BRIDGING THE GAP BETWEEN AUTONOMOUS SKILL LEARNING AND TASK-SPECIFIC PLANNING

A Dissertation Presented

by

SHIRAJ SEN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2013

Computer Science

BRIDGING THE GAP BETWEEN AUTONOMOUS SKILL LEARNING AND TASK-SPECIFIC PLANNING

A Dissertation Presented

by

SHIRAJ SEN

Approved as to style and content by:

Roderic Grupen, Chair

Shlomo Zilberstein, Member

Andrew Barto, Member

Maryjane Wraga, Member

Lori A. Clarke, Department Chair
Computer Science

To my family and friends.

ACKNOWLEDGEMENTS

This dissertation, while it bears only my name, is the culmination of the support I have received from a bunch of people. My interest in robotics started in my undergraduate years in India, while taking a course on Artificial Intelligence. It has been nearly nine years since I took that course, and my interest in this area has only increased over time. For this, I would like to thank Rod Grupen for guiding me over the years through our countless whiteboard discussions and spur-of-the-moment meetings, where I could drop by his office at any time of the day when he is available and exchange ideas.

I would like to thank Shlomo Zilberstein, Andy Barto, and Maryjane Wraga for their guidance that helped make my thesis a lot more focussed. I would like to thank Oliver Brock for the enthusiastic discussion we had the second day I was at UMass regarding why robotics is such an interesting field to study. I remember my discussion with him regarding the "Work Turing" test—where a robot performs certain manipulation task in a closed room, and the human is not able to differentiate correctly if the task was performed by a human or a robot. Many of the ideas I have worked on over the years have been inspired by this central goal. I would also like to extend my gratitude to Leeanne Leclerc and Priscilla Scott for pushing me to graduate and taking care of all the administrative things.

I would like to thank the members of the Laboratory of Perceptual Robotics for all the help they provided in developing and implementing the various ideas: Tom Billings, Steve Hart, Emily Horrell, Hee Tae Jung, Scott Kuindersma, Yun Lin,

Shichao Ou, Rob Platt, Dirk Ruiken, Grant Sherrick, Andrew Stout, John Sweeney, Takeshi Takahashi, and Dan Xie. I would specially like to thank Steve Hart, discussions with whom early on lead to the development of many of the ideas in this thesis.

I would like to thank my friends who stuck with me for the last few years and helping me forget about work through their exuberant enthusiasm and light-hearted jokes: Emmanuel Cecchet, Yariv Levy, Gal Niv, Margaret Sharron, and Borislava Simidchieva. I would like to thank my amazing housemates over the years for all their support: Matt Burak, Marc Cartright, Stacy Collins, Ranojoy Duffadar, Sam Huston, Sreekumar Kuriyedath, Ilene Magpiong, Dirk Ruiken, Tejinder Singh, and Susie.

I would like to thank many of my friends for their friendship, that were forged over the course of my stay at UMass: Aruna Balasubramanian, Niranjan Balasubramanian, Ethem Can, Stefan Christov, Francesca Colantuoni, David Cooper, Erin Cooper, Laura Dietz, Henry Feild, Jacqueline Feild, George Konidaris, Laura Sevilla Lara, Katerina Marazopoulou, Gaurav Mathur, Megan Olsen, Ashmita Sengupta, Sarah Osentoski, Tim Wood, and many more.

Finally, I would like to thank my family without whose patience and unyielding support this journey wouldn't have even gotten started in the first place.

ABSTRACT

BRIDGING THE GAP BETWEEN AUTONOMOUS SKILL LEARNING AND TASK-SPECIFIC PLANNING

FEBRUARY 2013

SHIRAJ SEN

B.Sc., INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

M.Sc., INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic Grupen

Skill acquisition and task specific planning are essential components of any robot system, yet they have long been studied in isolation. This, I contend, is due to the lack of a common representational framework. I present a holistic approach to planning robot behavior, using previously acquired skills to represent control knowledge (and objects) directly, and to use this background knowledge to build plans in the space of control actions.

Actions in this framework are closed-loop controllers constructed from combinations of sensors, effectors, and potential functions. I will show how robots can use reinforcement learning techniques to acquire sensorimotor programs (*skills*). The agent then builds a functional model of its interactions with the world as distributions over the acquired skills. In addition, I present two planning algorithms that can

reason about a task using the functional models. These algorithms are then applied to a variety of tasks such as object recognition and object manipulation to achieve its objective on two different robot platforms.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Humans interacting with their environment show a remarkable amount of *dexterity*. Humans do not have a single solution to a particular problem; rather over time, develop a suite of alternate solutions that can achieve the same task. Dexterity can be defined as the "ability to find a motor solution for any external situation, that is, to adequately solve any emerging motor problem correctly, quickly, rationally, and resourcefully" [8]. Bernstein argued that dexterity lies not in the motor act itself, but is revealed by its interaction with the changing external conditions, with uncontrolled and unpredictable influences from the environment.

Dexterity requires quickness of wits (noticing the environment has changed), quickness of resolution (quickly finding a solution), and qualitative quickness of movements. This requires an agent to be able to anticipate (have forward models) the effect of its actions and be able to react accordingly to achieve its objective. Robotics researchers working on dexterous robots usually focus their attention on only one aspect of the problem, rather then considering a holistic approach to the problem of achieving dexterous mobility and manipulation in robots. This dissertation is an attempt towards achieving dexterous behavior in robots. I build upon the work of Stephen Hart of learning skills autonomously by an intrinsically motivated agent [44] to show how robots can organize its knowledge about the dynamics of the world in a manner that supports high-level reasoning and knowledge re-use. I will show how agents can

act in the presence of uncertainty to quickly come up with multiple competing motor solutions that achieve its objective.

1.1 Representation for Planning and Control

An intelligent agent must reason about its own sensorimotor skills, and about the relationship between these skills and goals under run-time conditions. This requires the agent to represent knowledge about its interactions with the world in a manner that supports reasoning. Since the early 1970s, the AI and robotics community has been concerned with the design of efficient representations for automated robot control. However, most of these representations tend to tackle only one part of the problem—making either the control or planning problem easier.

One solution to the hybrid planning and reactive control problem is to adopt a two-level model: at the upper level, a planner sequences a set of subgoals to be achieved based on the available knowledge and the task at hand; at the lower level, a controller achieves these goals while dealing with the environmental contingencies (e.g., [36, 3]). The controller must be able to satisfy planned goals to the highest degree possible while trading off between multiple low-level goals (e.g., avoid joint limits and collision). It is, however, a challenge to develop a complex controller that juggles goals at both levels—the two control problems are treated as if they are uncoupled when that is clearly not the case.

Frustrated by such problems, many researchers are exploring other techniques for generating intelligent behavior without explicit representations of the kind used in symbolic AI. One of the most influential examples is the work of Brooks who outlined an approach to building robots based on the *subsumption architecture* [11, 12, 13]. Brooks stated that intelligence is an emergent property of certain complex systems

and can be generated without explicit representations and abstract reasoning. He stated that 'real' intelligence is situated in the world, and not in disembodied systems such as theorem provers or expert systems. In this proposition, intelligent behavior arises as a result of the agent's interaction with the environment and not based on some prior logic provided by a third party. This is also the basis for the knowledge structure and representation that I am presenting. The knowledge accumulated by the robot is not based on prior models constructed by third party knowledge engineers. It is based on models of the environment learned by the robot by direct interaction with the world. The model learned thus establishes only those aspects of the world that support controllable chains of inference. This naturally structures problem solving by ignoring parts of the state space that are not relevant to the robot or are expensive or difficult to discern. In Chapter 3, I present a knowledge representation grounded in robot's own interactions with the world combined with a control framework that supports multi-objective control.

1.2 Real World Planning

Planning has been closely related to the implementation of artificial agents since the birth of AI. It has been long understood that an intelligent agent needs to have some way of automatically designing a course of action that achieves its objective. Over the years, increasingly sophisticated planning algorithms have been developed for motion and manipulation planning. LaValle [70] and Ghallab *et al.* [38] present a comprehensive survey of the various planning techniques that have been developed for planning under uncertainty for both real and simulated worlds. Despite the immense volume of work, most researchers would accept that the problem is not solved. The underlying problem seems to be the expressiveness and precision of forward models in robotics as well as the complexity of searching a very high dimensional state space

efficiently.

The classical planning problem of finding a finite sequence of actions that will transform a given initial state to a state that satisfies a goal specification, is computationally difficult. In the traditional context, in which actions are represented using the STRIPS representation and the initial and goal spaces are specified as lists of literals, even restricted versions of the planning problem are known to be PSPACE-complete [32]. Although the complexity bounds sound disheartening, the worst case hardness result does not mean that computing plans is impossible. This is because many domains offer additional structure that can ease planning difficulties.

The focus of research on planning is often the design of efficient algorithms for use in structured domains that encode only the essential features. A lot of effort has been put into constructing implicit encodings of problems in the hope that the entire state space does not need to be explored to solve the problem. By assuming a task-specific representation, general-purpose planning algorithms have been designed and proved to be correct and complete in some cases [33, 17]. Logic frameworks are popular for constructing such representations, since they can represent certain kinds of planning problems very compactly. Also, the resulting representation is rational in that it produces outputs and explanations. Although these systems represent a significant technical breakthrough, the logic framework is severely limiting when applied to the real world. For example, these representations do not address the possibility that complete state knowledge about the world might not be available to the agent. Thus, it isn't possible to plan a complete sequence of actions from the present state to the goal in advance. Moreover, the world can change independently of actions taken by the agent, or there can be many situations when the planning agent isn't completely certain about the state of the world. A planner needs to adapt to run time feedback

by taking task directed exploratory actions that yield better predictions and improve planning performance while contributing simultaneously to improved forward models.

1.3 Contributions

This dissertation makes two main independent contributions to the field of robotics. A third contribution arises from collaborative work with others in the Laboratory for Perceptual Robotics (LPR) related to autonomous skill acquisition and demonstrations of empirically derived knowledge from these *skills*.

- *Skill-based Representation* : I present a functional representation for organizing a robot's knowledge about its environment in terms of its interaction statistics. The representation utilizes a uniform description of state, which is not specific to a particular task; it is domain general and applicable over a wide variety of tasks. Actions in this framework are closed-loop controllers constructed from combinations of sensors, effectors, and potential functions. In earlier work, Hart [44] showed that sensorimotor programs (schemas) can be acquired using intrinsically motivated reinforcement learning [4].

  I will show how a robot can utilize this uniform state representation to learn probabilistic models of its environment. The models capture the functional description of the environment as spatial and temporal distributions over the state of acquired skills. While the presented representation is rather general, my work will concentrate on models pertaining to rigid objects. Chapter 3 will show how such a model can be learned and used for various tasks such as object recognition and manipulation.

- *Task-specific Planning* : I will present two algorithms for planning in the space of control actions, supported by the functional object models. Chapter 4 presents

a planner that uses an information theoretic metric for planning tasks. However, the tasks that such a planner can perform is limited to those where the goal is to reduce uncertainty over state (For example, object recognition), as opposed to achieving a particular goal state. Chapter 5 presents a planner that alleviates this problem by allowing both recognition and goal state achievement tasks. I will show how a planner, in the presence of partial state information, can interact with the world in a task directed manner that leads to the discovery of control knowledge and dynamics of the world and concurrently uses the gained knowledge to make progress towards the goal.

CHAPTER 2

BACKGROUND

This chapter presents a survey of various representations that have been presented in the literature for skill learning and planning. We conclude with a description of our representational foundation for states and actions that will be used for modeling control knowledge and allows for seamless integration of probabilistic planning schemes with low level controllers.

2.1 Knowledge Representation

The problem of integrating low-level controllers with high-level planners introduces significant representation difficulties. This is because the requirements of controllers are different from those of traditional planners. Traditional closed loop controllers require high-bandwidth access to feedback from the environment. Deliberating about the outcomes of an action requires representing these possible outcomes and simulating their effects under run-time conditions. In practice, this is impossible for two reasons. First, all the information necessary for an accurate simulation may not be available. In real environments, many parameters are unknown or hidden and are not under the agent's control. Second, even if the robot has access to complete information for simulation, it might be so computationally expensive that the real world may change faster than that of the simulation. As a consequence, representations for planning often rely on a level of abstraction that is incompatible with the high-fidelity, high-bandwidth feedback required for control. In the next few subsec-

tions, I will present an overview of various representations that have been developed for performing planning and control.

2.1.1 Logic Based Representation

Logic based representations have been used in robotics and AI to represent planning and control problems. The first application of planning, in fact, was robot control: the STRIPS [33] was used to generate plans, i.e., sequences of abstract high level actions for the robot SHAKEY [82]. STRIPS takes a symbolic description of the world and the desired goal state, a set of action descriptions that include the initial and final conditions associated with an action, and then attempts to find a sequence of actions that will achieve the goal. The algorithm uses a rather simple means-ends analysis, which involves matching the post-conditions of an action against the desired goal.

Sacerdoti [94] represented the problem domain as a hierarchy of abstractions in which successively finer levels of detail are added to an abstract plan. The planner achieves significant increases in performance by first searching for a solution in the most abstract level of problem description, a simplified view of the problem space in which unimportant details are ignored. He further showed how the same logical representation can capture the essential non-linear nature of plans [95] by representing a plan as a partial ordering of actions in time. By avoiding premature commitments to a particular order for achieving the subgoals, this representation can easily and directly deal with problems that are otherwise very difficult to solve. However, as it turned out, it is difficult to transform abstract actions into motor controllers flexible enough to meet the goals of the abstract action given partial information and uncertainty in the real world implementation.

Saffiotti *et al.* [96] presented an approach for integrating planning and control based on *control schemas* that link physical movements to abstract action descriptions. Their approach is focussed on performing planning using the framework of multi-valued logic. Multi-valued logic can be viewed as logic of graded preference, where the truth value of a proposition P in a world can be interpreted as the utility, or *desirability*, of being in that world from the point of view of P. It represents degrees of truth on a numeric scale, thus providing an ideal framework to merge planning, typically expressed in symbolic terms, with control, typically expressed in numeric terms. They start from the definition of basic units of control that map each state to a measure of preference (or *desirability function*) over the space of all possible commands. The idea here is that different commands can generate, to a greater or a lesser extent, the same type of movement. *Control schemas* are composed by combining the corresponding desirability functions via the operators of multi-valued logic. These control schemas were then "lifted" to the level of abstract actions in an environment that can be used by a planner. Here, they used two key notions: that of "embedding" in the environment, by anchoring the agents internal state (used by the control schemas) to external objects (used by the planner) through perception, and contextual structure provided by the circumstances of execution. A control schema, together with a set of object descriptors and a contextual condition, is packaged into a *behavior*. Behaviors play the role of situated actions: they indicate which movements should be performed under what circumstances and with respect to which objects—bridging the gap between abstract action descriptions and physical control. Saffiotti and his colleagues showed how this logical representation can be used for automatic planning of complex behavior.

A robot that uses logical reasoning can sound highly compelling, since all the agent will then require is a representation of the knowledge expressed in logic and a

theorem prover as part of the problem-solver. However as Wooldridge [118] points out, to build an autonomous robot with such capabilities, two important problems need to be solved:

- The transduction problem: Translating the real world into an accurate and adequate symbolic representation.
- The representation/reasoning problem: How to represent information about complex real-world entities and processes symbolically, and how to reason based on partial information.

The failure to find solutions to these problems led to development of control techniques that don't depend on logical representations meant specifically for planning.

2.1.2 Configuration Space Representation

One of the most widely used representation for performing planning and control is to represent the problem in the configuration space ($\mathcal{C}$-*space*)—the space of all possible configurations of the robot. The planning problem then reduces to finding a solution in this space from the start state to the goal state.

Lozano-Pérez, Mason, and Taylor presented the *preimage planning* framework [75] to address manipulation planning problems in configuration space with bounded uncertainty. The most popular method within the preimage planning framework involves performing a backward search from the goal until it reaches the starting state. Although this sounds simple enough, the set of possible motion commands is infinite. Erdmann [31] showed that the preimage in general cannot be computed by any algorithm. It was later shown that the 3D version of preimage planning, in which the obstacles are polyhedral is NEXPTIME-hard [16].

This has caused research in this area to shift from exact, complete algorithms to sampling-based algorithms, such as Rapidly-exploring Random Trees (RRT) [69], that can rapidly find a feasible solution at the expense of completeness. However, these algorithms (RRT-Connect [65], Multipartite RRT [120]) waste a lot of their computational resources by randomly sampling a part of the state space that might not be relevant to the task. RRTs have been used extensively for various motion planning tasks for humanoids [64] and aerial-robots [61]. Miyazawa [77] used RRTs to accelerate planning the motion of fingertips for graspless manipulation. Zucker [120] presented a variant of the algorithm called Multipartite RRT (MP-RRT) that supported planning in unknown or dynamic environments. The algorithm combined the strengths of RRT with a biased sampling distribution and showed how branches from previous planning iterations can be used to re-plan quickly in dynamic environments. All these approaches were however constrained by the fact that the goal and initial state needed to be in the configuration space of the robot. Diankov [25] presented a planning algorithm called *BiSpace* that could plan in complex, high-dimensional spaces by simultaneously exploring multiple spaces (e.g., Cartesian and configuration space). Lately, this framework has been extended to handle a variety of constraints in manipulation planning including constraints on the pose of an object held by a robot, or constraints for following workspace surfaces [7].

Burridge *et al.* [14] showed how feedback motion planning can be considered as a sequential composition of locally valid feedback policies, or funnels, which takes an agent with a broad set of initial conditions to the goal region. The weakness of this approach was the difficulty of computing the region of applicability, or preimage, for a controller. Tedrake combined convex optimization-based techniques with randomized sampling of state space to create sequences of stabilizing controllers that probabilistically covers the reachable area of a state space ensuring that the goal state can be

reached from all initial conditions [108].

All these techniques reduce the problem of planning and control to a configuration space problem and search for a solution in that space. It is however a challenge to represent information about the world and task exclusively in configuration space.

2.1.3 Representation Free Planning and Control

Brooks [11, 13] proposed a reactive approach to robot control without explicit representations. He decomposed the problem into layers corresponding to levels of behavior. Within this setting, he introduced the idea of subsumption wherein the goals of higher-level layers subsume the roles of the lower, more reactive layers when they wish to take control. This approach employs neural mechanisms of inhibition and suppression to construct behavior as the structured interaction between primitive behaviors. Layers are able to substitute (suppress) the inputs to other layers and to remove (inhibit) the output from lower layers. The resulting architecture was one that could simultaneously make progress toward multiple, potentially conflicting goals in a reactive fashion, while giving precedence to higher priority goals. The ability of the robot to achieve its high level goal while still attending to its low level goals crucially depends on the programming of the interface. Brooks was successful in building robots for exploration, foraging and tracking using the above approach. However, subsumption-based robots cannot perform tasks requiring means-end reasoning. This is because the knowledge required for deliberation is not explicitly stored in the various layers. The focus of this approach is directed towards achieving robust behavior instead of correct or optimal behavior. Even then, the robustness depends critically on coefficients of inhibition and suppression.

From the above discussion, it is evident that none of the above representations allows a robot to seamlessly learn from interaction and plan using the learned models. The configuration space representation is good for geometric, non-contact based planning and control. On the other hand, logical representation provides powerful mechanisms for planning, but none for perception and autonomous learning. These challenges have led lately to the development of a representation that doesn't adhere to just logic or configuration. Instead of modeling the actions of the robot, it models the logic of discrete events generated by closed-loop control interactions (dynamical systems) in the context of partially observable systems and segments objects in the world in terms of the actions they support or *afford*. This functional representation has its roots in cognitive psychology and also provides the inspiration for the representation that I am presenting. The next section presents a survey of functional representations.

2.2 Functional Representation

Psychologist J. J. Gibson introduced the term affordance as "all *action possibilities* latent in the environment" [39]. He suggested that affordances are objectively measurable in relation to the actor and therefore dependent on their capabilities. Gibson presented an interactionist view of perception and action that focussed on the information that is available in the environment. According to this framework, *entities* surrounding an actor become useful *objects* by virtue of the actions that the actor can apply to them. The term *affordance* refers to the property of the environment that leads to a specific kind of interaction. It describes the attributes of an environment that support abilities/skills in the agent.

Following the formulation of the theory of affordances, there has been a lot of work done by the Ecological Psychology community that aimed at showing that humans

can perceive whether a specific action can be executed successfully in the environment. The hypothesis was that humans do not necessarily perceive objects (For example, box, stairs, ball), but the action possibilities (For example, liftable, climbable, throwable) in the world. Although the number of objects in the world can be infinite, the number of possible interactions is limited and is dependent upon the perceptual and motor capabilities of the human.

Warren's stair-climbing experiments [116] showed that organisms perceive their environment in terms of *intrinsic* or *body-scaled* metrics, not in absolute or global dimensions. He computed a constant, called π proportions, that depend on specific properties of the organism-environment system. For example, a human's judgement of whether he can climb a stair step is not determined by the global dimensions of the height of the stair step, but by its ratio to his leg-length. Oudejans *et al.*'s [86] study of street-crossing behavior and perception of *critical time-gap* for safe crossing shows that not only static properties of the organism, but also its dynamic state is important when deciding on actions.

Representing knowledge about the world in terms of affordances provides a powerful and computationally efficient way for an agent to encode its experiences. The use of affordances within autonomous robotics is mostly confined to behavior-based control, and their use in deliberation remains a largely unexplored area. This is not a coincidence, but indeed a consequence of the shortcomings in Gibson's theory. Gibson didn't view affordances as a representational unit that can be used by computational processes. Since the formulation of the theory of affordances by Gibson [39], a great deal of work has been done to formalize this concept in a manner that can be modeled computationally. Specifically, Stoytchev [106, 105] and Fitzpatrick [34] showed that affordance learning can be used to differentiate objects in the course of interaction

with the environment. Stoytchev's and Fitzpatrick's work uses affordance as a higher level concept that a developing cognitive agent learns about by interacting with objects in the environment. Montesano *et al.* [78] presented an affordance based model using Bayesian networks that linked actions and their effects to object features. In the next two subsections, I present a summary of two state-of-the-art projects that uses an affordance based representation for planning and control.

2.2.1 MACS - Affordance Inspired Robot Control

The MACS (Multi-Sensory Autonomous Cognitive Systems) project [97] presented an affordance based robot control architecture that can be used for both learning [26] and planning [73]. In their formalization, an affordance is an acquired relation between a certain *effect* and an (*entity, behavior*) tuple, such that when the agent applies the *behavior* on the *entity*, the *effect* is generated. Here the *entity* and *effect* descriptors are high dimensional features whose relationship through a pre-programmed discrete action is learned by the agent. Furthermore, they show that given a symbolic description of entity-action-effects, one can use standard propositional logic planner for reasoning. Though their research shows the benefits of using affordance based representation for planning and how such affordances can be learned, high-dimensional features employed to learn the relationships between the entity and effects are disconnected from the symbolic representations of actions and effects used for planning.

2.2.2 Object Action Complexes

Geib *et al.* [37] proposed a solution to the representational discontinuity by pairing actions and objects in a single representation that captures high-level action representations in terms of low-level control representations. This approach supports both learning behavior and reasoning about them. In the simplest case, the system has sensors, $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ where each sensor σ_i returns an observation $obs(\sigma_i)$ about some aspect of the world. The execution of a robot-level motor program may cause

changes to the world that can be observed through subsequent sensing. However, in reality, the full spectrum of effects caused by an action depends, in general, on aspects of the environment that can be outside the scope of the sensed information. Furthermore, each motor program can be executed on only a subset of objects in the world. They assume that the robot does not initially know about any objects and thus can't execute any motor programs. Instead the robot has a set of basic reflex actions that aren't dependent on particular objects and can be used as an initial means of exploring the world.

The planning level representation is based on a set of *fluents*, $f_1, f_2, \ldots, f_m$: first-order predicates and functions that denote particular qualities of the world, robot and objects. Fluents represent high-level (possibly abstract) counterparts of some of the properties that the robot is capable of sensing. In particular, the value of a fluent is a function of the observations returned by the sensor set, i.e., $f_i = \Gamma_i(\Sigma)$. Typically, each fluent depends on a subset of the sensor observations and not every sensor maps to a fluent. Fluents can also be parameterized by high-level versions of the objects known at the robot level. A state is a snapshot of the values of all instantiated fluents at some point during the execution of the system. States represent a point of intersection between the low-level and high-level representations, since states are induced from a set of sensor observations and the corresponding sensor/fluent mappings (Γ_i). It is however not at all evident as to how a robot can come up with a necessary set of fluents autonomously.

Given a state description in terms of fluents, the robot can observe a small portion of the world's state space (an object) and notice how it changes with the application of a motor program. This one instance of interaction is called the *instantiated state transition fragment (ISTF)*. Given, multiple instances of these, the robot learns struc-

tures referred to as *object-action complexes*, which are similar to ISTFs, but contain only the relevant instantiated state information needed to predict the applicability of an action and its effects, with all irrelevant information pruned away. Such a knowledge structure can then be used as a forward model for planning.

As is evident from the above two case studies, although affordance-based representation can be used for learning effects of actions as well as planning, there exists no knowledge representation that combines both these aspects seamlessly. One of the main contributions of this dissertation is the development of a representation that lets a robot accumulate control knowledge by direct interaction with the world. I present a methodology that extracts symbolic knowledge structures directly from interaction statistics. The representation allows objects in the world to be described not in terms of high dimensional features but instead, in terms of the set of actions that the object affords. In the next section, I describe our representational basis for learning environmental models and planning.

2.3 Representational Foundations

Our computational representation of knowledge is based on a framework called the *control basis* [48] 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 multi-objective control that is derived directly from the sensory, motor, and computational embodiment of an autonomous robot. These combinatorics provide a definition for action that is useful for organizing knowledge into structures that facilitates knowledge and transfer.

The control basis framework was originally introduced by Huber and Grupen as a means for robot systems to explore the combinatorics of sensory and motor control

circuits in an autonomous learning framework [50, 51, 49]. Primitive actions in the control basis are combinations of potential functions, sensory, and motor resources defined by three finite sets:

- Φ is a set of artificial potential functions,
- Σ is a set of feedback entities that can be computed by applying operators to sensory signals, and
- $\mathcal{T}$ is a set of motor units.

Potential functions $\phi \in \Phi$ are scalar navigation functions whose gradients lead asymptotically to fixed points. Artificial potential functions have been widely adopted for solving a number of path planning problems in robotics [21, 59, 60]. However, one of the greatest difficulties in using potential fields for robot control is satisfying the condition of a unique minimum. Rimon and Koditschek defined a class of *navigation functions* that had many desirable properties for an artificial potential used for robot control [62]. In addition to having a unique minimum at the goal configuration, these functions are continuously differentiable, have a finite gradient at all points, and have a non-singular Hessian at all critical points (a Morse function).

The potential functions used in our work are constructed so as to be *navigation functions* [62] ensuring that they are provably asymptotically stable on the domain in which they are defined. Examples of such functions include quadratic functions and solutions of Laplace's equation. The scalar potential can be viewed as a measure of the strain in the system between an observed situation and goal specified by percepts in Σ. The gradient of the potential function acts as a virtual force that decreases the distance between the present state of the system and a reference condition. Novel potential fields have been used in the past to avoid joint range limits [41] and form

grasps that fulfill force closure objectives [19, 53].

Feedback entities $\sigma \subseteq \Sigma$ are continuous functions in space and/or time that are published by sensors. The feedback entities can be a single feature of a single signal, it can refer to sets of such features describing composite kinematic structures and Cartesian features, or it can be temporal features depending on the history of observations. For example, a contact load cell on the fingertip of a robot hand may produce Cartesian contact position, and six axes of force and moment information of loads applied to the fingertip. Similarly, homogenous regions in a 3D point cloud signal can be used to locate features of various shapes in space and time.

Motor units $t \in \mathcal{T}$ are embedded controllers for actuating independent degrees of freedom in the robot. A motor unit consists of 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, $\tau \subseteq \mathcal{T}$. These references are used by higher-level controllers to descend the gradient of a potential function and achieve its objective.

Definition 1 (Controller)**.** *Let $c(\phi, \sigma, \tau)$, where $\phi \in \Phi$, $\sigma \subseteq \Sigma$, and $\tau \subseteq \mathcal{T}$ define a controller in the control basis.*

Primitive closed-loop controllers achieve their objective by following gradients in the scalar potential function $\phi(\sigma)$ with respect to changes in the value of the motor variables $\mathbf{u}_\tau$ as captured in the error Jacobian

$$J = \frac{\delta \phi(\sigma)}{\delta \mathbf{u}_\tau}. \tag{2.1}$$

Reference inputs to lower-level motor units are computed by

$$\Delta \mathbf{u}_\tau = \kappa J^{\#} \Delta \phi(\sigma), \tag{2.2}$$

where $J^{\#}$ is the pseudoinverse of J [79], $\Delta\phi(\sigma) = \phi(\sigma_{ref}) - \phi(\sigma_{act})$, the difference between reference and actual potential, and κ is a small positive gain.

The combinations of potentials Φ, and resources, Σ and $\mathcal{T}$ (given by the set $\Phi \times \Sigma \times \mathcal{T}$) defines all primitive closed-loop actions $a \in \mathcal{A}$ that the robot can employ. In this work, we use the shorthand notations c or ϕ_τ^σ to describe closed loop controllers.

2.3.1 Multi-objective Control

Multi-objective control actions are constructed by concurrently executing control primitives where subordinate controllers can be executed without destructively interfering with the primary controllers. Concurrency is achieved by projecting subordinate actions into the nullspace of superior actions [80]. If $c_1 = (\phi_1, \sigma_1, \tau_1)$ and $c_2 = (\phi_2, \sigma_2, \tau_2)$ are control actions that employ the same effector resources τ, then $c_2 \triangleleft c_1$ (read "c_2 *subject-to* c_1") denotes the linear projection

$$\Delta\mathbf{u}_\tau = \kappa_1 J_1^{\#} \Delta\phi_1(\sigma_1) + \left[I - J_1^{\#} J_1\right] \kappa_2 J_2^{\#} \Delta\phi_2(\sigma_2), \tag{2.3}$$

where, $\left[I - J_1^{\#} J_1\right]$ represents the null space of controller c_1. This prioritized mapping assures that inferior control inputs do not destructively interfere with superior objectives. The projection operation in Equation 2.3 can be extended to n-fold concurrency relations [80] with different motor unit sets.

2.3.2 Controller State

The time history (trajectory) of a dynamical system provides a highly informative basis for uncovering the parameters of the underlying stochastic process (Baum-Welch algorithm for HMMs [5]). Coelho [20] showed that the dynamics $(\phi, \dot{\phi})$ created when a controller interacts with the environment provides a natural discrete abstraction

of the underlying continuous state space[1]. He showed that trajectories representing the same control context can be combined to learn a generative model of a prototypical system behavior [20, 40]. This work is an early example of predictive state representation (PSR), which represent the state of a dynamical system by tracking occurrence probabilities of a set of future events (called *tests* or *characteristic events*) conditioned on past events (called *histories* or *indicative events*) [71, 100].

PSR is complete up to modeling resolution, but simpler/coarser dynamic models have proven to be useful in several independent studies. Huber used a binary state representation in which the state associated with the controller maps to '0' during the transient response of the controller and '1' when the controller converges to an attractor state in the potential [49]. The range between PSRs and Huber's binary state representation provide the full spectrum of state representation that explicitly links action and observations in control processes. We use a four-valued classifier presented by Hart [42] for discretizing the dynamics of a continuous time control action.

Definition 2 (Controller state)**.** *Let p^t define the state of a controller $c = (\phi, \sigma, \tau)$ at time t, where $p \in \{X, -, 0, 1\}$ such that :*

$$p^t(c) = \begin{cases} X & : \quad \phi \text{ state is unknown} \\ - & : \quad \phi \text{ has undefined reference - absorbing state} \\ 0 & : \quad |\dot{\phi}| > \epsilon \\ 1 & : \quad |\dot{\phi}| \leq \epsilon \end{cases} \tag{2.4}$$

where ϵ is a small positive constant.

[1] $\dot{\phi}$ is the observed change in potential as the robot interacts with the environment and should not be confused with the gradient of the field at that point, $\nabla\phi$.

In this state representation, 'X' indicates that the state of the action is not being evaluated (unknown control state), '$-$' indicates that the reference input percepts, σ, are not present in the feedback, '0' indicates the transient control response, 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 $\mathbf{s}^k = [p_1^k \cdots p_n^k] \in \mathcal{S}$ at time k.

Primitive control actions are guaranteed to achieve its objective provided the potential function has a defined gradient ($p \neq$ '$-$'). 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. This is because, the state of the controller evaluates to '$-$' (the potential function has no gradient). 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.

2.3.3 Sensorimotor Programs

Reinforcement Learning (RL) [107] is a natural paradigm for composing behaviors in autonomous agents because it can construct policies that does not require external supervision by using sequence of actions that lead to reward. Potential functions in discrete state and action spaces can be estimated using reinforcement learning techniques by modeling the dynamics as a Markov Decision Process (MDP) such that optimal control decisions that maximize reward can be made at any time knowing only the current state [107]. An MDP is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$ consisting of states

$\mathcal{S}$, actions $\mathcal{A}$, transition dynamics $\mathcal{T}$, and a reward function $\mathcal{R}$.

An agent may find an optimal policy for achieving its objective by learning the optimal value function. Mathematically, the optimal value at each state can be defined as:

$$V^*(s) = \max_a \sum_{s^{t+1}} T^a_{s^t s^{t+1}} [R^a_{s^t s^{t+1}} + \gamma V^*(s^{t+1})] \tag{2.5}$$

where $T^a_{s^t s^{t+1}}$ is the probability of arriving in state s^{t+1} after taking an action a in state s^t. $R^a_{s^t s^{t+1}}$ is the reward received when making that transition, and $\gamma \in [0, 1)$ is a discounting factor necessary to satisfy convergence criteria for infinite horizon tasks [6]. After the convergence of the value function, a greedy policy π at every decision point allows an agent to maximize reward, such that

$$\pi(s) = \arg\max_a \sum_{s^{t+1}} T^a_{s^t s^{t+1}} [R^a_{s^t s^{t+1}} + \gamma V^*(s^{t+1})] \tag{2.6}$$

2.3.3.1 SearchTrack schema

Sensorimotor programs can be acquired autonomously in our framework by defining the set of actions that the robot has access to and the rewarding state. 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^σ_τ 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. A $0 \rightarrow 1$ transition in the state of this action is considered rewarding since it leads to the discovery of controlled interactions between the agent and its environment. Search actions are of the form $\phi^{\tilde{\sigma}}_\tau$—their input, $\tilde{\sigma}$, is derived from probabilistic models describing distributions over effector reference inputs ($\mathbf{u}_\tau$) where rewarding Track-ing actions have been discovered in the past ($p(\phi^\sigma_\tau) = 1$). For example, such a controller can be used to direct the field of view of a robotic system to look at places where a

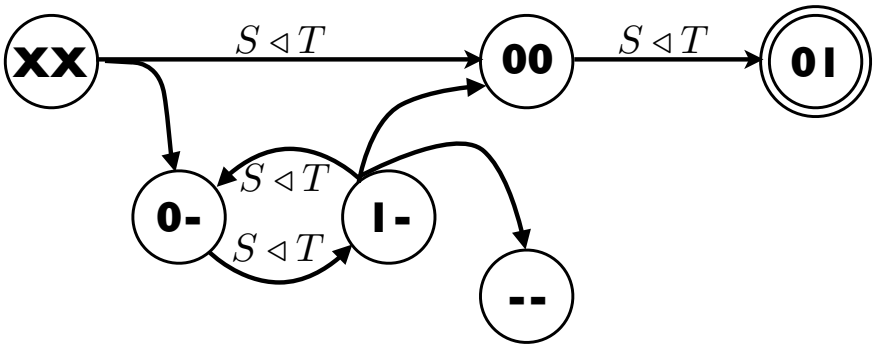
color feature has been found in the past.

The SEARCH actions can be thought of as *orienting* actions that orient a robot's sensor geometry, using prior knowledge, to increase the probability of convergence for a TRACK action. Initially the distribution $Pr(\mathbf{u}_\tau | p(\phi_\tau^\sigma) = 1)$ 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. These primitive probabilistic models describing the TRACK-able events capture concrete facts about the environment. In combination with many other such events, this representation forms the basis for a powerful, hierarchical model of the world.

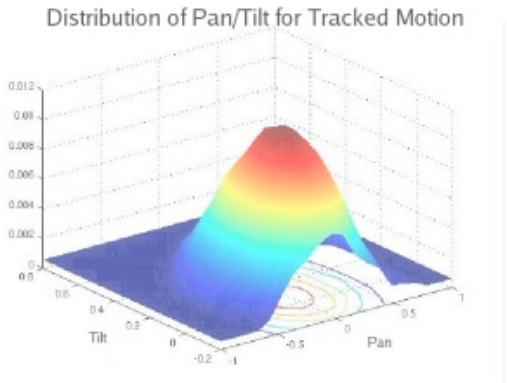
Hart [44] showed that restricting the sensory and effector resources to which the robot has access can lead to the acquisition of new and interesting behavior. In the simplest context, the robot was restricted to proprioceptive feedback from the pan/tilt head and large scale motion cues arising from a single camera. Effector resources are likewise restricted to motor controllers associated with the pan and tilt axes of the visual system. Under this developmental context, the robot has access to a small variety of SEARCH ($\phi_{pt}^{\widetilde{(u,v)}}$) and TRACK ($\phi_{pt}^{(u,v)}$) actions,

$$\mathcal{A} = \{\phi_{pt}^{\widetilde{(u,v)}}, \quad \phi_{pt}^{(u,v)}, \quad (\phi_{pt}^{\widetilde{(u,v)}} \triangleleft \phi_{pt}^{(u,v)}), \quad (\phi_{pt}^{(u,v)} \triangleleft \phi_{pt}^{\widetilde{(u,v)}})\}$$

where pt designates the pan and tilt axes of the head and (u, v) designates the centroid of the motion cue relative to the image center. The sensory reference $\widetilde{(u, v)}$ for the SEARCH action is sampled from the distribution $Pr((u, v) | p(\phi_{pt}^{(u,v)}) = 1)$. The only rewarding event that can be generated by these set of actions is the convergence of the TRACK-ing controller $\phi_{pt}^{(u,v)}$.



(a)



(b)

Figure 2.1. SEARCHTRACK behavior in terms of state $[p^{search} \quad p^{track}]$. A new SEARCH goal is sampled whenever SEARCH is executed from states for which $p^{search} \in \{X, 1\}$. Panel (b) shows the resulting distribution $Pr((u,v)|p(\phi_{pt}^{(u,v)}) = 1)$ after 50 presentations.

The state space defined by these actions is the vector of controller states $\mathbf{s} = [p^{search} \quad p^{track}]$. Figure 2.1(a) shows the SEARCHTRACK policy acquired after 25 learning trials in this developmental context using Q-learning. Action $S \lhd T$ is a concurrent combination of SEARCH and TRACK actions, where SEARCH is executed in the nullspace of TRACK. The policy begins by attempting to concurrently SEARCH for and TRACK a specific motion cue. If this cue exists in the signal, the policy attempts to continue TRACK-ing. If no target is immediately available, the policy samples new configurations from the search distribution until the target stimulus is found; at which point, the policy TRACK-s the feature. If no reference stimulus is found after sampling N times from the search distribution, the policy transitions to the absorbing state. The shorthand, $ST|_\tau^\sigma$ is used to describe a SEARCHTRACK schema for tracking a signal, σ, using effector resources, τ.

Definition 3 (SEARCHTRACK schema)**.** *Let $ST|_\tau^\sigma = (\pi, \mathcal{M})$ define a* SEARCHTRACK *schema, where $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy indicating the probability of taking an action in a state, and $\mathcal{M}$ is a set of probabilistic models of the form $Pr(\mathbf{u}_\tau | p(\phi_\tau^\sigma) = 1)$.*

A SEARCHTRACK schema forms the basis for control and modeling in our framework. This is because, over time, the model reflects the long term statistics of where a control program can be executed in the environment while the policy describes the action selection mechanism for achieving an objective.

The state of a SEARCHTRACK schema can be evaluated in a similar fashion as that of primitive controllers. However, unlike primitive controllers, the state of a schema is evaluated at discrete time intervals.

Definition 4 (SEARCHTRACK state)**.** *Let p^t define the state of a* SEARCHTRACK *program $ST|_\tau^\sigma$ at time t, where $p \in \{X, -, 0, 1\}$ such that :*

$$p^t(ST) = \begin{cases} X & : \quad \textit{state is unknown} \\ - & : \quad V \textit{ has no gradient - absorbing state} \\ 0 & : \quad |V^t - V^{t-1}| > \epsilon \\ 1 & : \quad |V^t - V^{t-1}| \leq \epsilon \textit{ - Goal state} \end{cases} \tag{2.7}$$

where V is the value function associated with the policy and ϵ is a small positive constant.

The state of the schema is inferred by a 4-valued classifier, where 'X' indicates unknown state (the schema is not being evaluated), '0' indicates that the policy is making progress, and '1' denotes achieving the goal state. The state of the schema is classified as '$-$' when the policy enters an absorbing state and can no longer make progress towards the goal. This will happen if the policy fails to find a trackable signal in the environment after sampling multiple times from its search distributions.

2.3.3.2 Hierarchical Programs

Representing the state of SearchTrack programs in the same way as primitive controllers allows a robot to learn hierarchical programs that sequences multiple SearchTrack programs. Hart *et al.* [43] presented a detailed description of the various manipulation programs (touching, grasping, picking up, placing, and inspecting objects) that can be learned in a hierarchical fashion using these conrol programs. All of these programs viewed abstractly as a sequence of SearchTrack programs can be used to Search and Track hierarchical generalization of visual and tactile features. Figure 2.2 shows a hierarchical schema to reliably track a reference force using its end effector. The learned program (ReachGrasp) involves tracking a visual stimuli followed by a SearchTrack schema that tracks forces using fingertip mounted tactile sensors. In this hierarchical schema, the Cartesian feature position tracker becomes part of the Search behavior that orients the robot to receive a

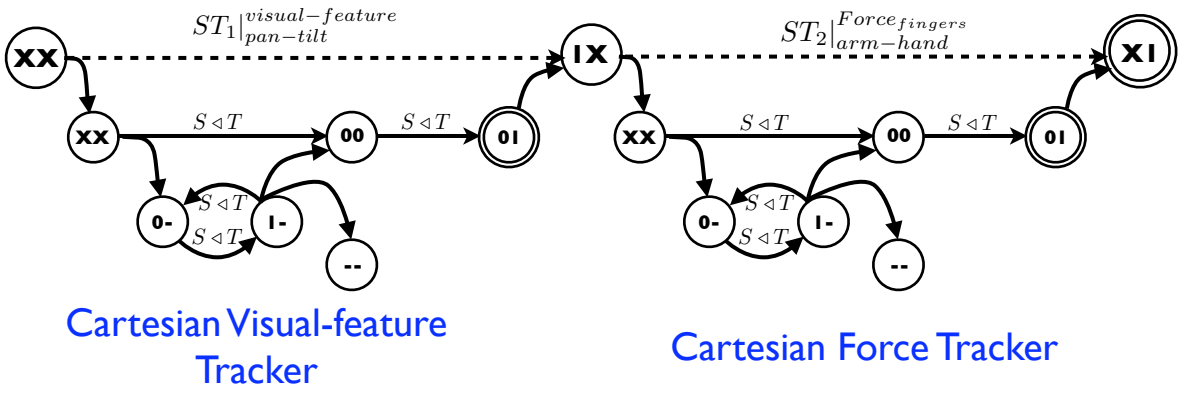


Figure 2.2. Sequential programs can be learned by sequencing a set of previously learned SEARCHTRACK schemas. The robot learns how to "grasp" by sequencing two different SEARCHTRACK schemas that establishes spatial features in $SE(3)$ followed by invariants in the force/moment domain associated with prehensile behavior.

TRACK-able force.

2.4 Discussion

The control basis framework provides a combinatoric means of assembling multi-objective closed loop control expressions by combining artificial potentials with elements from a set of sensory and motor resources. By utilizing an uniform representation of state, programs of control actions can be represented in this framework and acquired in an autonomous learning framework by using reinforcement learning techniques. We further showed that the learned programs can be composed hierarchically to acquire more complicated programs.

The use of the term "schema" was proposed by the German philosopher Immanuel Kant [57] as a way of mapping concepts to percepts over categories of objects. He talked about grounding concepts in sensations that would lend support to reasoning and intuition. One of the most influential theories of cognitive development was de-

veloped by Jean Piaget. His theory concerns the growth of intelligence [88], which for Piaget meant the ability to more accurately represent the world and perform operations on representations of concepts grounded in the world. His theory concerns the emergence and acquisition of schemata, schemes for perceiving the world in developmental stages—times when children are acquiring new ways of representing information. Jean Piaget suggested that schema are formed to meet new demands through a process of *accommodation* and that existing schema respond to new experiences through *assimilation* [88]. Piaget presents a 'constructivist' approach to the development of cognition, where he asserts that humans construct their cognitive abilities through self-motivated action in the world. Our computational framework acquires programs for controlling interaction with the environment and manages redundant sensory and motor resources to discover and maintain intrinsically rewarding relationships in dynamic environments. The acquired control programs and their long term statistics represent a domain general way of interacting with stimuli in the environment.

The schemas capture *common sense* knowledge acquired by the robot. The environment, however, presents important kinds of structure in terms of *objects*—sets of temporally related schemas. In the next chapter, I present a Bayesian framework for acquiring these knowledge structures in terms of distributions over SearchTrack schemas that can later be exploited by a planner.

CHAPTER 3

SKILL-BASED REPRESENTATION

In Section 2.3, we have been concerned primarily with an architecture for control and learning that can construct integrated sensorimotor behavior. The result is an autonomous learning method for composing closed-loop controllers. Guided by reward, this approach discovers new behavior and models the conditions under which the target stimuli are controllable. Although this representation is relatively simple, it is powerful—actions and states are automatically enumerated from a description of the resources comprising the embodied system, and knowledge is captured implicitly in the form of behavior and explicitly in the form of probability distributions over effector spaces to reflect structure latent in the environment. In this chapter, I describe how a robot can model its interactions with the environment in terms of the previously acquired programs.

The sensorimotor programs acquired by a robot model action-specific information. Each program in itself, however, captures a small part of the dynamics of the environment. This information is encoded in the state of the SearchTrack schemas and primitive controllers, where $p \in \{0, 1\}$ indicates the presence of a stimuli in the environment, and $p \in \{-\}$ indicates the absence of it. In the context of modeling the dynamics of the environment, we will be using a simpler binary state representation for capturing the state of either a SearchTrack schema or a primitive controller. In the rest of the dissertation, I will treat both of them uniformly and refer to them as control programs.

Definition 5 (Program state)**.** *Let γ^t define the state of either a* SearchTrack *schema $ST|_\tau^\sigma$ or a primitive controller ϕ_τ^σ at time t, where $\gamma \in \{-, +\}$ such that*

$$\gamma^t(a) = \begin{cases} - & : \quad ST|_\tau^\sigma \text{ or } \phi_\tau^\sigma \text{ has undefined reference } (p = \text{`-'}) \\ + & : \quad ST|_\tau^\sigma \text{ or } \phi_\tau^\sigma \text{ has a defined reference } (p \in \{0, 1\}) \end{cases} \tag{3.1}$$

The mapping function γ is a variant of the mapping presented in Equation 2.4 and Equation 2.7 that aggregates $p \in \{0, 1\}$ into the affirmative "+" token. Thus the state of a program signifies the presence or absence of a trackable external stimuli in the robot's environment.

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

$$z^t = \{\gamma(a) | a \in \mathcal{A}, \gamma \in \{-, +\}\} \tag{3.2}$$

where z contains the state of a set of control programs. The observations capture the event dynamics of a constellation of control programs—set of programs that can or cannot be executed successfully in the environment. In the next section, we describe how we can extract certain meaningful structures from the patterns of observations.

3.1 Environmental Structure

The environment provides a lot of structure regarding when sets of control programs have a trackable stimuli. The structure can arise from various contexts—a robot present in an office environment will receive stimuli corresponding to the context of an office (chairs, tables, cubicles), while a robot present in a household environment will receive stimuli corresponding to a different context (couch, TV, kitchen utensils). In this work, we will model the most basic environmental context—rigid body objects. An object to a robot represents an entity that allows certain sets of controllable interactions.

3.1.1 Aspects

Consider a set of observations, z made by the robot at any instant of time. Each element in the observation set describes the presence or absence of an externally reference stimuli with respect to a common sensor geometry. It denotes a "viewpoint" for the robot relative to the environment associated with a distinctive pattern of observable features.

The computer vision community has devised data structures called *aspect graphs* that represent such viewpoint dependencies explicitly to construct appearance-based models of objects. An aspect is the appearance of an object topologically from a specific viewpoint. An aspect graph is a graph with a node for every aspect and edges connecting adjacent aspects [63].

In spirit of these previous approaches, we will call a snapshot of simultaneously accessible features an observable *aspect* of the environment. Our representation, however, will generalize the representation to incorporate both haptic and appearance based visual features that are simultaneously accessible from a robot's viewpoint.

For example, a visual aspect, x_v, is a set of visual features, Σ_v (i.e., where either $\gamma(ST|_\tau^{\sigma\in\Sigma_v}) = +$ or $\gamma(\phi_\tau^{\sigma\in\Sigma_v}) = +$) that a sensor can detect reliably for a camera frame relative to the object. The features comprising a visual aspect are mutually consistent with line of sight constraints.

The concept is generalized to tactile observers. The haptic aspect, x_t, describes a set of surface patches, Σ_t (i.e., where either $\gamma(ST|_\tau^{\sigma\in\Sigma_t}) = +$ or $\gamma(\phi_\tau^{\sigma\in\Sigma_t}) = +$) that a tactile sensor can detect reliably for the hand frame relative to the object. Tactile features comprising a haptic aspect are mutually consistent with kinematic reacha-

bility constraints of the hand.

The Cartesian product of visual and haptic aspects ($x_v \times x_t$) creates a uniform framework for describing associated perspectives on the target object that emphasizes the coupling between "sight" and "touch" and, thus, reasoning about manual interactions that have visual and manual effects.

Elements of an aspect can be asserted by evaluating the state of control programs. Unlike observations in an open control context where features are located in a data driven stochastic search, aspects group features together that do not require modifications of the sensor geometry. As a result, aspects represent compelling geometrical and computational structure regarding the robot-world interaction.

Figure 3.1 is a Bayesian network that encodes the logical dependencies between the aspect X of an environment modeled as a multinomial random variable and a set of control events γ. We make the naive Bayes assumption that the presence (or absence) of a particular stimuli is unrelated to the presence (or absence) of any other stimuli, given the aspect class. Using the Bayes' theorem,

$$\begin{aligned} p(X|z) &= p(X|\gamma_1, \ldots, \gamma_N) \\ &\propto p(X) \prod_{i=1}^{N} p(\gamma_i|X) \end{aligned} \tag{3.3}$$

3.1.2 Objects

An aspect models reliable patterns over the state of control programs. An object is defined as a set of mutually exclusive aspects (X) related through control programs ($a \in \mathcal{A}$) that change the existing sensor geometry and, thus, influence the set of accessible aspects in $x_v \times x_t$. Figure 3.2 shows a graphical model that encodes the

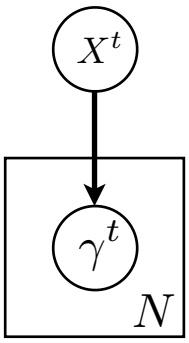


Figure 3.1. The graphical representation of an aspect as a spatial distribution over N control programs.

logical dependencies between the variables of the object model. An object O induces a distribution over a set of M mutually exclusive aspects. Each aspect x induces a distribution over the state of N control 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 (γ_j = '+', if the action has a reference).

The dependencies between the aspects (X) 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 describes how taking actions on an aspect influence the set of accessible aspects. For example, a hammer's handle allows the action of grasping, however if the handle is out of reach, the robot might have to pull 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

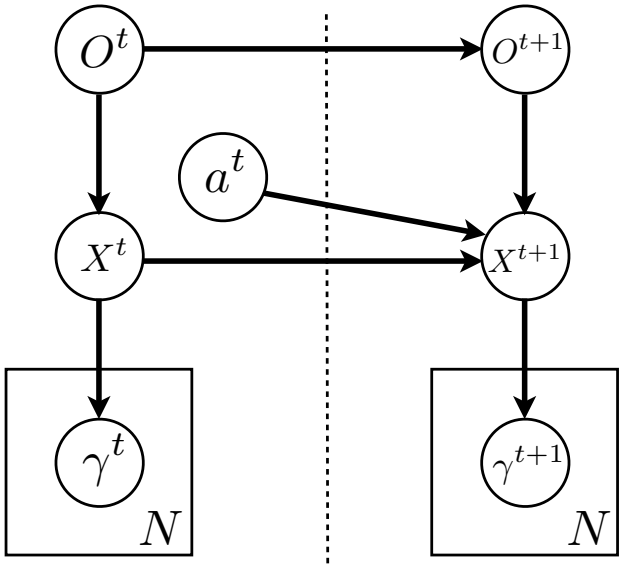


Figure 3.2. Figure shows a Bayesian network model representing objects O as a temporal distribution over aspects X. An aspect induces a distribution over the state of N programs (γ_j) as shown by the plate model. The two time slices in the model show the logical dependencies between aspects and an action a. O, X and a are modeled as multinomial random variables. γ_j is modeled as a Bernoulli random variable.

model for both interacting with objects as well as recognizing them.

The elements of the Bayesian network are modeled using discrete random variables. This allows a robot to model objects at a higher level of abstraction where the continuous state of a temporally extended control program is classified into a binary representation. A planner utilizing this model can build plans without caring about the runtime parameterizations of the control programs. This is because the policies (and models) or objective functions associated with the control programs determine the run time contingencies and parameterization.

3.2 Aspect Transition Graph

Objects represented in this framework provide a computationally efficient way of storing knowledge hierarchically, at the level of control programs, aspects, objects (for multi-object relationships), and so on. A planner can exploit the hierarchical knowledge structure by searching for plans at a level of abstraction in the hierarchy that ignores unimportant details. Figure 3.3 illustrates a fragment of the aspect-action transition model, as an Aspect Transition Graph (ATG), for a mallet that incorporates both visual and haptic features. Each node in the graph represents a pattern of observations regarding the dynamic status of six control programs :

- $ST_1|^{hue_{red}}_{pan-tilt}$: A sensorimotor program that visually tracks a red colored feature using the pan-tilt cameras.
- $ST_2|^{hue_{wood}}_{pan-tilt}$: A sensorimotor program that visually tracks a wooden colored feature using the pan-tilt cameras.
- $(ST_1|^{hue_{red}}_{pan-tilt})(ST_3|^{force_{fingers}}_{arm-hand})$: A sequential sensorimotor program that visually tracks a red colored feature and grasps it.
- $(ST_2|^{hue_{wood}}_{pan-tilt})(ST_4|^{force_{fingers}}_{arm-hand})$: A sequential sensorimotor program that visually tracks a wooden colored feature and grasps it.
- ${\phi_1}^{force_{wrist}}_{arm}$: A primitive controller that pulls the arm closer while maintaining a force reference. The effect of this controller is similar to a *pulling* action.
- ${\phi_2}^{force_{wrist}}_{arm}$: A primitive controller that lifts the arm while maintaining a force closure. The effect of this controller is similar to a *lifting* action.

Options for grasping this object include grasp controllers directed at visual segments associated with the red handle and the wooden head of the mallet. Power grasps on the head of the mallet do not succeed when used to lift the mallet from the supporting surface (table) due to insufficient friction between the robot hand and the

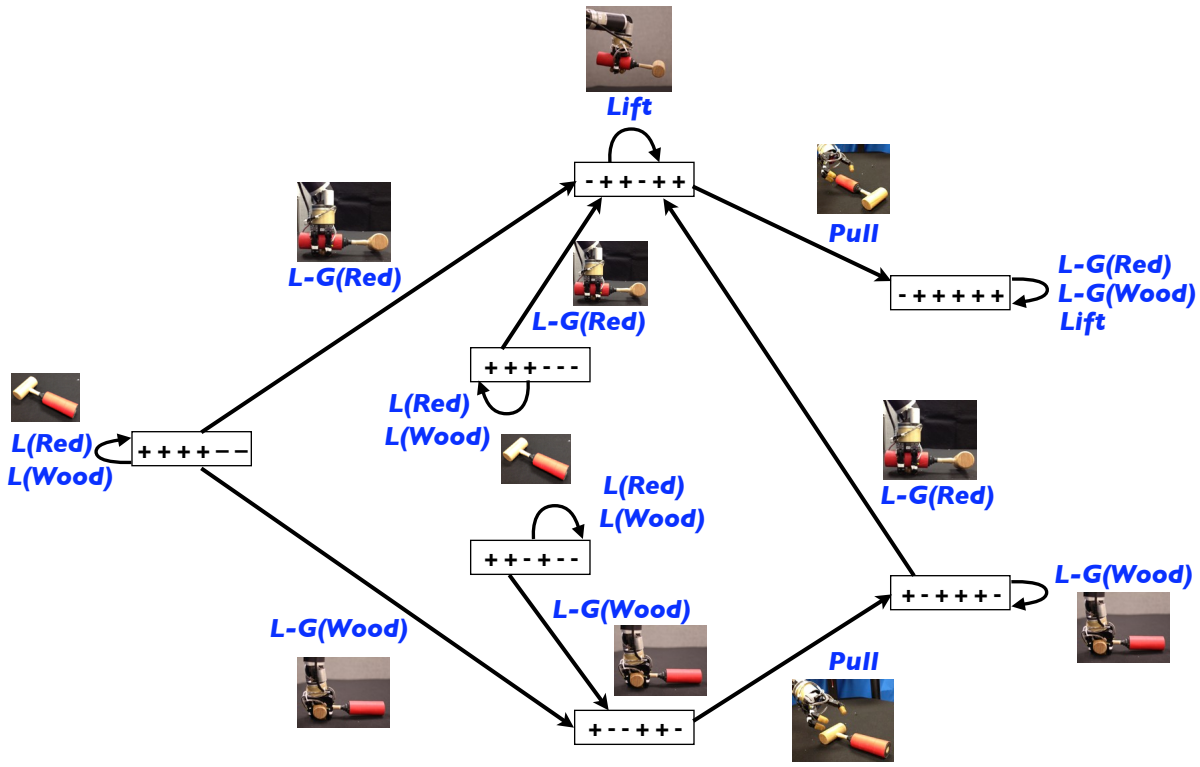


Figure 3.3. The Aspect Transition Graph (ATG) for the mallet/table. Each node (aspect) in the graph denote the state of 6 control programs along with their spatial distributions (not shown in figure). The control programs in each aspect are (in order) - 1. VISUALLY TRACK a red colored stimuli, 2. VISUALLY TRACK a wood colored stimuli, 3. GRASP the red feature, 4. GRASP the wood feature, 5. PULL the grasped feature on the table, and 6. LIFT the grasped feature from the table.

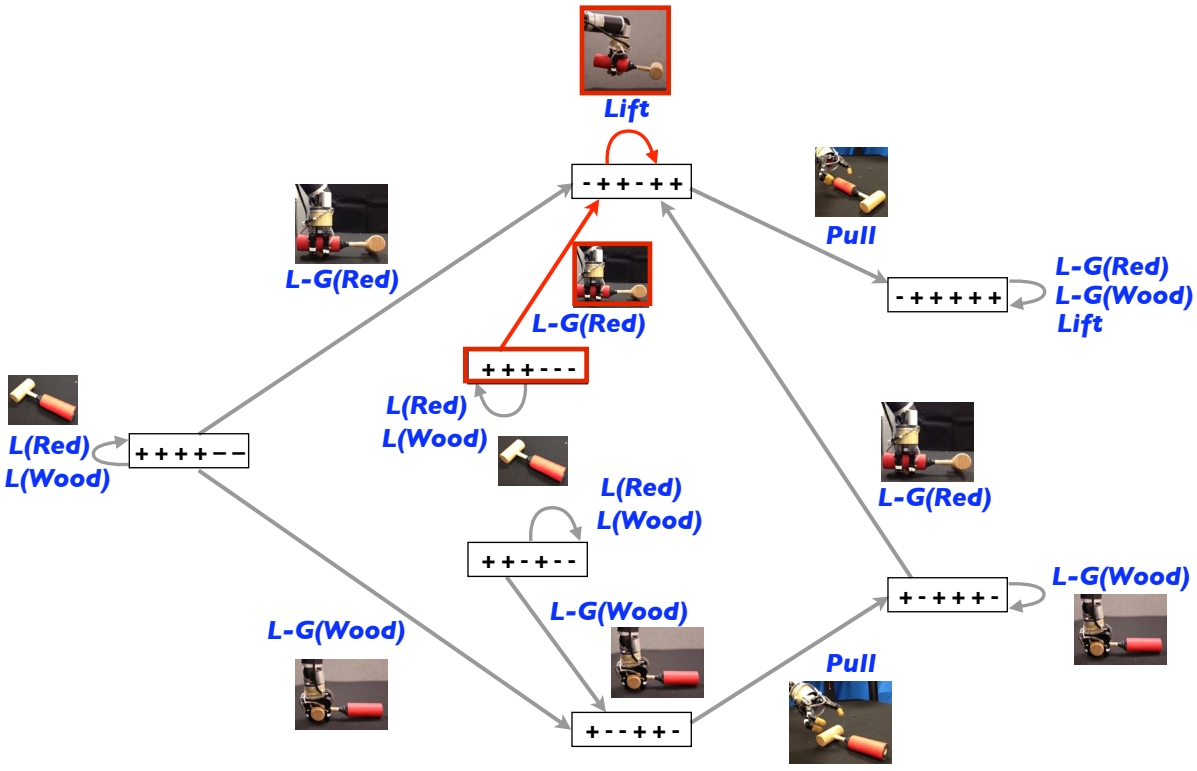


Figure 3.4. The red arrows indicate a plan in the ATG to lift the mallet when the mallet is presented within the reachable workspace. The solution entails grasping the red feature followed by lifting it.

hardwood surfaces of the mallet—only power grasps applied to the red handle are capable of fully immobilizing the mallet relative to the hand without the assistance of the supporting surface.

Consider the task of lifting the mallet from the table. When the entire state is directly observable, simple planners can exploit the aspect-transition model to generate interesting kinds of autonomous behavior. Perhaps the simplest version of this task is the case where the mallet is presented on the table in the reachable workspace. The trace of a plan is highlighted in red in Figure 3.4. If, however, the mallet is presented on the table such that the handle is initially out of reach, then the previous plan fails. Initially, the action to grasp the handle reveals that the handle is out of reach causing

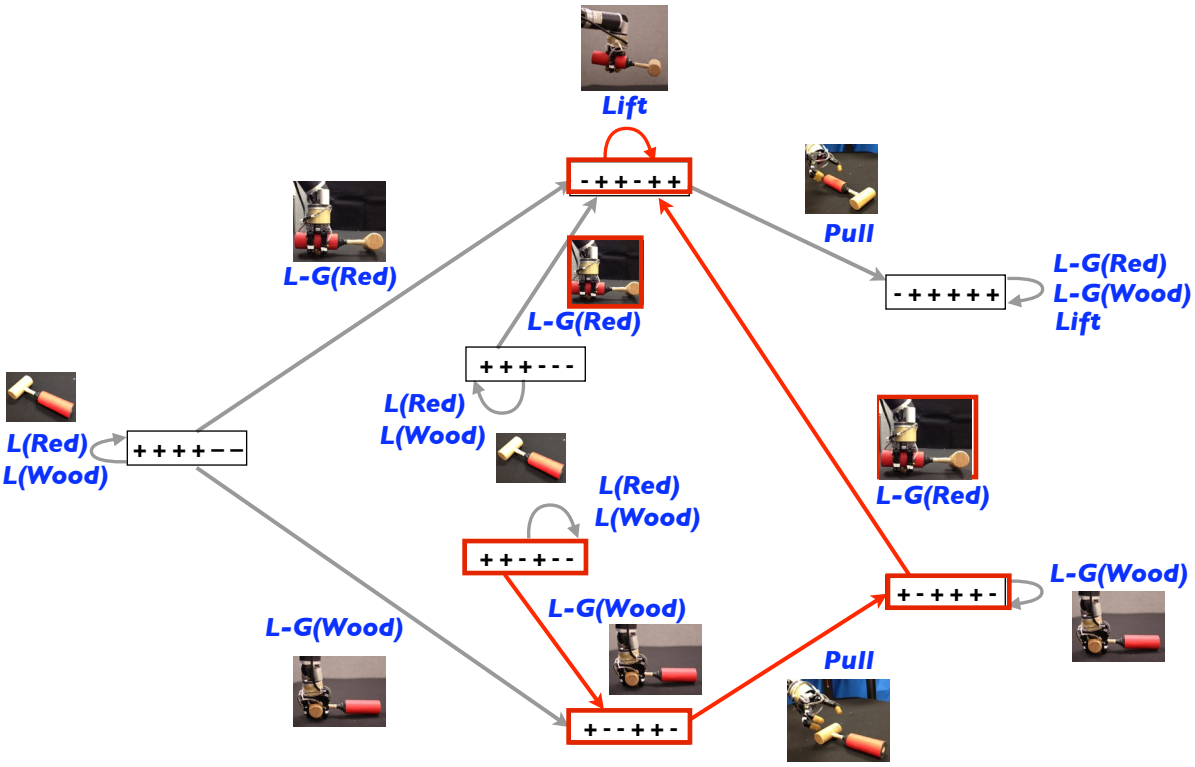


Figure 3.5. The red arrows indicate a plan in the ATG to lift the mallet when the mallet is presented in a region of the workspace where the LIFT is not achievable directly. The solution entails grasping the wooden feature followed by pulling the mallet closer. Once the mallet is closer, the plan requires the robot to regrasp the mallet and lift it.

the planner to compute a new course of action guided by the aspect transition model. The only recourse is to attempt to grasp the mallet head, and if successful, re-position the mallet on the table top, thus exposing an aspect of the mallet that affords grasp. At this point, the mallet is grasped and lifted from the table. This case is highlighted in red in Figure 3.5.

Both of these contingencies arise directly from the comprehensive model of the mallet. Furthermore the planner builds a plan at the level of aspects while ignoring the details of how to parametrize the control actions at runtime (for e.g., where to grasp, how to orient the hand relative to the object for grasping). Many simple planning algorithms can be used to navigate through the ATG illustrated in Figure 3.3. However in practice, the state necessary to construct manipulation strategies like this is only partially observable. A means of fusing information over time and making inferences on the basis of incomplete knowledge is required. In the next two chapters, we will present techniques that allow a robot to estimate the state from partial observations and use them for planning.

3.3 Experiments

We present two sets of experiments to show the efficacy of our representation. We demonstrate the applications of the above approach on our experimental platform, Dexter shown in Figure 3.6. Dexter is a bimanual robot with two 7-DOF Whole-Arm Manipulators (WAMs) from Barrett Technologies, two 3-finger 4-DOF Barrett Hands equipped with one 6-axis force/torque load cell sensor on each fingertip, a stereo camera pair and a Kinect mounted on a pan/tilt head.

Figure 3.6. Dexter is a bimanual upper-body humanoid.

3.3.1 Visual Object Recognition

In this section, we present an experiment where Dexter learns the functional models of objects and uses them for the task of object recognition. In this experiment, the robot uses only *visual actions* to recognize objects, meaning it cannot manipulate the object. Four objects were presented to the robot. Figure 3.7 shows the objects used in the experiment. The robot had access to a set of visual tracking actions for tracking color features as well as 3D blobs of various eccentricities. The robot learned a Bayesian model for each of the objects as described in Section 3.1.2. Once the models were learned, an object was presented at random in front of the robot and the task was to recognize the object. The inference proceeded by estimating the state of all the control programs and using the observation to infer the object.

Table 3.1 presents the confusion matrix for the task of object recognition when the only actions available to the robot were visual. The results imply that simply "looking" at an object from one pose may not be enough to disambiguate objects, unless the robot has access to more complicated visual features. The robot needs

to have access to actions that allow it to manipulate the objects in a manner that reduces its object uncertainty. In Chapters 4 and 5, we present two algorithms that allows a robot to select actions to manipulate and recognize objects in the presence of uncertainty.

Figure 3.7. The objects used in the object recognition experiment: crimper, mallet, hammer, and a toy.

3.3.2 Achieving a Goal State

In this set of experiments, we show the efficacy of our representation for goal-driven action selection when the state is completely observable. Models of objects were hand-built distributions of blobs (represented in terms of first and second moments) describing homogeneous hues, range image blobs, and hand goals in Cartesian space where "grasp" and "touch" actions have a defined reference. Grasp references are defined by TRACK-able force closure conditions while touch references are defined by

Object	Crimper	Hammer	Mallet	Toy
Crimper	19	0	0	16
Hammer	0	35	0	0
Mallet	0	9	14	12
Toy	0	6	8	21

Table 3.1. Table shows the confusion matrix for object recognition when the robot uses only visual features to reduce the uncertainty over objects.

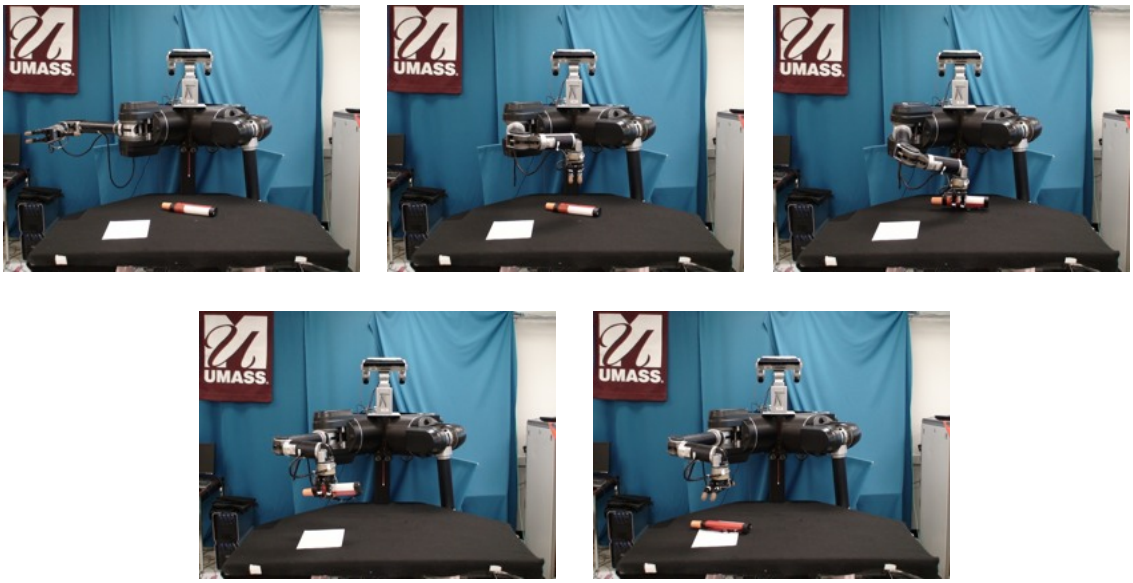


Figure 3.8. The robot performing a top grasp on the object and placing it on the goal.

small magnitude TRACK-able force events. For purposes of illustration, the middle of the mallet's handle and the middle of the emergency light were set to provide grasp references while their entire body provide references for touching. The temporal part of the object model captures the transitions between aspects when a manipulation action is executed in the context of each object. Ten trials were conducted for each object, in which the object was placed in the workspace in a variety of poses. In certain regions of the workspace, the object does not afford all haptic aspects, and additional manipulation actions have to be taken before grasp goals can be achieved.

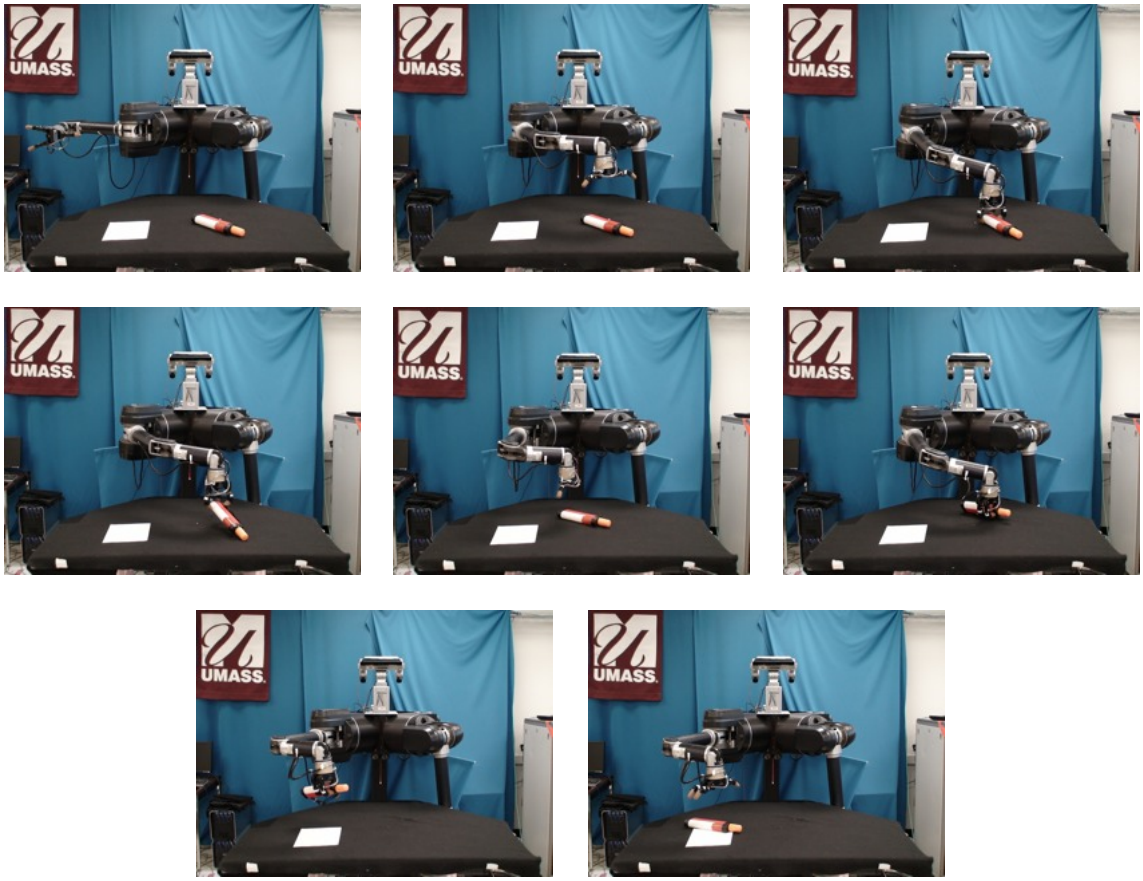


Figure 3.9. The robot pulling the object towards itself before performing a re-grasp on the object and placing it on the goal.

Figure 3.8 and 3.10 shows the case when the object is presented in a region where the robot can grasp successfully. In such a case, the control program associated with GRASP can select grasp locations from its search distributions where visually tracking the feature and grasping it converged simultaneously. However, when the region associated with grasp goals is out of reach (and hence the object aspect doesn't afford the goal—grasping in this case), the action selection proceeds by choosing a manipulation action that can change the aspect to one that affords grasping. Figure 3.9 and 3.11 shows the two scenarios where the robot chooses to PULL the object towards itself before executing the GRASP action. The above experiment shows the

power of learning and representing models of objects functionally, where the learned model can be directly used as a forward model by a planner for achieving a goal task.

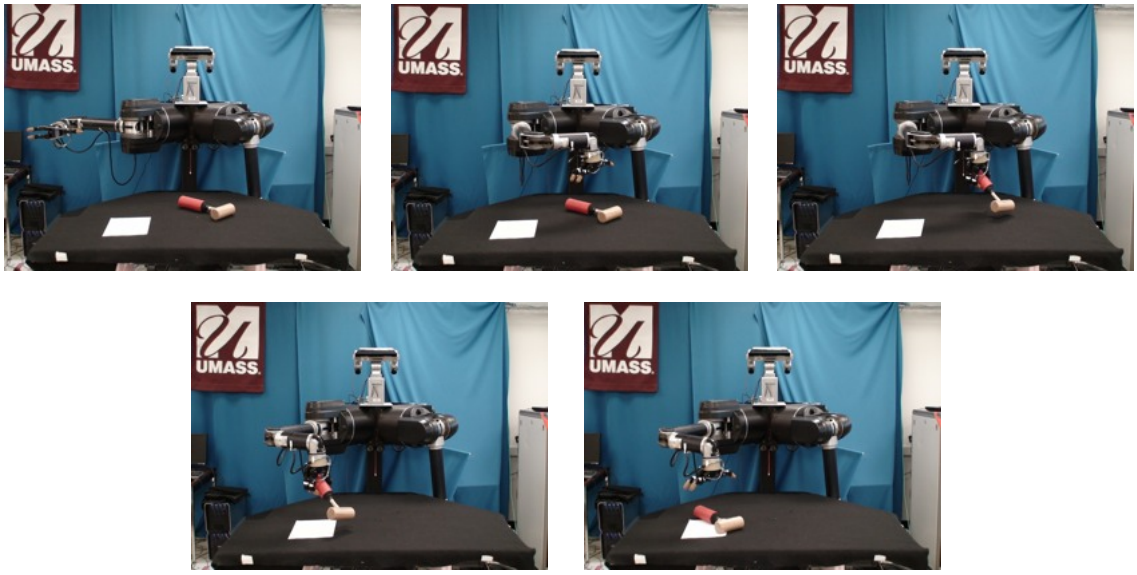


Figure 3.10. The robot performing a top grasp on the mallet and placing it on the goal.

3.4 Conlusions

This chapter introduced a functional representation for modeling the environment in terms of the state of its available *skills*. Objects in this framework were modeled as probabilistic distributions over the states of the behaviors. We presented some preliminary results on using this representation to learn object models and use the models directly for recognition tasks. We also showed how such a model can be used for goal driven action selection. Until now, we have assumed that the state of the world was completely observable. This reduces the planning process to a simple search task to find a path from the starting state to the goal state. However, in most real world tasks, the state is only partially observable. This requires the development of planning algorithms that can plan actions in the presence of uncertainty to achieve

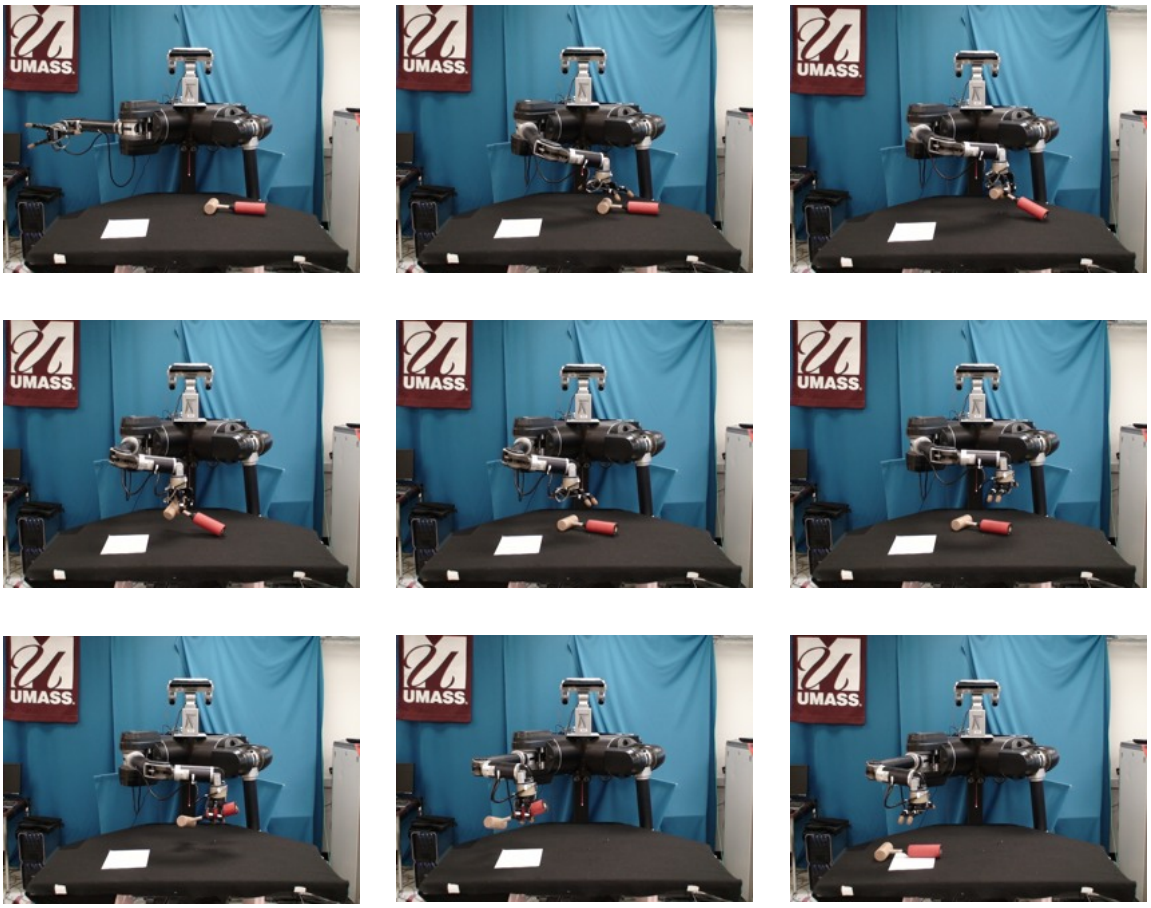


Figure 3.11. The robot pulling the mallet towards itself before performing a top grasp on the object and placing it on the goal.

its objective. In the next two chapters, I'll present two such planning algorithms that can be applied directly on the object models in the presence of uncertainty for task specific action selection.

CHAPTER 4

INFORMATION THEORETIC PLANNING

One of the key problems facing most planning systems today is that of guiding the search through an exponential state space. Planning must be performed in an abstract, and thus lower-dimensional state space. In such a space, the planner must be aware of the states from which a particular behavior can be initiated and the probabilistic distribution of outcome states that occur during execution. The planner needs to work out a probabilistically optimal strategy for a sequence of actions/skills that lead from the initial state to the goal while satisfying resource allocation constraints. Each action uses domain general *common sense* knowledge along with the high-level resource constraints to carry out the sub-plans. The planner provides a high-level switching mechanism among various actions that, in turn, create the preconditions that potentiate other actions. This ability to predict possible future states is *common sense*, and is also the missing link in computational AI/planning. In this dissertation, I hope to show that structuring representations in terms of object models and searching for a plan in that space leads to higher performance plans by avoiding computationally intensive sampling in the complete state-action space.

The problem of selecting actions in environments that are dynamic and not completely predictable or observable is a central problem in intelligent behavior. In AI, this translates into the problem of designing controllers that map sequences of observations into actions so that certain goals can be achieved. Planning algorithms usually follow three separate stages that are repeated until the objective is achieved.

- **State Estimation** : Making observations about the state of the world. The observations are then used to update the agent's *belief* of the state of the world.
- **Action Selection** : Selecting the next action to execute based on the estimate of the state and a planning metric. The action selection metric can be information theoretic or drawn from a policy.
- **Action Execution** : Parametrize the action based on runtime context and execute it.

In the next few sections, I'll explain in detail each of the planning components.

4.1 State Estimation

Bayesian filters have been applied successfully for state estimation for many years in robotics [111]. In general, a Bayesian filter estimates the partially observable dynamical system's state from a sequence of noisy observations and control actions. The state in case of the Dynamic Bayes Network illustrated in Figure 3.2 can be the object O or the aspect of the object X that the robot is interacting with.

Bayes filters represent the state at time t by a random variable x^t. At each point in time, a probability distribution over x^t, called *belief*, $bel(x^t)$, represents the agent's uncertainty. Bayes filters aim to sequentially estimate such beliefs over the state space conditioned on all information contained in the sensor data—the history of observations, $z^{1:t}$, and inputs, $a^{1:t}$. The belief over states is updated in a two step process: first a probabilistic forward model predicts the next state x^{t+1} according to

For all x^{t+1} do :

$$\begin{aligned} \overline{bel}(x^{t+1}) &= Pr(x^{t+1}|z^{1:t}, a^{1:t}) \\ &\approx \sum_{x \in X} Pr(x^{t+1}|a^t, x^t = x) bel(x^t = x) \end{aligned} \tag{4.1}$$

where $\overline{bel}(x^{t+1})$ is the predicted belief prior to fusing a new observation. Here $Pr(x^{t+1}|a^t, x^t = x)$ describes the system dynamics—that is, how the system changes over time.

The second step of the Bayes filter is called the *measurement update*. In this step, the filter corrects the predicted estimate by fusing new observations. It does so for each hypothetical posterior state x^{t+1}. The final belief at time $t+1$ is given by:

$$\text{For all } x^{t+1} \text{ do } :$$
$$bel(x^{t+1}) = \eta Pr(z^{t+1}|x^{t+1})\overline{bel}(x^{t+1}), \tag{4.2}$$

where η is a normalization constant.

Bayes filters are an abstract concept in that they provide only a probabilistic framework for recursive state estimation. Implementing Bayes filters requires specifying the perceptual model $Pr(z^{t+1}|x^{t+1})$, the state dynamics $Pr(x^{t+1}|a^t, x^t = x)$, and the representation of the belief $bel(x^{t+1})$. The properties of the different implementations of Bayes filters strongly differ in how they represent the probability densities over the state x^{t+1}. In our implementation, we use Particle filters to estimate the state of the system.

Particle filters [110, 27, 72] comprise a broad family of sequential Monte Carlo algorithms for approximate inference in partially observable Markov chains. In robotics, early successes of particle filter implementations can be found in the area of robot localization, in which a robot's pose has to be recovered from sensor data [109]. Particle filters have been used to solve problems involving global localization [9] and kidnapped robot problems, in which a robot has to recover its pose under global un-

certainty.

Particle filters represent beliefs by sets of samples called *particles* and are denoted:

$$\mathcal{X}^{t+1} := x_{[1]}^{t+1}, x_{[2]}^{t+1}, \ldots, x_{[N]}^{t+1} \tag{4.3}$$

Here each particle $x_{[i]}^{t+1}$ (with $1 \leq i \leq N$) is a concrete instantiation of the state at time $t+1$. Put differently, a particle is a hypothesis as to what the true world state may be at time $t+1$. Particle filters approximate the belief $bel(x^{t+1})$ by a set of particles $\mathcal{X}^{t+1}$. Ideally, the likelihood for a state hypothesis x^{t+1} to be included in the particle set $\mathcal{X}^{t+1}$ shall be proportional to its Bayes filter posterior $bel(x^{t+1})$:

$$x_{[i]}^{t+1} \sim Pr(x^{t+1}|z^{1:t+1}, a^{1:t}) \tag{4.4}$$

As a consequence, the denser a subregion of the state space is populated by samples, the more likely it is that the true state falls in this region. Particle filters realize Bayes filter updates according to a sampling procedure, often called *sequential importance sampling* or *resampling*. Initially the state is represented by an uniform distribution of samples. Each particle is associated with an importance factor $w_{[i]}^{t+1}$. Importance factors are used to incorporate the measurement z^{t+1} into the particle set. The importance, thus, is the probability of the measurement z^{t+1} under the particle $x_{[i]}^{t+1}$, given by $w_{[i]}^{t+1} = Pr(z^{t+1}|x_{[i]}^{t+1})$. If we interpret $w_{[i]}^{t+1}$ as the *weight* of the particle, the set of weighted particles represent (in approximation) the Bayes filter posterior $bel(x^{t+1})$.

Importance sampling proceeds by drawing with replacement N particles from the particle set where the probability of drawing each particle is given by its importance weight. This resampling procedure transforms a particle set of N particles into another particle set of the same size. By incorporating the importance weights into

the resampling process, the distribution of the particle changes: before the resampling step, they were distributed according to $\overline{bel}(x^{t+1})$, after the resampling they are distributed (approximately) according to the posterior

$$bel(x^{t+1}) = \eta Pr(z^{t+1}|x_{[i]}^{t+1})\overline{bel}(x^{t+1}) \tag{4.5}$$

4.2 Action Selection

Ideally the goals for an action can be sampled from the Bayesian model given the environment model and observations. However, in the presence of partial information, choosing an action given that it may be expensive or destructive (with respect to sensor measurements) requires safeguards to ensure that the robot chooses the next action that will optimally lead towards successfully completing its intended task.

The state of a system describes the relevant system parameters determined by the observations of the dynamics of control programs (γ in Equation 3.1). In our representation, actions and observations are strongly connected, since making a new observation is directly related to the dynamics of an action. We use an information theoretic formulation to tackle the problem of optimal action selection for state estimation. Many key problems in computer vision can be formulated as state estimation problems—object classification (estimating the class of an object), pose estimation, object tracking. Our goal is to provide a mechanism for action selection that reduces the state uncertainty and ambiguity. Uncertainty arises from the noise in the sensor data, while ambiguity is based on the inherent structure of the problem, e.g., objects which are identical in different views.

In contrast to classical approaches for state estimation, our approach does not optimize a metric related to state estimator, like its variance. Instead, we make use

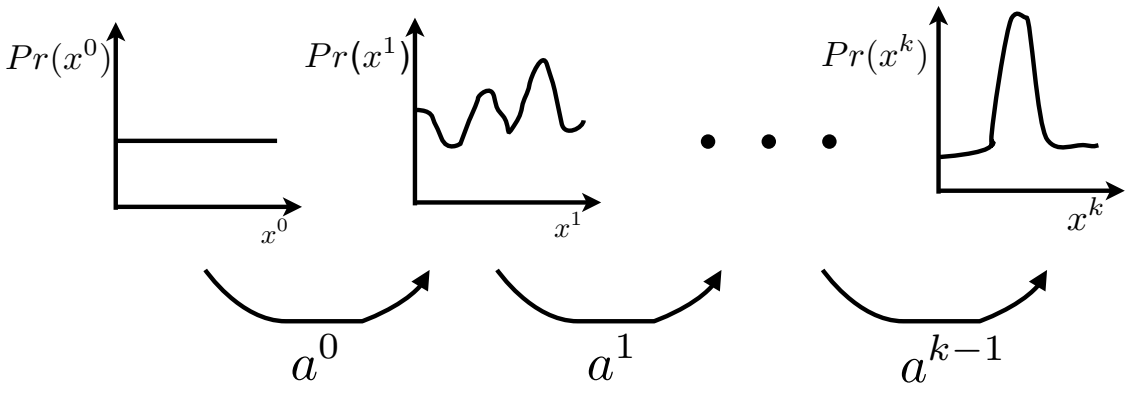


Figure 4.1. Uncertainty and ambiguity in the posterior distribution of the state x^t is reduced by choosing appropriate information-acquisition actions a^t.

of the knowledge that is encoded in the state estimator as conditional probability densities. The general principle of our work is depicted in Figure 4.1. A sequence of actions a^t is chosen in order to transform a prior distribution $Pr(x^t)$ over the state space to an unimodal distribution. Initially $Pr(x^t)$ is uniform if no knowledge about the state is available.

The problem of action selection reduces to selecting the next action that maximally reduces the uncertainty of the state. Entropy measures the amount of uncertainty in the value of a random variable x^t.

$$H(x^t) = -\sum_{x^t} Pr(x^t) \log(Pr(x^t)) \tag{4.6}$$

The entropy is zero if the outcome is unambiguous; it reaches its maximum if all outcomes are equally likely.

Since the true state x^{t+1} of the system cannot be observed, it needs to be inferred from the observations z^{t+1} made after taking an action a^t. The observation is related to the state by the likelihood function $Pr(z^{t+1}|x^{t+1}, a^t)$, which is proportional to

the probability that an observation z^{t+1} is made if an action a^t is taken to reach a particular state x^{t+1}. The likelihood function also serves as a model of the noise component of the stochastic actions. The probability density function $Pr(z^{t+1}|a^t)$ of the observation is defined as

$$Pr(z^{t+1}|a^t) = \sum_{x^{t+1}} Pr(z^{t+1}|x^{t+1}, a^t) Pr(x^{t+1}) \tag{4.7}$$

An entropy $H(z^{t+1}|a^t)$ can also be associated with the distribution $p(z^{t+1}|a^t)$. The important quantity in this formalism is the chosen action a^t. Since the likelihood function $Pr(z^{t+1}|x^{t+1}, a^t)$ is conditioned on the action, the action itself influences the observation. The goal is to estimate the true state x^{t+1}, given the observation z^{t+1}.

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

$$I(x^{t+1}; z^{t+1}|a^t) = H(x^{t+1}) - H(x^{t+1}|z^{t+1}, a^t) \tag{4.8}$$

Using the definitions of the entropies $H(x^{t+1})$ and $H(x^{t+1}|z^{t+1}, a^t)$,

$$I(x^{t+1}; z^{t+1}|a^t) = \sum_{x^{t+1}} \sum_{z^{t+1}} Pr(x^{t+1}) Pr(z^{t+1}|x^{t+1}, a^t) \log \left(\frac{Pr(z^{t+1}|x^{t+1}, a^t)}{Pr(z^{t+1}|a^t)} \right) \tag{4.9}$$

Since the goal of the planner is to reduce the uncertainty, it has to maximize the mutual information. The optimal action $\hat{a}^t$ to execute next, given a belief over states $Pr(x^t)$ and observation model $Pr(z^{t+1}|x^{t+1}, a^t)$ is

$$\hat{a}^t = \arg\max_{a^t} I(x^{t+1}; z^{t+1}|a^t) \tag{4.10}$$

4.3 Action Execution

The runtime parameterization of an action selected for execution is governed by either the policies of the sensorimotor programs (SearchTrack schema) or the objective function of the primitive controllers. In case of a sensorimotor program, its policy describes the sequence of actions that achieves the objective. The program contains contingencies for various runtime scenarios that can occur in the presence of a particular environmental context. Having a rich suite of sensorimotor programs that can deal with the run time requirements of a task allows a planner to plan actions at a level of abstraction where many of the low level details about an action can be ignored.

4.4 Experiment

In this section, we present an experiment that uses information theoretic action selection to reduce uncertainty over the state of an object. The experiment was conducted using uBot-5, shown in Figure 4.2. The uBot-5 is a dynamically balancing mobile manipulator with 4 degrees of freedom in each arm, a torso rotation, and two wheels [23]. The balancing is performed using a LQR controller that compensates for forces exerted upon it during navigation. Each arm of the robot terminates on a small ball.

The control architecture of the robot is implemented in Robot Open Source (ROS) [92]. Perception is performed using a Kinect sensor, and the ARToolkit augmented reality tags (ARTags) [58]. The ARTags are placed in known configurations on the object, allowing the robot to localize the pose of each feature reliably.

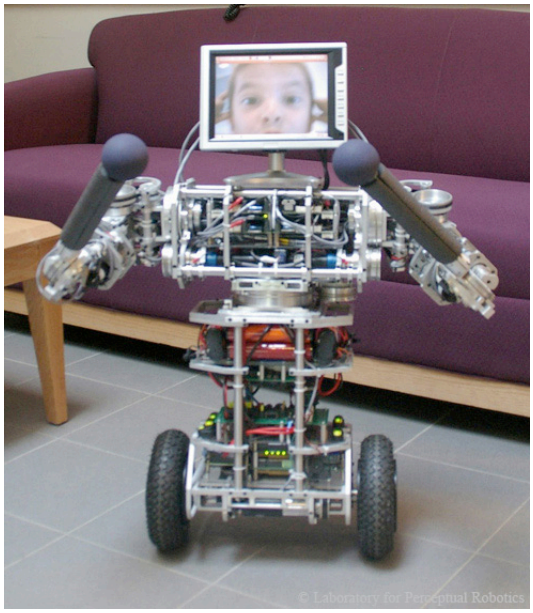


Figure 4.2. The uBot-5, a dynamically balancing mobile manipulator.

4.4.1 Object Recognition

Object recognition is still an open problem. From the choice of features to the actual classification problem, we are still far from the 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 challenges that arises from using such databases is the inability to look at an object from different poses that would provide different, and probably more discriminative views of objects that can remove the ambiguity in recognition. We hypothesize that having a robot that can interact with objects allows the system to deal with partial observability arising from the inability of the robot to see the whole object from a single viewpoint.

The objects used in this task were two boxes. Each face of the box has an ARTag feature associated with it. Figure 4.3 shows a flattened image of all the faces of each box with the associated ARTag. The features marked by the red and blue colors are

repeated. Having ambiguous features leads to partial observability of state, whereby seeing either the features in red or blue is not enough for the robot to completely determine the object state. Figure 4.3 also highlights the discriminating feature in each box. Both the objects look visually similar except for the one discriminating face. The model of the objects contain a set of visual tracking actions which track the ARtag features. Figure 4.4 shows the various visual actions present in the model. The models also contain a set of actions that manually interact with the objects :

- PULL : Brings the arms closer to the robot while maintaining a contact force in its end effector.
- GRASP : Moves the arms to obtain a contact load in its end effector.
- ROTATE-X : Rotates the object counterclockwise around the X-axis.
- ROTATE-Z : Rotates the object counterclockwise around the Z-axis..

Figure 4.5 shows the effect of the ROTATE-X and ROTATE-Z actions on a box.

Figure 4.6 and Figure 4.7 shows the action selection process for two instances of the object recognition task. In both cases, the planning algorithm proceeds by first estimating the state of the environment based on the observations. The estimated state is used to compute the best next action to execute that maximally reduces uncertainty over objects. This procedure leads to different plans being executed based on the agent's belief over objects. Figure 4.6 shows a plan in which the robot's uncertainty over objects is completely reduced after executing the action sequence PULL→ROTATE-X. Figure 4.7 presents a plan in which the robot had to execute a longer action sequence PULL→ROTATE-Z→PULL→ROTATE-Z before it can recognize the object.

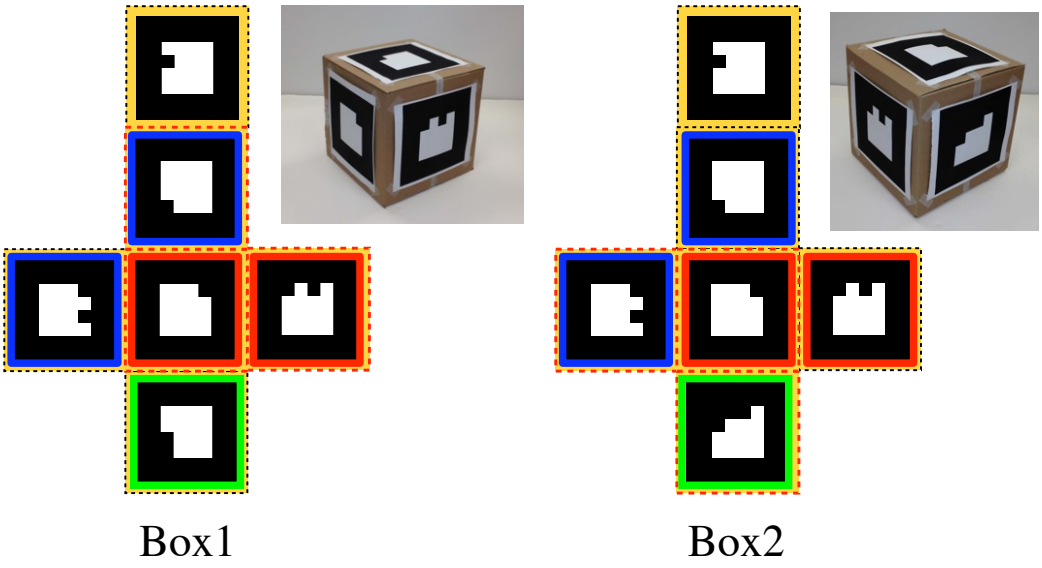


Figure 4.3. A flattened image of the two boxes showing the various ARtag features on each of its faces. The red and blue colors indicate the symmetry in the features present in each box. The green colors indicate the discriminating face for each box.

Figure 4.4. The set of visual actions being used to model the objects. The visual actions track a set of ARtag features.

This experiment shows that using mutual information as a metric for action selection, a planner at every iteration of the planning loop can select actions that maximally reduces its uncertainty over a particular state. Furthermore, the planner fuses information about the executed action and its observations to maintain an updated belief over the state. Thus the action selection policy while being greedy also maximizes the probability of the planner to achieve its objective given the information state.

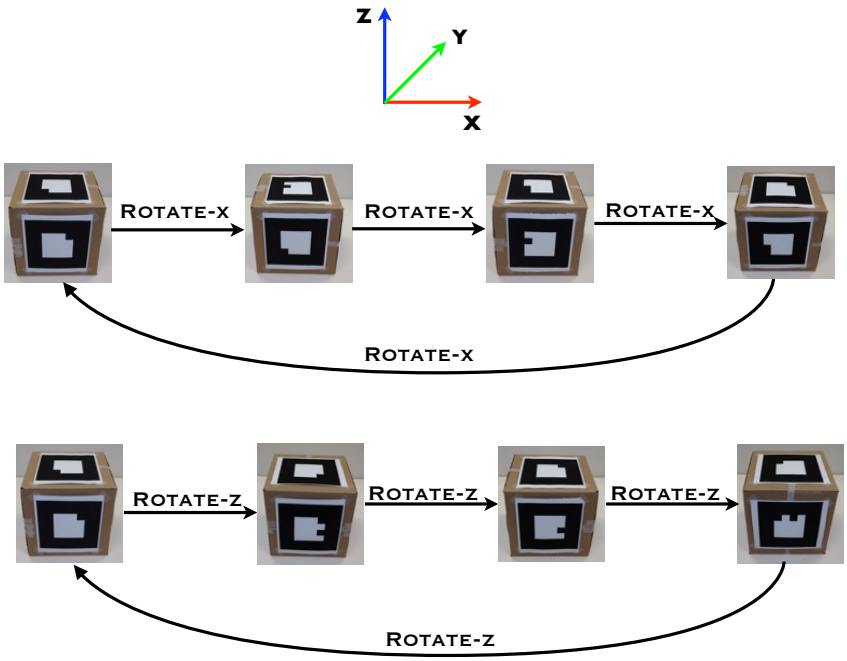


Figure 4.5. The effect of taking the ROTATE-X and the ROTATE-Z actions on Box1. ROTATE-X rotates the box counterclockwise around the X-axis. ROTATE-Z rotates the box counterclockwise around the Z-axis.

4.5 Related Work

Manipulation planning requires determining a goal configuration for possibly several objects, and generating a sequence of manipulation actions that result in the desired configuration [1, 74]. The planning community either uses geometric planners that plan in the configuration space of the robot and the world, or a propositional logic based planner that expresses the state of the world as logical assertions engineered by humans [70]. Planning in hybrid spaces, combining discrete mode switching with continuous geometry has also been used to sequence robot motions involving different contact states or dynamics [45]. Cambon *et al.* [15] showed how linking symbolic description to its geometric counterpart can lead to an integrated planning process that is able to deal with intricate symbolic and geometric constraints. Plaku and Hager [91] extended this approach to handle robots with differential constraints and provide a utility-driven search strategy. Choi and Amir [18] use a hand-built geometri-

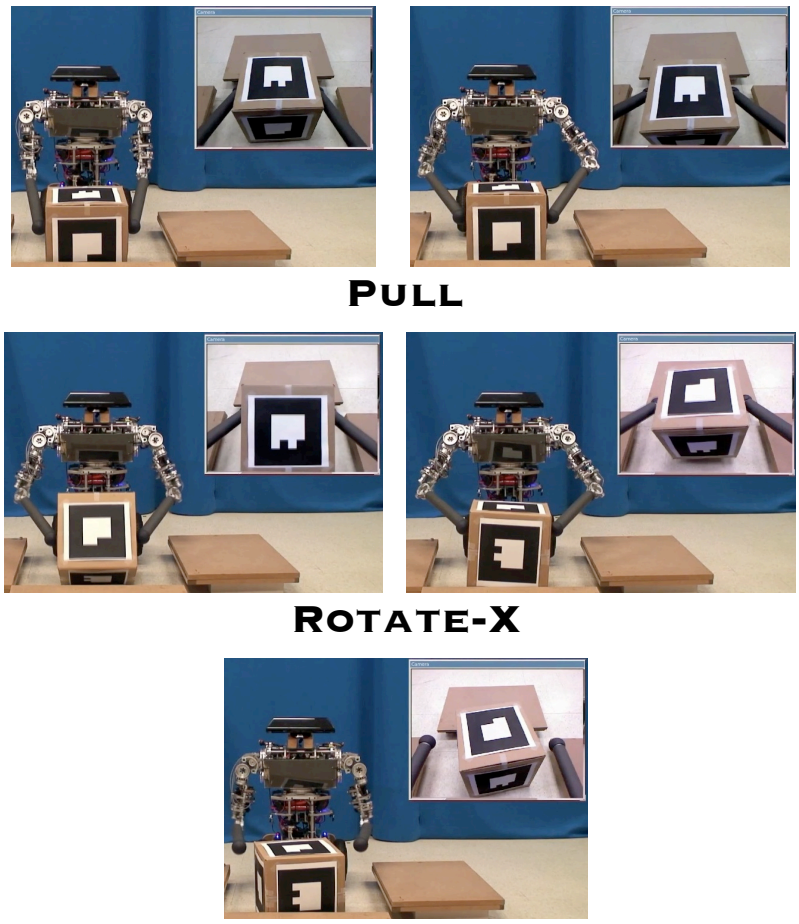


Figure 4.6. The robot performs the action sequence: PULL→ROTATE-X as part of the action selection process to recognize Box1.

cal roadmap and "lift" the representation to form a symbolic description for planning.

Hierarchical approaches to planning have been proposed to speed up the search for plans. Since the work of Sacerdoti [94] 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 [84]. Wolfe *et al.* [117] provided a task and motion planner based on hierarchical transition networks (HTNs) [81]. Kaelbling and Lozano-Perez [55] 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 information state. 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).

The usefulness of information theoretic concepts have been recognized recently, specially in the field of computer vision, with applications like image registration [115], viewpoint selection in object recognition [10, 98, 24], and feature extraction [52]. In [98], an active object recognition scheme uses mutual information to place receptive fields optimally over the object of interest. Denzler [24] used mutual information in a sequential decision process to take actions that explicitly changes the prior distribution. Our work is closest in spirit to this approach, in which the robot selects the best possible action based on the current information state. However, our actions are not limited to visual tracking actions, but also involves manipulating the object to reduce uncertainty.

Information theoretic concepts have also been used for active vision and action selection. Examples are active localization of robots [35], active viewpoint selection for object recognition [2], and sensor planning for active object search [119]. In [103], the concept of entropy was used for active scene exploration, where the uncertainties of each object in the scene is decreased by computing the utility of pan/tilt/zoom settings. In the control literature, entropy has been used for optimal state estimation from sensor fusion. Noonan *et al.* [83] used entropy for sensor fusion in linear dynamic systems. They performed sensor fusion by applying the relationship between

the Fisher information matrix and Cramer-Rao lower bound on the error in state estimation.

4.6 Conclusions

This chapter presented a information theoretic planning algorithm that can be directly applied to the Bayesian models for task specific planning. The planner builds plans at a level of abstraction where the run-time parameterization of control programs can be ignored. We showed how mutual information can be used for selecting greedy actions that reduce uncertainty over hidden states.

However, not all planning tasks involve reducing the uncertainty over a particular random variable. Many tasks, especially in robotics, require the robot to achieve a particular goal state. For example, the goal can be to achieve grasp on an object or to manipulate an object to make some feature visible. Action selection for such tasks cannot be performed by using a mutual information metric. In the next chapter, we present a POMDP based planning algorithm that can handle both uncertainty reduction tasks as well as those requiring achieving a goal state.

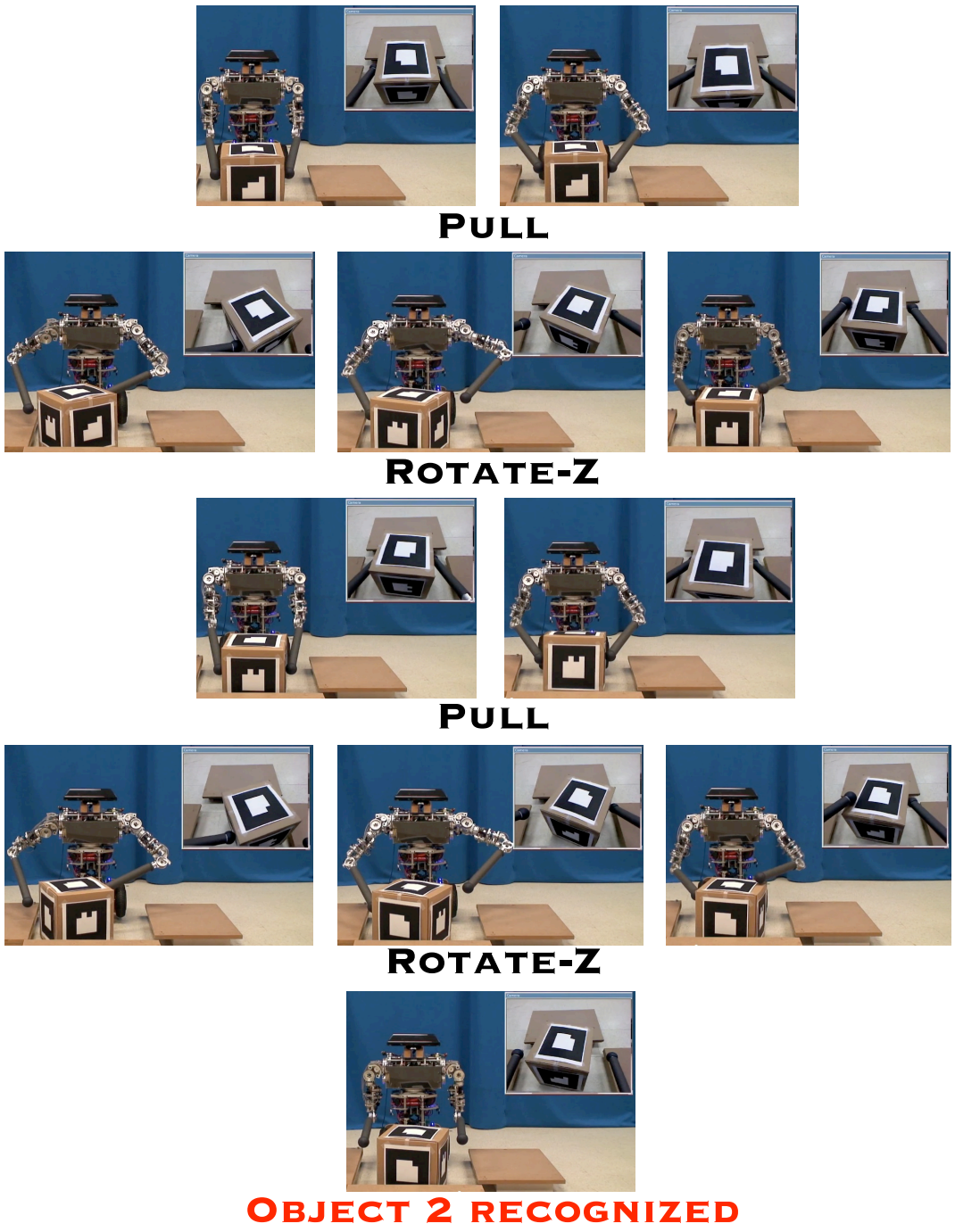


Figure 4.7. The robot performs the action sequence: PULL→ROTATE-Z→PULL→ROTATE-Z as part of the action selection process to recognize Box2.

CHAPTER 5

POMDP-BASED PLANNING

While information completeness and determinism are useful approximations for planning domains at some abstract level, they may not help in most cases. Planning based on Markov Decision Processes (MDPs) is designed to deal with nondeterminism and partial observability. Its key idea is to represent planning as an optimization problem, in which planning algorithms search for a plan that maximizes a utility function.

A great deal of progress has been made on the problem of planning motions for robots with many degrees of freedom through free space [68, 70]. These methods enable robots to move through complex environments, as long as they are not in contact with the objects in the world. However, as soon as the robot needs to contact the world, in order to manipulate objects, open-loop strategies are no longer robust. The fundamental problem with planning for motion in contact is that the configuration of the robot and the objects in the world is not exactly known at the outset of execution, and, given the resolution of the sensors, it cannot be exactly known. In such cases, traditional open-loop plans are not reliable.

In this chapter, we build on those ideas, addressing the weaknesses in the approach via probabilistic representation. By modeling the initial uncertainty using a probability distribution, and doing the same for uncertainties in action, one can choose plans that optimize a variety of different objective functions including the plan most likely

to achieve the goal. The probabilistic representation also affords an opportunity for computational savings by focusing on parts of the space that are most likely to be encountered.

By building an abstraction of the underlying continuous state and action spaces, we lose the possibility of acting optimally, but gain an enormous amount in computational simplification, making it feasible to compute solutions to real problems. We will be using the methods of model minimization and state abstraction (described in Chapter 3) to create an abstract model of objects, and then model the problem of choosing actions under uncertainty as a partially observable Markov decision process (POMDP) [101].

5.1 Approach

Partially observable Markov decision processes are the primary model for formalizing decision problems under uncertainty. A POMDP model is given by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R} >$,where

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.
- $\mathcal{O}$ is a set of observations.
- $\mathcal{T}$ is the state-transition model $P(s^{t+1}|s^t, a^t)$ that specifies a probability distribution over the resulting state s^{t+1} given an initial state s^t and action a^t.
- Ω is the observation model $P(o^{t+1}|s^{t+1}, a^t)$ that specifies the probability of making an observation o^{t+1} in state s^{t+1} after executing action a^t.
- $\mathcal{R}$ is the reward function mapping state-action pairs to an immediate reward.

Problems that are naturally described by having a goal state can be encoded in this framework by assigning the goal states a high reward.

Given the model of a POMDP, the problem of optimal control can be broken into two parts: state estimation, in which the probability distribution over the underlying state of the world, or *belief state*, is recursively estimated based on the actions and observations of the agent; and the policy execution in which the current belief state is mapped to the optimal control action.

Belief-state update is a straightforward instance of a Bayesian filter as explained in Section 4.1. The problem of deriving an optimal policy is much more difficult. The policy for a POMDP with n states is a mapping from the n-dimensional simplex (the space of all possible belief states) into the action set. Although a policy specifies only the next action to be taken, the actions are selected by virtue of their long-term effects on agent's total reward. Generally, we seek policies that choose actions to optimize the expected total reward over the next k steps (finite-horizon) or the expected infinite discounted sum of reward, in which each successive reward after the first is devalued by a discount factor.

These policies are quite complex because, unlike in a completely observable MDP, in which an action has to be specified for each state, in a POMDP, an action has to be specified for every probability distribution over states in the space. Thus, the policy will know what to do when the robot is completely uncertain about its state, or when it has two competing possibilities.

5.1.1 Action Selection

Computing the exact optimal finite or infinite-horizon solution of a POMDP is generally computationally intractable. However, it is often possible to derive good approximate solutions by taking advantage of the fact that the set of states that are reachable under a reasonable control policy is typically dramatically smaller than the original space [89, 102, 104].

We used a novel POMDP planning algorithm called Heuristic Search Value Iteration (HSVI) [102]. HSVI, a form of point-based value iteration, is an anytime algorithm that returns a policy by sampling belief states that have a relatively high probability of being encountered, and concentrates its representational and computational power in those parts of the belief space. It get its power by combining two well known techniques: attention-focusing search heuristics and piecewise linear convex representations of the value functions.

HSVI stores the upper and lower bounds on the optimal value function V^*. It performs a local update at a specific belief, where the beliefs to update are chosen by exploring forward in the search tree according to certain heuristics that select actions and observations. HSVI makes asynchronous updates to the value function bounds by basing its heuristics on the most recent bounds when choosing which successor to visit. This technique uses a depth-first exploration strategy, because a breadth-first heuristic search typically employs a priority queue, and propagating the effects of asynchronous updates to the priorities of the queue elements would create substantial extra overhead.

HSVI returns policies in the form of a set α-vectors and associated actions. The expected discounted sum of values when executing this policy from some belief state

b is

$$V(b) = \max_{\alpha_i} b \cdot \alpha_i \tag{5.1}$$

and the best action is the action associated with the maximizing alpha vector. The alpha vectors define hyperplanes in the belief space, and the maximization over them yields a value function that is piecewise-linear and convex. By construction, each of the α-vectors is maximal over some part of the belief space. The space is partitioned according to which α-vector is maximizing over that region.

To execute a policy, we apply a state estimator as described earlier. The state estimator starts in some initial belief state, and then consumes successive actions and observations, maintaining the Bayes optimal belief state. To generate an action, the current belief state is dotted with each of the α-vectors, and the action associated with the maximal α-vector is executed.

5.2 Experiments

We present two tasks that show the advantages of using a POMDP based planner. The first task requires the robot to select actions to achieve a goal state. The second task is the same as the one presented in Chapter 4 where the goal is to reduce uncertainty over objects.

5.2.1 Achieving a Goal State

Most planning tasks in robotics require an agent to achieve a particular goal state. The goals can be rather general (grasp a cup), or can be specific (grasp the handle of a cup). Our representation framework allows a planner to express both these kinds of goals—from very specific to very general. The goal observation z_g is given by:

$$z_g = \{(\gamma)_a | a \in \mathcal{A}, \gamma \in \{-, +\}\} \tag{5.2}$$

As the number of elements in z_g increase, the goals become more specific.

In this experiment, the task for the robot is to successfully track a particular visual feature—$ARtag_4$. Figure 5.1 shows the goal for the planner. In terms of our state representation, this goal can be expressed as:

$$\gamma(ST|^{ARtag_4}_{pan-tilt}) = \text{“}+\text{”}$$

The reward function R for such a task can be computed directly from the object model.

$$R = Pr(z_g^{t+1}|x^t, a^t) \tag{5.3}$$

where x^t is the hidden state of the system and a^t is the action being executed.

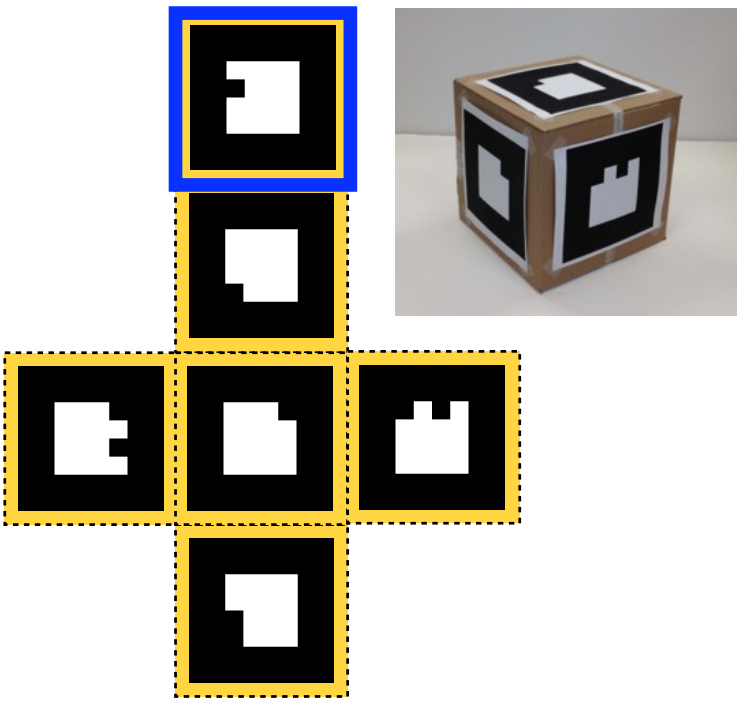


Figure 5.1. The goal for the planning task is to select actions to interact with the box to make the goal face (shown in blue) visible.

Given the reward function, we used Heuristic Search Value Iteration as described in Section 5.1.1 to compute the approximate policy. Figure 5.2 and Figure 5.3 show two runs of the computed policy. The policy execution proceeds in a similar manner as in the case of the information theoretic planner. The planner makes observations to update its belief of the system state. It then selects the action associated with the maximal alpha vector. This process proceeds until the goal state is reached. The policy returned by the planner directly handles uncertainty by selecting "exploration" actions when its belief over the state is ambiguous. Figure 5.2 shows an execution of the policy where the planner executed the action sequence PULL→ROTATE-X→PULL→ROTATE-X to orient the object in a manner where the goal state was achieved (the goal feature can be seen on the top of the box in the final state). Figure 5.3 shows another run of the same policy. Here, the planner selects a different action sequence (ROTATE-Z→PULL→ROTATE-Z) to achieve the goal state.

The two runs of the above experiment show that the plan contains all contingencies (given a complete object model) for achieving the goal task. The planner proceeds by executing the action associated with robot's present belief of the state. As the belief gets updated with the current observations, the action selected by the planner changes to reflect its present belief.

Expressing goals as binary assertions over the space of actions allows a planner to search for plans in a discrete observation space, without worrying about the runtime parameters of the action. However, if the task explicitly defines a goal in the level of spaces, as opposed to the level of actions (For example, grasping the cup at a particular position and orientation), the planning problem becomes computationally intractable, since the number of possible states become infinite.

5.2.2 Object Recognition

The formalization of active object recognition as a POMDP has not been well studied until recently. Eidenberger [29] linked POMDP rewards to the expected minimization of information entropy (uncertainty over a state variable) from the next observation. In practice, the reward of future actions needs to be computed online, since it depends on the entropy of the current state. The dependency of rewards on the current state means that the robot has to solve a POMDP after each observation, which is a very costly process. In our approach, since our object models capture all possible state transitions that the planner can encounter, it allow us to define the reward apriori. This enables us to solve the POMDP problem offline.

In order to transform a recognition problem into a problem of action selection, one that can be solved using a POMDP, we introduce a set of *dummy* actions called RECOGNIZE that correspond to the act of recognizing the object. The recognition of an object is equivalent to the recognition of one of the states that best disambiguates the object. The set of actions is thus defined as:

$$\begin{aligned}\mathcal{A} = \{&ST|_{pan-tilt}^{ARtag_1}, ST|_{pan-tilt}^{ARtag_2}, ST|_{pan-tilt}^{ARtag_3}, ST|_{pan-tilt}^{ARtag_4}, ST|_{pan-tilt}^{ARtag_5}, Pull, \\ &Grasp, Rotate-X, Rotate-Z, Recognize_1, \ldots, Recognize_N\}\end{aligned}$$

where N is the total number of objects being considered.

We have an extra state, the *Sink*, where the robot enters after an attempt to recognize an object. The state transition function T for the *Sink*-state is given by

$$T(sink^{t+1}|x^t, a^t) = \begin{cases} 0 & : \quad a^t \notin \{recognize_1, \ldots recognize_n\} \\ 1 & : \quad a^t \in \{recognize_1, \ldots recognize_n\} \end{cases} \tag{5.4}$$

This ensures that the RECOGNIZE action is taken only when the planner is completely sure about the object.

The robot receives reward when it recognizes an object correctly. The recognition of the object corresponds to the recognition of at least one of its corresponding states. This is encoded in the rewards the robot receives.

$$R(recognize_i|x^t) = Pr(o_i|x^t) \tag{5.5}$$

where $Pr(o_i|x^t)$ is the probability of the object presented being object i given the state.

Figure 5.4 shows the objects being used in the object recognition task. Both objects look visually similar except for one discriminating face. Figure 5.5 and Figure 5.6 show two runs of the computed policy. In each run, one of the objects was presented in front of the robot in an unknown orientation. The policy selects actions in a manner that reduces its uncertainty over objects with the final action selected being the RECOGNIZE action.

The above experiment shows that expressing the problem of uncertainty reduction as a problem of action selection over a POMDP allows a planner to choose actions based on its belief while simultaneously making progress towards disambiguating the object. Though the task described above showed how to use POMDPs for object recognition, the same technique can be used directly for other tasks involving uncertainty reduction. If the goal of the planner is to reduce uncertainty over objects to the extent that it can infer if an object is graspable or not, the reward function and the RECOGNIZE action can be appropriately defined.

5.3 Related Work

A Partially Observable Markov Decision Process (POMDP) is a generalization of a Markov Decision Process that provide techniques for calculating optimal control actions under uncertainty. They extend MDPs to domains where considerations of hidden state are crucial for task performance. They have proved useful for a wide range of real world domains such as robot navigation [93, 99, 85] and robot interaction [22, 90]. Unfortunately, POMDPs of the size necessary for optimal robot control are an order of magnitude larger than what today's best, exact POMDP algorithms can tackle [54]. However, robotic applications can yield highly structured POMDPs, where certain actions are only applicable in certain situations. To exploit this structure, Pineau [90] developed a hierarchical version of POMDPs that breaks down decision making into a collection of smaller problems that can be solved more efficiently. Their use of POMDPs at the highest level of behavioral control is in contrast to existing applications. This approach has been applied to the task of guiding elderly people with mild physical and cognitive disabilities in a nursing home [90]. It required the robot to navigate to a person's room, alert them, inform them of an upcoming event or appointment, and then guide the person, while carefully monitoring the person's progress and adjusting the robot's velocity and path accordingly.

Uncertainty is a key issue when determining object and action parameters. Ek [30] presented a system that is able to infer the commanded task and reason about action selection given information derived from partial observations. In this work, an optimal perceptual action is defined to be the action that will maximally disambiguate (reduce entropy over) the state-space. In [28], Dragiev utilizes Gaussian processes to dilate expectation for object pose in the context of reaching and grasping tasks. Recently, Petrovskaya [87] presented the Guaranteed Recursive Adaptive Bounding (GRAB) algorithm for efficient approximate inference that was tested in the context

of a manipulation environment by accurately localizing an object's pose from a set of relative sensor measurements.

While the use of POMDPs for both planning and control does sound very promising for real world tasks such as navigation, their use in manipulation tasks has been relatively unexplored until recently. Hsiao [46] provided a method for planning under uncertainty for simple grasping problems by partitioning the configuration space into a set of regions that are closed under compliant motions. She further showed how this approach can be used for task driven manipulation of objects where there is uncertainty about the relative pose of the robot and the objects [47]. She presented a decision theoretic framework in which the robot iteratively minimizes its uncertainty in object pose by probing an object.

Kaelbling and Lozano-Pérez [55, 56] developed an approach to integrated task and motion planning that integrates geometric and symbolic representations in an aggressively hierarchical planning architecture, called Hierarchical Planning in the Now (HPN). They showed that the hierarchical decomposition allows efficient solutions to problems with very long horizons while the symbolic representations support abstraction in complex domains with large number of objects. They handled uncertainty over future states by planning in approximate deterministic models, performing careful execution monitoring, and replanning when necessary. The uncertainty in current state is handled by planning in the belief space. Their approach to making planning tractable is to construct a temporal hierarchy of short-horizon problems, thus reducing the complexity of the individual planning problems to solve. The hierarchical approach is not guaranteed to produce optimal plans. It is, however, complete in domains for which the goal is reachable from every state.

5.4 Conclusions

This chapter presented a POMDP based planning algorithm for action selection. We showed that POMDPs can be used for both recognition tasks as well as achieving a goal state. The reward function for both tasks can be computed directly from the probabilistic forward models of objects. This allows the planner to compute a policy at runtime given the task description. Since the computed policy specifies an action for every probability distribution over states in the space, it has all contingencies computed for selecting actions when the robot is completely uncertain about the state. The actions selected can be exploratory in nature or goal directed based on planner's estimate of the state and the goal.

It is worthwhile to note that the robot was not trained to perform a particular task, and hence the representation was not chosen to fit the task. However, our functional representation of knowledge allows a robot to reuse its knowledge for different tasks by using the Bayesian models as forward models for planning and extracting the reward function directly from the models.

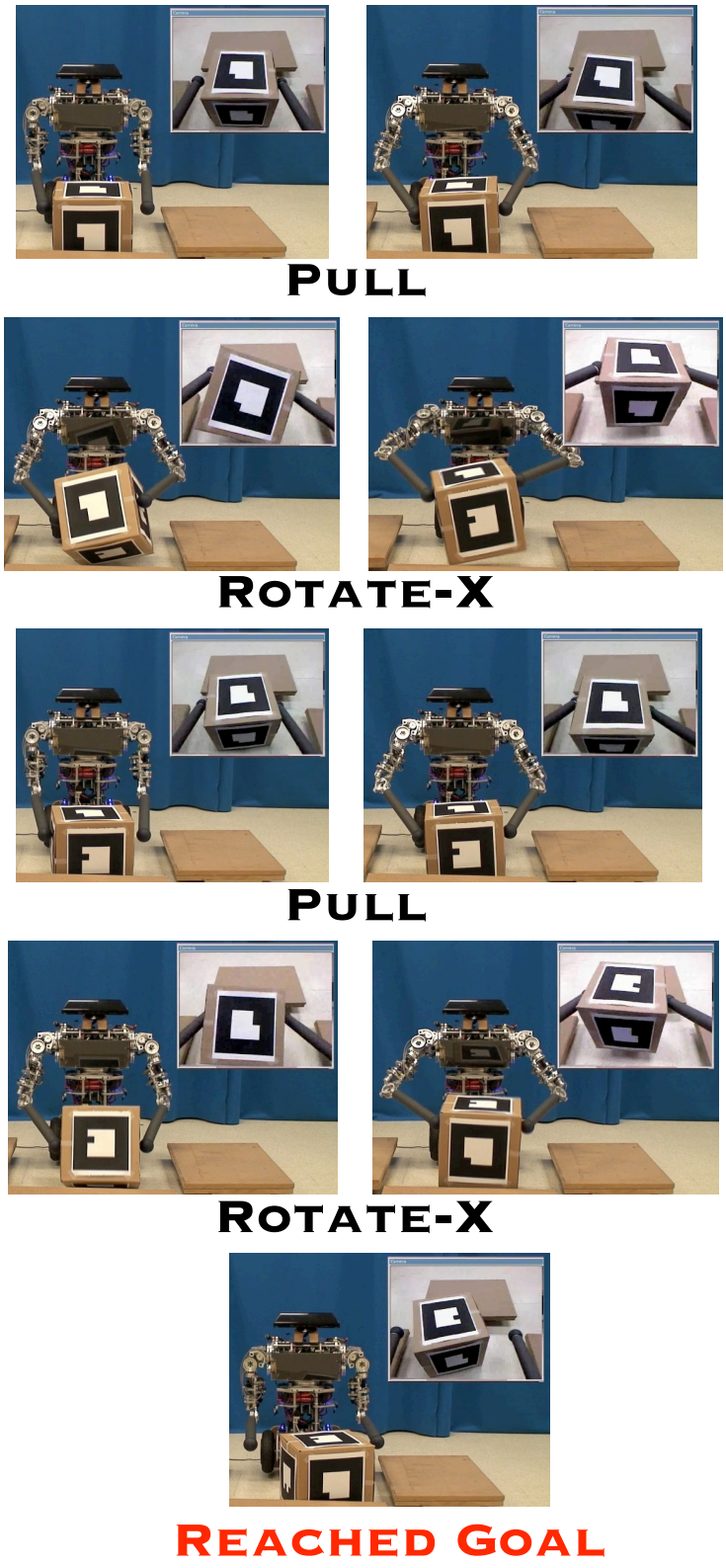


Figure 5.2. The robot performing the action sequence PULL→ROTATE-X→PULL→ROTATE-X to reach the goal state. After reaching the goal state, the goal feature can be seen on the top face of the box.

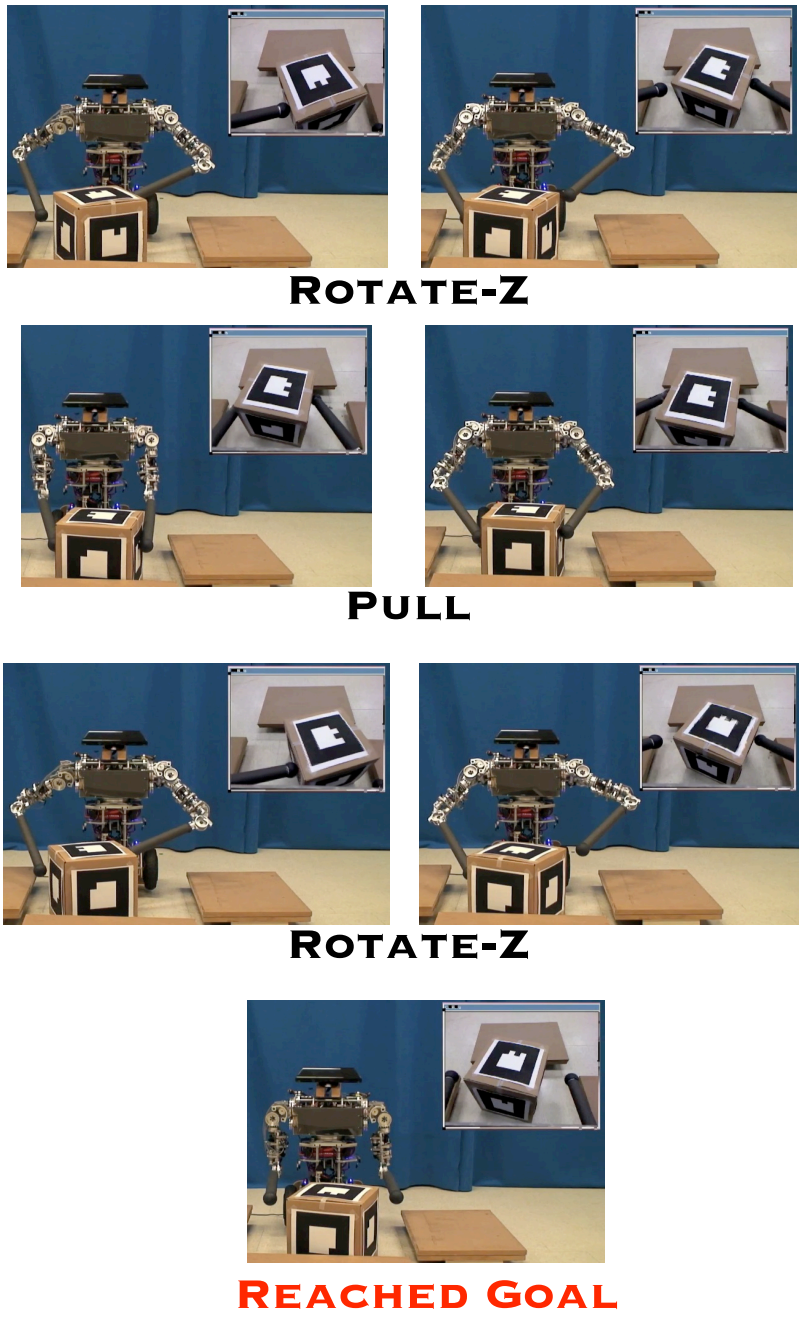


Figure 5.3. The robot performing the action sequence ROTATE-Z→PULL→ROTATE-Z to reach the goal state. After reaching the goal state, the goal feature can be seen on the front face of the box.

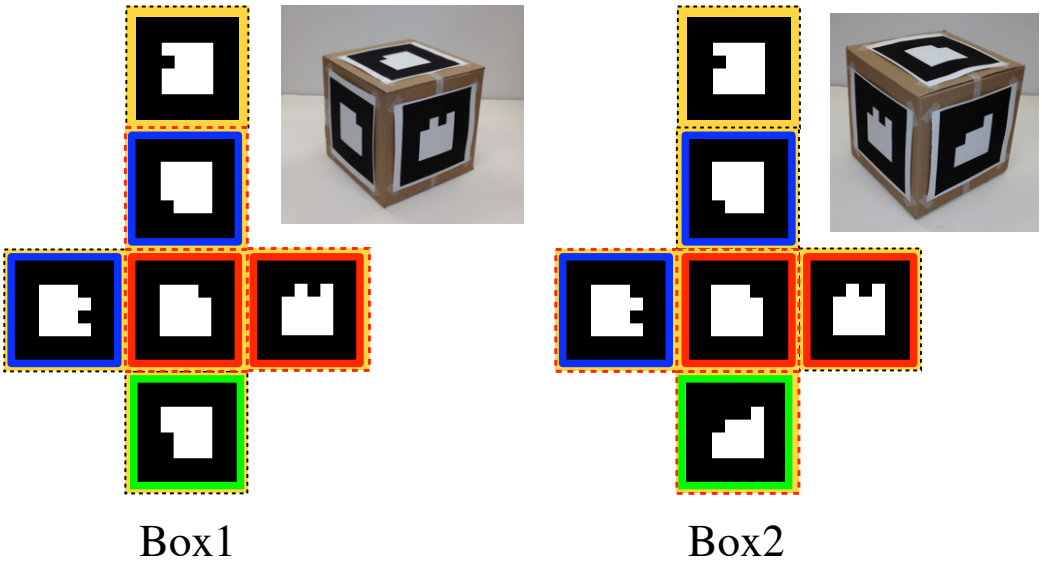


Figure 5.4. A flattened image of the two boxes showing the various ARtag features on each of its faces. The red and blue colors indicate the symmetry in the features present in each box box. The green colors indicate the discriminating face for each box.

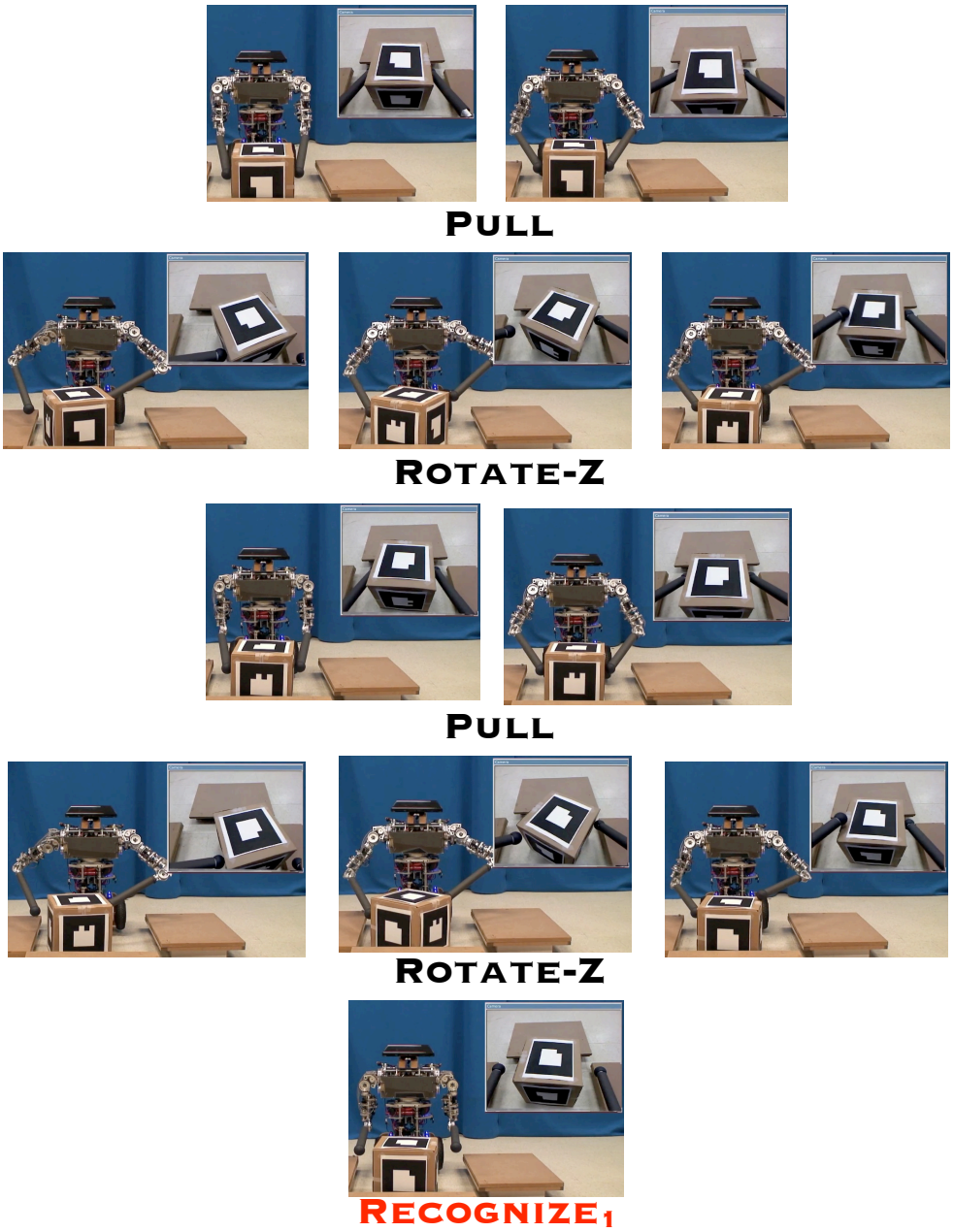


Figure 5.5. The robot performs the action sequence: PULL-ROTATE-Z-PULL-ROTATE-Z-RECOGNIZE$_1$ as part of the policy to recognize Box1.

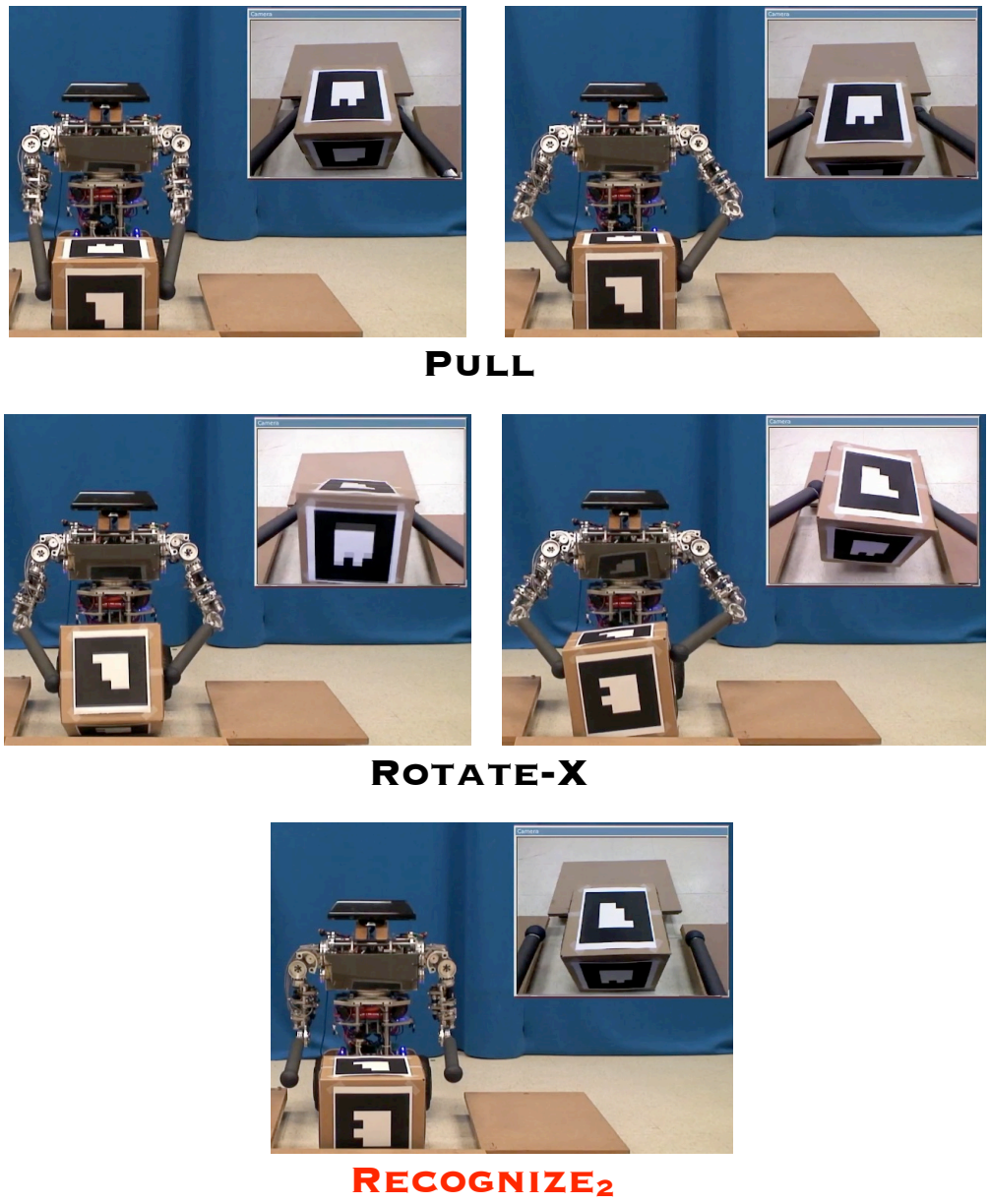


Figure 5.6. The robot performs the action sequence: PULL-ROTATE-X-RECOGNIZE$_2$ as part of the policy to recognize Box2.

CHAPTER 6

CONCLUSIONS AND DISCUSSIONS

The goal of this thesis was to present a holistic approach to planning robot behavior, using previously acquired skills to represent control knowledge directly, and use this background knowledge to build plans in the space of control actions.

Chapter 2 presented the representational foundations that lets a robot to accumulate control knowledge by direct interaction with the world. I showed how a robot can make use of low-level controllers and their dynamics to learn sensorimotor programs (*schemas*). I showed how such programs can be composed hierarchically to learn programs that can interact reliably with multiple stimuli from the environment. These sensorimotor programs provided the basis for control and modeling in our framework. The schemas capture common sense knowledge acquired by the robot, where the acquired programs and their long term statistics represent a domain general way of interacting with stimuli in the environment.

Chapter 3 presented a Bayesian framework that was used by a robot to model its environment in terms of distributions over the state of control programs. The knowledge accumulated by the robot models the dynamics of the environment as learned by direct interaction with the world. I showed how such a model can be used for both object recognition as well as a forward model to build plans.

Chapters 4 and 5 presented two planning algorithms that can be applied to the learned models for task-specific planning. The planner uses the probabilistic forward models to come up with a high level plan. I showed how the planner adapts to run time feedback by taking task directed exploratory actions that yield better predictions and improve planning performance.

6.1 Future Work

While the knowledge representation framework presented in this dissertation provides a powerful way to bridge the gap between autonomous skill acquisition and model-based planners, the work can be extended in several directions.

6.1.1 Learning Robust Object Models

The object models used for the various planning tasks used fiducial markers such as ARtags, since they are a lot more robust to sensory noise and variations in lighting. A future direction of this work can be to learn object models that uses primitive visual features such as color, shape, and texture. The models can be made robust by learning them in varying lighting conditions and environmental context. Since the learned models should reflect patterns over reliably trackable stimuli, control actions having high variance for an object can be ignored as they are not very informative. This would allow robot to learn models about its environment directly from interaction while ignoring the parts of its state space that are not reliable and hence can't be used for inference.

6.1.2 Learning Multi-Object Relationships

The Bayesian formulation of objects as spatially structured schemas provides a powerful mechanism for autonomous learning and planning for a robot performing manipulation tasks. However, until now, we have only considered the case where one object is present in the environment. This is almost always not the case. For

example, to grasp a tool lying on a table, a robot needs to interact with two objects—tool and table. While each object in isolation can be described by their own model, the model distributions change when objects are interacting (or maintaining certain spatial relationships) with one another. An action that reaches for the tool lying on the table cannot choose any grasp goal on the tool that is in contact with the table. Our presented Bayesian formulation needs to be extended in order to provide a principled way of re-computing these distributions based on observations that are consistent with multiple objects.

6.1.3 Probabilistic Inference on POMDPs

Toussaint *et al.* [113] proposed a new approach to planning and goal-directed behavior by using probabilistic inference on graphical models that represent states, actions, constraints, and goals. They showed that by using graphical models to specify the dependencies across multiple time-steps, one can reason about the effects of actions in the future. In their framework, inference was viewed as an internal simulation for control, planning, and decision making. This idea has been applied to low-level motor control [112], as well as high level planning [66, 67]. In [114], these methods were integrated to let a robot perform goal-directed behavior at various levels of abstraction. These inference techniques to directly extract plans from a Dynamic Bayes Net can also be applied to our knowledge representation to learn policies. This would allow a robot to compute policies *on-demand* based on the task requirements.

6.1.4 Dexterous Mobility and Manipulation

Humans exhibit remarkable *motor resourcefulness* with respect to tasks where the same task, based on environmental constraints, utilizes different motor policies to achieve its objective. These constraints can be resource (availability of effectors), accuracy (fine manipulation vs. coarse manipulation), or energy. One of the best examples of motor resourcefulness is an human's ability to use its arms for mobility

(rock climbing), or its legs for manipulation (use the pedals in the car). Since our goal is to develop robots that exhibit dexterity, a robot needs to be able to select both mobility and manipulation actions (not restricted to a particular group of effectors) based on the task. A future extension of our work will be to build object models where the actions available include both mobility and manipulation behaviors. For example, in our experiments involving manipulating the box, the action of rotating the box to re-orient it could very well have been performed by re-orienting the robot itself, by moving the robot with respect to the box. Having available such alternate motor solutions can allow a planner to select different plans based on environmental constraints.

6.2 Discussions

The central goal of this dissertation was to take a small step towards achieving dexterous behavior in robots. Learning needs to be the core feature of any robot working in unstructured environments. Nonetheless, present-day learning approaches fail to take into account that learning is never a finished process but an everyday task for biological systems. In the future, robots should be able to acquire new skills by exploration and *play* (much like animals), and use its acquired knowledge to act purposefully in the world. Having a unified representation for planning and learning will not only allow a robot to build and execute plans, but also to find deficiencies in its skills and models.

McCarthy and Hayes [76] first presented the frame problem in the context of the background necessary to predict the change in state expected as a consequence of the actions of an agent. The problem states that it is a practical impossibility to describe or infer all the necessary preconditions and all the possible consequences of a given action. It is not only difficult to determine what changes and what does not, it is

also difficult to determine what is relevant. I propose that a failure in the execution of a plan indicates that implicit knowledge structures in the form of hierarchical behavior used as forward models by the planner do not capture salient features of the environment that are necessary for "framing" this task. Thus, the failure of a plan can be used by the learning agent to direct subsequent exploration to skills that need further improvement. This *seeds* a learner to either explore a part of robot's state space to improve its skills or build more complete models.

BIBLIOGRAPHY

[1] Alami, R., Laumond, J. P., and Siméon, T. Two manipulation planning algorithms. In *Algorithms for Robotic Motion and Manipulation* (1997).

[2] Arbel, T., and Frank, F. P. Viewpoint selection by navigation through entropy maps. In *ICCV* (1999), pp. 248–254.

[3] Arkin, Ronald. Integrating behavioral, perceptual and world knowledge in reactive navigation. *Robotics and Autonomous Systems 6* (1990), 105–122.

[4] Barto, A., Singh, S., and Chentanez, N. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the International Conference on Developmental Learning (ICDL)* (LaJolla, CA, 2004).

[5] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics 41*, 1 (1970), 164–171.

[6] Bellman, R. E. *Adaptive control processes: a guided tour*. Rand Corporation Research studies. Princeton University Press, 1961.

[7] Berenson, D., Srinivasa, S., Ferguson, D., and Kuffner, J. Manipulation planning on constraint manifolds. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on* (may 2009), pp. 625 –632.

[8] Bernstein, Nicholai. *On Dexterity and Its Development*. Lawrence Erlbaum Associates, 1996.

[9] Borenstein, Johann, Everett, H. R., and Feng, Liqiang. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Natick, MA, USA, 1996.

[10] Borotschnig, H., Paletta, L., Prantl, M., and Pinz, A. Active object recognition in parametric eigenspace. In *In Proceedings of the 9th British Machine Vision Conference* (1998), pp. 629–638.

[11] Brooks, R.A. A robust layered control system for mobile robot. *IEEE Journal of Robotic Automation 2*, 1 (1986), 14–23.

[12] Brooks, R.A. Elephants don't play chess. *Robotics and Autonomous Systems 6* (1990), 3–15.

[13] Brooks, R.A. Intelligence without representation. *Artificial Intelligence Journal 47* (1991), 139–159.

[14] Burridge, R. R., Rizzi, A. A., and Koditschek, D. E. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research 18* (1999), 534–555.

[15] Cambon, S., Alami, R., and Gravot, F. A hybrid approach to intricate motion, manipulation and task planning. *Int. J. Rob. Res. 28* (January 2009), 104–126.

[16] Canny, J., and Reif, J. New lower bound techniques for robot motion planning problems. In *In Proceedings of IEEE Symposium on Foundations of Computer Science* (1987), pp. 49–60.

[17] Chapman, D. Planning for conjunctive goals. *Artificial Intelligence 32* (1987), 333–378.

[18] Choi, J., and Amir, E. Combining planning and motion planning. In *Proceedings of International Conference on Robotics and Automation* (2009).

[19] Coelho, J., and Grupen, R. Online grasp synthesis. In *Proceedings of the 1996 Conference on Robotics and Automation* (Minneapolis, MN, April 1996), IEEE.

[20] Coelho, J., and Grupen, R. A control basis for learning multifingered grasps. *Journal of Robotic Systems 14*, 7 (1997), 545–557.

[21] Connolly, C.I., Burns, J.B., and Weiss, R. Path planning using laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on* (may 1990), pp. 2102 –2106 vol.3.

[22] Darrell, T., and Pentland, A. Active gesture recognition using partially observable markov decision processes. In *Proceedings of 13th IEEE International Conference on Pattern Recognition (ICPR)* (1996).

[23] Deegan, P., Thibodeau, B., and Grupen, R. Designing a self-stabilizing robot for dynamic mobile manipulation. In *Workshop on Manipulation for Human Environments* (Philadelphia, Pennsylvania, 2006).

[24] Denzler, J., and Brown, C. An information theoretic approach to optimal sensor data selection for state estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24* (2002).

[25] Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., and Kuffner, J. Bispace planning: Concurrent multi-space exploration. In *Proceedings of Robotics: Science and Systems (RSS)* (2008).

[26] Dogar, M. R., Ugur, E., Sahin, E., and Çakmak, M. Using learned affordances for robotic behavior development. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Pasadena, CA, USA, 2008).

[27] Doucet, A., de Freitas, N., and Gordonr, N. J. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[28] Dragiev, Stanimir, Toussaint, Marc, and Gienger, Michael. Gaussian process implicit surfaces for shape estimation and fluent grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2011), IEEE.

[29] Eidenberger, Robert, and Scharinger, Josef. Active perception and scene modeling by planning with probabilistic 6d object poses. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan* (2010), IEEE, pp. 1036–1043.

[30] Ek, C.H., Song, D., Huebner, K., and Kragic, D. Task modeling in imitation learning using latent variable models. In *In Proceedings of 10th IEEE-RAS International Conference on Humanoid Robots* (2010).

[31] Erdmann, M. A. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[32] Erol, K., Nau, D. S., and subrahmanian, V. S. On the complexity of domain-independent planning. In *Proceedings of the American Association for Artificial Intelligence (AAAI) Conference* (San Jose, CA, 1992), pp. 381–386.

[33] Fikes, R. E., and Nilsson, N. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence 5*, 2 (1971), 189–208.

[34] Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. Learning about objects through action: Initial steps towards artificial cognition. In *IEEE International Conference on Robotics and Automation* (Taipei, May 2003).

[35] Fox, D., Burgard, W., and Thrun, S. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research 11* (1999), 391–427.

[36] Gat, E. Integrating planning and reacting in heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the 10th Conference on Artificial Intelligence* (1992), pp. 1–7.

[37] Geib, C., Mourao, K., Petrick, R., Pugeault, N., Steedman, M., Krueger, N., and Worgotter, F. Object action complexes as an interface for planning and robot control. In *Proceedings of the Humanoid-06 Workshop: Towards Cognitive Humanoid Robots* (Genoa, Italy, 2006).

[38] Ghallab, M., Nau, D., and Traverso, P. *Automated Planning - Theory and Practice*. Morgan Kaufmann Publications, San Francisco, CA, USA, 2004.

[39] Gibson, J. The theory of affordances. In *Perceiving, acting and knowing: toward an ecological psychology* (Hillsdale, NJ, 1977), Lawrence Erlbaum Associates Publishers, pp. 67–82.

[40] Grupen, R., and Coelho, J. A. Acquiring state from control dynamics to learn grasping policies for robot hands. *International Journal on Advanced Robotics* (2002), 427–443.

[41] Hart, S., and Grupen, R. Natural task decomposition with intrinsic potential fields. In *Proceedings of 2007 International Conference on Intelligent Robots and Systems(IROS)* (San Diego, CA, 2007).

[42] Hart, S., and Grupen, R. Learning generalizable control programs. In *Transactions on Autonomous Mental Development* (Zaragoza, Spain, 2010), pp. 1–16.

[43] Hart, S., Sen, S., and Grupen, R. Intrinsically motivated hierarchical manipulation. In *Proceedings of 2008 IEEE Conference on Robotics and Automation* (Pasadena, CA, 2008).

[44] Hart, Stephen. *The Development of Hierarchical Knowledge in Robot Systems*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2009.

[45] Hauser, K., and Latombe, J. Integrating task and prm motion planning: Dealing with many infeasible motion planning queries, 2009.

[46] Hsiao, K., Kaelbling, L. P., and Lozano-Perez, T. Grasping pomdps. In *Proceedings of 13th IEEE International Conference on Robotics and Automation* (2007).

[47] Hsiao, K., Kaelbling, L. P., and Lozano-Perez, T. Task-driven tactile exploration. In *Proceedings of Robotics: Science and Systems* (2010).

[48] Huber, M. *A Hybrid Architecture for Adaptive Robot Control*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2000.

[49] Huber, M., and Grupen, R. Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)* (Nagoya, Japan, 1997).

[50] Huber, M., and Grupen, R.A. A hybrid discrete dynamic systems approach to robot control. Tech. Rep. 96-43, Department of Computer Science, University of Massachusetts Amherst, October 1996.

[51] Huber, M., MacDonald, W., and Grupen, R. A control basis for multilegged walking. In *Proceedings of the Conference on Robotics and Automation* (Minneapolis, MN, April, 1996).

[52] Iii, J. Fisher, Fisher, J. W., Jos, I., and Principe, C. A nonparametric methodology for information theoretic feature extraction. In *in Proc. DARPA97* (1997), pp. 1077–1084.

[53] Jr., R. Platt, Fagg, A., and R.A, Grupen. Nullspace composition of control laws for grasping. In *Proceedings of the International Conference on Intelligent Robots and Systems* (2002).

[54] Kaelbling, L. P., Littmann, M. L., and Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence 101* (1998), 99–134.

[55] Kaelbling, Leslie Pack, and Lozano-Perez, Tomas. Hierarchical task and motion planning in the now. In *IEEE Conference on Robotics and Automation Workshop on Mobile Manipulation* (2010).

[56] Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Pre-image backchaining in belief space for mobile manipulation. In *International Symposium on Robotics Research (ISRR)* (2011).

[57] Kant, I. *Critique of Pure Reason, Translated by Norman Kemp Smith.* Macmillan and Company, Ltd., 1965.

[58] Kato, Hirokazu, and Billinghurst, Mark. 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* (Washington, DC, USA, 1999), IWAR '99, IEEE Computer Society, pp. 85–.

[59] Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on* (mar 1985), vol. 2, pp. 500 – 505.

[60] Khatib, O. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of 3*, 1 (february 1987), 43 –53.

[61] Kim, J., and Ostrowski, J. Motion planning of aerial robots using rapidly-exploring random trees with dynamic constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2003).

[62] Koditschek, D. E., and Rimon, E. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics 11*, 4 (1990), 412–442.

[63] Koenderink, J. J., and Doorn, A. J. The internal representation of solid shape with respect to vision. *Biological Cybernetics 32* (1979), 211–216. 10.1007/BF00337644.

[64] Kuffner, J. J., K.Nishiwaki, Kagami, S., Inaba, M., and Inoue, H. Motion planning for humanoid robots. In *Proceedings of the International Symposium on Robotics Research (ISRR)* (2003).

[65] Kuffner, J. J., and LaValle, S. M. Rrt-aonnect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2000), pp. 995–1001.

[66] Lang, Tobias, and Toussaint, Marc. Approximate inference for planning in stochastic relational worlds. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, ACM, pp. 585–592.

[67] Lang, Tobias, and Toussaint, Marc. Relevance grounding for planning in relational domains. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I* (Berlin, Heidelberg, 2009), ECML PKDD '09, Springer-Verlag, pp. 736–751.

[68] Latombe, J. C. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Mass, 1991.

[69] LaValle, S. M. Rapidly-exploting random trees: A new tool for path planning. Tech. Rep. TR 98-11, Computer Science Dept., Iowa State University, October 1998.

[70] LaValle, S. M. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[71] Littman, M. L., Sutton, R. S., and Singh, S. Predictive representations of state. In *In Advances In Neural Information Processing Systems 14* (2001), MIT Press, pp. 1555–1561.

[72] Liu, Jun S., and Chen, Rong. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association 93* (1998), 1032–1044.

[73] Lorken, C., and Hertzberg, J. Grounding planning operators by affordances. In *Proceedings of the 2008 International Conference on Cognitive Systems* (Karlsruhe, Germany, 2008).

[74] Lozano-Perez, T., Jones, J., and Mazer, E. Handey: a robot system that recognizes, plans and manipulates. In *Proceedings of International Conference on Robotics and Automation* (1987).

[75] Lozano-Pérez, T., Mason, M. T., and Taylor, R. H. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research 3*, 1 (1984), 3–24.

[76] Mccarthy, John, and Hayes, Patrick J. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence* (1969), Edinburgh University Press, pp. 463–502.

[77] Miyazawa, K., Maeda, Y., and Arai, T. Planning of graspless manipulation based on rapidly-exploring random trees. In *Proceedings of 6th IEEE International Symposium on Assembly and Task Planning* (Montreal, Canada, 2005), pp. ITP–3.

[78] Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. Modeling affordances using bayesian networks. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems* (San Diego, CA, 2007).

[79] Nakamura, Y. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[80] Nakamura, Yoshiihiko, Hanafusa, Hideo, and Yoshikawa, Tsuneo. Task-priority based redundancy control of robot manipulators. *Int. J. Rob. Res. 6*, 2 (1987), 3–15.

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

[82] Nilsson, N. Shakey the robot. *SRI International, Technical Report*, 323 (1984).

[83] Noonan, C., and Oxford, K. Entropy measures of multi-sensor fusion performance. In *In Proceedings of the IEE Colloquium on Target Tracking and Data Fusion* (1996), pp. 15/1–15/5.

[84] Nourbakhsh, I. Using abstraction to interleave planning and execution. In *Proceedings of the Third Biannual World Automation Congress* (1998).

[85] Nourbakhsh, Illah, Powers, Rob, and Birchfield., Stan. Dervish: An office-navigating robot. *AI Magazine 16*, 2 (1995).

[86] Oudejans, R., Michaels, C., van Dort, B., and Frissen, E. To cross or not to cross: The effect of locomotion on street-crossing behavior. *Ecological Psychology 8*, 3 (1996), 259–267.

[87] Petrovskaya, Anna, Thrun, Sebastian, Koller, Daphne, and Khatib, Oussama. Towards dependable perception: Guaranteed inference for global localization. Dependable Robots in Human Environments Workshop at Robotics: Science and Systems, June 2010.

[88] Piaget, J. *The Origins of Intelligence in Childhood*. International University Press, 1952.

[89] Pineau, J., Gordon, G., and Thrun, S. Point-based value iteration: An anytime algorithm for pomdps, 2003.

[90] Pineau, J., and Thrun, S. High-level robot behavior control using pomdps. In *In AAAI Workshop notes* (2002).

[91] Plaku, E., and Hager, G. D. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *ICRA* (2010), pp. 5002–5008.

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

[93] Roy, N., and Thrun, S. Coastal navigation with mobile robots. In *Proceedings of Neural Information Processing Systems (NIPS)* (2000).

[94] Sacerdoti, E. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence 5* (1974), 115–135.

[95] Sacerdoti, E. The non-linear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI)* (Stanford, CA, 1975), pp. 206–214.

[96] Saffiotti, A., Konolige, K., and Ruspini, E. H. A multivalued logic approach to integrating planning and control. *Artificial Intelligence 76* (1995), 481–526.

[97] Sahin, E., Çakmak, M., Dogar, M. R., Ugur, E., and Uçoluk, G. To afford or not to afford: A new formalization of affordances towards affordance-based robot control. *Adaptive Behavior 15*, 4 (2007), 447–472.

[98] Schiele, B., and Crowley, J. L. Transinformation for active object recognition. In *In Proceedings of the Sixth International Conference on Computer Vision* (1998), pp. 249–254.

[99] Simmons, Reid, and Koenig, Sven. Probabilistic navigation in partially observable environments. In *In: Proceedings of the fourteenth international joint conference on artificial intelligence* (1995), pp. 1080–1087.

[100] Singh, S., Littman, M. L., Jong, N. K., Pardoe, D., and Stone, P. Learning predictive state representations. In *Proceedings of the Twentieth International Conference on Machine Learning* (August 2003).

[101] Smallwood, Richard D., and Sondik, Edward J. The optimal control of partially observable markov decision processes over a finite horizon. In *Operations Research* (1973), vol. 21, pp. 1071–1088.

[102] Smith, T., and Simmons, R. Heuristic search value iteration for pomdps, 2004.

[103] Sommerlade, Eric, and Reid, Ian. Information theoretic active scene exploration. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)* (May 2008).

[104] Spaan, M. T. J., and Vlassis, N. Perseus: Randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research 24* (2005), 195–220.

[105] Stoytchev, A. Behavior-grounded representation of tool affordances. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Barcelona, Spain, 2005).

[106] Stoytchev, A. Toward learning the binding affordances of objects: A behavior-grounded approach. In *Proceedings of the AAAI Spring Symposium on Developmental Robotics* (Stanford University, 2005).

[107] Sutton, R., and Barto, A. *Reinforcement Learning*. MIT Press, Cambridge, Massachusetts, 1998.

[108] Tedrake, R. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems* (Seattle, USA, June 2009).

[109] Thrun, S., Fox, D., Burgard, W., and F., Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence 128*, 1-2 (2001).

[110] Thrun, Sebastian. Particle filters in robotics. In *in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI* (2002).

[111] Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[112] Toussaint, Marc, and Goerick, Christian. Probabilistic inference for structured planning in robotics. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA* (2007), IEEE, pp. 3068–3073.

[113] Toussaint, Marc, Harmeling, Stefan, and Storkey, Amos. Probabilistic inference for solving (po)mdp's, Dec 2006.

[114] Toussaint, Marc, Plath, Nils, Lang, Tobias, and Jetchev, Nikolay. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *IEEE International Conference on Robotics and Automation (ICRA)* (2010).

[115] Viola, P. A. Alignment by maximization of mutual information. In *International Journal of Computer Vision* (1995), pp. 16–23.

[116] Warren, W. H. Perceiving affordances: Visual guidance of stair climbing. *Journal of Experimental Psychology 105*, 5 (1984), 683–703.

[117] Wolfe, J., Marthi, B., and Russell, S. Combined task and motion planning for mobile manipulation. In *ICAPS* (2010).

[118] Wooldridge, M., and Jennings, N. R. Intelligent agents: Theory and practice. *Knowledge Engineering Review 10*, 2 (1995), 115–152.

[119] Ye, Y., and Tsotsos, J. K. Sensor planning for 3d object search, 1996.

[120] Zucker, M., Kuffner, J., and Branicky, M. Multipartite RRTs for rapid replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2007).