximum likelihood estimate over aspects, x^t_{ML} is utilized to compute the observation likelihood, $\Pr(x^{t+1}|o^t, a^t) = \Pr(x^{t+1}|o^t, a^t, x^t_{ML})$.

B. Belief over Aspects

The goal of certain planning tasks require manipulating objects to achieve a particular pose. A pose of an object implicitly defines a viewpoint on the object with respect to the robot. This information is modeled by the latent aspect variables in our probabilistic representation. Thus, the task of re-orienting objects is described in our framework as reducing belief over aspects to achieve a goal aspect. The optimal action that maximally reduces the uncertainty over aspects is given by Equation 6, where

$$I(x^t; z^{t+1}|a^t) = \sum_{x^t} \sum_{z^{t+1}} \Pr(x^t) \Pr(z^{t+1}|x^t, a^t) \times \log \left(\frac{\Pr(z^{t+1}|x^t, a^t)}{\Pr(z^{t+1}|a^t)} \right) \quad (10)$$

Once the uncertainty over aspects is completely reduced, the object model is used as a forward model to sequence a set of actions to achieve the goal aspect.

C. Belief over Program Status

Certain planning tasks require achieving a particular state for a control program without the need to recognize the object. For example, if the task is to achieve a grasp, it is not necessary to completely disambiguate the object before performing the grasp action. Since the state of a control program is observable, given a goal observation $\hat{z}^t$ at time t, the optimal action to choose is given by

$$\hat{a}^t = \arg\max_{a^t} I(\hat{z}^t; z^{t+1}|a^t) \quad (11)$$

The conditional Mutual Information can once again be computed from our object models. This shows that expressing tasks as manipulating beliefs over certain random variables provides a consistent way of expressing tasks and computing plans.

IV. Experiments

In this paper, we explore in detail the capabilities of our model and planner for the task of object recognition. The experiment was conducted using the UMass uBot platform—a two-wheeled dynamically balancing mobile manipulator with two 4 degrees-of-freedom (DOF) arms and a trunk rotation. The robot's sensor package includes encoder feedback on all 11 DOF and an ASUS RGB-D camera on a 1 DOF head. Control is implemented in Robot Open Source (ROS) publish-subscribe operating system [15].

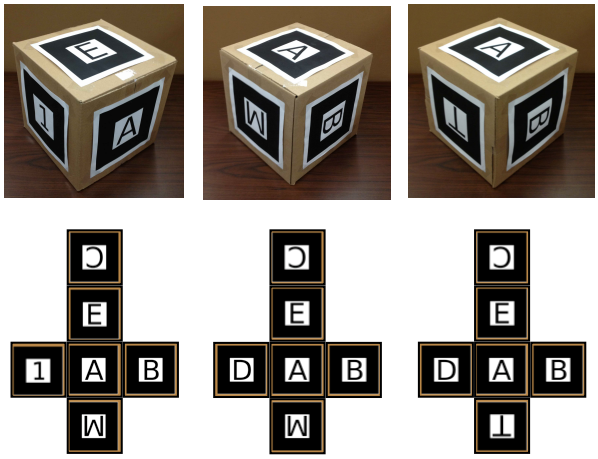


Fig. 2. A flattened image of three of the boxes used in the experiments showing the various ARtag features on each of its faces. It should be evident from the images that certain features are repeated over multiple objects.

To illustrate the approach to belief-space planning in its simplest form, an extremely simple experimental object called the ARcube is employed. It consists of a roughly 28 cm cube made out of cardboard box, weighing about 0.25 kg with distinctive ARtag surface markings chosen from a set of 13 alternatives (A, B, C, D, E, M, T, 1, 2, 3, 4, 5, 6). These ARtags were trained using the ARToolkit [16]. Each of the ARcubes used in these demonstrations incorporate an unique combination of six of these ARtags. Though the use of ARtag features sound simplistic, the above 13 alternatives provide a rich set of possible objects (in the order of 10^6) that can be generated by choosing 6 of the ARtags. In our experiments, the number of generated ARcube models are varied from 10 to 1000. Figure 2 unfolds the surface geometry of three of the ARcubes to illustrate the faces of each box with the associated ARtag. Some of the tags are repeated in multiple objects. These ambiguous features lead to partially observable state, therefore, detecting single tags is not enough for the robot to completely differentiate these objects.

The model of the objects contain a set of visual tracking actions (TRACK$_A$, ... TRACK$_T$, TRACK$_1$, ... TRACK$_6$) that track each of the ARtag features. The models also contain a set of actions that manually interact with the objects :

- GRASP : Moves the robot base and the arms to obtain and track grasp forces in the end effector.
- ROTATE+Y : Rotate the box counterclockwise around the Y-axis.
- ROTATE-Y : Rotate the box clockwise around the Y-axis.
- ORBIT+Z : Orbit the box by moving counterclockwise around the Z-axis of the box.
- ORBIT-Z : Orbit the box by moving clockwise around the Z-axis of the box.

GRASP, ROTATE+Y, and ROTATE-Y are TRACK-ing actions that depend on probabilistic search distributions describing the placement of contacts that cause force closure and a

net moment on the object respectively. These distributions are defined spatially in terms of sufficient combinations of visual trackers. ORBIT+Z and ORBIT-Z are SEARCH actions that reliably change the visual aspects by driving the mobile manipulator to a new position and heading relative to the object's pose estimate. Figure 3 and Figure 4 show the effect of executing the ROTATE+Y and ORBIT-Z actions respectively on a box.

The task employs model-based belief dynamics coupled with mutual information to select a pattern of actions that optimally discriminate between multiple objects. The planning algorithm proceeds by first estimating the distribution of belief over states defined by the objects. The estimated state is used to compute the action that maximally reduces uncertainty. Figure 5 shows how the *a posteriori* probability and entropy over 10 objects change during one such execution of the planner. Initially, when the robot has made no observations, it has an uniform belief over objects. However, as the planner executes actions and makes new observations, it fuses these observations with its belief to compute a new posterior over objects. This process gets repeated until the object has been recognized—which is given by the entropy falling below a particular threshold.

This procedure leads to different plans based on the initial presentation of the object and the history of the agent's observation. We performed 20 runs of the experiment with an object chosen at random and placed in front of the robot at a random pose. The robot was able to successfully recognize the object among these small set of distractors 100% of the time. When the number of possible objects was increased from 10 to 100, there wasn't any noticeable decrease in performance of the planner. Figure 6 shows how the entropy and the belief over 100 objects changed during the execution of one such trial. Initially, when no observations have been made, the planner assumes an uniform prior over all objects ($\Pr(o_i) = 1/100$). However, the agent's belief very quickly converges on a few objects after the execution of the first action.

Figure 7 shows how the planner's uncertainty over the 100 objects change with the number of executed actions. For the sake of clarity, only 5 trials are shown on the plots. As is evident from the plots, the entropy decreases monotonically with increasing actions. This indicates that the information theoretic planner is making optimal decisions to reduce uncertainty at every iteration.

Furthermore, since the planner only computes the next best action to execute, the computation time required is minimal. Figure 8 shows how the average planning time varies with the number of available object models. All experiments were conducted on a single quad-core machine with 16GB of RAM. The machine performed both the vision processing as well as the action selection. The planning code was written in Python. As can be seen from the table, the planner can build plans in real-time even for 1000 objects. This is because after every observation, the planner searches for plans in a reduced set of possible objects. Thus the planning speeds up with increased number of observations. Figure 8 also tabulates the average number of actions required to recognize the objects. The actions are averaged over 20 trials. The table shows that as the number of models increase, the performance of the planner degrades. This can be attributed to the greedy nature of the plans where only the action that maximally reduces uncertainty over the next time step is chosen (as opposed to planning a sequence of actions that optimally reduces uncertainty).

V. CONCLUSION

This paper presents a Bayesian framework that can be used by a robot to model its environment in terms of distributions over the status of control programs. The knowledge accumulated by the robot models the dynamics of the environment that can be learned by direct interactions with the world. Since the focus of the paper was on showing the strengths of Belief Space Planning and action-based representations, the simple models employed were not learned but hand-coded. The experiments show that an information theoretic planner coupled with a model for constructing belief space dynamics efficiently generates all contingencies (given a complete object model) for achieving the goal by executing the action associated with robot's present belief of the state. Expressing goals as discrete assertions over the space of actions allows a planner to search for plans in a discrete observation space. As part of our future work, we are interested in studying how such a planner can be extended to plan multi-step plans (as opposed to one-step planning). We are also interested in extending the plans as well as the models to multi-object assemblies.

ACKNOWLEDGMENTS

This material is based upon work supported under Grants NASA-GCT-NNH11ZUA001K and ONR-MURI-N000140710749.

REFERENCES

[1] C. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Math. Oper. Res.*, vol. 12, no. 3, pp. 441–450, Aug. 1987. [Online]. Available: http://dx.doi.org/10.1287/moor.12.3.441

[2] E. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence*, vol. 5, pp. 115–135, 1974.

[3] I. Nourbakhsh, "Using abstraction to interleave planning and execution," in *Proceedings of the Third Biannual World Automation Congress*, 1998.

[4] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *ICAPS*, 2010.

[5] D. Nau, O. Ilghami, U. Kuter, J. W.Murdock, D. Wu, and F. Yaman, "Shop2: An htn planning system," *Journal of Artificial Intelligence Research*, vol. 20, pp. 379–404, 2003.

[6] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[7] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE Conference on Robotics and Automation Workshop on Mobile Manipulation*, 2010.

[8] ——, "Pre-image backchaining in belief space for mobile manipulation," in *International Symposium on Robotics Research (ISRR)*, 2011.

[9] M. Huber, "A hybrid architecture for adaptive robot control," Ph.D. dissertation, Department of Computer Science, University of Massachusetts Amherst, 2000.

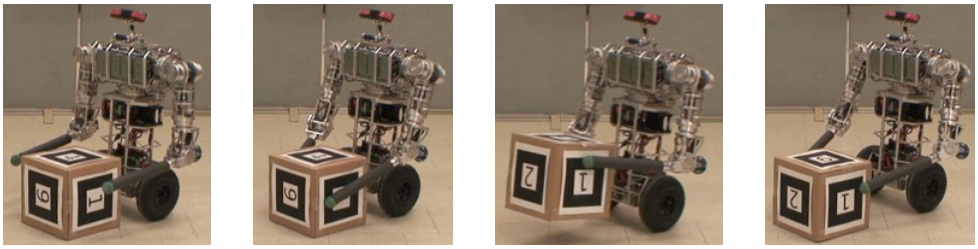


Fig. 3. ROTATE+Y rotates the box counterclockwise around the Y-axis. The effect of taking the ROTATE+Y action is to re-orient the box making the ARTag feature "6" move to the top of the box.

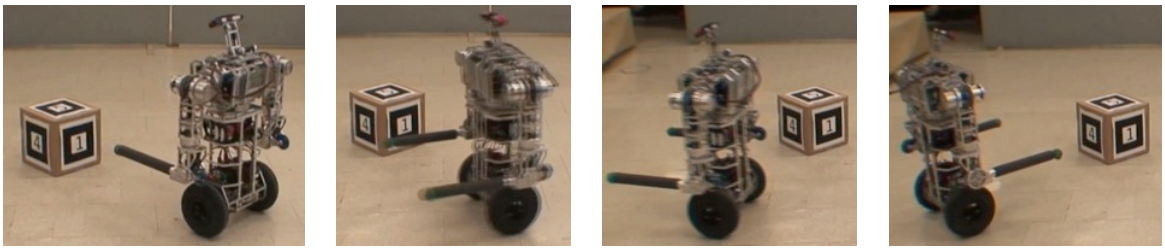
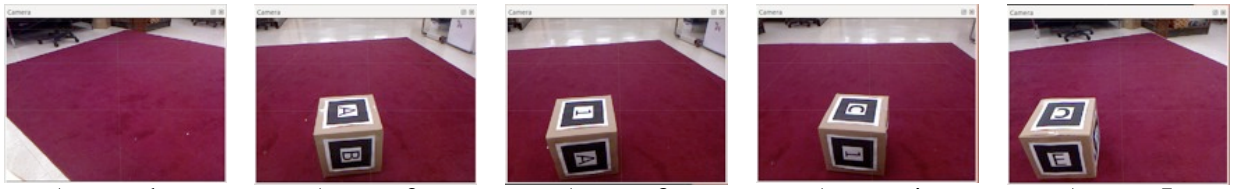


Fig. 4. ORBIT-Z re-orients the robot by moving it clockwise around the box. The effect of taking the ORBIT-Z action on this box is that the ARTag feature "4" becomes visible.



Aspect	Obj_1	Obj_2	Obj_3	Obj_4	Obj_5	Obj_6	Obj_7	Obj_8	Obj_9	Obj_10	Entropy
Aspect 1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	1.000
Aspect 2	0.20	0.20	0.20	0.00	0.20	0.00	0.20	0.00	0.00	0.00	0.698
Aspect 3	0.00	0.00	0.33	0.00	0.33	0.00	0.33	0.00	0.00	0.00	0.477
Aspect 4	0.00	0.00	0.00	0.00	0.50	0.00	0.50	0.00	0.00	0.00	0.301
Aspect 5	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.000

Fig. 5. The robot performs the action sequence: TRACK-A→ROTATE-Y→ROTATE-Y→ORBIT+Z as part of the policy to reduce uncertainty over 10 objects. The planner fuses information from interactions with the object to update distributions of belief over objects until no uncertainty remains. The table shows the posterior probability over objects after each new action-observation. The rightmost column of the table shows the entropy over the object distribution after every state estimate.

[10] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[11] S. Hart and R. Grupen, "Learning generalizable control programs," in *Transactions on Autonomous Mental Development*, Zaragoza, Spain, 2010, pp. 1–16.

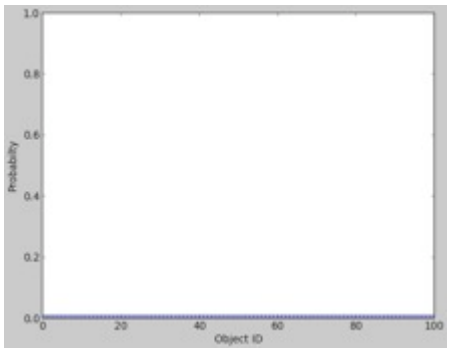
[12] S. Hart, S. Sen, and R. Grupen, "Intrinsically motivated hierarchical manipulation," in *Proceedings of 2008 IEEE Conference on Robotics and Automation*, Pasadena, CA, 2008.

[13] ——, "Generalization and transfer in robot control," in *Proceedings of 8th International Conference on Epigenetic Robotics*, Brighton, UK, 2008.

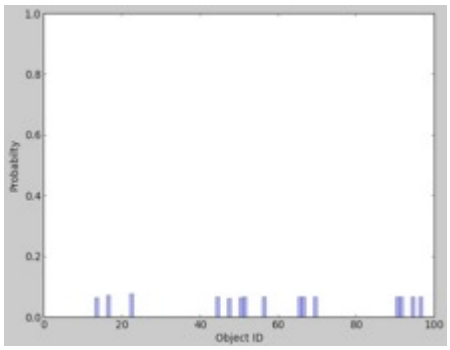
[14] S. Sen, "Bridging the gap between autonomous skill learning and task-specific planning," Ph.D. dissertation, Department of Computer Science, University of Massachusetts Amherst, 2013.

[15] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

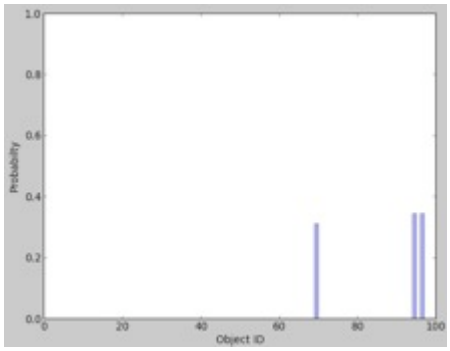
[16] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, ser. IWAR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 85–. [Online]. Available: http://dl.acm.org/citation.cfm?id=857202.858134



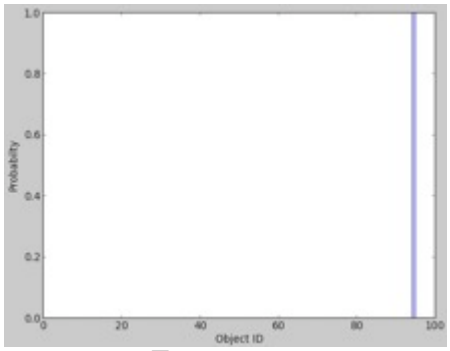
Entropy: 2.0



Entropy: 1.175



Entropy: 0.476



Entropy: 0.0

Fig. 6. The robot performs the action sequence: TRACK-A→ROTATE-Y→ORBIT-Z as part of the policy to reduce uncertainty over 100 objects and recognize the object. The bottom row shows the entropy over the object distribution after every state estimate.

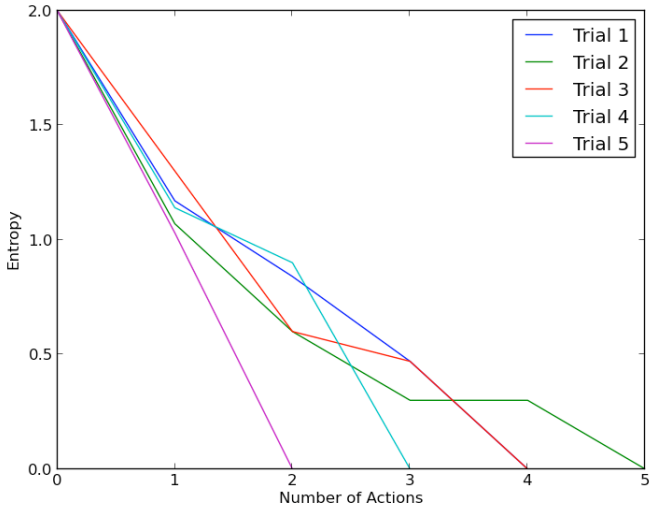


Fig. 7. The plot shows the change in object entropy as a function of the number of actions executed by the robot. The entropy is computed over 100 objects.

Number of Objects	Average number of actions	Average planning time (in secs)
10	3.8	0.26
100	4.2	0.43
1000	5.8	2.29

Fig. 8. The table shows how the variation in performance of the planner with the number of objects. The second column shows the average number of actions required to recognize the object. The third column shows the average time required by the planner to select each action.