THE DEVELOPMENT OF HIERARCHICAL KNOWLEDGE IN ROBOT SYSTEMS

A Dissertation Presented

by

STEPHEN W. HART

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

DOCTOR OF PHILOSOPHY

September 2009

Computer Science

THE DEVELOPMENT OF HIERARCHICAL KNOWLEDGE IN ROBOT SYSTEMS

A Dissertation Presented

by

STEPHEN W. HART

Approved as to style and content by:

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help and support of many people. Most of all, I would like to extend my gratitude to Rod Grupen for many years of inspiring work, our discussions, and his guidance. Without his support and vision, I cannot imagine that the journey would have been as enormously enjoyable and rewarding as it turned out to be. I am very excited about what we discovered during my time at UMass, but there is much more to be done. I look forward to what comes next! In addition to providing professional inspiration, Rod was a great person to work with and for—creating a warm and encouraging laboratory atmosphere, motivating us to stay in shape for his annual half-marathons, and ensuring a sufficient amount of cake at the weekly lab meetings. Thanks for all your support, Rod!

I am very grateful to my thesis committee—Andy Barto, David Jensen, and Rachel Keen—for many encouraging and inspirational discussions. Their comments and feedback significantly contributed to the form of this document. I would especially like to thank Andy for organizing the Intrinsic Motivation seminar I attended a few years ago. This seminar helped focus the direction of my research and led me to pursue many of the topics explored in this dissertation.

I would like to thank the many friends and colleagues who helped make my graduate years as enjoyable as they were. Thanks especially to Aron Culotta, Sarah Osentoski, Ashvin Shah, Maryanne Olson, Trek Palmer, Ricky Chang, Heather Perkins, and Shiraj Sen for their friendship and kindness, and for their company during those many, many nights at the Moan & Dove. I would also like to extend my gratitude to the members of the Laboratory for Perceptual Robotics and the Autonomous Learning Laboratory for our discussions and collaborations—specifically, Emily Horrell, Rob Platt, Shichao Ou, Dirk Ruiken, George Konidaris, and John Sweeney. Priscilla Scott and Leeanne Leclerc deserve special acclimations for making sure all of those nasty administrative details got taken care of and for being patient with all of us flaky academics.

My sincere appreciation goes out to my friends at WMUA, especially the members of the weekday morning Jazz Block—Glenn Siegel, Ron Freshley, Ken Irwin, and Michael Ehlers—and to Hank Berry. My years as the Thursday morning DJ of "Soul Station," provided me with a welcome and necessary outlet that helped balance my work and hobbies. The music I discovered during this period provided an additional level of creative inspiration that has, no doubt, profoundly shaped my being at its core. To all those who stood up for their convictions, ran against the grain, and followed their own paths—John Coltrane, Albert Ayler, Bob Dylan, Cecil Taylor, Peter Brötzmann, Robert Wyatt, and many others—the passion, creativity, and love that you brought (and bring) to your work is an inspiration to us all.

Most of all I would like to thank my fianceé, Meg, for her love, her kindness, and her patience—especially during these last few stressful months. As I anticipate the journey we are just beginning, I am filled with excitement more than words can express. I would also like to thank my parents, Bill and Jean, and my sister and brother-in-law, Catherine and Adam, for their support and their love as I chose to stay in school for one degree after another. Finally, special gratitude should be extended to Maximus Felinus and Doris Day for their uncanny tolerance.

ABSTRACT

THE DEVELOPMENT OF HIERARCHICAL KNOWLEDGE IN ROBOT SYSTEMS

SEPTEMBER 2009

STEPHEN W. HART

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic Grupen

This dissertation investigates two complementary ideas in the literature on machine learning and robotics—those of *embodiment* and *intrinsic motivation*—to address a unified framework for skill learning and knowledge acquisition. "Embodied" systems make use of structure derived directly from sensory and motor configurations for learning behavior. Intrinsically motivated systems learn by searching for native, hedonic value through interaction with the world. Psychological theories of intrinsic motivation suggest that there exist internal drives favoring open-ended cognitive development and exploration. I argue that intrinsically motivated, embodied systems can learn generalizable skills, acquire control knowledge, and form an epistemological understanding of the world in terms of behavioral *affordances*.

I propose that the development of behavior results from the assembly of an agent's sensory and motor resources into state and action spaces that can be explored autonomously. I introduce an intrinsic reward function that can lead to the open-ended learning of hierarchical behavior. This behavior is factored into declarative “recipes” for patterned activity and common sense procedural strategies for implementing them in a variety of run-time contexts. These skills form a categorical basis for the robot to interpret and model its world in terms of the behavior it affords. Experiments conducted on a bimanual robot illustrate a progression of cumulative manipulation behavior addressing manual and visual skills. Such accumulation of skill over the long-term by a single robot is a novel contribution that has yet to be demonstrated in the literature.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation

Computational approaches for accumulating long-term control knowledge have eluded psychologists and roboticists since the beginning of artificial intelligence research. I argue that in order for an organism to learn over a lifetime, it must figure out how to take actions that affect the world in measurable ways, be motivated to employ such actions in a goal-directed manner, and understand the consequences of these actions when they are performed. To accomplish these objectives in a computational system, I suggest that (1) an agent must have knowledge representations that effectively capture controllable interactions with its environment, (2) an agent must have a means of organizing and adapting knowledge structures in flexible ways to describe complicated tasks in comprehensive ways, and (3) an agent must be endowed with an *intrinsic* means of motivating behavior and knowledge acquisition that is not particular to any one task, but leads to categorical and epistemological models of how the agent can interact with its environment.

The ability to apply accumulated knowledge and flexible decision making processes in unpredictable environments is what the Russian biomechanist Nikolai Bernstein defined as physical and cognitive "dexterity" (Bernstein, 1996). Dexterous manipulation, therefore, is a rich domain in which to study behavioral and epistemological organization in manual systems. Manipulation skills in humans develop during the sensorimotor play stage occurring in the first 18-24 months of infancy, ending when infants begin to create multi-object structures such as *stacks* and followed by the first "conceptual" stage of development (Piaget, 1952). For many decades, psychologists have looked at this early stage of human (and animal) development and attempted to characterize play behavior (c.f., (Power, 2005)). Although many studies carefully tabulate the observed play behavior common to most primates, they provide few hypotheses of what exploratory and organizational mechanisms underly this stage. More focus is given to the conceptual period that is more naturally expressed using computational reasoning and planning techniques.

The sensorimotor stage is worth considering in detail because it creates foundational knowledge for subsequent development and because this is when most manual skills are acquired in rudimentary forms. In these few months, infants gain increasingly comprehensive knowledge about objects and interactions to expand their growing capabilities. They become bored with objects they have seen, seek out new ones, and explore how these objects might relate to each other.

Computational models that exhibit similar behavior and learn over the long-term have not been demonstrated. Such computational models would solve many open questions in artificial intelligence, such as: (1) how an agent can learn efficiently from real-world experience, (2) how knowledge and behavior should be organized, and (3) what motivational mechanisms can allow an agent to learn and gain expertise. It is worth the challenge. Understanding mechanisms for learning and accumulating knowledge is critical, not only for understanding how our minds work, but also for understanding how artificial agents can have working minds as well.

1.2 Approach

Much of the literature in robotics has been devoted to examining how mobile agents can navigate in low-dimensional spaces while avoiding collisions with obstacles. However, many interesting forms of behavior involve direct physical interaction with the world. The range of behavior exhibited by the human hand is immense because it facilitates the acquisition of implicit knowledge regarding the use of utensils, handles, containers, keyboards, and multi-object interactions like stacking, insertion, and other assembly relationships of various complexity between bodies.

I propose a novel approach to programming dexterous robotic systems by guiding the development of critical control knowledge. I advance a framework for self-motivated discovery and the organization of behavior derived explicitly in terms of a robot's sensory and motor resources. An action in this framework is a decision to activate a set of multi-objective closed-loop control circuits. The state-space for decision making arises from the dynamic status of these circuits. I introduce an intrinsic motivation function that provides reward to states and actions in a way that is grounded explicitly in the robot's ability to discover what behavior its environment *affords.*

This dissertation takes a *dynamical systems* approach to the organization of behavior in contrast to the sampling-based or planning techniques common in the literature that place unrealistic assumptions on the availability of complete environmental knowledge. Dynamical systems approaches provide a biological and psychological theory of behavior and development (e.g., Mussa-Ivaldi and Bizzi (2000), Waddington (1943), Thelen and Smith (1994)) and employs analytical tools from physics, engineering and artificial intelligence (e.g., optimal control theory and reinforcement learning techniques (Sutton & Barto, 1998)). This dissertation examines a characterization of certain dynamical systems—feedback control systems that follow potential fields—to provide a natural substrate for adaptive-optimal behavior in robot systems.

Potential fields assign value to states that a learning agent can explore. Although potential field approaches are common in robotics and machine learning, they are usually used to address problems on a task-by-taks basis. The result is that no commonly accepted description of how to uniformly represent multiple tasks in the same potential field framework has yet been proposed. However, there has been much recent attention in the robotics and machine learning communities on how to represent the *intrinsic* value for taking certain actions in certain situations (Singh et al., 2004a; Lungarella et al., 2003; Oudeyer et al., 2007). This work has been inspired by earlier work in the psychology literature on internal drives by Hull (1943), and intrinsic motivation by Berlyne (1960), White (1959), and others. In this dissertation, I will examine an intrinsic reward framework that can be applied to many different real-world tasks.

1.3 Contributions

This thesis makes a number of contributions to the literature. It provides (1) a representation for integrated control programs—or *skills*—in a manner that facilitates the representation of "knowledge" that can easily be retrieved, re-used, and adapted, (2) a framework for intrinsic motivation with which a robot can learn behavior *and* discover when that behavior is afforded by the environment, and (3) a means of categorizing the world in terms of behavioral affordances that can drive long-term exploration. The research presented in this document extends the *control basis* framework for assembling behavioral programs from a combinatoric set of sensory and motor resources (Huber, 2000; Coelho, 2001; Platt, 2006). It builds upon the existing theory by providing mechanisms for generalizing programs to new contexts, and by introducing a natural intrinsic reward function for learning control basis programs that rewards a robot for uncovering behavioral affordances.

The theoretical contributions of this thesis are demonstrated on a sequence of robot manipulation tasks: visual and tactile search and tracking, reaching out and touching (sometimes grabbing) objects, visual inspection, object segmentation (in terms of affordances), and simple multi-object tasks such as "pick-and-place" and stacking.

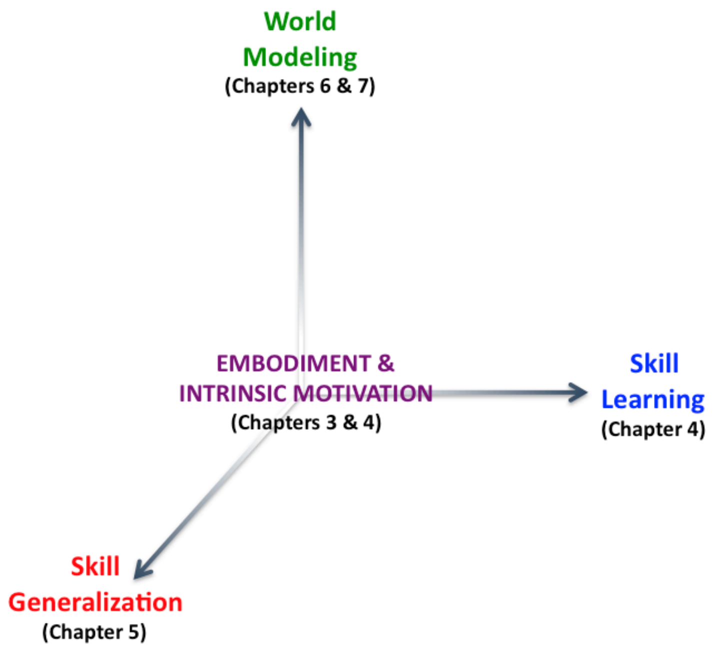


Figure 1.1. Document Overview.

1.4 Document Overview

In this document, I propose a framework for robot development in which techniques for *skill learning*, *skill generalization*, and *world modeling* arise from principles of embodiment and intrinsic motivation. I propose a single intrinsic motivator that biases a robot to assemble sensory and motor circuits that couple with the environment in measurable ways. This motivator allows a robot to learn behavioral skills in simple learning situations, generalize these skills to new contexts, and then use these skills to model the world in terms of collections of control affordances. The proposed framework is illustrated in Figure 1.1. Although the dimensions are discussed in different chapters to facilitate a comprehensive evaluation, it is important to keep in mind that these processes can (*and should be*) run concurrently in an integrated learning system.

The document is organized as follows. Chapter 2 provides background on dynamical systems theories in the psychology and engineering literature that provide the underpinnings of the approach taken in the rest of the document. A brief overview of psychological and computational theories of intrinsic motivation, embodiment, and developmental robotics is also included. Chapters 3-7 constitute the main technical chapters of this dissertation.

- Chapter 3 provides an introduction to the *control basis* framework and how it can be used to create action spaces from combinations of sensory and motor systems. The control basis provides structure from embodiment that make the application of stochastic machine learning algorithms tractable in complex robot systems.

- Chapter 4 addresses how natural state and reward descriptors arise from the dynamics of control basis actions. An intrinsic motivation function is introduced that is used to learn control basis programs that uncover behavioral affordances. The chapter presents a number of hierarchical programs that are learned using this intrinsic motivator.

- Chapter 5 shows how to generalize control basis programs into dexterous strategies (called *schema*) that can be applied in many situations. Generalization is accomplished by factoring programs into *declarative* and *procedural* components. The declarative structure captures the abstract intentions of a program. The procedural structure captures how to parameterize these intentions based on run-time context.

- Chapter 6 examines how a robot can use its skill set to model its world in terms of the behavior it affords. Collections of environmental affordances are assembled into memory structures called "catalogs." This chapter demonstrates how a robot can explore a number of catalogs for some simple objects it encounters.

- Chapter 7 shows how affordance catalogs can be extended to model multi-object assemblies when the robot acquires the ability to perform a "pick-and-place" task. This chapter ends with a demonstration of how a robot can build hierarchical "meta-catalogs" that form the basis for long-term behavioral exploration and world modeling.

Finally, Chapter 8 provides a summary of the work presented and discusses areas of future investigation.

CHAPTER 2
BACKGROUND

In this dissertation, we are interested in robot behavior and knowledge that evolves and changes over the course of long-term development. The proposed framework uses a unique dynamical systems approach based on control theoretic methods. Sensory and motor resources are attached to control objectives (or "intentions") at run-time to respond dexterously to interactions with the world. In this chapter, related work on dynamical and developmental systems is presented. We begin by examining dynamical systems in physics and engineering, and follow with a discussion of dynamical systems theories for biological development and behavior. We conclude with an examination of three areas of developmental robotics: embodiment, intrinsic motivation, and affordance modeling. Individual chapters in the remainder of this document will provide more specific summaries of related work as they are relevant.

2.1 Dynamical Systems

In the nineteenth–centrury, Henri Poincaré examined *dynamical systems* in which a set of system (or state) variables change over time in accordance with a set of governing rules. Poincaré studied gravitational systems, but derived a general theory for how *potential fields* define interaction forces between physical bodies without direct mechanical interaction. At each point in the state space, the system responds to forces equal to the negative gradient of the potential field until it arrives at one of many possible stable *attractor* states where the gradient of the potential becomes zero if the flow-field is unobstructed.

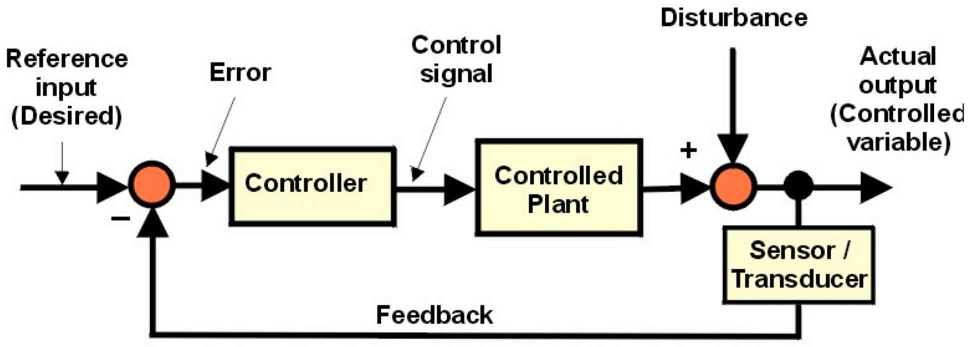


Figure 2.1. A special purpose dynamical system for closed-loop feedback control.

In addition to describing forces in natural systems, dynamical systems can be used to describe the ontogenetic, epigenetic, and epistemological processes of development in biological systems. Ontogenetic development addresses how an individual develops as it grows and interacts with its environment. Epigenetic development examines how an individual's gene expression is effected by mechanisms other than changes in the underlying DNA. Epistemological development addresses how an individual's cognitive knowledge structures grow and adapt over time with experience. These theories provide insight into how developing organisms change over time. This dissertation provides a framework for grounding development in the existing mathematical dynamical system formalism.

2.1.1 Feedback Control

Feedback control systems, like that illustrated in Figure 2.1, are dynamical systems whose performance depends on properties of the environment as well as inputs specified by a control function. In closed-loop control systems, feedback from sensors is compared to a reference value in order to compute an error that is regulated by control inputs applied to the plant. Feedback control loops can be used to track time-varying reference signals called *forcing functions* that shape the system's movements. Forcing functions can originate from computational processes like a trajectory generator, or from the dynamics of signals originating from external environmental stimuli. The time history of the overall response of a feedback loop reflects the pattern of disturbances that are applied to the controlled system.

Tools from control theory describe how to analytically design control functions for linear and non-linear systems in order to guarantee certain performance criteria. The Russian physicist Aleksandr Lyapunov studied one such criterion related to the asymptotic behavior of dynamical systems near attractors. A dynamical system is said to be asymptotically stable if, as $t \to \infty$, the system tends toward an attractor state and stays there when it gets there. Mathematically, we can define Lyapunov stability as follows:

Definition 2.1: If a function $V(\boldsymbol{q}, t)$, with state $\boldsymbol{q}$ and time t, exists where

$$\begin{aligned} V(\mathbf{0}, t) &= 0, \text{ and} \\ V(\boldsymbol{q}, t) &> 0, \text{ for } \boldsymbol{q} \neq \mathbf{0} \text{ (positive definite), and} \\ dV/dt &< 0, \text{ (negative definite),} \end{aligned}$$

then that function is a *Lyapunov function* and the state space $\boldsymbol{q}$ is asymptotically stable in the neighborhood of the origin.

2.1.2 Artificial Potential Fields

Potential fields define the forces acting on a system. For example, the gravitational potential near the surface of the earth is $\phi = mgh$, where g is the gravitational constant, h is the height of a body, and m is the mass of the object. Gravity creates a constant force field that acts upon an object equal to $-\nabla\phi = -mg\hat{\mathbf{z}}$. Similarly, Hooke described a linear force field that governs the behavior of springs such that $-\nabla\phi = -kx$, where k is a constant and x is the amount the spring is extended from its equilibrium position. Hooke's law defines the potential energy stored in the spring as $\phi = \frac{1}{2}kx^2$.

The above formulae can be applied to artificial systems, such as a robot manipulator, to shape how that manipulator moves to goal positions and how it responds to environmental disturbances. Simple formulations of artificial potential fields can often result in local minima where $-\nabla\phi = 0$ and the gradient vanishes short of true global minima (Latombe, 1991). However, it is possible to apply constraints to the shape of potential functions in order to eliminate local minima. In particular, avoiding *critical* points where the gradient of the field becomes zero. Critical points occur in three situations: type (0) minima, type (1) saddle points, and type (2) maxima. Artificial potential fields in robotics were first developed in the context of path planning tasks by Krogh (1984) and Khatib (1986) to move the hypothetical "particle" (i.e., the robot) toward goal configurations. Examples of artificial potential functions appropriate for robot control are discussed in the next chapter.

2.1.3 Potential Fields for Discrete State/Action Spaces

Potential functions in discrete state and action spaces are called *value functions.* Value functions can be estimated using reinforcement learning (RL) techniques for a class of systems called Markov Decision Processes (MDPs) (Sutton & Barto, 1998). An MDP is a tuple:

$$M \;=\; <\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}> \tag{2.1}$$

consisting of states $\mathcal{S}$, actions $\mathcal{A}$, transition dynamics $\mathcal{P}$, and a reward function $\mathcal{R}$. In a Markov system, optimal control decisions that maximize value—or *reward*—can be made at any time knowing only the current state $s \in \mathcal{S}$. The information in state s is sufficient; it is not necessary to know how the system arrived at s. Although this assumption is rarely true in practice, MDPs have been successfully applied to many domains.

Reinforcement learning techniques estimate value functions that satisfy the Bellman optimality equation. This criterion states that the action that produces the maximum value at every state can be estimated as the sum of possible (discounted) values at successor states and the reward gathered along the way. This sum is weighted by the likelihood of arriving at each of the next possible states. Mathematically, the optimal value at each state can be defined as:

$$V^*(s) \quad = \quad \max_a \sum_{s'} P^a_{ss'} \big[R^a_{ss'} + \gamma V^*(s')\big], \tag{2.2}$$

where $P^a_{ss'}$ is the probability of arriving in state s' after taking action a in state s. $R^a_{ss'}$ 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. Value functions that satisfy Bellman's equation (Bellman, 1961) are useful because they allow an agent to maximize reward by following a greedy policy π at every decision point that maps states to actions, such that:

$$\pi(s) \quad = \quad \text{argmax}_a \sum_{s'} P^a_{ss'} \big[R^a_{ss'} + \gamma V^*(s')\big]. \tag{2.3}$$

It is also possible to estimate state/action value functions, $\Phi(s, a)$, using techniques such as Q-learning (Watkins & Dayan, 1992)[1]. Q-learning estimates the value function through trial-and-error experience using the update-rule:

$$\Phi(s, a) \quad \leftarrow \quad \Phi(s, a) + \alpha\big(r + \gamma \ \max_{a'} \Phi(s', a') - \Phi(s, a)\big) \tag{2.4}$$

[1]The $\Phi(s, a)$ notation is used instead of the customary $Q(s, a)$ to emphasize the relation of discrete value functions to the continuous potential fields ϕ.

where $\gamma \in [0, 1]$ is the discount rate, r is the reward received, and $\alpha > 0$ is a step-size. With sufficient experience, this estimate is guaranteed to converge to the optimal value Φ^*. In this case, the optimal policy π^* maps states to actions by maximizing the expected sum of discounted future reward, such that

$$\pi^*(s) \quad = \quad \text{argmax}_a \Phi^*(s, a). \tag{2.5}$$

To balance exploration and exploitation, many techniques for selecting actions from Φ are available. One such approach suggests choosing actions according to an ϵ-greedy policy in which the action with the highest value for the current state is chosen with probability $(1 - \epsilon)$, and a random (exploratory) action is chosen the remainder of the time.

2.2 Dynamical Systems Theories of Development

Dynamical system models have also been suggested for development in biological systems. These models provide useful insight into how a series of interacting "forces" might result in epigenetic, ontogenetic, and epistemological development for organisms operating over long time-scales.

2.2.1 Epigenetic Development

In the 1940's, C.H. Waddington proposed an "epigenetic landscape" for the genetic development of biological organisms in the face of external influence, such as mutations. His landscape metaphor, depicted in Figure 2.2, draws clear parallels to dynamical systems theory (Waddington, 1943). As development occurs, the organism takes different paths (like the marble) through the landscape, as goverened by biological "rules," to develop new genetic products at the attractor states (the wells at the bottom of the figure). According to Slack's characterisation of the landscape, "all the cells in the embryo would evolve according to the same [dynamical] laws, but because

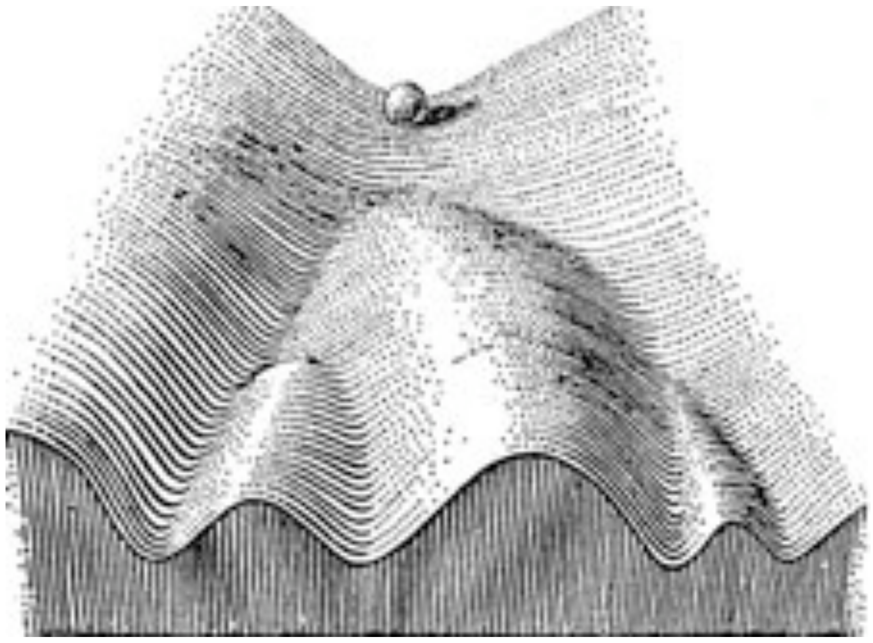


Figure 2.2. Waddington's "Epigenetic Landscape" (Waddington, 1943).

of the existence of inducing signals, cells in different regions would follow different pathways...and end up at different attractors, which can be elegantly associated with different states of terminal differentation" (Slack, 2002).

As an example, Waddington described how the bristle on the distant end of the antenna of the Drosophila (a type of fruit fly) could change roles to become a leg segment when subjected to a mutant version of an allele of the aristapedia gene. Such hard turns in the course of development led Waddington to hypothesize that the genetic makeup of the organism controls discrete stages with multiple bifurcations, or pathways, realized through the formation of the various "folds" in the landscape that make different end-states possible. Although, mainly meant as a metaphor for epigenetic development, Waddington's landscape supplies a fundamental—albeit qualitative—view of development at the biological level.

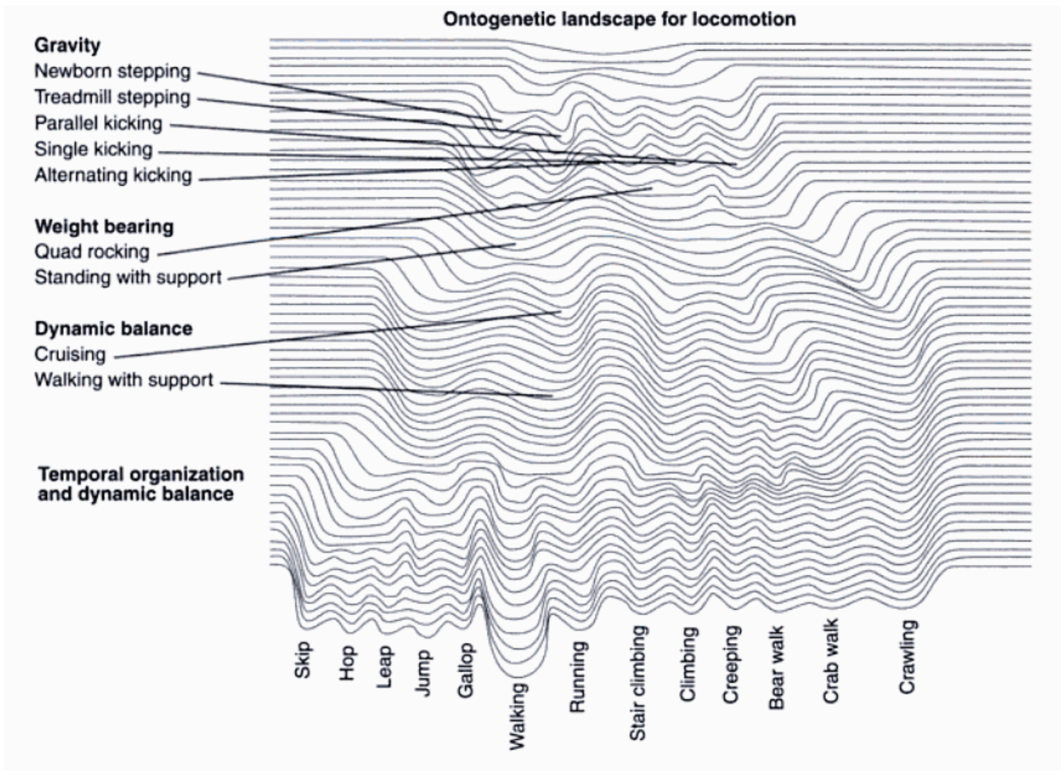


Figure 2.3. Thelen's "Ontogenetic Landscape" for infant locomotion (Muchisky et al., 1996).

2.2.2 Dynamic Field Theory

The developmental psychologist Esther Thelen provided a dynamic systems theory of congition and action (Thelen & Smith, 1994). She argued that development, both at the ontogenetic and epistemological levels, can be viewed as the assembly of many interacting subsystems. In a dynamical sense, Thelen proposed that paths to different attractor regions—representing stable behavioral or cognitive states—are traversed in the context of physiological or environmental influences. Furthermore, these paths, as well as the "deepness" of the attractor regions are reinforced from experience, biasing the organism in similar situations.

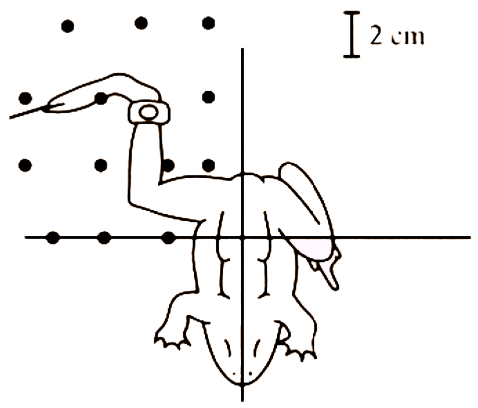
Thelen applied Waddington's epigenetic landscape metaphor to ontogenesis, a specific example of which is shown in Figure 2.3 for infant locomotion. She argued that the dynamic influences of many interacting components, such as the onset/offset of motor reflexes, physiological and maturational changes such as the strength-to-weight ratio, environmental influences like gravity, or even reinforcement from repetition, all influence the development of behavor such as locomotion. In the diagram, time proceedes downward as various behaviors (represented as valleys in the landscape) emerge in the precense of influences displayed on the left hand side.

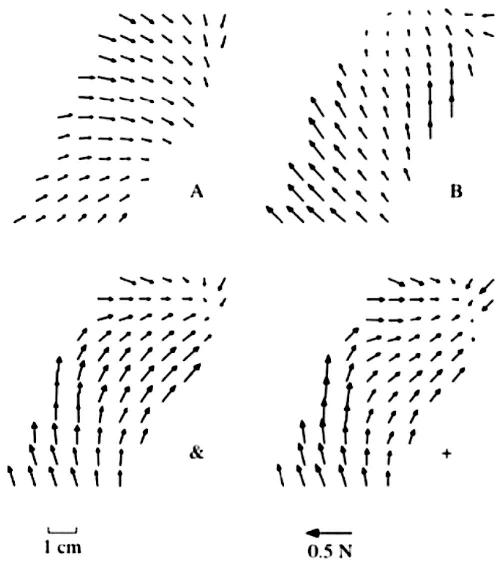
One early example demonstrating Thelen's theory examines the stepping reflex of newborn infants that seemingly dissappears after the first 3 months—reappearing later shortly before the infant begins walking—even as kicking increases. Thelen found that chubbier babies tended to lose the stepping reflex earlier than more slender babies, as well as that, under certain situations such as when the infant is held upright with its lower torso submerged in water, the stepping reflex can be triggered even after it had seemingly dissappeared. From this, Thelen hypothesized that behavior emerges from the interaction of many interacting components. For stepping, the posture of the infant, the fat-to-muscle ratio of the infant's legs (which increases around 3 months of age), and the pull from the environment all influence the resulting behavior (Spencer et al., 2006).

Thelen attempted to formalize her theory mathematically (c.f., (Schöner & Thelen, 2006)), but maintained that "the specific form of the model is...less important than the general principles...on which it is based" (Thelen & Bates, 2003). Nevertheless, her theories provide a useful metaphor for how development can be approached from a dynamical systems perspective.

2.2.3 Composable Motion Primitives

There also exists growing interest in the neuroscience literature positing potential field approaches to the dynamical movements of animals. Mussa-Ivaldi, Gizster, and Bizzi proposed a new perspective on how the central nervous system (CNS) rep-





(a) (b)

Figure 2.4. (a) the hindlimb of the frog was placed in a number of locations (the dots) and a stimulus was applied to its spinal cord. Resulting forces were measured to estimate the force fields governing the limb's movements. (b) shows two spinal activation fields, A and B, observed in frogs when two areas are artificially stimulated. The vector sum of the field (+) is highly correlated to the observed behavior when the two areas are co-stimulated (&). These figures are adapted from (Mussa-Ivaldi & Bizzi, 2000)

resents and solves some of the most fundamental computational problems of motor control (Gizster et al., 1993; Mussa-Ivaldi & Bizzi, 2000). The authors provide a hypothesis in which desired motions are assembled from a "grammar" of inherent force fields that can push an organism's limbs in various directions with respect to a reference attractor point. They provided an analysis showing that motor primitive modules reside in the spinal cord of frogs and rats. These modules are adapted to meet the demands of both the environment and the changing mechanical properties of the animal's limbs as they grow, thus implicitly solving the inverse dynamics problem.

Spinal force fields were measured by placing the hindlimb of a frog in various planar locations, as seen by the dots in Figure 2.4(a), while the spinal cord was stimulated. The resulting Cartesian force response of the frog was measured using a transducer. The authors demonstrate how observed motions are consistent with linear, weighted sums of low-level modules (or potential fields) combined to achieve the task behavior. Figure 2.4(b), from (Mussa-Ivaldi et al., 1994), shows how two observed spinal force fields (A) and (B) when added together (+) correlate approximately 87% to the observable patterns when the two activation areas are co-stimulated (&).

This series of experiments are compelling and suggest that motor behavior in frogs is assembled from a discrete set of adapting primitives that reside in the animal's nervous system. Such an approach avoids the necessity of building complete or accurate internal dynamic models, but rather allows the organism to assemble complex, co-articulated movements using only local information. Work by Mussa-Ivaldi, Bizzi, and Gandolfo have demonstrated that such an approach could be applied to the planning and execution of a visuomotor tasks (Mussa-Ivaldi & Bizzi, 2000; Gandolfo et al., 1996), and thus provides biological evidence for the framework presented in this document.

2.3 Developmental Robotics

Computational methods for developmental learning in robotic systems was proposed by a number of researchers including Sandini et al. (1997), Grupen et al. (Piater & Grupen, 2000; Coelho et al., 2000; Huber, 2000), Berthouze et al. (1996), and Asada et al. (2001), and have recently enjoyed a great deal of attention in the literature. Researchers are discussing (1) how to leverage theories of development to build better autonomous, adaptable, and capable robots (Asada et al., 2001; Metta, 2000) and (2) how to apply computational mechanisms to gain a deeper understanding of the processes of development that arise in natural systems (Braitenberg, 1984;

Reeke et al., 1990; Pfeifer, 2002; Sporns, 2003). Developmental robotics includes methods for a robot to learn about itself (e.g., learning about the visual appearance of the robot's own limbs (Natale, 2004)), as well as about what things the robot can actively control (i.e., its hand and fingers) (Kemp & Edsinger, 2006; Edsinger & Kemp, 2006). Other methodologies show the advantages of providing time-varying developmental constraints to guide algorithms that learn by exploring (Gomez, 2004; Lee & Meng, 2005). Constraints are relaxed as the robot gains more competency in stages. Staged learning also provides a means for grounding knowledge in an agent that learns increasingly complex skills (Asadi et al., 2006; Kaplan & Hafner, 2005; Cohen et al., 2007). Edsinger and Kemp showed how a humanoid may develop a sense about its appendages in a proximal-to-distal fashion—learning about its hands in one stage, and its fingers in the next—by using information theoretic techniques for discovering discriminating perceptual categories (Kemp & Edsinger, 2006; Edsinger & Kemp, 2006).

The field of developmental robotics has expanded to examine the processes of motor control, cognitive development, longitudinal learning, and the role of embodiment in intelligence (Lungarella et al., 2003). In part, this dissertation is examining the utility of this more integrated computational view. In particular, I examine how theories of sensorimotor play behavior and intrinsically motivated systems can naturally be represented using computational tools. Three specific areas of developmental robotics relevant to the proposed framework are discussed next.

2.3.1 The Role of Embodiment

Developmental robotics focuses on the integration of machine learning, psychological theories of development, and computational models of behavior in *situated* systems. This perspective adds to the debate on long-standing issues of artificial intelligence. One prominent issue concerns the role of embodiment in the acquisition of behavioral and cognitive competence. For many years, knowledge in computational systems was treated as a process independent of a body.

Knowledge representations in artificial systems have been studied since the earliest days of artificial intelligence for rule-based systems (Newell & Simon, 1961; Minsky, 1974; Davis et al., 1993). In the 1980's, a great deal of research focused on the formal representation of common sense knowledge in symbolic AI systems (Hayes, 1978; Hobbs & Moore, 1985; Davis, 1990). More recently, ambitions large-scale projects such as CYC (Lenat, 1995) and OpenMind (Singh, 2001) have been undertaken to collect such knowledge through textual analysis, but have yet to prove any deep, common sense understanding of consequences or an ability to scale to real-world competence in any behavioral task.

Even traditional robotics techniques relied on having a completely deterministic world—such as a factory setting—in which non-reactive and symbolic planning techniques could be applied with reasonable success. For example, natural language descriptors are often used to describe objects and object relationships in what are called "Blocks Worlds" (Winograd, 1971).

In contrast to such systems, I believe that complex behavior is really the result of situated learning systems that use sensory and motor apparati to explore the unstructured environment around them. How important is the character of embodiment to the expected outcome of development? Margaret Wilson proposed six "views" in order to argue that embodiment plays a central role in the cognitive processes of any learning agent (Wilson, 2002). Wilson argued that (1) *cognition is situated*, inherently involving the interaction between sensory and motor systems, (2) *cognition is time-pressured* because it must inform decisions that react to a dynamic, changing world, (3) *cognitive work is "off-loaded" onto the environment* because, due to the limitations of our processing capabilities, we must exploit the world on a "need to know" basis, (4) *the mind and the world form a closed-loop system* that make it difficult to study cognition in isolation, (5) *cognition is for action* and must be understood in terms of its ability to allow for interaction with a dynamic world, and (6) *"off-line," mental cognition is body based* and grounds the mechanisms for thought and understanding. In conjunction, these views form a compelling argument that cognitive development should be studied in embodied systems, such as a robots, that are designed to interact with and manipulate the world.

In *Vehicles*, the psychologist Valentino Braitenberg examined how non-trivial behavior can emerge in simple embodied systems that interact with the world (Braitenberg, 1984). Later, Brooks showed how the emergent behavior of a reactive robot system does not have to rely on complex world models internal to the robot (Brooks, 1986; Brooks, 1991), suggesting that "the world is its own best model" (Ballard, 1991) that might be better left unmodeled. These developments have led to a subfield of *behavior-based* robotics (Arkin, 1998). A number of studies illustrate the development of both language (Steels & Vogt, 1997; Roy, 1999; Oates, 2001) and knowledge representations (Cohen et al., 2001; Kuipers, 2000; Kupiers et al., 2005; Papudesi & Huber, 2006) in computational systems that depend on the physical sensory and motor configurations of the robot to engage the environment actively. This is often referred to as a process of "grounding" empirical knowledge.

2.3.2 Affordance Modeling

Psychologists argue that intelligence is inseparably grounded in an organism's behavioral interaction with the world, shaping our processes of development and cognition (Wilson, 2002), our conceptual thought (Johnson & Lakoff, 1980), and our understanding of the world in terms of behavior it affords (Gibson, 1977). Affordances create ontologies grounded in agent-world interactions implying that knowledge acquisition is inextricably related to embodiment.

There has been a large amount of recent computational work on robot learning techniques for extracting environmental affordances. Chamero defined affordances as a relationship between an agent and an object in terms of the potential for action (Chamero, 2003). Fitzpatrick examined affordances for pushing and grasping objects (Fitzpatrick et al., 2003; Sweeney & Grupen, 2007; Saxena et al., 2007; Stark et al., 2008). Stoytchev et al. looked at the affordances of robot tool use and the sounds objects make when manipulated (Stoytchev, 2005; Sinapov et al., 2009). Affordance learning for navigation tasks has also been applied to the domain of mobile robots (Rome et al., 2006; Modayil & Kupiers, 2007).

The Multi-Sensory Autonomous Cognitive Systems (MACS) group proposed a general model of affordances in terms *entities*, *behaviors*, and *effects* (Şahin et al., 2007). In this work, machine learning algorithms are used to predict the perceptual effects of taking actions that interact with objects (or entities). The MACS formulation has been explored on a mobile robot platform to learn affordances for traversability, pushability, and liftability (by means of a magnetic gripper) (Doğar et al., 2007; Uğur et al., 2009), as well as to ground planning operators for extended tasks (Uğur et al., 2009; Lörken & Hertzberg, 2008).

A number or researchers have provided a formalism of affordances called "Object-Action Complexes" (or OACs), also grounded in the robot's sensory data, but ultimately used for higher-level planning tasks (Krüger et al., 2009; Geib et al., 2006). The OACs framework has been used to predict grasp affordances from the visual appearance of an object for a robot gripper (Kraft et al., 2008; Detry et al., 2009).

Common to these approaches is that robot actions are typically pre-programmed activities that require no sensory feedback to measure "success." These activities make no attempt to ground behavioral objectives in the robot's closed-loop dynamics. In contrast, this document examines a formulation of affordances that arises from a robot's dynamic sensorimotor interactions with its environment.

2.3.3 Intrinsic Motivation

One key tenet of the developmental robotics paradigm is that organisms should be self-motivated to acquire increasingly complex skills over the long term with minimal intervention from human programmers or demonstrators. As a result, developmental roboticists have turned to psychological theories of *intrinsic motivation* to explore theoretical models of self-guided exploration and knowledge acquisition.

Psychological theories of motivation go back over half a century. In the 1940's, Hull introduced the concept of "drives" such as hunger, pain, sex, or escape in human behavior, in terms of deficits that the organism wishes to reduce to achieve homeostatic equilibrium (Hull, 1943). Later researchers extended the Hullian theory by introducing drives for manipulation (Harlow, 1950), and for exploration (Montgomery, 1954).

In contrast, White argued that fundamental drives were not the whole story, suggesting that exploratory behavior often occurs for the sole purpose of gaining knowledge necessary to achieve competent autonomy (White, 1959). Such behavior, White claims, is meant to produce knowledge for its own sake as this information might be useful later in goal-directed (or extrinsically motivated) behavior.

Berlyne proposed a number of intrinsically motivating factors—what he called the *collative variables*—novelty, habituation, curiosity, surprise, challenge and incongruity (Berlyne, 1960). He also observed that an organism is rewarded most when the level of novelty is somewhere between "familiar" and completely "new." Berlyne stated that the spontaneous exploratory behavior observed in infants is explained by these internal drives (Berlyne, 1965).

A number of psychologists examined how an organism may be motivated to improve its cognitive models. Festinger, for example, suggested that there is a drive to reduce the cognitive dissonance between internal knowledge structures and the current perception of the organism's world (Festinger, 1957). Similarly, Kagan introduced primary motivators to reduce the discrepancy between cognitive structures and experience (Kagan, 1972). Linking perception and action, Hunt suggested that children, in fact, seek out to reduce such incongruity by taking actions to gather more information (Hunt, 1965). Such an active strategy for exploring an agent's own cognitive models seems reasonable for robots that develop over the long-term as well and is related to the computational mechanisms introduced in Chapter 6 of this document.

There has been more recent work in the neuroscience community examining the biological basis for drives or motivators, both intrinsic and extrinsic. It has been observed that similarities exist between the activity of midbrain dopamine cells and existing computational models of reinforcement learning (Houk et al., 1995; Schultz & Dayan, 1997). Initial results showed the similarity when considering extrinsically rewarding domains, but more recent studies have argued that this dopamine activity is more indicative of intrinsic motivators (e.g., novelty or exploration) (Dayan & Belleine, 2002; Kakade & Dayan, 2002), as well as "prediction error" (Horvitz, 2000). This research has found that neuromodulator activity in dopamine cells show strong congruence between sensory stimuli that is novel and unpredicted rewards. It has been noted that this fact may be due to the underlying rewarding characteristics of novelty itself (Reed et al., 1996).

Early computational models of theories such as Berlyne's were explored by Csikszentmihalyi who argued that organism strives for experience just beyond their comfort zone (Csikszentmihalyi, 1991; Csikszentmihalyi, 1996). He created a system which was rewarded for reaching situations that presented a new—but *modest*—learning challenge.

Other implementations considered models that were concerned with growing the complexity of their machines (Herrmann et al., 2000). Integrated approaches to novelty and curiosity in developmental learning began in work by Weng (Weng et al., 2001; Huang & Weng, 2002; Weng, 2002), and later pursued by Marshall et al. (2004), and Kaplan and Oudeyer (Kaplan & Oudeyer, 2003; Kaplan & Hafner, 2005; Oudeyer et al., 2005; Oudeyer & Kaplan, 2008). Saunders explored issues of *creativity* by creating an agent that pushes the boundaries of its own "artistic" creations (Saunders, 2002). Much of this work provided interesting initial results, but did not focus on learning skills that could be re-used in later tasks.

Schmidhuber examined how intrinsic motivation techniques can be used to shape reward in reinforcement learning systems (Schmidhuber, 1991b; Schmidhuber, 1991a; Schmidhuber & Storck, 1993). Barto et al. demonstrated how to intrinsically motivate a learning agent to learn a *re-usable set of skills* that can be applied in different, but possibly related, tasks (Barto et al., 2004; Singh et al., 2004a). Typically this work focuses on learning *options* that can be transfered to new tasks (Sutton et al., 1999). This work includes that by Barto and Singh (Barto et al., 2004; Singh et al., 2004a), Digney (1998), McGovern (2002), Şimşek and Barto (2004), and Konidaris and Barto (2007).

Following Schmidhuber, Oudeyer et al. make two useful distinctions in statistical models of intrinsic motivation (Oudeyer et al., 2007): those that predict the consequences of an agent's action (Thrun, 1995; Huang & Weng, 2002; Barto et al., 2004), and those that predict the errors in those predictions (a *meta*-predictor) (Herrmann et al., 2000; Kaplan & Oudeyer, 2003). The former approach, the authors argue, results in systems that will focus on challenging situations that may be too complex to ever get a handle on. The latter approach remedies this issue by using a meta-predictor to recognize that certain situations are too complex, or *unlearnable.* These situations can be ignored at the expense of situations in which more learning progress is possible. These approaches, however, may still stagnate from pathological situations. The authors introduce a more effective means of recognizing learning progress by not only looking at learning rate, but also considering the contextual *similarity* between learning situations to prevent those pathological problems. This work is important because it introduces the idea of an implicit *memory* in the computational intrinsic motivation literature.

Intrinsically motivated computation systems are related to earlier approaches in the statistical and machine learning communities. Federov introduced the idea of "optimal experiment design" in which the next sample is chosen in order to minimize the number of samples and gain the maximal amount of information or return (Federov, 1972). In the machine learning community, such approaches became popular under the name "active learning" (Thrun, 1995; Cohn et al., 1996). This work has provided much of the tools and inspiration for the underlying mechanisms of other intrinsically motivated computational systems. However, it does not speak directly to the creation of re-useable skills in robot behavior.

2.4 Discussion

This thesis proposes a framework in which a robot can represent and organize its knowledge structures dynamically and developmentally to support robust strategies for object manipulation. Manipulation tasks provide a rich domain for studying developmental learning because they require a grounded understanding of how a robot can reason about how to employ motor resources into behavior that responds to multiple domains of sensory data. Furthermore, although competent manipulation strategies will be imperative for any robot operating in unstructured environments, little research has focused on how such strategies might be acquired in adaptive robot systems designed to accumulate knowledge over the long-term. To address these issues, this dissertation proposes a framework to investigate how a robot can use techniques of embodiment, intrinsic motivation, and affordance discovery to bias its exploration to incrementally develop hierarchical manipulation structures that can be used in a variety of tasks.

CHAPTER 3
A COMBINATORIC BASIS FOR ROBOT CONTROL

The **control basis** framework was orignally 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 (Huber, 2000). These combinatorics provide a definition for action that is useful for organizing knowledge into structures that facilitate generalization and transfer. Huber formulated robot learning in a *discrete event dynamic systems* (DEDS) framework, to control the complexity of a state/action spaces and to ensure that safety and performance specifications are satisfied during on-line exploration.

The control basis is used to construct state and actions spaces from feedback controllers by combining elements of a set of artificial potential functions Ω_ϕ, with elements of the robot's sensory and motor resources Ω_σ and Ω_τ, respectively. The control basis provides a representation in which reinforcement learning techniques can be used to create value functions, Φ, that define discrete action policies. These policies are the hierarchical generalization of the primitive control laws that comprise the control basis. The result is a naturally recursive description of knowledge in terms of functions (*intentions*) that underlie behavior. Hierarchical control and the construction of value functions, Φ, will be discussed in Chapter 4.

All control expressions constructed from the control basis provide force or velocity reference commands to lower-level closed-loop motor units that actuate the robot. This framework is diagrammed in Figure 3.1. The process of assembling feedback control loops from potential fields is discussed in the remainder of this chapter. How

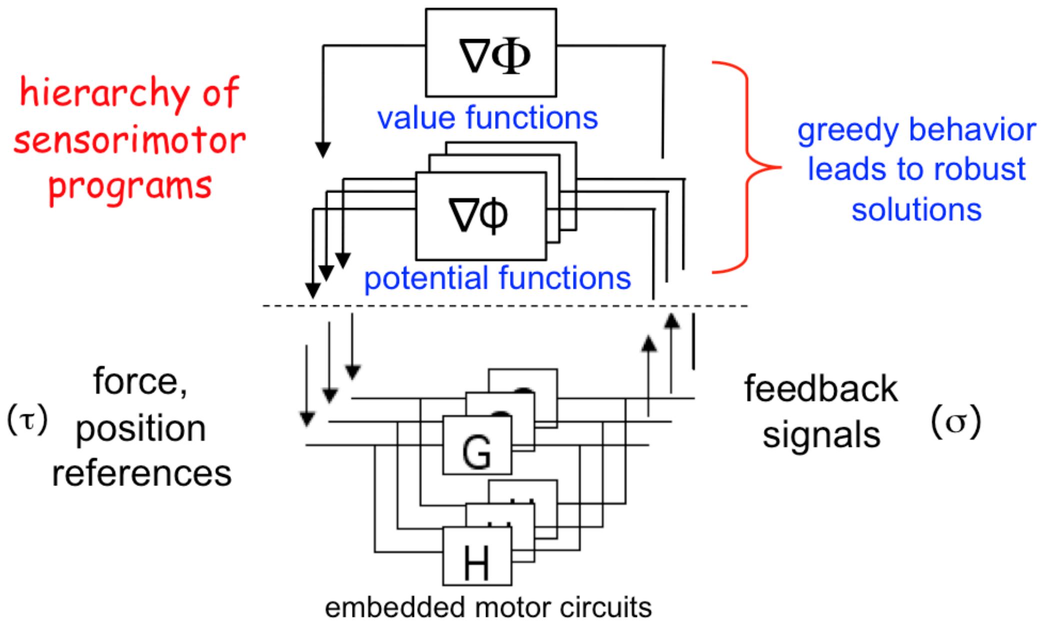

hierarchy of sensorimotor programs
$\nabla\Phi$
value functions
$\nabla\Phi$
potential functions
greedy behavior leads to robust solutions
(τ) force, position references
feedback signals (σ)
G
H
embedded motor circuits

Figure 3.1. The hierarchical control basis architecture. Control actions descend potential functions by submitting reference inputs to lower-levels and ultimately, to embedded motor circuits. Motor circuits are fixed position and force referenced controllers that produce stable behavior. Programs written using these control actions ascend value functions describing optimal sequential behavior. H and G represent feedback and feedforward transfer functions, respectively.

to assemble higher-level programs on top of these feedback controllers is discussed in the next chapter.

3.1 Artificial Potential Functions

The control basis exploits the fact that a small alphabet of primitive control elements $\Omega_\phi \times \Omega_\sigma \times \Omega_\tau$ can yield a large variety of hierarchical control circuits. How do we define an efficient basis, Ω_ϕ, for low-level (or native) control actions? In this section, we examine a set of artificial potential functions that have formal constraints (e.g., no local minima) that are of general use in adaptive control systems. All of the control actions employed in this dissertation can be characterized by these potential functions, as can many other control actions for mobile robots or robot manipulators.

The control basis provides a general and unified framework for assembling feedback control laws when these potential functions are combined with sensory and motor resources.

Potential function approaches to control require constraints on the shape of the potential function in order to guarantee asymptotically stable behavior. Rimon and Koditschek enumerated the conditions for a class of *navigation functions* that can formally be used as control functions (Koditschek & Rimon, 1990; Rimon & Koditschek, 1992):

- **Analytic -** Potential ϕ is an analytic function if it is infinitely differentiable (i.e., it can be written as a Taylor series). This property guarantees that the function has a gradient that points toward a minimum.
- **Polar -** The gradient of the navigation functions create streamlines that terminate at a unique minimum.
- **Admissible -** The gradient $\nabla\phi$ of a navigation function must be bounded so that it can serve as a realistic control input without gain scheduling or scaling.
- **Morse -** Navigation functions are Morse if they contain no degenerate critical points (e.g., saddle points).

Careful use of topological constraints on the shape of the potential function can provide a basis for control that minimizes the problems with potential field approaches that are commonly cited in the literature (e.g., local minima). We now examine a number of artificial potential functions that form a basis for the experimental work in this thesis and describe the conditions under which they can be used to construct closed-loop controllers.

3.1.1 Quadratic Potential Functions

Hooke's law is an example of a quadratic potential field that describes the strain energy stored in a spring. It can be employed in the control basis to induce virtual spring-like properties on feedback errors observed in many domains. Hooke's law is defined as:

$$\phi_s(\sigma_{ref}, \sigma_{act}) \quad = \quad \frac{1}{2}(\sigma_{ref} - \sigma_{act})^T(\sigma_{ref} - \sigma_{act}) \qquad (3.1)$$

where the difference between the actual and the reference feedback signals, $\sigma_{act}, \sigma_{ref} \subseteq \Omega_\sigma$, captures virtual errors between two features of the same type. This function is convex and, if the input error is bounded, then it also a navigation function and can be used for control[1]. We use this kind of primitive "intention" repeatedly in the control basis framework to, for instance, build tracking controllers in force and position domains. Hooke's law can be employed for:

a) Configuration Control: where the objective is to reduce a relative error between two n-dimensional robot configurations in the space $\mathcal{C}_n$. Configuration-space tracking controllers can provide smooth movements to via points defined by a trajectory generator, or can move a robot's degrees of freedom (DOFs) to locations where other interesting events occur (e.g., configurations where contact is often made with objects). Configuration-space control can be useful for maintaining a robot's end-effector pose or for tasks such as moving a pan/tilt camera to foveate on a visual feature.

b) Spatial Control: where the objective is to reduce a relative error between Cartesian goals in subsets of $SE(3)$. Spatial error tracking controllers can position

[1]We ensure that errors are bounded in the experiments discussed in this dissertation.

a robot's hand at the location of an object or control its orientation, move a mobile robot to a desirable goal location (such as a door), or to follow a leader robot (Sweeney et al., 2002). It also can be used to guide visual servoing tasks that minimize the visual distance between a robot's hand and an object (Hager et al., 1996).

c) Force Control: where the objective is to reduce a relative error between contact forces, torques, or moments in subsets of $SE^*(3)$. Force-domain controllers can be used to maintain a grasp on an object or to actively reduce the magnitude of contact forces with the environment. Regulation tasks that remove residual force and moment errors can be used to achieve grasp conditions such as wrench-closure (Coelho, 2001; Platt et al., 2002) or to perform assembly tasks such as "peg-in-hole."

3.1.2 Harmonic Functions for Collision-Free Motion

Much attention in the robotics literature concerns how to move robot manipulators from one configuration to another along a collision-free path. This problem is known to be PSPACE-hard in the number of degrees-of-freedom of the robot (Canny, 1988), and is challenging also because it often requires knowing the location of relevant obstacles that may be dynamic. Many approaches to robot navigation have been examined over the last few decades (Latombe, 1991). Often, these techniques utilize sample-based planning methods such as the Probabilistic Road Map (PRM) (Kavraki et al., 1996) and Rapidly-Exploring Random Tree (RRT) approaches (LaValle, 1998).

Due to the complexity of sample-based methods, however, artificial potential field approaches have also been developed for robot path planning that make more efficient use of sensor feedback. Harmonic functions describe many physical processes that rely on minimum energy configurations such as soap films, laminar fluid flow, the temperature dissipation in thermally conductive media, and the voltage distribution in resistive networks. They have been applied in robot systems to find path plans with no local minima and that produce the minimum probability of collisions (Connolly et al., 1990; Connolly & Grupen, 1994; Akishita et al., 1990). Harmonic functions satisfy Laplace's equation,

$$\nabla^2 \phi_h = \frac{\partial^2 \phi}{\partial q_1^2} + \frac{\partial^2 \phi}{\partial q_2^2} + \cdots + \frac{\partial^2 \phi}{\partial q_n^2} = 0, \tag{3.2}$$

for an n-dimensional state-space $\mathbf{q}$. Goals and obstacles define the boundaries of the navigatable free space in this approach. A harmonic potential field has no local minima or maxima (type 0 and type 2 critical) points in the interior of this space. Laplace's constraint only permits saddle points (type 1 critical points) in the interior of the free space. The critical points constitute a set of measure zero, so that any small random perturbation will move the system away from the saddle points and into areas where a gradient exists. Numerical relaxation methods (e.g., Successive Over-Relaxation (SOR), Jacobi iteration, or Gauss-Seidel iteration (Burden et al., 1972)) can solve for harmonic functions quickly in tasks for low-dimensional systems.

Harmonic functions, unfortunately, do not meet all four of the criteria for navigation functions. As a result, they produce paths that minimize hitting probability, but are not asymptotically stable. This limitation can be overcome by allowing the system to follow the direction of the potential's gradient, but shaping its magnitude before sending it to the plant.

3.1.3 Kinematic Conditioning Functions

Conditioning actions are useful in multi-objective control tasks and can provide a natural way for an embodied system to get the most out of its sensory and motor resources (Hart & Grupen, 2007). Metrics based on the scalar condition number of the manipulator Jacobian have been used to optimize the kinodynamic configuration of a robot mechanism (Salisbury & Craig, 1982). These metrics are useful for avoiding singular locations where a robot cannot be controlled in certain directions. Yoshikawa proposed a field for biasing the manipulator Jacobian toward configurations that optimize the "manipulability" index and avoiding singularities (Yoshikawa, 1985). Examples of isotropic conditioning techniques that allow a mechanism to be equally sensitive to input displacements in all directions can be found in (Nakamura, 1991; Asada & Granito, 1985; Angeles & Lopez-Cajun, 1992; Grupen & Souccar, 1993; Ranjbaran et al., 1996).

Using similar techniques, anisotropic conditioning can be applied when the task is well-specified ahead of time. Chiu demonstrated how to utilize the redundancy of a manipulator to optimize the application of velocities and forces along known task directions (Chiu, 1987). In this work, a technique was proposed to increase force/velocity amplification or precision along specific directions. While techniques for anistropic conditioning are useful for the high-performance transmission of displacements in highly sensitive tasks, Chiu's approach does not lead directly to a useful navigation function.

Several conditioning fields have been devised in the course of this dissertation—each of which captures some independent prerogative of a kinematic system.

a) Range Limits: This field is useful for keeping a manipulator away from joint range limits. It can provide a greater likelihood that global objectives are met because it considers the physical limitations of the robot. One possible choice for an n-dimensional system, implements an n-dimensional cosine field around the center of each joint's range of motion. For a set of joint angles $\boldsymbol{\theta} \in \mathcal{C}_n$, we define

$$\phi_r(\boldsymbol{\theta}) = n - \sum_{i}^{n} cos(g_i(\theta_i)) \tag{3.3}$$

where

$$g_i(\theta_i) = \frac{\theta_i - \bar{\theta}_i}{\theta_{i,max} - \theta_{i,min}}\pi. \tag{3.4}$$

In this equation, $\theta_{i,max}$ and $\theta_{i,min}$ represent the upper and lower limits of joint i's range of motion, and $\bar{\theta}_i = \theta_{i,min} + (\theta_{i,max} - \theta_{i,min})/2$. This field provides a convex potential that is centered in the middle of the robot's range of motion over all degrees of freedom.

Function $\phi_r(\boldsymbol{\theta})$ is the sum of independent cosines over the domain $-\pi \leq \theta_i \leq +\pi$ for all i. The diagonal elements of the Hessian are of the form $d^2\phi/d\theta_i^2 = cos(g_i(\theta_i))k_i$ where $k_i = (\frac{\pi}{\theta_{i,max}-\theta_{i,min}})^2$ and the off-diagonal elements are equal to zero. The field, therefore, is positive definite and has only one maximum. Because trigonometric functions are analytic (i.e., they can be represented as a Taylor series), then there are no interior critical points and the field is Morse. The gradient magnitude $\nabla\phi_r(\boldsymbol{\theta})$ is bounded (between 0 and $\sqrt{k_i}$) at every point, and thus the function is admissible. It follows that $\phi_r(\boldsymbol{\theta})$ is a navigation function because it satisfies the four necessary requirements.

b) Manipulability: Yoshikawa's *measure of manipulability* (MoM) field moves a kinematic mechanism into configurations that allow for a tradeoff in the ability to perceive (input) errors and to impart (output) movements (Yoshikawa, 1985). Informally, it conditions the manipulator Jacobian in a manner that preserves flexibility and least commitment for unknown future circumstances. Such a methodology is useful when programming dexterous robots that must behave well in uncertain real-world environments. In addition, it provides a natural kinematic "sweet-spot" in the system, and pushes the manipulator away from singular locations. Whether the manipulability field for a particular robot is a navigation function or not depends on that robot's specific configuration. However, it is often a useful objective to maximize in practice to provide smooth and natural movements for redundant manipulators. The manipulability field is defined as:

$$\phi_m(\boldsymbol{\theta}) \quad = \quad -\sqrt{det(\mathbf{J}(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta})^T)} \tag{3.5}$$

where $\boldsymbol{\theta} \in \mathcal{C}_n$ is an n-dimensional subset of joints that form a kinematic chain and $\mathbf{J}$ is the manipulator Jacobian.

c) Localizability: Kinematic conditioning can be applied to any linear transformation and has been generalized to incorporate acceleration and inertial measures (Chiacchio et al., 1992; Chiacchio, 2001; Nakamura, 1991; Rosenstein & Grupen, 2002), and also to evaluate viewpoint quality in a stereo system in order to maximize localization precision (Uppala et al., 2002; Hart & Grupen, 2007). Let us define a *measure of localizability* (MoL) field to achieve this latter objective. This field is defined in terms of the oculomotor Jacobian $\mathbf{J}(\boldsymbol{\gamma}^l, \boldsymbol{\gamma}^r)$, where $\boldsymbol{\gamma}^l$ and $\boldsymbol{\gamma}^r$ represent the headings toward a feature viewed by both the left and right cameras. The oculomotor Jacobian transforms visual displacements into Cartesian displacements. The localizability field is defined as:

$$\phi_l(\boldsymbol{\gamma}^l, \boldsymbol{\gamma}^r) = \frac{1}{\sqrt{det(\mathbf{J}(\boldsymbol{\gamma}^l, \boldsymbol{\gamma}^r)\mathbf{J}(\boldsymbol{\gamma}^l, \boldsymbol{\gamma}^r)^T)}}. \tag{3.6}$$

This potential field describes how the Jacobian of the stereo triangulation equations amplify imprecision. Optimizing for localizability allows for high precision in stereo-triangulation tasks where the objective is to recover the Cartesian location of a feature from its visual appearance.

The collection of potential functions presented in this section,

$$\Omega_\phi \quad = \quad \{\phi_s, \phi_h, \phi_r, \phi_m, \phi_l\}, \tag{3.7}$$

provides a number of ways for a robot to re-code its sensory signals into artificial gradients that precipitate behavior. I argue that this set of potentials can support a rich set of behavioral prerogatives in an embodied system. In the remainder of this dissertation, I support this argument with demonstrations in which a bimanual robot employing only this set of potentials learns how to perform a number of hierarchical manipulation tasks, including stimuli tracking and multi-object assembly.

3.2 Sensory and Motor Signals

We now discuss the sources of sensory and motor signals that provide the specific means for an embodied system to observe and interact with its environment. These signals form the sensor and effector sets, Ω_σ and Ω_τ, that, along with a set of potential functions Ω_ϕ, define the primitives of the control basis framework.

Although, any given set of sensorimotor resources is particular to a specific robot, many standard types of signals are used in the robotics literature to govern behavior. Sensor signals may come directly from "raw" physical devices (e.g., encoders, cameras, microphones, force/torque strain gauges, etc.), they may arise through mathematical transformations applied to the information returned by multiple of these devices (e.g., via forward kinematics functions, functions for signal localization, etc.), or they may be sampled from statistical models that the robot builds as it learns about its world.

A robot's set of motor signals is defined by the degrees of freedom that accept command inputs. Typically, this set consists of configuration variables or motor torques, but may consist of "virtual" DOFs defined by sensory transformations (e.g., the position of a robot's hand calculated from the forward kinematics). It is often useful to group subsets of a robot's effector variables together into "synergies" that are controlled concurrently. For example, the motor variables that control a robot's arm form a synergy that can allow for reaching movements.

As we examine the following types of sensorimotor signals, we will provide concrete examples for the robot Dexter seen in Figure 3.2. These sets will define the basis for all of the experimental work in this document. Dexter has a two degree of freedon

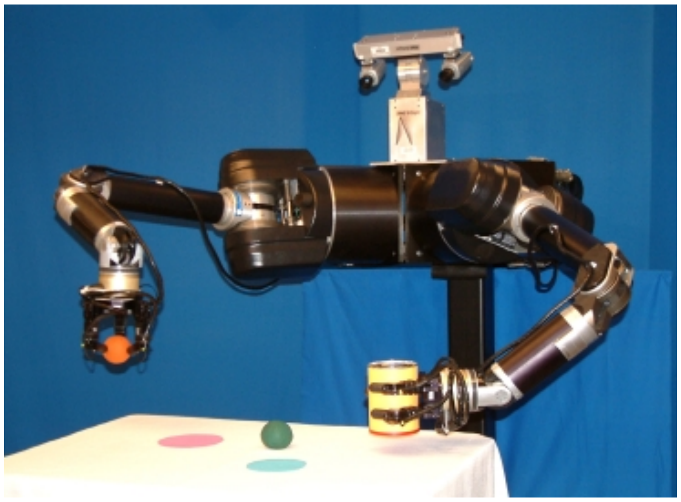


Figure 3.2. The bimanual robot "Dexter."

pan/tilt head equipped with two Sony color cameras and two 7-DOF Whole-Arm Manipulators (Barrett Technologies, Cambridge MA). Each WAM is equipped with a 3-finger Barrett Hand with a F/T load-cell on each fingertip. Each hand has four degrees of freedom (one for each finger, and one for the spread angle between two of these fingers).

3.2.1 Directly Measured Signals

Robots often provide a number of channels of directly measurable sensory data (e.g., a robot's joint encoders or accelerometers). Dexter incorporates the following types of measurable signals:

a) Proprioceptive signals provide a robot with a sense of the configuration of its own body by measuring the position of its joints. These variables can capture the pose of a robot, or can be modified to move the robot in certain directions. Dexter has a total of twenty-four degrees of freedom that are grouped into five useful motor synergies to form the set:

$$\Omega_\theta \quad = \quad \{\boldsymbol{\theta}_{i,arm}, \boldsymbol{\theta}_{i,hand}, \boldsymbol{\theta}_{head} \mid i \in \{l, r\}\}. \tag{3.8}$$

where $\boldsymbol{\theta}_{l,arm}$, $\boldsymbol{\theta}_{r,arm} \in \mathcal{C}_7$ measure the configuration variables of the robot's left and right arms, $\boldsymbol{\theta}_{l,hand}$, $\boldsymbol{\theta}_{r,hand} \in \mathcal{C}_4$ measure the configuration variables of the robot's left and right hands, and $\boldsymbol{\theta}_{head} \in \mathcal{C}_2$ measures the configuration variables of the robot's pan/tilt head. This set of resources are also independently actuatable, therefore and define the effector resources available to Dexter.

b) Force-Domain signals provide a robot with a means of measuring when it makes contact with objects in its environment (including itself). Forces and torques can be measured from load-cells, strain gauges, capacitive surfaces, or from examining the motor currents of a robot's joints. Dexter has 6-axis load cells on each of its six fingertips. It will often be useful to consider the forces and torques on each of the robot's fingers in isolation as well as the net forces and torques measured on each of the hands. This set of signals is

$$\Omega_f \quad = \quad \big\{\mathbf{f}_{h,i}, \boldsymbol{\tau}_{h,i}\ \mathbf{f}_{h,net}, \boldsymbol{\tau}_{h,net} \big|\ h \in \{l, r\}, i \in \{1, 2, 3\}\big\}, \tag{3.9}$$

where $\mathbf{f}_{h,i} \in \mathbb{R}^3$ and $\boldsymbol{\tau}_{h,i} \in SO^*(3)$ are the force and torque values measured on finger i of hand h, respectively, $\mathrm{f}_{h,net} = \sum_i^3 \mathbf{f}_{h,i}$, and $\boldsymbol{\tau}_{h,net} \sum_i^3 \boldsymbol{\tau}_{h,i}$. These signals are represented in the robot's world coordinate frame for simplicity.

c) Visual channels of data provide rich sources of information. However, due to the high information content in visual imagery, it is often useful to consider sub-channels of information obtained by pre-processing or filtering. For example, a typical camera image can be decomposed into its RGB or YUV color spaces, or into channels of hue, saturation, or intensity (HSI). Further pre-processing on these spaces allow for a system to compute regions-of-interest, (or ROIs) of similar values, or to compute visual flow fields estimating the observed motion of a scene. Techniques for computing the texture or shape of objects in an image (e.g., N-jets (Koenderink, 1984; Piater, 2001) , SIFT-descriptors (Lowe, 2004), scale-space operators (Lindeberg, 1994)) can also be applied to compute a variety of visual invariants. The stereo camera pair located on Dexter's pan/tilt head provides the robot with a number of visual features of various types. In the experimental work in this document, Dexter uses a subset of these features—those pertaining to headings toward ROI centroids of the observed net motion and of the response from a set of 30 discretized hue, saturation, and intensity channels (10 each)[2]. This set is defined as:

$$\Omega_\gamma \;=\; \left\{\gamma^c_{motion}, \gamma^c_{hue,i}, \gamma^c_{sat,i}, \gamma^c_{int,i} \mid i \in \{1, ..., 10\}, c \in \{l, r\}\right\}, \tag{3.10}$$

where $\gamma^l_i, \gamma^r_i \in SO(2)$ are headings toward ROI features on channel i (e.g., motion, hue-color 7, saturation channel 3, etc.) in the left and right camera image, respectively. Of course, depending on the scene viewed on a robot's cameras, many of these features may not be well-defined at any given time. For example, if no net motion is observed on the robot's left image plane, then $\boldsymbol{\gamma}^l_{motion}$ is evaluated as undefined.

3.2.2 Kinematic Transformations

Transformations on raw sensory information allow for a number of virtual sources of data. For example, many robot tasks involve the manipulation of entities in Cartesian space, something a robot such as Dexter does not have direct access to. However, Cartesian quantities are easily recovered from other sensor sources through kinematic transformations. We discuss two such transformations next.

[2]In the experiments, environments are kept simple to make conspicuous the appearance of objects we wish to teach Dexter about.

a) Forward kinematic functions map joint angle configurations of a robot arm to Cartesian space (Craig, 2004). These functions require knowledge of the robot's geometry (e.g., via Denavit-Hartenberg (DH) parameters). It is natural to express many kinds of subtasks as references in $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{R} \in SO(3)$, or combinations thereof. Therefore, we define two standard kinematic transformations:

$$f_p(\boldsymbol{\theta}_i) = \mathbf{x}_i \tag{3.11}$$

$$f_r(\boldsymbol{\theta}_i) = \mathbf{R}_i \tag{3.12}$$

where $\mathbf{x}_i \in \mathbb{R}^3$ is a position and $\mathbf{R}_i \in SO(3)$ is an orientation of an articulated mechanism with configuration variables $\boldsymbol{\theta}_i$. The following set describes the positions and orientations of Dexter's hands (or *end-effectors*):

$$\Omega_p = \big\{\mathbf{x}_{i,arm}, \mathbf{R}_{i,arm} \mid i \in \{l, r\}\big\}, \tag{3.13}$$

where $\mathbf{x}_{i,arm} = f_p(\boldsymbol{\theta}_{i,arm})$ and $\mathbf{R}_{i,arm} = f_r(\boldsymbol{\theta}_{i,arm})$ for $i \in \{l, r\}$. Differentiating these functions with respect to the configuration variables $\boldsymbol{\theta}_i$ produces the manipulator Jacobian that describes the sensitivity of $\mathbf{x}_i$ to changes in $\boldsymbol{\theta}_i$.

b) Stereo triangulation functions can recover the Cartesian position of features seen on multiple cameras by considering the viewing geometry. The triangulation function takes as input the headings towards a feature i viewed by both the left and right cameras, $(\boldsymbol{\gamma}_i^l, \boldsymbol{\gamma}_i^r)$, and returns that feature's Cartesian position,

$$f_t(\boldsymbol{\gamma}_i^l, \boldsymbol{\gamma}_i^r) = \mathbf{x}_i. \tag{3.14}$$

Differentiating this function with respect to the input headings produces the oculomotor Jacobian. Dexter uses this equation to compute Cartesian positions corresponding to the thirty-one potential channels of visual data in Ω_γ:

$$\Omega_v = \big\{\mathbf{x}_{motion}, \mathbf{x}_{hue,i}, \mathbf{x}_{sat,i}, \mathbf{x}_{int,i} \mid i \in \{1, ..., 10\}\big\}. \tag{3.15}$$

3.2.3 Internal Models

As a robot interacts with its world, it can learn statistical models of configurations that tend to lead to rewarding events and use these models to inform future searches. For example, a mobile robot can learn that certain locations in a room (e.g., a book shelf or a toy bin) have often contained play objects in the past. When it desires to play with these objects again, the robot can drive to these locations and to see if they are present.

We denote the set of models for the robot using set Ω_m. Because the number of such models may grown continually over a robot's lifetime, it is difficult to estimate a reasonable upper bound on the size of this set. In the next chapter, we will examine how elements of Ω_m can improve a robot's ability to achieve reward.

3.2.4 Typing

This section introduces a number of sources of sensory information and abstractions of these signals that can be used in control programs. Control circuits, however, require strictly typed inputs and outputs to behave as intended (Henderson & Shilcrat, 1984). Typing is important for guaranteeing correct control expressions as well as for enabling behavioral abstraction. For example, kinematic conditioning metrics, such as those presented in Section 3.1.3, can be applied to any collection of configuration variables, regardless of the specific mechanism they represent. Similarly, "reaching" tasks that move a robot's end-effector to a Cartesian position defined by triangulation can be implemented for any combination of hue, saturation, and intensity features.

Sensory signals of one type can sometimes be transformed—or *typecast*—into signals of a different type, via the forward kinematics or stereo triangulation equations (or their inverses). Typecasting creates dexterous alternatives for controlling robots, and allows combinations of control tasks to be constructed in different state spaces and combined in some joint space. The notion of typing underlies the approach for the generalization and transfer of control programs that is presented in Chapter 5, and will be discussed in more detail at that point.

Enforcing typing constraints and allowing automatic typecasting between signals makes it possible to implement a formal programming specification for the control basis that can facilitate code re-use and control construction, and can generate search spaces for machine learning algorithms. A Control Basis Applications Programming Interface (or CBAPI) was implemented using Microsoft Robotics Developer's Studio (Microsoft Co., 2008) middleware and used to perform all of the experimental work in this dissertation (Hart et al., 2009).

This section also introduced a set of sensory signals that can provide a large set of basic feedback information for the robot Dexter. Let us define the control basis sets of sensor and motor resources for Dexter used herein as:

$$\Omega_\sigma \triangleq \{\Omega_\theta, \Omega_f, \Omega_\gamma, \Omega_p, \Omega_v, \Omega_m\}, \tag{3.16}$$

and

$$\Omega_\tau \triangleq \{\Omega_\theta, \Omega_p\}, \tag{3.17}$$

where the elements are all of the sets defined for Dexter in this section. The effector set includes both the configuration variables Ω_θ because they can be controlled directly

Type	**Space**
Configuration variables	$\mathcal{C}_n$
Wrench coordinates	$SE^*(3)$
Force vectors	$\mathbb{R}^3$
Torques	$SO^*(3)$
Headings	$SO(2)$
Cartesian coordinates	$SE(3)$
Cartesian positions	$\mathbb{R}^3$
Cartesian orientations	$SO(3)$

Table 3.1. Dexter's Resource Types.

as well as "virtual" effector variables Ω_p that can accept input commands via the manipulator Jacobians. The set of types supported by these resource sets is displayed in Table 3.1.

The sets Ω_ϕ, Ω_σ, and Ω_τ define a large combinatoric space, but other resource allocations could be used for Dexter. Nevertheless, we will demonstrate how this particular specification supports a vast amount of manipulation behavior.

3.3 Typed Control Expressions

The triples formed by combining one element from Ω_ϕ with subsets of Ω_σ and Ω_τ provide the basis of all primitive control actions a robot can use to build integrated behavioral programs. In this section, we will show how to (1) construct control expressions from these triples, and (2) how to combine control expressions into multi-objective laws that do not sacrifice the shape properties of the individual controllers (i.e., the navigation function conditions).

3.3.1 Primitive Control Actions

Primitive actions in the control basis framework are closed-loop feedback controllers constructed by combining a potential function singleton $\phi \in \Omega_\phi$, with feedback signals $\sigma \subseteq \Omega_\sigma$, and motor variables $\tau \subseteq \Omega_\tau$. In any such configuration, $\phi(\sigma)$ is a navigation function defined to satisfy properties that guarantee asymptotic stability and no local minima, as discussed in Section 3.1.

A closed-loop controller in the control basis, $c(\phi, \sigma, \tau)$, describes a circuit that iteratively computes reference inputs to low-level motor units. The sensitivity of the potential to changes in the value of motor variables is captured in the task Jacobian $\mathbf{J} = \partial\phi(\sigma)/\partial\tau$, where $\mathbf{J}^{\#}$ is the Moore-Penrose pseudoinverse (Nakamura, 1991). Control signals are computed by the expression:

$$\Delta\tau \;=\; -\kappa\left(\mathbf{J}^{\#}\phi(\sigma)\right), \tag{3.18}$$

where κ is a positive gain[3]. Under this control law, the system follows the negative gradient of the potential toward stable attractor states where $\nabla_{\tau}\phi(\sigma) = 0$. All controllers in the control basis framework define linear dynamical systems that suppress disturbances from the environment.

3.3.2 Co-Articulation

Multi-objective control actions are constructed by combining control primitives in a prioritized fashion. Consider two control actions, a higher priority controller $c(\phi_1, \sigma_1, \tau_1)$ and a lower priority controller $c(\phi_2, \sigma_2, \tau_2)$. To ensure that the lower priority objective does not destructively interfere with the progress of the higher priority objective, we combine control objectives using *nullspace projection* (Nakamura, 1991). If the effector variables τ_1 and τ_2 are of different types, the lower priority objective must first be transformed into the space of the higher priority objective. Let the configuration variables of $\tau_1 = \mathbf{q}_1 = [q_1 \ldots q_m]^T$. Let $g(\cdot)$ by a function that maps τ_2 into the same space as τ_1 so that $g(\tau_2) = \mathbf{q}_2 = [q_k \ldots q_n]^T$, where, in general, k may be less than m, and moving $\mathbf{q}_2$ has an effect on $\mathbf{q}_1$. We now show how to compute a

[3]In all of the experiments in this document $\kappa = 1$ for simplicity.

composite output signal in the space $\tau = \mathbf{q}_1 \cap \mathbf{q}_2 = [q_1 \ldots q_n]^T$. First we define the individual task Jacobians

$$\mathbf{J}_1 = \frac{\partial \phi_1}{\partial \mathbf{q}_1} = \left[\frac{\partial \phi_1}{\partial q_1} \cdots \frac{\partial \phi_1}{\partial q_k} \cdots \frac{\partial \phi_1}{\partial q_m} \right] \in \mathbb{R}^{1 \times m}, \tag{3.19}$$

and

$$\mathbf{J}_2 = \frac{\partial \phi_2}{\partial \mathbf{q}_2} = \left[\frac{\partial \phi_2}{\partial q_k} \cdots \frac{\partial \phi_2}{\partial q_m} \cdots \frac{\partial \phi_2}{\partial q_n} \right] \in \mathbb{R}^{1 \times (n-k+1)}. \tag{3.20}$$

We pad these Jacobians with zeros in order to represent them in τ, so that

$$\bar{\mathbf{J}}_{11} = \frac{\partial \phi_1}{\partial \mathbf{q}_1} = \left[\frac{\partial \phi_1}{\partial q_1} \cdots \frac{\partial \phi_1}{\partial q_k} \cdots \frac{\partial \phi_1}{\partial q_m} \; \mathbf{0}_{n-m} \right] \in \mathbb{R}^{1 \times n}, \tag{3.21}$$

and

$$\bar{\mathbf{J}}_{22} = \frac{\partial \phi_2}{\partial \mathbf{q}_2} = \left[\mathbf{0}_{k-1} \; \frac{\partial \phi_2}{\partial q_k} \cdots \frac{\partial \phi_2}{\partial q_m} \cdots \frac{\partial \phi_2}{\partial q_n} \right] \in \mathbb{R}^{1 \times n}. \tag{3.22}$$

With this padding each transformation reflects its insensitivity to changes in the values of variables outside the domain of the corresponding control objective. The change in output of the superior action in the domain of τ is

$$\Delta \tau = -\kappa_1 \left(\bar{\mathbf{J}}_{11}^{\#} \phi_1(\sigma_1) \right). \tag{3.23}$$

The change in output of the subordinate action, however, must be projected into the nullspace of the task Jacobian of the superior objective in τ,

$$\mathcal{N}_{12} = \left(\mathbf{I} - \bar{\mathbf{J}}_{12}^{\#} \bar{\mathbf{J}}_{12} \right), \tag{3.24}$$

where

$$\bar{\mathbf{J}}_{12} = \frac{\partial \phi_1}{\partial \mathbf{q}_2} = \left[\mathbf{0}_{k-1} \; \frac{\partial \phi_1}{\partial q_k} \cdots \frac{\partial \phi_1}{\partial q_m} \cdots \frac{\partial \phi_1}{\partial q_n} \right] \in \mathbb{R}^{1 \times n}, \tag{3.25}$$

before it can be added to the output. The resulting prioritized composite control law is

$$\Delta\tau = -\kappa_1 \left(\bar{\mathbf{J}}_{11}^{\#}\phi_1(\sigma_1)\right) - \kappa_2\, \mathcal{N}_{12} \left(\bar{\mathbf{J}}_{22}^{\#}\phi_2(\sigma_2)\right). \tag{3.26}$$

This control law can be extended to all combinations of n-fold concurrency relationships between control basis actions. We use the "subject-to" operator "$\triangleleft$" to represent the prioritized combination between any two such control actions (Huber & Grupen, 1996; Huber, 2000). The control expression $c_2 \triangleleft c_1$—read, "c_2 *subject-to* c_1"—provides a useful shorthand notation for Equation 3.26.

3.3.3 Examples

We now provide two examples in which co-articulated control basis expressions are implemented on the robot Dexter to improve its performance in two different tasks. These expressions are assembled from Dexter's task-independent—*native*—resource sets, but are used in the context to reach out to objects and to facilitate visual classification. Both examples emphasize the utility of "uncommitted" conditioning actions in task-directed behavior.

3.3.3.1 Kinematically Conditioned Reaching

In this first set of demonstrations, we examine the effect of moving Dexter's right arm to an object placed in locations in the robot's workspace while optimizing kinematic conditioning objectives in the nullspace. By doing so, the robot can perform actions while trying to keep away from joint limits and singular configurations. These demonstrations are adapted from those provided in Hart and Grupen (2007). Consider three controllers:

PosturalBias is a kinematic conditioning control action that biases the posture of a robot mechanism toward the middle of its range of motion via ϕ_r. In this example, it is employed for the robot's right arm and is defined as

$$\text{PosturalBias}(\text{RightArm}) \triangleq c(\phi_r, \boldsymbol{\theta}_{r,arm}, \boldsymbol{\theta}_{r,arm}). \tag{3.27}$$

Substituting this parameter set into Equation 3.18, motor variable displacements can be computed as:

$$\Delta\boldsymbol{\theta}_{r,arm} = -\left(\frac{\partial\phi_r(\boldsymbol{\theta}_{r,arm})}{\partial\boldsymbol{\theta}_{r,arm}}\right)^{\#} \phi_r(\boldsymbol{\theta}_{r,arm}). \tag{3.28}$$

Manipulability is a kinematic conditioning control action that optimizes the manipulability metric according to ϕ_m. In this example, we optimize this metric with respect to the robot's right arm, defined as

$$\text{Manipulability}(\text{RightArm}) \triangleq c(\phi_m, \boldsymbol{\theta}_{r,arm}, \boldsymbol{\theta}_{r,arm}). \tag{3.29}$$

Control outputs from this controller can also be computed by substituting this parameter set into Equation 3.18.

Reach is a tracking control action that uses the virtual spring potential function ϕ_s to reduce the Cartesian error between the end-effector and a reference position. In this example, Reach action is implemented to move Dexter's right arm, $\mathbf{x}_{r,arm} \in \mathbb{R}^3$ (computed from the robot's forward kinematics), toward the position of a highly-saturated object $\mathbf{x}_{sat,10}$ (computed by triangulating the visual feature $(\boldsymbol{\gamma}^l_{sat,10}, \boldsymbol{\gamma}^r_{sat,10})$ viewable on both of Dexter's left and right camera images), and is defined as:

$$\text{Reach}(\text{Sat-10},\text{RightArm}) \triangleq c(\phi_s, (\mathbf{x}_{sat,10}, \mathbf{x}_{r,arm}), \mathbf{x}_{r,arm}). \tag{3.30}$$

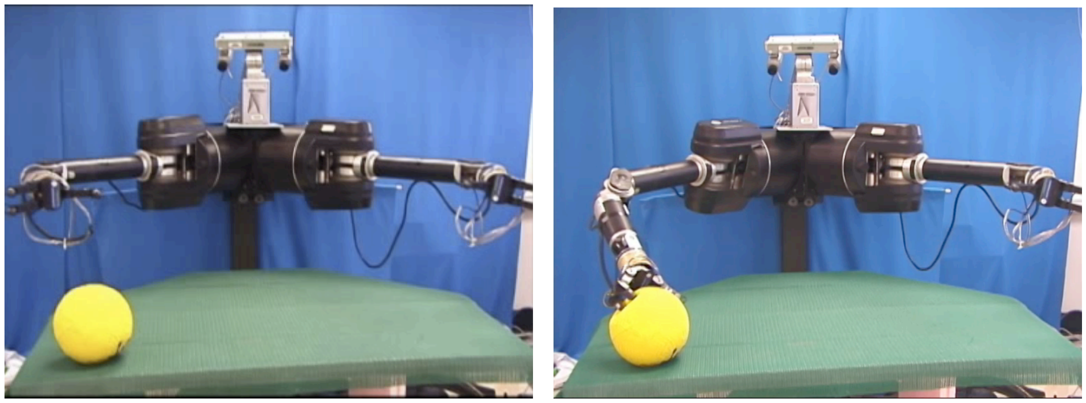

(a) (b)

Figure 3.3. Frames (a) and (b) show the robot before and after reaching to a highly-saturated object with its right arm.

Let us define an error vector $\boldsymbol{\epsilon}=(\mathbf{x}_{sat,10}-\mathbf{x}_{r,arm})$. Using Equation 3.18, effector variable displacements for Reach are computed as follows:

$$\Delta\mathbf{x}_{r,arm} = -\left(\frac{\partial\phi_s(\mathbf{x}_{sat,10},\mathbf{x}_{r,arm})}{\partial\mathbf{x}_{r,arm}}\right)^{\#}\phi_s(\mathbf{x}_{sat,10},\mathbf{x}_{r,arm}) \tag{3.31}$$

$$= \left((\mathbf{x}_{sat,10}-\mathbf{x}_{r,arm})^T\right)^{\#}\left(\frac{1}{2}(\mathbf{x}_{sat,10}-\mathbf{x}_{r,arm})^T(\mathbf{x}_{sat,10}-\mathbf{x}_{r,arm})\right) \tag{3.32}$$

$$= (\boldsymbol{\epsilon}^T)^{\#}\left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) = \frac{1}{2}\boldsymbol{\epsilon}. \tag{3.33}$$

In the following demonstration, a highly-saturated object was placed in twenty-five, uniformly distributed locations on the table in front of Dexter in a $0.4m \times 0.8m$ square. Three composite control laws are constructed and executed, each with the reaching controller as the superior objective. For one control law, this was the only controller employed. The other two laws employed each of the kinematic conditioning controllers as an inferior objective. These three laws are defined as:

$$\begin{aligned}
c_0 &\triangleq \textsc{Reach}(\textsc{Sat-10},\textsc{RightArm})\\
c_1 &\triangleq \textsc{PosturalBias}(\textsc{RightArm}) \triangleleft \textsc{Reach}(\textsc{Sat-10},\textsc{RightArm})\\
c_2 &\triangleq \textsc{Manipulability}(\textsc{RightArm}) \triangleleft \textsc{Reach}(\textsc{Sat-10},\textsc{RightArm})
\end{aligned}$$

The change in potentials for all of the controllers under all three laws were recorded over the twenty-five reach actions. Figure 3.3 illustrates this behavior starting from the center of the arm's range of motion. An example reach is seen in Figure 3.3(a) and Figure 3.3(b).

Figure 3.4(a) shows the average potential ϕ_s for the reaching action over all runs. We see how the quadratic error function causes the system to decrease steadily to the goal. Figure 3.4(b) shows the average value of the postural bias potential over the course of the reaching actions performed under control laws c_0 and c_1. Because the robot begins each action at the minima of this field, the potential only increases as the reaching action is performed. However, in the case where the range of motion controller is optimized in the nullspace, the increase in this metric grows less, on average, than the case where this objective is not optimized, thus staying further away from joint limits. Figure 3.4(c) shows the manipulability potential over the course of the reaching actions performed under control laws c_0 and c_2. In the case where the manipulability controller is optimized in the nullspace, the increase in this metric grows less, on average, than the case where this objective is not optimized, thus keeping the robot further away from undesirable singular configurations.

3.3.3.2 Conditioning for Sensor Acuity

In visual classification tasks accuracy can be improved by optimizing the viewing geometry of the object to be classified. For Dexter, this can be accomplished by picking up the object and moving it closer to the robot's cameras via the localizability metric. Consider the following two controllers:

TOUCH uses the virtual spring potential ϕ_s to apply a small magnitude force in a desired reference direction. In this example, Dexter uses TOUCH to apply a net force of $0.2N$ along the normal direction of its palms (the z-direction of each arm's end-effector coordinate frame), such that ${}^{l}\mathbf{f}_{l,ref} = {}^{r}\mathbf{f}_{r,ref} = [0\ 0\ 0.2]^T$. Rotating these force references into the world frame, we have $\mathbf{f}_{r,ref} = (\mathbf{R}_{r,arm})({}^{r}\mathbf{f}_{r,ref})$ and $\mathbf{f}_{l,ref} = (\mathbf{R}_{l,arm})({}^{l}\mathbf{f}_{l,ref})$, where $\mathbf{R}_{l,arm}$ and $\mathbf{R}_{r,arm}$ are the rotation matrices capturing the orientation of the left and right end-effectors computed from Equation 3.12. If these force references are tracked on both of Dexter's hands simultaneously, this action maintains a simple bimanual hold on an object. TOUCH minimizes the error between $\mathbf{f}_{ref} = \begin{bmatrix} \mathbf{f}_{l,ref} \\ \mathbf{f}_{r,ref} \end{bmatrix}$ and the perceived net forces $\mathbf{f}_{net} = \begin{bmatrix} \mathbf{f}_{l,net} \\ \mathbf{f}_{r,net} \end{bmatrix}$ by moving the position of each of the robot's arms $\mathbf{x}_{arms} = \begin{bmatrix} \mathbf{x}_{l,arm} \\ \mathbf{x}_{r,arm} \end{bmatrix}$. This controller is defined as:

$$\text{TOUCH}(\text{BOTHARMS}) \triangleq c(\phi_s, (\mathbf{f}_{ref}, \mathbf{f}_{net}), \mathbf{x}_{arms}). \qquad (3.34)$$

TOUCH achieves its objective by adjusting the reference Cartesian endpoint position of a virtual spring in response to force errors. If we define the error vector $\boldsymbol{\epsilon}=(\mathbf{f}_{ref} - \mathbf{f}_{net})$, then, by Equation 3.18,

$$
\begin{aligned}
\Delta \mathbf{x}_{arms} &= -\left(\frac{\partial \phi_s(\mathbf{f}_{ref}, \mathbf{f}_{net})}{\partial \mathbf{x}_{arms}}\right)^{\#} \phi_s(\mathbf{f}_{ref}, \mathbf{f}_{net}) && (3.35)\\
&= -\left(\frac{\partial \phi_s(\mathbf{f}_{ref}, \mathbf{f}_{net})}{\partial \mathbf{x}_{arms}}\right)^{\#} \left(\frac{1}{2}(\mathbf{f}_{ref} - \mathbf{f}_{net})^T(\mathbf{f}_{ref} - \mathbf{f}_{net})\right) && (3.36)\\
&= -\left(\frac{\partial \phi_s(\mathbf{f}_{ref}, \mathbf{f}_{net})}{\partial \mathbf{f}_{net}} \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{x}_{arms}}\right)^{\#} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) && (3.37)\\
&= -\left(-(\mathbf{f}_{ref} - \mathbf{f}_{net})^T \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{x}_{arms}}\right)^{\#} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) && (3.38)\\
&= \left(\boldsymbol{\epsilon}^T\right)^{\#} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) = \frac{1}{2}\boldsymbol{\epsilon}. && (3.39)
\end{aligned}
$$

Touch allows Dexter to hold simple objects like balls or boxes when they are placed between the robot's hands.

A similar Touch strategy can be used for the individual fingers of Dexter's hands. Because of the geometric structure of Dexter's hand, simultaneously "Touch-ing" the same object with all three fingers often forms a primitive grasp. This is the control basis analog of the palmer grasp reflex in humans, and demonstrates how an embodied system's morphology can be exploited to accomplish behavior. We will call this primitive grasp a "grab." It should be noted that more robust grasp strategies exist that achieve wrench closure on objects (Coelho, 2001; Platt et al., 2002), but we will be satisfied with Touch in this dissertation due to its simplicity.

Localizability is a conditioning action that optimizes the localizability metric ϕ_l defined by Equation 3.6. It can be used as a primitive form of viewpoint control—or *inspection*—for objects held by a robot. In Figure 3.5, Dexter uses this controller to optimize its viewpoint on a white (intensity channel 10) box observed by the heading features, $(\boldsymbol{\gamma}^l_{int,10}, \boldsymbol{\gamma}^r_{int,10})$. This action moves the centroid of the hand positions $\mathbf{x}_{avg} = \frac{1}{2}(\mathbf{x}_{l,arm} + \mathbf{x}_{r,arm})$ subject-to a bimanual grab. Localizability is defined as:

$$
\text{Localizability(Int-10,BothArms)} \triangleq c(\phi_l, (\boldsymbol{\gamma}^l_{int,10}, \boldsymbol{\gamma}^r_{int,10}), \mathbf{x}_{avg}). \quad (3.40)
$$

Using Equation 3.18, effector variable displacements for LOCALIZABILITY are computed as follows:

$$\Delta \mathbf{x}_{avg} = -\left(\frac{\partial \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10})}{\partial \mathbf{x}_{avg}}\right)^{\#} \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10}) \quad (3.41)$$

$$= -\left(\frac{\partial \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10})}{\partial \mathbf{x}_{int,10}} \frac{\partial \mathbf{x}_{int,10}}{\partial \mathbf{x}_{avg}}\right)^{\#} \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10}) \quad (3.42)$$

$$= -\left(\frac{\partial \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10})}{\partial \mathbf{x}_{int,10}}\right)^{\#} \phi_l(\gamma^l_{int,10}, \gamma^r_{int,10}). \quad (3.43)$$

We define a composite control action that optimizes localizability for the view of an object while maintaining a grab on the object via the TOUCH controller:

$$c_3 \triangleq \text{LOCALIZABILITY}(\text{INT-10}, \text{BOTHARMS}) \triangleleft \text{TOUCH}(\text{BOTHARMS}) \quad (3.44)$$

To show the quantitative effect of localizability, we demonstrate a classification task in which Dexter reads the barcode on a package. A barcode pattern belonging to one of three different sets was placed on the side of a box facing the robot. Each set is displayed as a row in Figure 3.6(a). Each of these sets contains three test-patterns (for a total of nine) that have the same maximum spatial frequency bandwidth, designated as “low,” “medium,” and “high.” Table 3.2 shows the width of the smallest period T of each barcode pattern, as well as their expected pixel resolution on the robot’s cameras and the value of the localizability metric ϕ_l at three different depths x_d. Note how the localizability metric approaches zero as the position moves closer to the robot’s cameras (i.e., the depth decreases). These values were calculated for the 640x480 images using a pinhole camera model with a focal length of 837 mm. The ranges correspond to the location of the box (1) placed in the center of the table in

Table 3.2. Pixels per Period and ϕ_l at Three Object Locations

	$x_d = 100$ cm	$x_d = 75$ cm	$x_d = 50$ cm
$T_{high} = 2$ mm	1.67	2.32	3.35
$T_{medium} = 6$ mm	5.02	6.97	10.04
$T_{low} = 30$ mm	25.11	33.48	50.02
ϕ_l	0.95	0.43	0.14

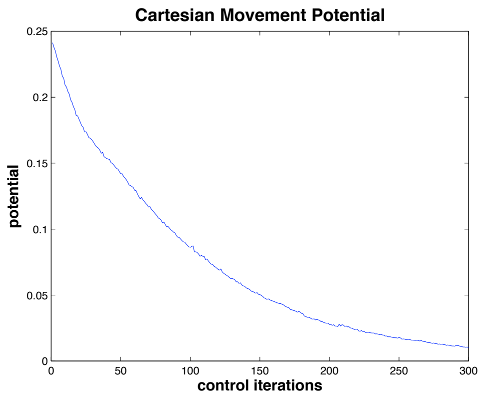
front of the robot (Figure 3.5(a)), (2) held in front of the robot, (Figure 3.5(b)), and (3) held at the optimal localizability configuration, (Figure 3.5(c)). The views from the robot's left camera at each stage of these locations are seen in Figures 3.5(d)-3.5(f).

In each configuration, Dexter converted the appearance of the viewed barcode into a string of intervals based on color (black or white) and bar width. Classification was performed by finding the lowest mean-squared error match between the perceived test pattern and the three patterns in the same (known) frequency class. Figure 3.6(b) shows the accuracy in classification for each pattern set at each of the three locations. Each classification bar is the average result over ten trials for each pattern set. For the table location, only the low-frequency patterns were classifiable. When the box was held in the location in front of the robot, the low- and medium-frequency patterns were classifiable over 80% of the time. The high-frequencies patterns were accurately classifiable just under 80% of the time when the box was moved to the localizability "sweet-spot."

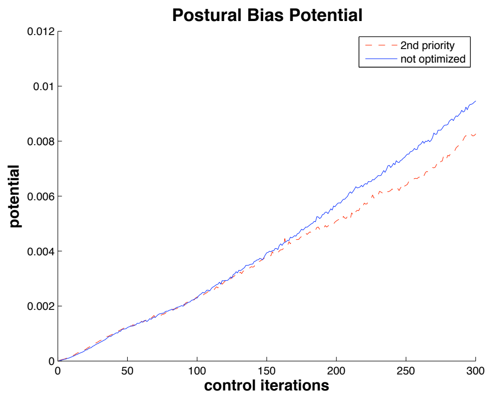
3.4 Discussion

The control basis framework provides a combinatoric means of assembling co-articulated closed-loop control expressions by combining artificial potentials with elements of a set of sensory and motor resources. Demonstrations of the control basis were presented in this chapter on Dexter showing uncommitted task objectives—

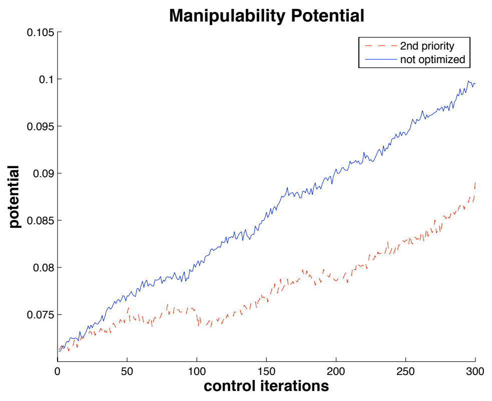
localizing an object, reaching to it, and controlling its viewpoint—can improve the detection of barcodes recognition task. The value of conditioning tasks depend on the tradeoffs between the cost of constructing and computing these control gradients versus the added precision that the conditioning affords given the run-time context. In the next chapter, we examine how programs of control actions can be represented in the control basis and acquired in an autonomous learning framework in which a robot is motivated to discover control affordances. We also show how such programs can be composed hierarchically.



(a)



(b)



(c)

Figure 3.4. Frame (a) shows the potential of the Cartesian movement controller averaged over 25 uniformly distributed locations on the table in front of Dexter. Frame (b) shows the potential for the postural bias controller for the same 25 locations. When that controller is run as a subordinate control action to the Cartesian movement controller compared to the same metric when that controller is not run. Frame (c) shows the same comparison for the manipulability controller run as the secondary objective to the movement action.

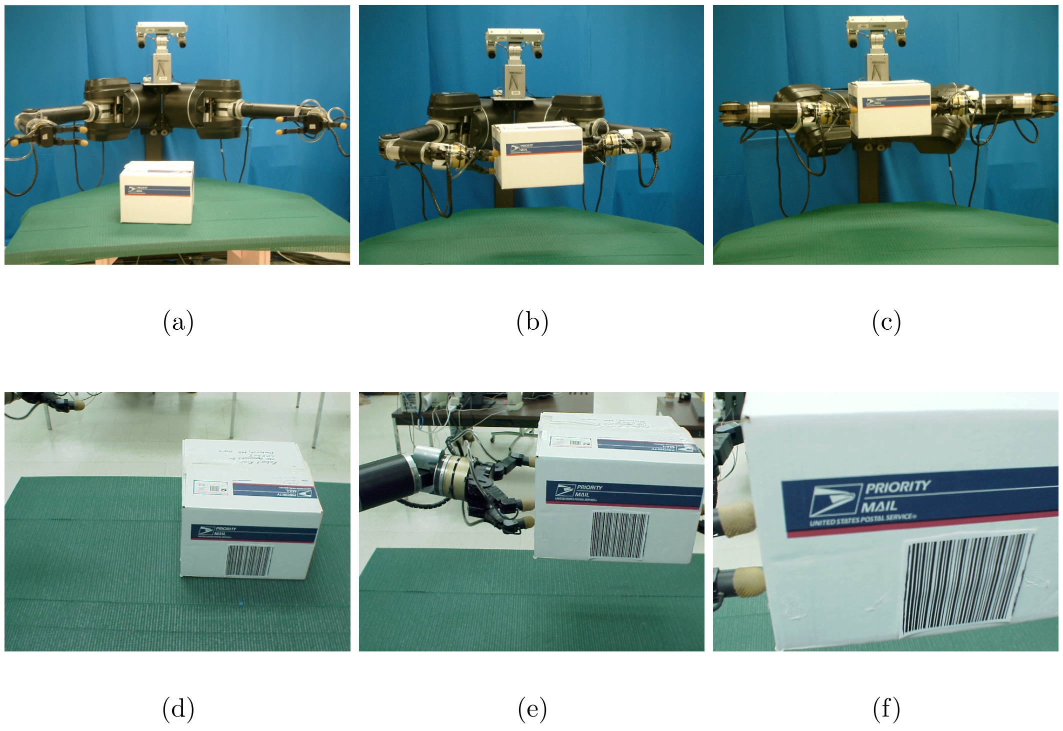
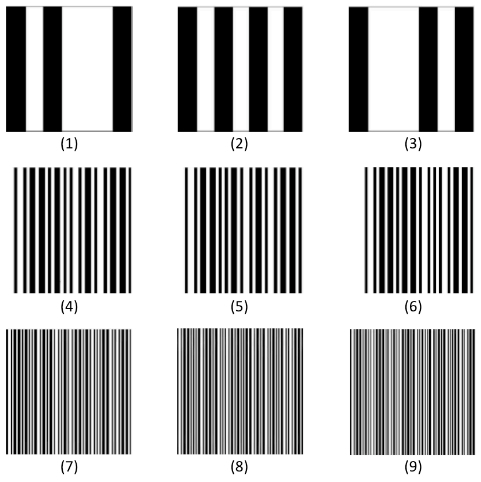
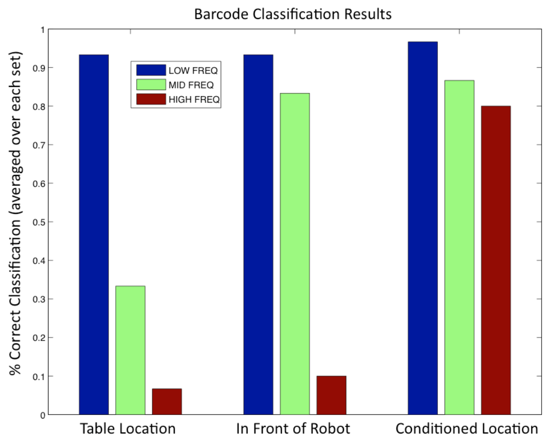


Figure 3.5. Panels (a), (b), and (c) show the three places where the barcode pattern was classified. Panels (d), (e), and (f) show a screenshot from the robot's left camera for the same three locations.



(a)



(b)

Figure 3.6. Panel (a) shows the barcode patterns used for classification experiments. Patterns (1-3) are randomly generated low-frequency patterns. (4-6) are medium-frequency patterns, representing the characters A, B, and C, respectively, in the Code-39 barcode standard. The high-frequency patterns (7-9), are the Code-39 encoded strings ROBOT, DEXTER and AMHERST, respectively. Each pattern is 10 cm square. Panel (b) shows the classification error for each view of the barcode patterns in panel (a) for each different box location. Results for each test set are averaged.

CHAPTER 4
SKILL LEARNING

To behave effectively in unstructured environments, robots must interpret an enormous amount of data and decide how to issue coordinated commands to many degrees of freedom. The control basis framework provides typing constraints that lend structure to the creation of legal control actions and allow for the efficient organization of behavior and knowledge. In particular, control expressions are created by allocating *discrete* sensory and motor resources to dynamic processes that are robust to environmental disturbances. This discrete nature of control composition makes the control basis framework amenable to stochastic search. In this chapter, we describe how machine learning algorithms such as reinforcement learning can explore the structured control basis in order to create domain-general knowledge. We present (1) a novel definition of state that captures convergence events in the run-time dynamics of control actions, and (2) an intrinsic motivator that rewards the discovery of environment affordances.

The proposed definitions of state, action, and reward allow a robot to learn hierarchical control basis programs. Hierarchical representations provide efficient encoding of complex behavioral programs in a manner that hides complexity and bounds the size of the state and action spaces explored. They also encourage behavioral re-use by making available temporally extended policies as single, invokable actions. In the proposed framework, hierarchical programs are built from the bottom up. When a robot learns a new program, it creates an additional means for exploring its environment to acquire additional behavioral capabilities.

To show the efficacy of the proposed framework, a longitudinal learning experiment is performed on Dexter in which the robot acquires a collection of hierarchical programs in developmental *stages*. Each training stage is designed by the programmer to make intrinsic rewards conspicuous. In the stages discussed, the robot explores a program space for a fixed number of learning episodes designated by the programmer. This experimental methodology is used to illustrate the specific learning processes discussed in this chapter. In general, however, a robot should make its *own* decisions about when it has learned all it can from the training context. Chapter 6 introduces a methodology for habituation that can callow a robot to autonomously make such decisions.

4.1 A State Representation for Dynamic Processes

Complex systems that must learn from on-line experience are best served by efficient state representations that capture real-valued sensory and motor signals in compact forms. Useful structure arises when this data is represented as a series of interacting dynamical systems either generated by the robot (e.g., control processes), or observed by the robot's sensors (e.g., environmental forcing functions). The *discrete event dynamic systems* (DEDS) approach leverages this fact by capturing state in terms of discrete events observed in the continuous data (Ostroff & Wonham, 1985).

In the case of control processes, the dynamics $(\phi, \dot{\phi})$ created when a controller interacts with the world provides a natural discrete abstraction of the underlying continuous state space. Huber provided a binary state representation in which the predicate $p_i(\phi, \dot{\phi})$ associated with controller $c_i(\phi, \sigma, \tau)$ is 0 during the transient response of the controller and transitions to 1 when the controller converges to an attractor (Huber & Grupen, 1997). The change in potential, $\dot{\phi}$, is the observed, time-based derivative and is not to be confused with the gradient of the field at that point, $\nabla\phi$.

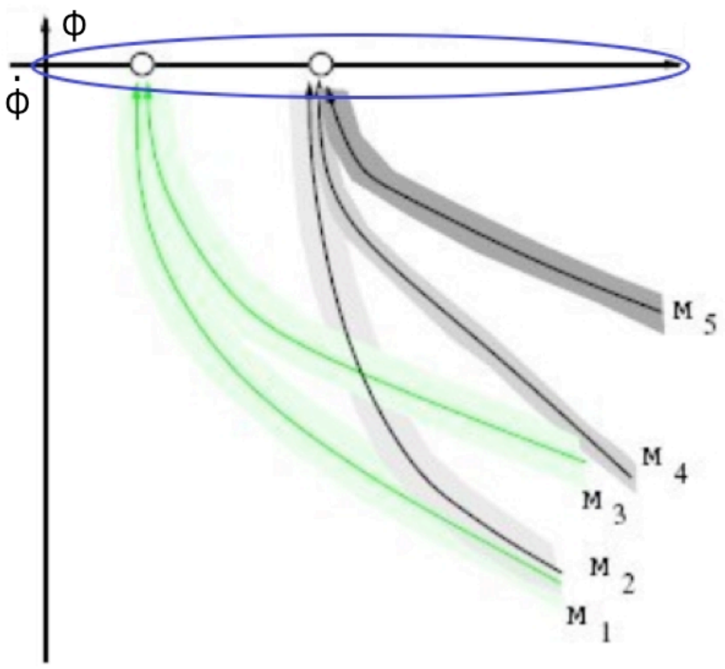


Figure 4.1. Five trajectories of the error dynamics of a controller as they reach convergence (the blue circled region). Each trajectory can be modelled from experience and matched against future controller executions. For asymptotically stable controllers, ϕ is non-negative and $\dot{\phi}$ is negative definite and so these trajectories reside in the lower-right quadrant. This plot is adapted from (Coelho, 2001).

Coelho provided a deeper description of controller state. He developed a predictive state representation that considers the trajectory of the system over time, fitting that trajectory to probabilistic models that were learned for a particular task (Coelho, 2001). A discrete state representation for each controller captured the likely pattern of membership in these learned models that the controller generates. Figure 4.1 shows a phase portrait for five controller trajectory models that converge to two distinct attractor states. Using such a representation, a robot learned a family of dynamic models that could parse the run-time events occurring during a grasping task to recognize the shape of the object being grasped.

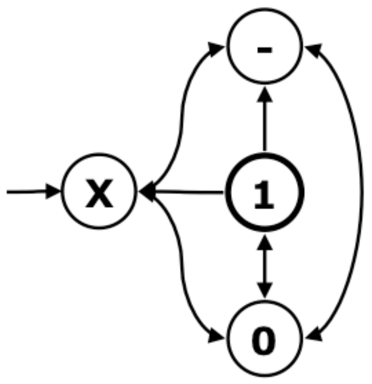


Figure 4.2. This figure shows an iconic representation of the state transitions for a dynamic process.

Coelho's work provided an early example of what was later given a more general formulation in the machine learning community in the form of predictive state representations[1]. PSRs enrich an agents state space with a (potentially large) vector of real-valued observations accumulated as the agent takes actions (Littman et al., 2002; Singh et al., 2004b). The range between PSRs on one side and Huber's binary predicate space on the other provide a large spectrum of state representations that explicitly link actions and observations in controlled processes.

In this dissertation, we will use a simple discrete state definition—more closely related to Huber—based on *quiescence events* capturing controller convergence. Quiescence events occur when either a controller reaches an attractor state in its potential or when a lack of progress along the gradient of that potential is observed. In Figure 4.1, quiescence occurs when the dynamics trajectory reaches the blue-circled region at $\dot{\phi} = 0$, whether the potential has been minimized ($\phi = 0$) or not. Formally, we can define a predicate $p(\phi, \dot{\phi})$ associated with controller $c(\phi, \sigma, \tau)$, such that:

[1]This relationship becomes clear if one considers the control command as the action, the model membership as the state, and the potential dynamics as the observations.

$$p(\phi, \dot{\phi}) = \begin{cases} \text{X} & : \ \phi(\sigma) \text{ controller is not activated} \\ - & : \ \phi(\sigma) \text{ undefined feedback reference } \sigma \\ 0 & : \ |\dot{\phi}| > \epsilon, \text{ transient response} \\ 1 & : \ |\dot{\phi}| \leq \epsilon, \text{ quiescence,} \end{cases} \tag{4.1}$$

where ϵ is a small positive constant. When the controller is not activated, the state evaluates to "X." If the controller is activated, the state predicate will evaluate to "0" if there is a target stimuli present in the feedback signal (and $\phi(\sigma)$ can be computed), or "−" if there is not. The controller runs in the transient state "0" until it loses the target stimuli or quiesces in state "1." Figure 4.2 shows an iconic representation of the evolution of a controller as it interacts with the task domain. Given a collection of n distinct primitive control actions a discrete state space $\mathcal{S}$ can be formed where $\mathbf{s} \in \mathcal{S}$ is defined as $\mathbf{s} = (p_1 \dots p_n)$. In this chapter, Dexter learns a number of control programs that utilize this four-predicate state logic.

4.2 Affordance Discovery

The state representation of Equation 4.1 registers convergence events that occur in the dynamics of a robot's control circuits. When a convergence event occurs for a controller that tracks an environmental forcing function measured by elements of the subset we will call $\Omega_{\sigma(env)} \subseteq \Omega_\sigma$, it creates a closed-loop coupling between the robot and the world that has a special designation. We call this coupling an *environmental affordance.* Affordances are thus measured not only in terms of a perceptual stimuli, but also in terms of the robot's ability to engage that stimuli with its motor resources in a stable control configuration. By modeling affordances defined in this way, a robot can gain a sense of what it can control and thus increase its ability to perform tasks in its world. I argue that a robot designed to learn adaptive behavior in unstructured environments over the long-term must be equipped with an imperative to seek out affordances and the conditions in which they occur.

We now define an intrinsic motivation function for affordance discovery (Hart et al., 2008b). In the control basis, the discovery of an affordance is measured by the convergence event

$$b_i^k = \left((p_i^{k-1} \neq 1) \wedge (p_i^k = 1)\right), \tag{4.2}$$

where p_i^k is the state of a controller $c_i = c(\phi_i, \sigma_i, \tau_i)$ at step k. The intrinsic motivation function provides a unit of reward for all controllers that converge at k according to the function

$$r_i^k = \begin{cases} 1 & : \text{ if } \left(b_i^k \wedge (\sigma_i \subseteq \Omega_{\sigma(env)})\right) \\ 0 & : \text{ otherwise} \end{cases} \tag{4.3}$$

$$r^k = \sum_i r_i^k. \tag{4.4}$$

The restriction that the controller's feedback signal σ_i must be an element of $\Omega_{\sigma(env)}$ prevents reward from occurring for random movements, kinematic conditioning actions, actions that track transformed signals, or actions that respond to internal models the robot may have. As a result, the affordance discovery reward function partitions control expressions derived from the control basis into two disjoint subsets; those that track forcing functions originating in external environmental stimuli and those that do not.

What kinds of controllers produce rewarding events by the affordance discovery motivator? For Dexter, controllers that respond to feedback signals in the set of visual headings Ω_γ and the set of force/torque measurements Ω_f monitor forcing functions originating in external environmental stimuli are rewarding, such that $\Omega_{\sigma(env)} = \{\Omega_\gamma, \Omega_f\}$. Consider a visual tracking controller that moves Dexter's stereo pair of cameras to foveate on a brightly colored object. Because the appearance of the object can be monitored by an element in Ω_γ, a quiescence event for this tracking controller will represent the discovery of a new "trackable" affordance with respect to that object. If that object also affords a controlled touch response using a controller that tracks a force-domain signal in the set Ω_f when the robot reaches out to it (and it does not roll away), that object also has a "touchable" affordance. It is important to note that not all controllers referenced to feedback signals in $\Omega_{\sigma(env)}$ will always provide control affordances. An affordance represents a tight coupling between a perceptual stimulus and the robot's body. If a visual feature, for example, is moving too fast for the robot to track given the limitations of its motor systems, the controller will not produce a convergence event and thus not provide a reward.

The procedure for estimating the state/action value function Φ for a set of control basis actions $\mathcal{A}$ and the reward function provided in Equation 4.4 is shown in Algorithm 1. This procedure is called ACCOMMODATE() because it shapes Φ to provide strategies for uncovering affordances in the environment. It executes m reinforcement learning episodes of no more than T state transitions[2] over the state and action space defined by $\mathcal{A}$. The Q-Learning update rule is shown in Line 16. ACCOMMODATE() also estimates probability distributions of the form $Pr(\tau|\sigma, reward)$ when rewarding conditions are met (Line 19). Distributions of this form provide primitive memory structures that encode configurations where the robot has achieved reward in the past. As described in Section 3.2.3, they can be used as sensory signals in the internal model set Ω_m to inform future searches.

We now present two simple programs Dexter learned using the affordance discovery reward function to find visual and tactile affordances. These programs were learned using Algorithm 1. The first program, called SEARCHTRACK, moves Dexter's pan/tilt head to locations where the robot has previously observed highly-saturated

[2]For all the experiments in this document, $T = 100$. A slight modification of the ACCOMMODATE() procedure could be made to allow it to run until the policy that maximizes return converges, rather than for a number of episodes that is fixed *a priori*.

Algorithm 1 ACCOMMODATE(m, $\mathcal{A}$, T)

```
1: Let $\mathcal{A}'$ be the set of non-composite actions in $\mathcal{A}$
2: Let $\mathcal{S}$ be the state space formed from the predicates of the actions in $\mathcal{A}'$
3: $n \leftarrow |\mathcal{A}'|$, $\gamma = 0.8$, $\alpha = 0.1$, $\epsilon = 0.2$
4: for $i = 1$ to $m$ do
5:     $k \leftarrow 0$
6:     $reward \leftarrow$ false
7:     while $(reward = \text{false}) \wedge (k < T)$ do
8:         $\mathbf{s}^k \leftarrow (p_1^k \ldots p_n^k)$
9:         $a \leftarrow \pi(\mathbf{s}^k)$ (via $\epsilon$-greedy selection)
10:        repeat
11:            execute $a$ for one iteration
12:            $k \leftarrow k + 1$
13:            $\mathbf{s}^k \leftarrow (p_1^k \ldots p_n^k)$
14:        until $\mathbf{s}^k \neq \mathbf{s}^{k-1}$
15:        evaluate $r^k$ according to Equation 4.4
16:        $\Phi(\mathbf{s}^{k-1}, a) \leftarrow \Phi(\mathbf{s}^{k-1}, a) + \alpha\big(r^k + \gamma \max_{a'} \Phi(\mathbf{s}^k, a') - \Phi(\mathbf{s}^{k-1}, a)\big)$
17:        if $r^k > 0$ then
18:            $reward \leftarrow$ true
19:            update $Pr(\tau|\sigma, reward)$ for all rewarding control actions $c(\phi, \sigma, \tau)$,
20:        end if
21:    end while
22: end for
```

pixel regions on its cameras' image planes, and then tracks these regions to determine if they afford quiescence. The second program, called TACTILEPROBE, flexes the robot's hand to configurations where tactile stimuli tend to occur, and then determines whether the environment affords quiescence in a force tracking controller. In both cases, the visual and tactile tracking control actions cause reward according to the affordance discovery motivator. Both learning problems orient the robot to uncover a single rewarding control event, and are taught to Dexter in simple, constrained environments designed to make those events conspicuous.

4.2.1 SearchTrack

In the first learning stage, Dexter acquired a simple skill called SEARCHTRACK for discovering visual affordances. The training context focused exclusively on the most highly-saturated pixels in the field of view. We provide a comparison of learning results from two scenarios, one in which the robot explored co-articulated control actions, and one in which it did not. This comparison demonstrates the quantitative advantage of co-articulated policies over sequential policies that execute only a single action at a time. Both scenarios employed the following two primitive control actions:

TRACK is a control action that pursues a saturation cue on the robot's left camera by changing the reference head posture, $\boldsymbol{\theta}_{head}$, according to the virtual spring potential function ϕ_s. The goal is to keep the coordinate of a highly-saturated visual cue, $\boldsymbol{\gamma}^l_{sat,10}$, at the left camera's image center, $\boldsymbol{\gamma}^l_0 = [0\ 0]^T$. This controller is defined as:

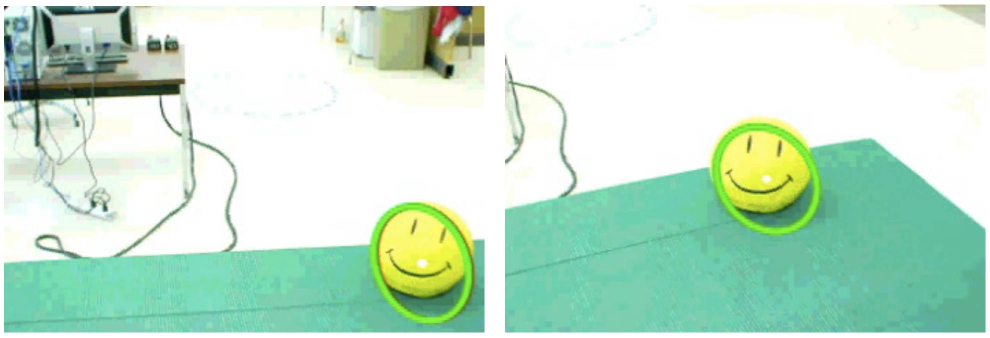
$$\textsc{Track}(\textsc{Sat-10}) \ \triangleq\ c(\phi_s, (\boldsymbol{\gamma}^l_0, \boldsymbol{\gamma}^l_{sat,10}), \boldsymbol{\theta}_{head}). \tag{4.5}$$

Defining an error vector $\boldsymbol{\epsilon}=(\boldsymbol{\gamma}^l_0 - \boldsymbol{\gamma}^l_{sat,10})$, and plugging these resources into Equation 3.18:

$$\begin{aligned}
\Delta\boldsymbol{\theta}_{head} &= -\left(\frac{\partial\phi_s(\boldsymbol{\gamma}^l_0, \boldsymbol{\gamma}^l_{sat,10})}{\partial\boldsymbol{\theta}_{head}}\right)^{\#} \phi_s(\boldsymbol{\gamma}^l_0, \boldsymbol{\gamma}^l_{sat,10}) && (4.6)\\
&= -\left(\frac{\partial\phi_s(\boldsymbol{\gamma}^l_0, \boldsymbol{\gamma}^l_{sat,10})}{\partial\boldsymbol{\gamma}^l_{sat,10}} \frac{\partial\boldsymbol{\gamma}^l_{sat,10}}{\partial\boldsymbol{\theta}_{head}}\right)^{\#} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) && (4.7)\\
&\approx -\left(-\boldsymbol{\epsilon}^T\mathbf{I}_{2\times 2}\right)^{\#} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) = \frac{1}{2}\boldsymbol{\epsilon}, && (4.8)
\end{aligned}$$

where the Jacobian capturing how pan/tilt displacements effect the view of headings perceived in the robot's left camera is approximately the identity matrix $\mathbf{I}_{2\times 2}$ due to the fact that the pan and tilt axes intersect at the camera's optical center. Given this physical relationship, the perceived heading errors in the image plane correspond directly to variations in the robot's pan and tilt configuration.

The quiescence of TRACK is rewarding according to the affordance discovery motivator because $\boldsymbol{\gamma}^l_{sat,10} \in \Omega_{\sigma(dir)}$. We see how TRACK(SAT-10) regulates the position



(a) (b)

Figure 4.3. Frames (a) and (b) show the image from Dexter's camera before and after the execution of TRACK(SAT-10). The highly-saturated region that it moves to center in its image plane is marked in a green ellipse.

of the observed highly-saturated feature on the image plane in Figure 4.3. These pictures show the view from Dexter's camera before and after the execution of the track controller. We see how the saturated region corresponding to the yellow ball (circled in green) begins in the periphery of the image, but ends up in the center upon controller quiescence, such that $\boldsymbol{\gamma}_0^l = \boldsymbol{\gamma}_{sat,10}^l$.

SEARCH constructs and reduces a feedback error from two signals—the current value of the head's pan/tilt angles $\boldsymbol{\theta}_{head}$, and a head reference posture $\boldsymbol{\theta}_{head,ref}$—according to the gradient of the potential function ϕ_s, This controller is defined as:

$$\text{SEARCH}(\text{SAT-10}) \triangleq c(\phi_s, (\boldsymbol{\theta}_{head,ref}, \boldsymbol{\theta}_{head}), \boldsymbol{\theta}_{head}). \tag{4.9}$$

$\boldsymbol{\theta}_{head,ref}$ is sampled from a probabilistic model of priors for the search target in terms of pan and tilt head angles,

$$\boldsymbol{\theta}_{head,ref} \sim Pr(\boldsymbol{\theta}_{head} | \boldsymbol{\gamma}_{sat,10}^l = \boldsymbol{\gamma}_0^l). \tag{4.10}$$

$\boldsymbol{\theta}_{head,ref} \in \mathcal{C}_2$ is thus sampled from a distribution of head configurations where the environment is likely to have afforded Track(Sat-10) quiescence in the past. It is easy to see that $Pr(\boldsymbol{\theta}_{head}|\boldsymbol{\gamma}^l_{sat,10} = \boldsymbol{\gamma}^l_0)$ is closely related to $Pr(\tau|\sigma, reward)$ for the Track controller that is updated at Line 19 of the Accommodate() procedure. These distributions are thus used interchangeably in this program.

Search orients the head to postures where the target saturation is *likely* to be found on the left camera image plane. It is not rewarding by the affordance discovery intrinsic motivator because the reference for the action is not derived from the environment, but rather from probabilistic models of past environments. Note that similar models capturing where other features in Ω_γ occur could be used to inform many additional Search actions.

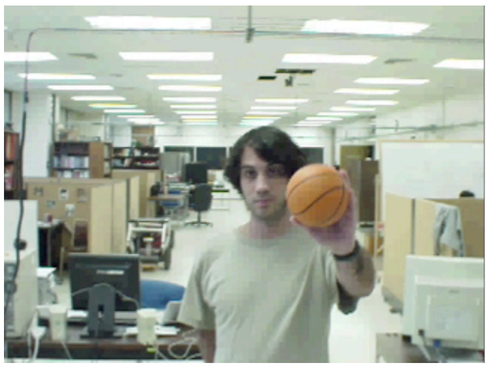
In conjunction, the Search(Sat-10) and Track(Sat-10) controllers support the construction of the state space $\mathcal{S}_{st}$ where each state $\mathbf{s} \in \mathcal{S}_{st}$ is evaluated such that $\mathbf{s} = (p_{search}\ p_{track})$, and two possible action sets (dropping the parameter values for notational simplicity) $\mathcal{A}^1_{st} = \{$Search, Track$\}$ and $\mathcal{A}^2_{st} = \{$Search, Track, Search $\triangleleft$ Track, Track $\triangleleft$ Search$\}$, depending on whether co-articulation is allowed. The following experiment compares polices for asserting the track affordance using each of the candidate action sets. Dexter learns policies for SearchTrack according to the procedure shown in Algorithm 1. Ten trials of 50 learning episodes were performed for each experiment. At the beginning of each trial, $\Phi(s, a)$ was initialized to zero. The average reward per state transition was recorded for each episode and averaged over the trials. Each episode ended when a rewarding event occurred (i.e., Track quiesced). The distribution $Pr(\boldsymbol{\theta}_{head}|\boldsymbol{\gamma}^l_{sat,10} = \boldsymbol{\gamma}^l_0)$ was estimated as a non-parametric distribution with a small Gaussian smoothing kernel and was initialized at the beginning of each trial to uniform.

In half of all of the episodes, the experimenter presented a highly-saturated object at a position in front of the robot (in the camera's initial field of view) as seen in the image from Dexter's camera in Figure 4.4(a). In the other half of the training episodes, no object was presented to the robot. However, other saturation cues were available in the robot's environment if the robot "looked around" (e.g., toward the window to its left, as seen in Figure 4.4(b)).

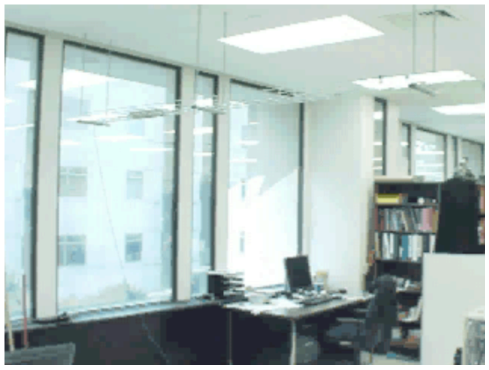
Figure 4.4(c) shows one of the learned non-parametric distributions at the end of a trial for SEARCHTRACK estimating $Pr(\boldsymbol{\theta}_{head}|\boldsymbol{\gamma}^{l}_{sat,10} = \boldsymbol{\gamma}^{l}_{0})$. The large peak in the center of the robot's pan range corresponds to locations where the experimenter held objects in front of the robot (Figure 4.4(a)), the smaller peak corresponds to the configuration where the saturated window region could be seen (Figure 4.4(b)). This distribution reflects the robot's knowledge at the end of the trial concerning were TRACK affordances occur. This model can thus be used as a prior by the SEARCH controller to inform future executions to orient the robot to efficiently achieve reward by the affordance discovery motivator.

For the case in which the action set did not include co-articulated actions, the robot learned a policy that resulted in the transitions shown in Figure 4.5(a). From the start state (XX), the policy chooses action TRACK. If a saturation stimulus is absent, the state transitions to (X$-$), thereafter entering a loop that iteratively searches using SEARCH and then tests for stimuli using TRACK. When the saturation cue is detected, the robot enters state (X0) and then continues to execute TRACK until it quiesces in state (X1) and receives reward. The average reward graph for these experiments is shown in red in Figure 4.6(a). It appears that a stable policy is learned after about 10 episodes.

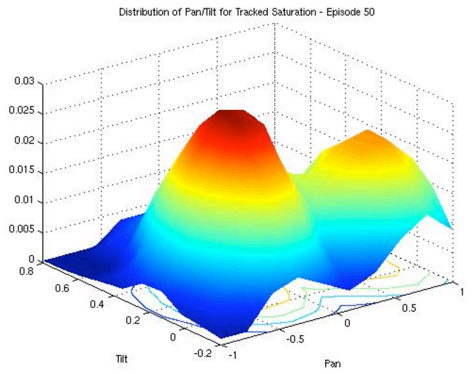
For the experiment in which the action set included co-articulated actions, the robot learned the policy that resulted in the transitions shown in Figure 4.5(b). This policy allows the robot to "interrupt" the search process if a saturated stimuli appears



(a)

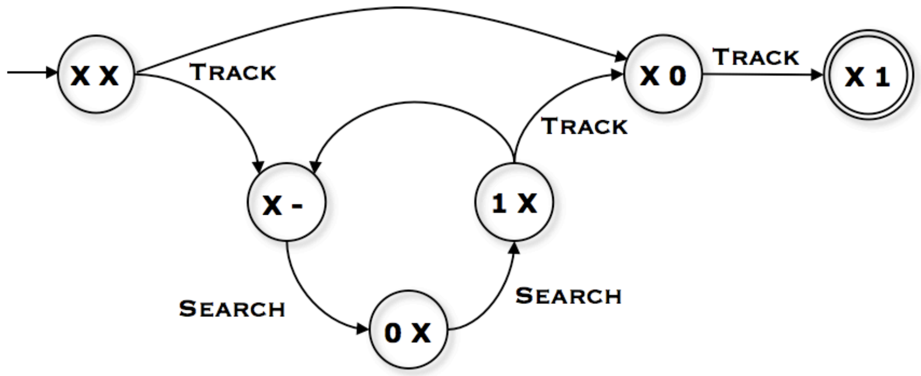


(b)

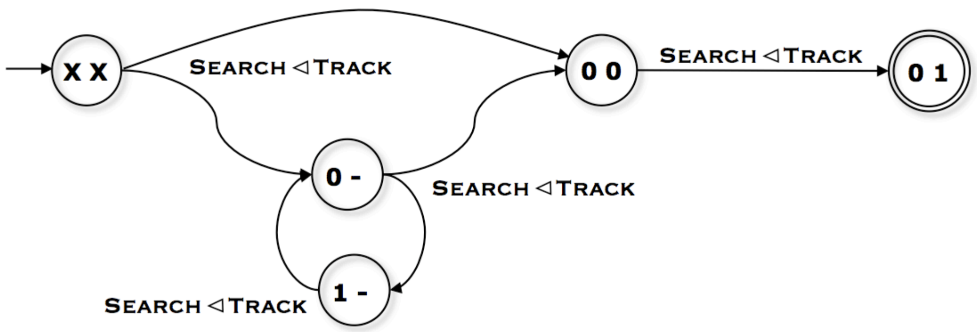


(c)

Figure 4.4. Frame (a) shows an image from Dexter's left camera when a saturated object is presented in front of the robot. Frame (b) shows an image from Dexter's camera while viewing window to its left. Frame (c) shows the non-parametric distributions after 25 training episodes summarizing the pan/tilt configurations where Dexter expects to observe this range of saturation in the visual feedback.

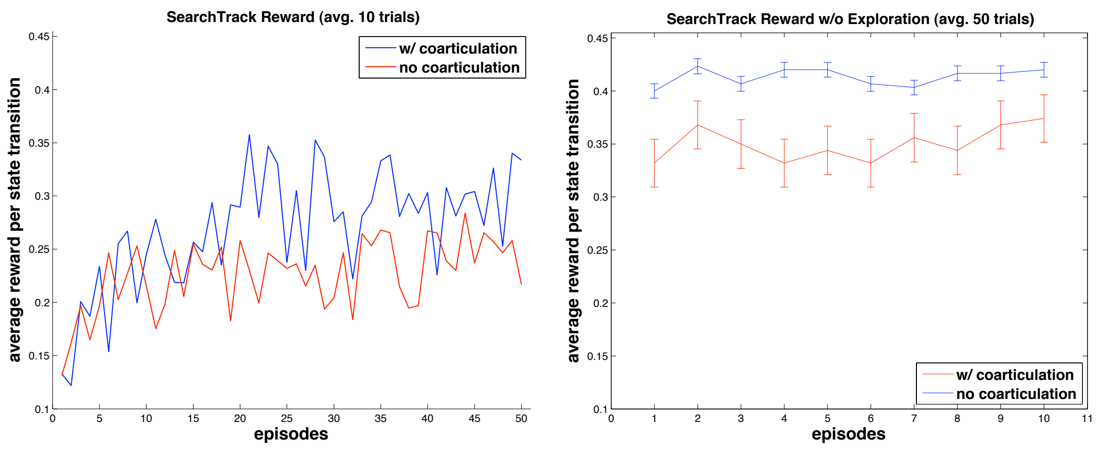


(a)



(b)

Figure 4.5. SearchTrack transition diagrams for the policies acquired on Dexter in the first stage of learning. Transitions are shown if they occurred with a probability greater than 20%. The diagrams are characterized by states $\mathbf{s} \in \mathcal{S}_{st}$ where $\mathbf{s} = (p_{search}\ p_{track})$. The policy employs Track first, and Search is chosen only when no stimuli is immediately present. The state diagram in (a) shows the policy when only single actions are allowed in the action set. The state diagram in (b) shows the policy when composite actions are allowed.



(a) (b)

Figure 4.6. Reward plots for the SEARCHTRACK policies learned on Dexter in stage 1. Plot (a) shows the learning curves for each experiment, averaged over 10 trials of 50 episodes each with an exploration rate of 20%. Plot (b) shows the reward for the learned policies over 50 trials of 10 additional episodes in which there is no exploration and the stimuli is guaranteed to be found from a search. The error bars shows a clear statistical advantage of the co-articulated policy (blue) over the sequential action policy (red).

as the robot moves to the reference location. The policy suggests using action Search $\triangleleft$ Track in all states, allowing the higher priority action, Track, to dictate the robot's behavior if the stimuli is present, while performing successive searches in the track controller's nullspace, if it is not. The resulting co-articulated policy requires fewer state transitions on average than the sequential search-then-track policy seen in Figure 4.5(a). This is apparent by the fact that the blue line (the average reward for the co-articulation policy) in Figure 4.6(a) seems to be slightly higher, on average, than the red line (the average reward for the sequential policy). However, because there are more actions in the action set (four, as opposed to two), it takes about twice as long for the robot to learn a stable policy.

Due to the high level of exploration in the learning episodes (20%), as well as the stochasticity in the search process, the improvement of the co-articulated policy over the single-action policy is not statistically significant, despite an apparent advantage seen in Figure 4.6(a). To show that the co-articulated policy is in fact better, a simulation experiment was performed in which both learned policies were run under similar environmental conditions, except that (1) exploration was turned off, and (2), in the 50% of the cases where the robot had to employ Search to find the saturation stimuli, it is guaranteed to find it on the first try. Under these conditions, fifty trials of ten episodes were performed, and the resulting average reward graphs are shown in Figure 4.6(b), along with the data's variance. From this graph it is clear that the co-articulated policy is in fact better (i.e., it achieves more reward per state transition) than the sequential policy for the affordance discovery reward function.

4.2.2 TactileProbe

SearchTrack provides a general *orient-response* control sequence in which the robot reacts to stimuli in a controlled way if that stimuli is present, or searches for it if it is not. Although this behavior was conveyed to Dexter in the context of visual tracking, the strategy is applicable to other domains in which a control reference must first be uncovered before it can be controlled. For example, this strategy can allow a robot to perform searches with each of its fingers to find tactile contact. We call such a behavior TactileProbe and teach it to Dexter in a second stage of learning. TactileProbe employs two primitive control actions. The first is a Touch controller, similar to that of Equation 3.34, constructed to control a small reference force on each of the three fingers of Dexter's right hand so that the robot can grab simple objects,

$$\textsc{Touch}(\textsc{RightHand}) \triangleq c(\phi_s, (\mathbf{f}_{r,ref}, \mathbf{f}_{r,hand}), \boldsymbol{\theta}_{r,hand}), \tag{4.11}$$

where

$$\mathbf{f}_{r,hand} = \begin{bmatrix} \mathbf{f}_{r,1} \\ \mathbf{f}_{r,2} \\ \mathbf{f}_{r,3} \end{bmatrix}, \quad \text{and} \quad \mathbf{f}_{r,ref} = \begin{bmatrix} \mathbf{f}_{ref,1} \\ \mathbf{f}_{ref,2} \\ \mathbf{f}_{ref,3} \end{bmatrix}$$

and $\mathbf{f}_{ref,i}$ is a reference vector of $0.2N$ pointing in the inward (palm) direction of finger i, rotated into the world frame. Because Touch regulates forces observed on the fingertips derived from contact with the environment it is rewarding by the affordance discovery motivator. The second action is a Search action in the tactile domain, and is defined as follows:

Search constructs a feedback error from two signals: the configuration variables of the robot's right-hand $\boldsymbol{\theta}_{r,hand}$, and a reference posture $\boldsymbol{\theta}_{r,ref}$ drawn from an internal model. In this experiment, this reference, $\boldsymbol{\theta}_{r,ref} \in \mathcal{C}_4$, is sampled from a distribution of hand configurations where a force response is felt on all three of the robot's right hand fingers, where

$$\boldsymbol{\theta}_{r,ref} \sim Pr(\boldsymbol{\theta}_{r,hand} \mid ||\mathbf{f}_{r,i}|| > \epsilon, i = 1,2,3), \tag{4.12}$$

where ϵ is a small force reference ($0.2N$). SEARCH reduces the error between $\boldsymbol{\theta}_{r,ref}$ and $\boldsymbol{\theta}_{r,hand}$ by moving the robot's fingers according to the gradient of the potential function ϕ_s, such that:

$$\textsc{Search}(\textsc{RightHandForces}) \triangleq c(\phi_s, (\boldsymbol{\theta}_{r,ref}, \boldsymbol{\theta}_{r,hand}), \boldsymbol{\theta}_{r,hand}). \tag{4.13}$$

SEARCH orients the hand to postures where contact is likely to be found—it increases the probability that force cues will be found on the fingertips. Note that a similar control action could be constructed for the robot's right hand.

SEARCH(RIGHTHANDFORCES) and TOUCH(RIGHTHAND) provide a 2-predicate state vector $\mathbf{s} \in \mathcal{S}_{tp}$ where $\mathbf{s} = (p_{search}\ p_{touch})$ with four possible actions $\mathcal{A}_{tp} = \{\textsc{Search}, \textsc{Touch}, \textsc{Search} \triangleleft \textsc{Touch}, \textsc{Touch} \triangleleft \textsc{Search}\}$. A learning stage was conducted in which Dexter acquired a policy for TACTILEPROBE by means of the ACCOMMODATE() procedure shown in Algorithm 1. A single trial of 25 learning episodes was performed. Each episode ended when a rewarding event occurred (i.e., the touch controller quiesced). At the end of each episode, the one-dimensional postures of the fingers on the robot's right hand were added to three Gaussian distribution models (one for each finger) so that this knowledge can be used in future episodes to provide samples according to Equation 4.12.

During the learning episodes, the experimenter provided various balls and boxes to the robot. These objects ranged in size from $7cm$ in radius to $12cm$ in radius. The object was held in place until the robot successfully completed the TACTILEPROBE episode, resulting in a three-fingered touch.

Not surprisingly, the resulting policy for TACTILEPROBE learned after 25 episodes resembles that of SEARCHTRACK, substituting the TOUCH action for the TRACK

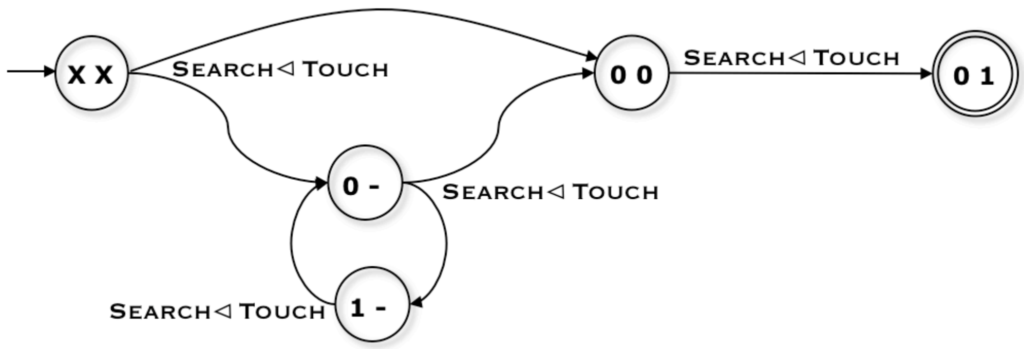


Figure 4.7. The transition diagram for the policy learned for TACTILEPROBE using the robot's right hand, characterized by the state vector $\mathbf{s} = (p_{search}\ p_{touch})$. Transitions are shown if they occurred with a probability greater than 20%. The policy employs TOUCH first, and SEARCH is chosen only when no stimuli is immediately present.

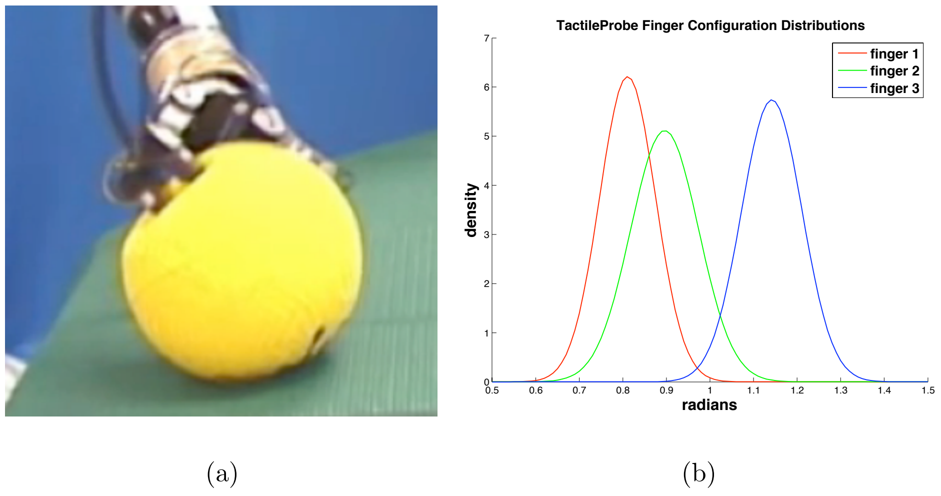


Figure 4.8. Panel (a) shows an image of Dexter's hand after a rewarding TACTILEPROBE episode. Panel (b) shows the Gaussian distributions after 25 training episodes estimating $Pr(\boldsymbol{\theta}_{r,hand} \mid ||\mathbf{f}_{r,i}|| > \epsilon, i = 1, 2, 3)$ and summarizing the finger configurations where Dexter expects to make contact.

action. The transition diagram for this policy is shown in Figure 4.7. The resulting Gaussian distributions of finger configurations learned after the trial are shown in Figure 4.8. TactileProbe provides a useful behavior to allow a robot to position its fingers in contact with the object that it wishes to grab. Variations of this behavior could also be used to perform grasp *pre-shaping* to increase the likelihood and the efficiency of more sophisticated grasping actions. We will see in the next chapter how TactileProbe can be generalized to allow for a robot to probe with both its fingers and its hands to seek out controllable tactile stimuli.

This similar structure of SearchTrack and TactileProbe suggest a methodology in which abstract—or *declarative*—policies can be transfered to different *procedural* contexts by re-allocating the resources of control actions. The subject of program generalization will be addressed in Chapter 5. We will restrict our discussion to a methodology in which we allow only re-allocations that maintain the original *type* specifications of the original program. We now turn to examine how control basis programs, like SearchTrack and TactileProbe, can be used as single actions in hierarchical programs that seek to uncover further environmental affordances.

4.3 Hierarchical Composition

As described in Chapter 2, value functions provide a natural hierarchical generalization of potential functions for discrete state/action spaces. Furthermore, performing greedy ascent on a value function will lead an agent toward maximally rewarding states where $\dot{\Phi}=0$. The basis for hierarchy in the control basis framework depends on the abstraction of sensorimotor programs, with all the internal state they require, in terms of the single, four-predicate state logic of Figure 4.2. Although a program can have a significant amount of internal structure, the hierarchical learning agent views this program as a single, temporally extended control action, whose state is represented the same way as that of all other control actions. Control basis programs are similar to reinforcement learning *options* (Sutton et al., 1999) with their own state/action spaces.

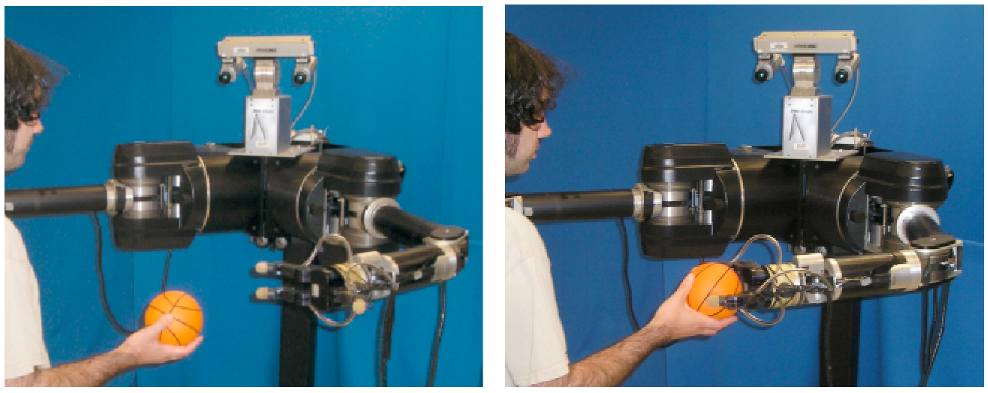
In the remainder of this chapter, we present the results of a number of additional learning stages in which Dexter used the affordance discovery motivator to acquire hierarchical control basis programs. In each stage, Dexter learned a policy to uncover a new affordance using the ACCOMMODATE() procedure. Each of these programs employs at least one other control basis program hierarchically. When a program is used as a hierarchical single action in a new learning program, we will treat it as having "habituated," and will fix its policy in place. Furthermore, these programs will provide no reward from the intrinsic motivator.

4.3.1 ReachTouch

The first hierarchical program Dexter learned asserts the TOUCH affordance by combining SEARCHTRACK with arm control tasks. We call this program REACH-TOUCH, and it allows Dexter to engage objects that are not physically presented to the robot (as they were in the TACTILEPROBE example). This program uses the SEARCHTRACK program hierarchically to find highly-saturated visual features that it can triangulate, reach to, and touch with its right hand using REACH(SAT-10,RIGHTARM) and TOUCH(RIGHTHAND) actions.

A learning stage for Dexter was constructed to enable Dexter to learn a policy for REACHTOUCH. The robot was provided with three actions SEARCHTRACK, REACH(SAT-10,RIGHTARM), and TOUCH(RIGHTHAND). These three actions construct an action space $\mathcal{A}_{rt} = \{$SEARCHTRACK, REACH, TOUCH, REACH $\triangleleft$ TOUCH, TOUCH $\triangleleft$ REACH$\}$[3] and a state vector $\mathbf{s} \in \mathcal{S}_{rt}$ where $\mathbf{s} = (p_{st}\ p_{reach}\ p_{touch})$ and p_{st} is the state predicate value of the entire SEARCHTRACK program. Quiescence of the

[3]Co-articulated actions between primitives and programs were not allowed. How to co-articulate discrete state/action policies is an open research question, but see (Rohanimanesh & Mahadevan, 2005) for one possible approach.



(a) (b)

Figure 4.9. Panel (a) shows Dexter reaching to a highly-saturated object and panel (b) shows Dexter holding that object with the force controllers in the hand.

Touch(RightHand) controller is rewarding by the affordance discovery motivator because it signifies an affordance in the environment. One trial of 25 learning episodes was conducted in this learning stage to allow Dexter to learn the ReachTouch program using the Accommodate() procedure.

During approximately half of the learning episodes, a human presented highly-saturated objects to Dexter, as seen in Figure 4.9(a). For the other half of these episodes, the human delivered a highly-saturated object in front of the robot in a location initially out of view of the robot's two cameras. During a small number of episodes (10%) no object was presented to the robot. In the cases where the object was presented to the robot, it was able to grab it using Touch, as seen in Figure 4.9(b).

By the end of the learning trial, Dexter had learned a policy for ReachTouch to achieve reward by uncovering Touch affordances. This transition diagram showing the most likely state transitions that occur under the greedy policy for this program is shown in Figure 4.10. The robot starts out in state (XXX) and chooses the action Reach $\triangleleft$ Touch. If the object is initially in the field of view, the state transitions

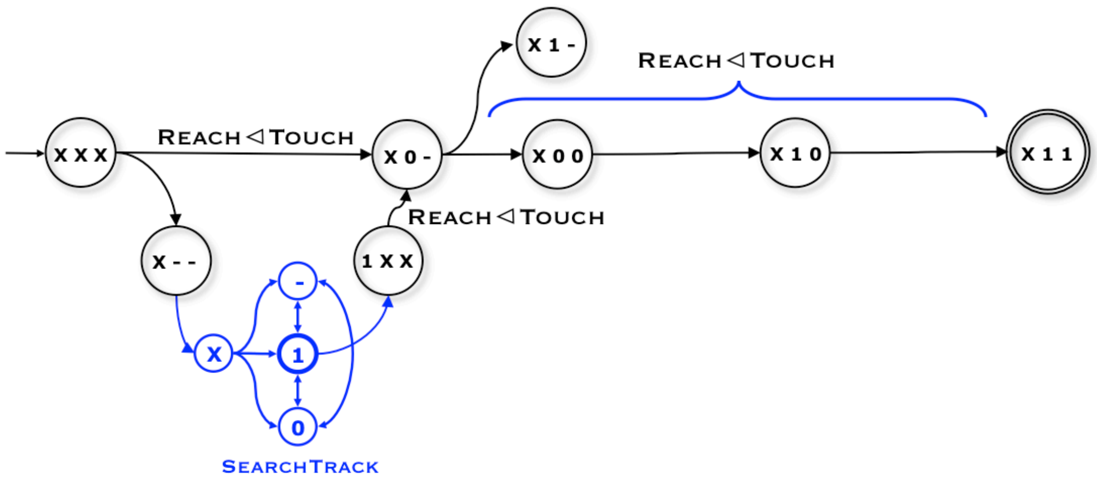


Figure 4.10. This diagram shows the transition diagram for the policy learned for ReachTouch after 25 episodes. The state vector is $\mathbf{s} = (p_{st}\ p_{reach}\ p_{touch})$ where p_{st} is the state value of the SearchTrack program. The hierarchical use of SearchTrack is indicated by the abstract transition icon introduced in Figure 4.2.

to (X0−) in which there is a reach goal, but no reference to Touch. If the object is not initially in the robot's field of view, the robot transitions to state (X−−), from which it employs SearchTrack to find a reach goal. When a goal is found, Reach $\triangleleft$ Touch is tried once again, resulting in a transition to state (X0−). From this state, the robot continues executing Reach $\triangleleft$ Touch until it either reaches state (X11), in which reward is received by the intrinsic motivator, or the reach action converges without bringing the robot's hand into contact with an object, resulting in a transition to state (X1−). This latter case occurred in the small number of episodes in which the robot reached toward visual stimuli that did not pertain to an object within its reachable workspace (e.g., the window in Figure 4.4(b)).

The simple policy for ReachTouch provides a robust way for Dexter to turn visual cues into spatial and tactile cues it can engage in different ways. We will next show two programs in which it is used hierarchically to accumulate still more reward from the affordance discovery motivator.

4.3.2 VisualInspect

In the next learning stage, Dexter acquired a policy for picking up an object and inspecting it for additional visual Track affordances that might not be initially visible, either because they are too small or because they are initially on a side of the object facing away from the robot. We call this program VisualInspect because it employs a controller similar to that defined in Equation 3.40 to bring highly-saturated objects grasped in the robot's right hand to places where the robot optimizes visual acuity. As new visual features become present through this process of "inspection," the robot can move to track them and receive additional reward. A specific training context is constructed to teach Dexter how it might be able to inspect objects for new Track affordances.

Three actions are employed in VisualInspect. The first is the hierarchical ReachTouch program that can grab objects. The second is the Localizability controller that maximizes the localizability metric to bring objects grasped by the robot to the stereo sweet-spot. This controller is defined as:

$$\textsc{Localizability}(\textsc{Sat-10},\textsc{RightArm}) \triangleq c(\phi_l, \boldsymbol{\gamma}_{sat,10}, \mathbf{x}_{r,arm}), \tag{4.14}$$

where $\boldsymbol{\gamma}_{sat,10} = (\boldsymbol{\gamma}^l_{sat,10}, \boldsymbol{\gamma}^r_{sat,10})$. Because this controller is used to inspect grasped objects, it is only defined when the object is held in the robot's hand. The third controller is a visual tracking controller Track that samples and tracks additional hues on the robot's left camera measured by $\boldsymbol{\gamma}^l_{hue,i}$, where $i \in [1, 10]$. This controller tracks visual features in the environment and is thus rewarding by the affordance discovery motivator.

These actions construct the action set for VisualInspect, $\mathcal{A}_{vi} = \{$ReachTouch, Localizability, Track, Track $\triangleleft$ Localizability, Localizability $\triangleleft$ Track$\}$, with states $\mathbf{s} \in \mathcal{S}_{vi}$ where $\mathbf{s} = (p_{rt}\ p_{loc}\ p_{track})$ and p_{rt} is the state of the ReachTouch program. Dexter explored this state and action space for 25 episodes using the Ac-

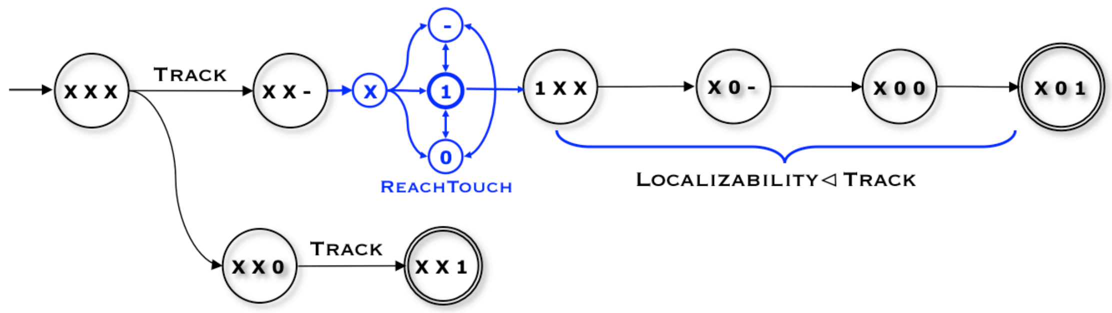
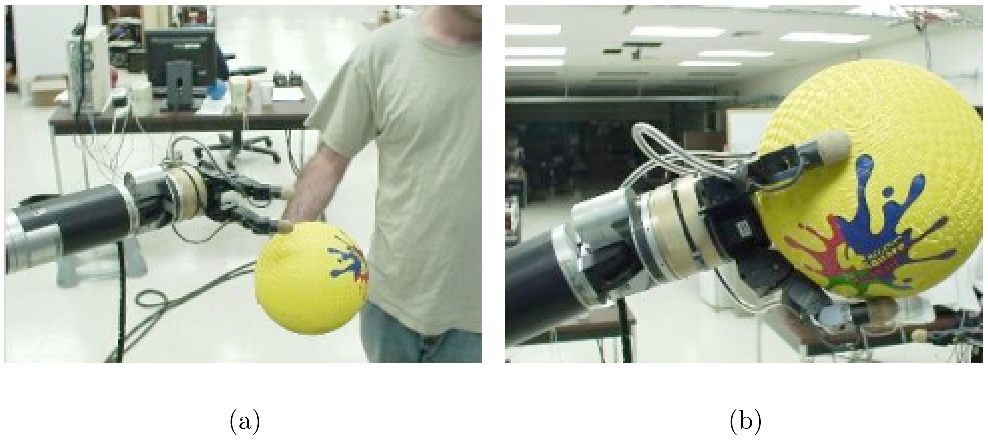


Figure 4.11. This diagram shows the transition diagram for the policy learned for VISUALINSPECT after 50 episodes. The state vector $\mathbf{s} = (p_{rt}\ p_{loc}\ p_{track})$, where p_{rt} is the state value of the REACHTOUCH program.

COMMODATE() procedure, receiving reward when the environment affords quiescence of the TRACK.

During the 25 episodes, a highly-saturated yellow object with a small blue feature on one of its sides was presented to the robot as seen in the left camera image shown in Figure 4.12(a). Sometimes the blue feature was initially facing the robot (25% of the time), sometimes it was facing away (75% of the time). The transitions that occur under the policy learned after these training episodes is shown in Figure 4.11. The robot begins by activating the TRACK controller. In the cases where an additional color feature is initially visible (e.g., the blue patch on the ball) the robot will track that feature through state (XX0) and receive reward in state (XX1). In the cases where an addition color feature is not initially visible, the robot transitions to state (XX−). From this state, the policy dictates that the robot use the REACHTOUCH program to grab hold of a highly-saturated object and then visually condition its appearance with the composite action LOCALIZABILITY $\triangleleft$ TRACK. As this action executes, it brings any additional hue features on the object into view, causing a transition from state (X0−) to state (X00). Continuing this action, the robot will track the new feature and receive reward when it arrives in state (X01). The appear-



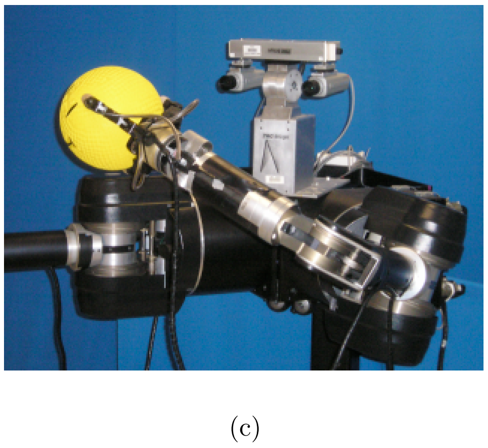


Figure 4.12. Frames (a) and (b) show Dexter's left camera image before and after the execution of the control law LOCALIZABILITY $\triangleleft$ TRACK. Frame (c) shows Dexter after the execution of that law.

ance of Dexter as it performs this sequence of actions for the yellow ball with the blue patch is seen in Figures 4.12(b) and 4.12(c).

4.3.3 BimanualTouch

In the final stage of learning we present in this chapter, Dexter acquired a program called BIMANUALTOUCH. This program allows the robot to get more tactile reward from an object by picking it up with one hand and bringing it into contact with the other. The ablily to transfer objects between hands is a useful skill for a bimanual mechanism because it overcomes constraints on the reachable workspace of any single arm. For example, consider a "pick-and-place" task in which an object far to the robot's left must be placed far to its right. In this situation, the robot can pick up the object with its left hand, transfer it to its right hand, and deliver it to the place goal. Although there may be multiple, redundant ways for a robot to bring an object into contact with both of its hands (e.g., two concurrent ReachTouch programs), we provide a solution here that takes advantage of the symmetric morphology of Dexter. Because of the particular structure of Dexter's body, optimizing Yoshikawa's manipulability metric (Yoshikawa, 1985) for both arms simultaneously causes the robot's hands to come close together. If the robot performs this action while maintaining a grab on an object with one hand, it will position the robot to discover an additional Touch affordance with its other hand.

We allow Dexter to explore three actions. The first is the ReachTouch program, used by BimanualTouch hierarchically. The second is a composite controller called ConditionedHold(RightHand) that optimizes manipulability on both arms concurrently subject to maintaining a hold on the object using the Touch(RightHand) controller. ConditionedHold(RightHand) is defined as

$$\begin{aligned}\textsc{ConditionedHold}(\textsc{BothArms},\textsc{RightHand}) &\triangleq \\ &\textsc{Manipulability}(\textsc{BothArms}) \;\triangleleft\; \textsc{Touch}(\textsc{RightHand}),\end{aligned}$$

where

$$\textsc{Manipulability}(\textsc{BothArms}) \;\triangleq\; c(\phi_m, \boldsymbol{\theta}_{arms}, \boldsymbol{\theta}_{arms}). \tag{4.15}$$

This composite controller will ensure that a hold is maintained on the object while the robot conditions the arm configurations. This controller is undefined when the Touch controller does not have a reference (the hand is not in contact with an object).

The third controller used in BIMANUALTOUCH is another TOUCH controller, this time parameterized by the robot's *left*-hand finger forces and motor variables,

$$\textsc{Touch}(\textsc{LeftHand}) \triangleq c(\phi_s, (\mathbf{f}_{l,ref}, \mathbf{f}_{l,hand}).\boldsymbol{\theta}_{l,hand}), \tag{4.16}$$

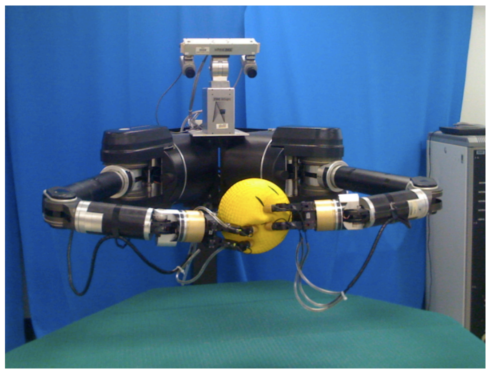
defined similarly as the TOUCH(RIGHTHAND) controller defined in Equation 4.11. In this program, TOUCH(LEFTHAND) is rewarding via the intrinsic motivator. These controllers construct the action set $\mathcal{A}_{bt}$ = {REACHTOUCH, CONDITIONEDHOLD, TOUCH, TOUCH $\triangleleft$ CONDITIONEDHOLD, CONDITIONEDHOLD $\triangleleft$ TOUCH} and state vector $\mathbf{s} \in \mathcal{S}_{bt}$ where $\mathbf{s} = (p_{rt}\ p_{hold}\ p_{touch})$ and p_{rt} is the state of the REACHTOUCH program and p_{hold} is the state of the CONDITIONEDHOLD composite controller. The robot explored this state and action space for 25 episodes using the ACCOMMODATE() procedure, learning a new policy to uncover left-handed TOUCH affordances.

During the 25 learning episodes, a highly-saturated object was placed in various locations on the right side of the robot (not always in the robot's initial field of view). The transition diagram for the policy learned after these episodes is shown in Figure 4.13. The robot begins by invoking REACHTOUCH—which in turn invokes SEARCHTRACK if the object is initially out of view—to grab the object with its right hand. After REACHTOUCH completes in state (1XX), the composite action CONDITIONEDHOLD $\triangleleft$ TOUCH is run until completion—passing through state (X0$-$) as the hands are brought together, state (X00) as the left hand comes into contact with the object, state (X10) as the manipulability action completes, and state (X11) as the left-handed TOUCH quiesces (and reward is received).

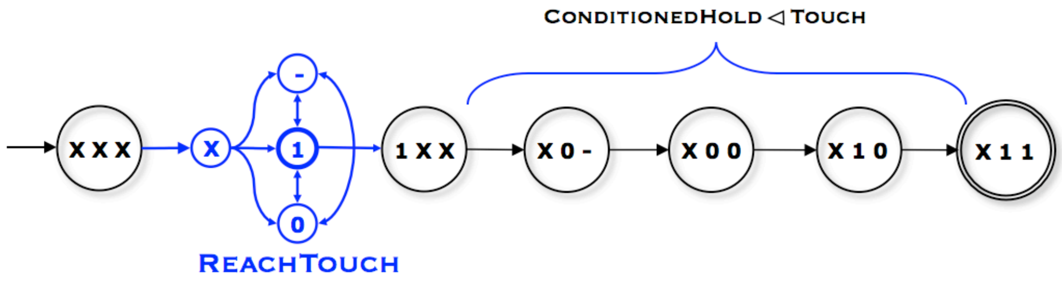
4.4 Discussion

This chapter introduced a novel state and reward representation useful for organizing a robot's resources into behavioral programs. An intrinsic reward function for affordance discovery was introduced to reward the robot for creating controllable interactions with its environment. This reward function is designed to encourage a robot to discover where and how it can use its control actions to engage the world in a way that is grounded in the robot's sensory and motor subsystems.

Policies for each of the programs in this chapter were learned in stages using reinforcement learning in a small number of episodes. Each stage was specifically designed by the human "programmer" to make a particular rewarding event conspicuous. During these stages, however, the robot explored its state/action spaces to learn a policy to achieve that reward. The focus in these learning stages was not to provide a completely "hands-off" approach to robot learning, but, on the contrary, to demonstrate how a robot programmer can easily teach a robot a useful skill by making certain artifacts conspicuous. In the next chapter, we will show how a robot can autonomously generalize and adapt these hierarchical programs to new contexts to achieve reward in more complex situations.



(a)



(b)

Figure 4.13. Frame (a) shows Dexter after the completion of BIMANUALTOUCH. Frame (b) shows the transition diagram for the learned policy after 25 episodes. Transitions are shown if they occurred with a probability greater than 20%. The state vector is $\mathbf{s} = (p_{rt}\ p_{hold}\ p_{touch})$, where p_{rt} is the state of the REACHTOUCH program and p_{hold} is the state of the CONDITIONEDHOLD composite controller.

CHAPTER 5
SKILL GENERALIZATION

The behavioral programs discussed in the last chapter were conveyed to Dexter by restricting the environmental context and the control expressions that the robot could explore. Such *behavioral scaffolding* is an appropriate way for a human teacher to bootstrap intrinsically motivated behavior, but it is necessary to consider how the robot can transfer what it has learned to more general situations. I argue that any robot designed to provide dexterous solutions for a wide variety of tasks must be able to adapt its behavioral knowledge to new contexts, different from those in which that knowledge was initially acquired.

In this chapter, I address how a control basis program can be transformed into a unit of behavior called a *schema* to provide dexterous contingency plans in a variety of environmental contexts. This generalization is accomplished by factoring existing control programs learned by the affordance motivator into *declarative* and *procedural* components (Hart et al., 2008a). The declarative structure of a program— capturing abstract information concerning which combination of objectives are required to meet a behavioral goal—can be transfered to different contexts. The procedural structure examines the environmental conditions under which reward is received and dictates how resources should be allocated to the (abstract) declarative objectives at run-time.

Specifically, our approach allows a robot to find the statistically reliable parameterizations of control basis actions that maintain the original typing constraints and transition dynamics of policies that achieve reward. After a brief discussion of schema and related computational approaches, we will examine how control basis programs can be factored into abstract policies that can be re-allocated with different sensory and motor resources in different contexts. The performance gains for this generalization technique are demonstrated in simulation and on Dexter.

5.1 Background

Algorithms that allow for the autonomous acquisition of general robot behavior have continued to pose a significant challenge in artificial intelligence research. Two recent trends in the literature have addressed the ability to transfer skills learned in one context to another and to create generalizable representations from context-specific experience to facilitate transfer. Techniques for skill transfer and generalization enable the re-use of knowledge and are often presumed to accelerate learning in new, related tasks. I argue that a key limiting factor for skill transfer in robot systems has been a tendency to approach generalization from a task-level perspective; finding the means to take strategies learned for one task and transfer it to other related, but human-designated tasks. In contrast, I propose that a robot should autonomously seek to build task-independent—*common sense*—strategies for manipulating the world in increasingly complex and general ways. I hypothesize that such common sense strategies can only be built from the bottom up as a robot learns how to adapt—or *re-parameterize*—its existing policies to novel situations.

In this chapter, two criteria for re-parameterization of control programs are investigated. The first requires that procedural choices adhere to the transition dynamics of the initial program that is being generalized. Such an approach has been used in reinforcement learning systems by Ravindran (2004) and has been applied to control basis programs by Platt (2006). The second technique imposes re-parameterization constraints by maintaining type compatibility among sensory and motor resources, and is a formalization of the techniques presented in previous work presented in Hart et al. (2005).

5.1.1 Piagetian Schema

The concept of generalizable units of behavior is related to Piaget's discussion of sensorimotor *schema* (Piaget, 1952). Piaget suggested that schema are formed to facilitate new agent-environment interactions through a process of *accommodation* and that existing schema generalize to new experiences through a process of *assimilation*. This chapter addresses how to combine these processes into a unified computational framework. Declarative strategies are learned in an accommodation phase where only one procedural choice is relevant. During a potentially open-ended assimilation phase, these strategies are used to bootstrap learning in a variety contexts where many procedural choices may exist.

Arbib introduced the notion of *perceptual* and *motor* schema as a theory of cognitive organization that could be useful in artificial intelligence (Arbib, 2003). Computational schema that provide contingency plans to accomplish a desired behavior have been demonstrated in rule-based control systems (Nilsson, 1994), and empirical cause-and-effect systems in discrete (Drescher, 1991) and continuous domains that can be explored using computational mechanisms for active learning and intrinsic motivation (Mugan & Kuipers, 2007; Mugan & Kuipers, 2008).

5.1.2 Computational Approaches

In the machine learning literature, transferring skills from one context to another has attracted recent interest. Wilson et al. (2007) present an approach for learning shared structures in Markov Decision Processes that can be applied to multiple tasks. Mehta et al. (2005) assumes the reward functions to be linear combinations of rewarding features with only the feature weights varying among otherwise fixed MDPs. Ravindran (2004) exploits graph homomorphisms in an MDP to learn general policies in an abstract space. These approaches exploit the underlying structure in a large class of MDPs, but are hard to transfer to real robots because they require large amounts of training data.

In contrast, Cohen et al. (2007) provided a promising example of how the dynamics of control actions can be used to learn schema that can be transfered to new situations. Similarly, the control basis approach also makes use of low-level controllers and their dynamics to learn robot-specific knowledge structures that can be generalized to accommodate context-specific contingencies (Coelho & Grupen, 1997; Huber & Grupen, 1997). In this work, general control policies for grasping and mobility are extended to assimilate new contexts (e.g., 2-fingered grasps vs. 3-fingered grasps) to provide a greater wealth of behavior.

The framework presented in this chapter is similar to Konidaris' *agent-space* options (Konidaris & Barto, 2007). Konidaris and Barto introduce the *problem space* and the *agent space*. The problem space is related to the procedural structure of control basis programs in that it captures information concerning the run-time context of a task. The agent space is related to the declarative structure in that it captures re-usable policies that an agent can apply to different tasks. In Konidaris's approach, the agent and problem state spaces are both constructed *a priori*. In the design proposed in this chapter, relevant features in the state vector are recovered incrementally and as necessary as a robot learns increasingly rich procedural policies.

5.2 Controller Abstraction

As discussed in Section 3.2.4, control expressions in the control basis provide typing constraints on the input sensors and output effectors that are used to compute control inputs. As a result, a potential function $\phi \in \Omega_\phi$, when combined with a sensory signal $\sigma \subseteq \Omega_\sigma$ with a characteristic input type (CIT) $t_{in} \in \mathcal{T}$, and an effector resource $\tau \subseteq \Omega_\tau$ with characteristic output type (COT) $t_{out} \in \mathcal{T}$, represents a family of functionally equivalent controllers we will call an *abstract action*, $a(\phi, t_{in}, t_{out})$,

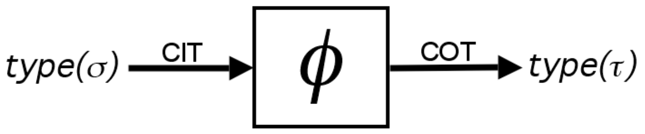


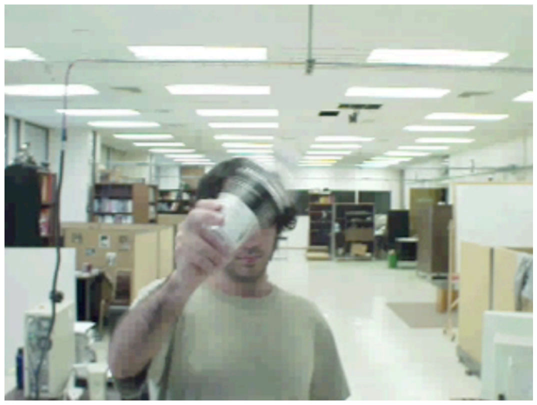
Figure 5.1. Abstract actions consist of objective functions $\phi \in \Omega_\phi$ coupled with a characteristic input type (CIT) and a characteristic output type (COT).

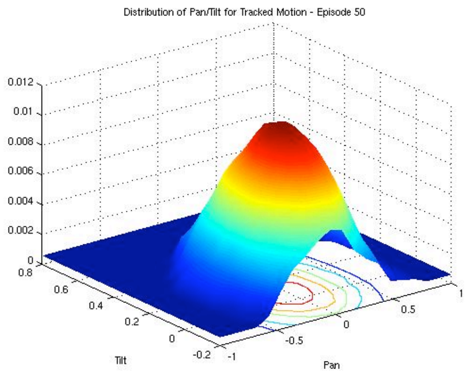
illustrated in Figure 5.1. For example, the abstract action using a harmonic potential ϕ_h represents a class of control actions that provide collision-free motion plans in $\mathbb{R}^3$ to a manipulator configuration output in $\mathcal{C}_n$. However, goals and obstacles in ϕ_h can be observations $\mathcal{O} \in \mathbb{R}^3$ derived from a laser scanner, a stereo vision system, a tactile probe, or any other equivalent sources of position information.

This method of controller abstraction allows us to re-parameterize control actions in the control basis. We have already seen a number of examples of control basis re-parameterization in which the Touch and Track controllers, for example, were applied to different combinations of Dexter's force sensors, arm variables, or visual features. Typing constraints provide a large amount of structure pertaining the intentions of a control basis program that make re-parameterizations preserve abstract behavioral goals.

5.2.1 Abstracting SearchTrack

The SearchTrack program that Dexter learned in the previous chapter created models of where training features (highly-saturated pixel regions) occurred and tracked them on the center of its left camera image plane. The strategy acquired, however, could equally well be applied to different visual features. In a new learning stage, Dexter executed its SearchTrack policy for an additional twenty-five training episodes, this time building a model of where regions of *pixel motion* occur in its pan/tilt configuration space and tracking such motions. This was accomplished by


(a)



(b)

Figure 5.2. Panel (a) shows Dexter's left camera view while tracking motion during a typical programming trial, and (b) shows the non-parametric distribution of pan/tilt configurations learned for motion cues after 25 training episodes. The single peak corresponds to the place where the experimenter presented motion cues to the robot during the acquisition of SEARCHTRACK.

"swapping out" the feedback signals to SEARCH and TRACK with the signals pertaining to high-saturation signals $\gamma^l_{sat,10} \in \Omega_\gamma$, with signals pertaining to motion cues, $\gamma^l_{motion} \in \Omega_\gamma$. For 50% of these episodes, an object was shaken in front of the robot, as seen in Figure 5.2(a). The other half of the time, no object was presented to the robot. Figure 5.2(b) shows the non-parameteric distribution Dexter learned during these episodes encoding which pan/tilt configurations track motion cues.

To show the wide applicability of re-parameterization for SEARCHTRACK, Dexter was directed to explore headings towards regions of interest in the *thirty* hue, saturation, and intensity channels in Ω_γ (10 channels each). The result of this exploration was that Dexter was able to gain a comprehensive "understanding" of the affordances in its primitive visual environment. For this training situation, Dexter cycled through each of the thirty HSI channels, parameterizing SEARCHTRACK accordingly, and gathering data regarding where the environment affords tracking each $\sigma \in \Omega_\gamma$. Dexter attempted to acquire fifty positive samples of each channel, but gave up if no valid heading was found after ten samples. Figures 5.3, 5.4, and 5.5 show histograms of pan/tilt locations where regions of each of the thirty channels were trackable. Most channels produced some response from the environment, although a few did not (e.g., saturation channels 6 and 9, hue channels 7 and 8, etc.).

In conjunction, these visual signals can be used by Dexter as a primitive background model for what it expects to see from its cameras. Given that Dexter is a stationary robot, such a model provides a prior on the entire visual environment the robot can expect to observe. Furthermore, deviations from this model can direct the robot's attention towards new possible affordances. For example, any object placed in front of the robot, which will itself be comprised of a combination of the above thirty channels, will result in some deviation from the robot's prior model. It is easy to see how this deviation could be used to "trigger" further exploration—for example, the robot could try reaching out and touching the object, picking it up, etc.

It should be noted that a basis of hue, saturation, and intensity features to describe the visual affordances of a scene will provide for only the most simple characterization of the environment. However, the above technique can be applied equally well using more robust feature descriptors such as SIFT-descriptors (Lowe, 2004) or differential Gaussian invariants (Lindeberg, 1994) if they are added to the set of heading channels in Ω_γ.

5.2.2 Abstracting TactileProbe

Given an understanding of the robot's visual background, Dexter can begin to explore other affordances of objects it encounters. For example, Dexter can explore locations where force-domain TOUCH affordances occur to create priors for TACTILEPROBE. Because TOUCH is invoked not only by TACTILEPROBE, but also by REACHTOUCH and BIMANUALTOUCH, Dexter can build a comprehensive understanding by a number of strategies.

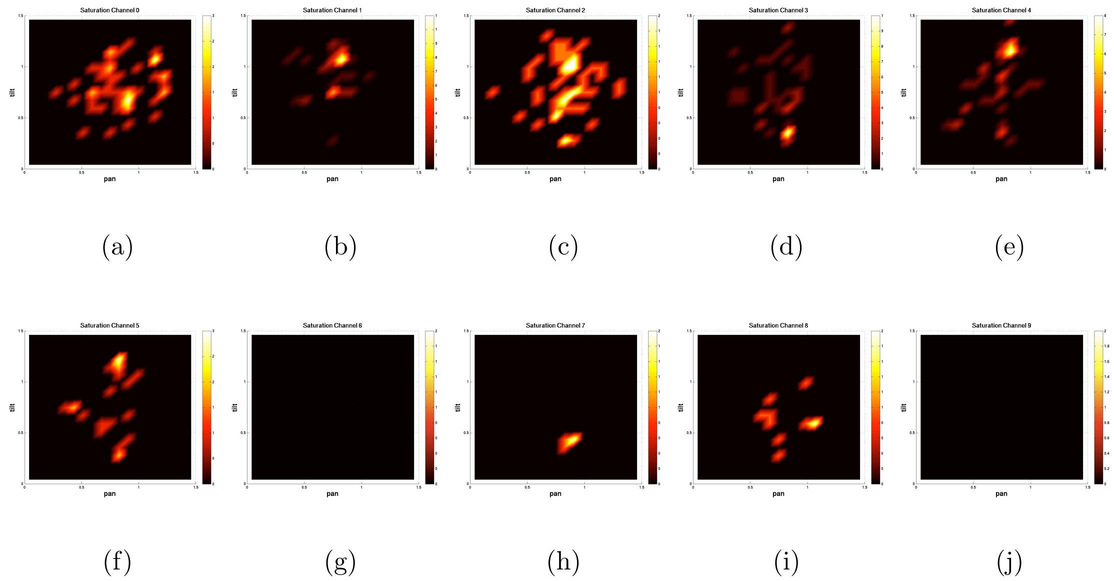


Figure 5.3. Trackable Configurations for Channels of Saturation

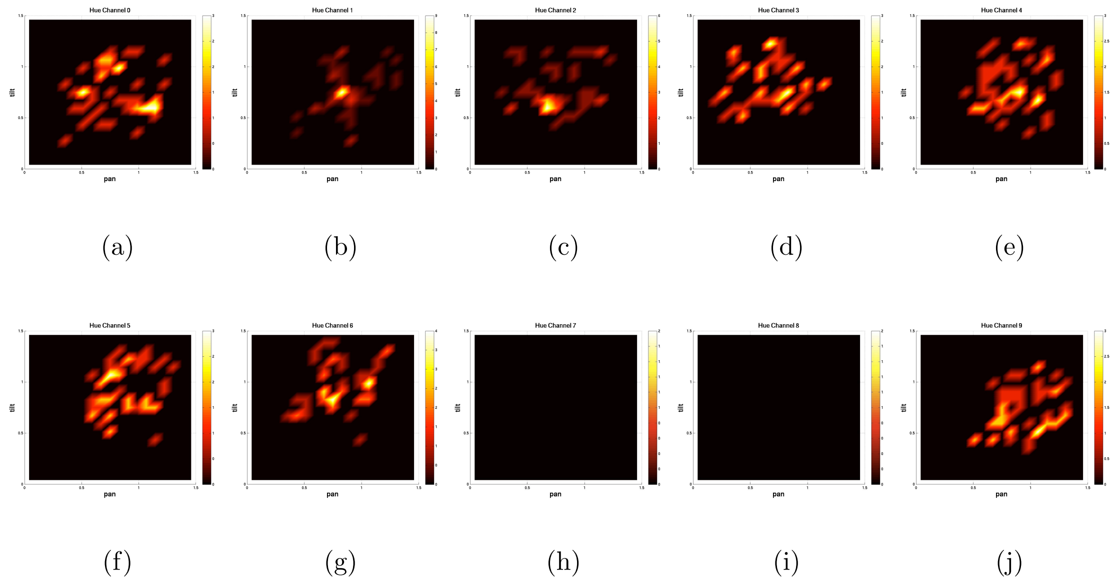
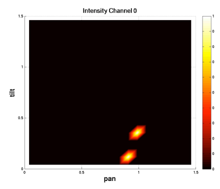
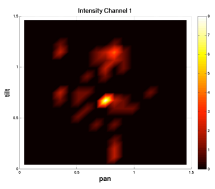


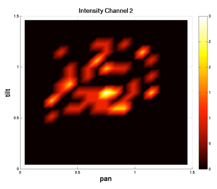
Figure 5.4. Trackable Configurations for Channels of Hue



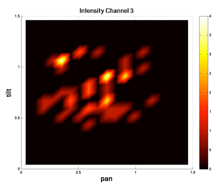
(a)



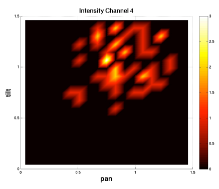
(b)



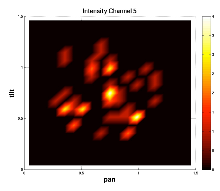
(c)



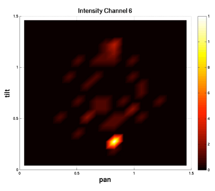
(d)



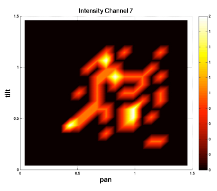
(e)



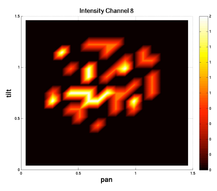
(f)



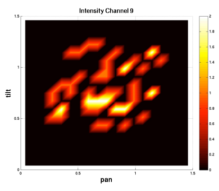
(g)



(h)



(i)

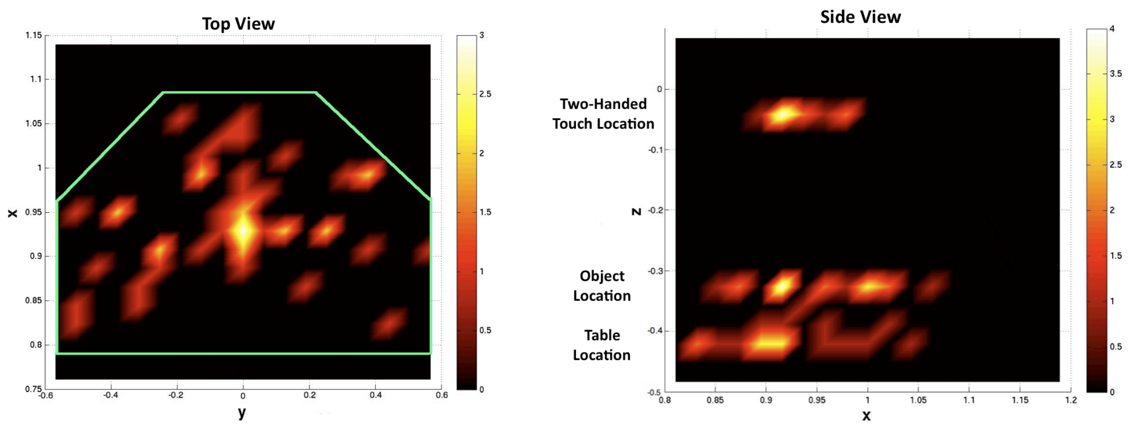


(j)

Figure 5.5. Trackable Configurations for Channels of Intensity

To explore areas of force-domain reactions, Dexter gathered experience concerning the Cartesian locations where its environment afforded Touch by extracting fifty samples using either ReachTouch or BimanualTouch. For twenty of these actions, the robot executed the ReachTouch program with either of its arms to reach out to a large green table placed in front of the robot. Because the table is large, goal locations for the reach action were sampled from the observed surface of the table. For an additional twenty actions, Dexter used the ReachTouch program to reach out and touch a highly-saturated object placed on this table. In the other ten actions, Dexter executed the BimanualTouch program on the object with highly-saturated hue.

Cross-sectional histograms of the gathered touch samples are shown in Figure 5.6. Figure 5.6(a) shows the top-down view of places that afford Touch. The outline of the green table from this view is imposed on the image in green. We see that all the responses occur on or above this table region. The large peak in the center of the image is the location where the BimanualTouch brings the object to be touched by both hands, more clearly seen in the side view shown in Figure 5.6(b). We see



(a) (b)

Figure 5.6. (a) shows the top-down (xy-plane) view of the Cartesian locations where tactile responses occur. The outline of the table is shown in green. (b) shows the side view (xz-plane). The coordinate system has its origin in Dexter's chest.

samples at the table height (about $-0.42m$ in the z-direction), slightly above the table height (where the objects were placed), and at the BimanualTouch location (slightly below $0m$ in the z-direction).

The resulting samples may be used to build models for the Search action in TactileProbe even though they were gathered using other programs. These models can be used to provide Cartesian references for the Reach action in ReachTouch that are likely to produce a Touch affordance even when no object is visible (maybe due to occlusion, the camera being damaged, or the lights being out).

These examples should make clear, that by maintaining the typing constraints imposed by policies, a robot can efficiently and autonomously explore re-parameterizations to gather a large amount of experience about the world. This experience forms the robot's first cognitive models and function as a simple internal *memory* characterizing locations and configurations of where rewarding behavioral events tend to occur. We next examine how a robot can learn how to intelligently adapt control basis programs to new contexts to provide additional opportunities to achieve reward.

5.3 Control Basis Schema

In the last chapter, we demonstrated how a robot can learn specialized hierarchical skills in successive training stages. We now describe a process by which a robot can generalize these skills to new situations in subsequent stages of learning that preserve the transitional structure (or "intentions") of the original policies. As a robot generalizes its programs to support increasingly dexterous procedural contingency plans, these programs are transformed into knowledge structures we call *schema* that can produce reward in many contexts.

The proposed methodology for generalizing control basis programs into schema is illustrated in Figure 5.7. A policy π is first learned over a state and action space $\mathcal{A}$ and $\mathcal{S}$ using specific sensory and motor allocations that work in the training context. The policy is then factored into declarative (abstract) and procedural components. The abstract actions are then allocated with type-constrained resources based on the environmental context $f \in \mathcal{F}$ in order to preserve the original transition structure of π. Enforcing strict typing constraints reduces the combinatorial space of possible resource combinations, making the search space more efficient for machine learning algorithms to explore. We now discuss how this process can be represented computationally.

Let the (ordered) declarative and procedural parts of a prioritized control law $c_i = c(\phi_0, \sigma_0, \tau_0) \lhd \cdots \lhd c(\phi_n, \sigma_n, \tau_n)$ be defined, respectively, as follows:

$$\begin{aligned} declarative(c_i) &= (a_0, \cdots, a_n) && (5.1) \\ procedural(c_i) &= (\omega_0, \cdots, \omega_n) && (5.2) \end{aligned}$$

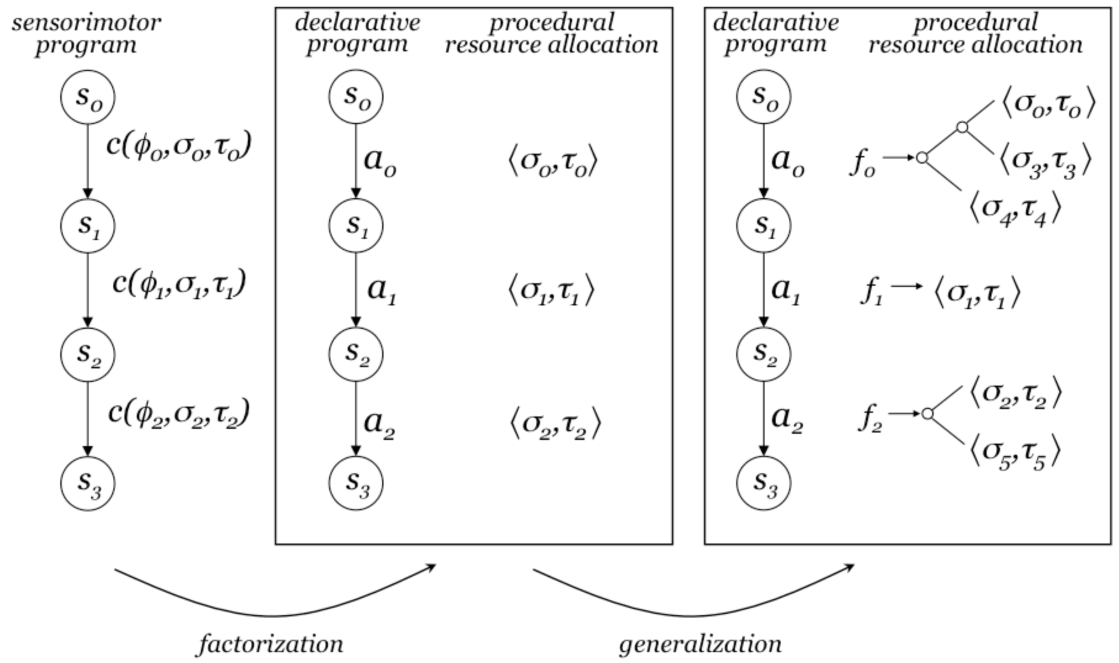

sensorimotor program
s_o
$c(\phi_o,\sigma_o,\tau_o)$
s_1
$c(\phi_1,\sigma_1,\tau_1)$
s_2
$c(\phi_2,\sigma_2,\tau_2)$
s_3
declarative program
procedural resource allocation
a_o
$\langle\sigma_o,\tau_o\rangle$
a_1
$\langle\sigma_1,\tau_1\rangle$
a_2
$\langle\sigma_2,\tau_2\rangle$
declarative program
procedural resource allocation
f_o
$\langle\sigma_o,\tau_o\rangle$
$\langle\sigma_3,\tau_3\rangle$
$\langle\sigma_4,\tau_4\rangle$
f_1
$\langle\sigma_1,\tau_1\rangle$
f_2
$\langle\sigma_2,\tau_2\rangle$
$\langle\sigma_5,\tau_5\rangle$
factorization
generalization

Figure 5.7. Sensorimotor programs in the control basis can be factored into procedural and declarative components and generalized to new environmental contexts, by means of the policy $\psi(a_i, f_j)$, where $a_i \in \mathcal{A}$ and $f_j \in \mathcal{F}$.

where each a_m is a single-objective abstract action consisting of a potential function with characteristic input and output types, such that $a_m = a(\phi_m, type(\sigma_m), type(\tau_m))$, ω_m is set of a sensor and effector resources, such that $\omega_m = \langle\sigma_m, \tau_m\rangle$ that meet the original CIT and COT typing constraints of c_m, and $m = 0 \ldots n$.

At run-time, each abstract action must be allocated with sensorimotor resources. Let procedural policy ψ for a schema be a mapping from abstract action $a \in \mathcal{A}$, and context $f \in \mathcal{F}$, to a set of sensorimotor resources for each controller,

$$\psi : (a, f) \mapsto (\omega_0, \cdots, \omega_n). \tag{5.3}$$

One useful procedural policy for a single-objective abstract action $a(\phi_i, t_{in}, t_{out})$ is defined as:

$$\psi(a, f) \quad = \quad argmax_{\omega_i} Pr(\mathbf{s}_{cur}, \mathbf{s}_{des} | c_i, a, f) \tag{5.4}$$

where $\omega_i = \langle \sigma_i, \tau_i \rangle$, c_i is a controller parameterized by ϕ_i and resource model ω_i (that obeys the CIT and COT typing constraints of original control action, such that $type(\sigma_i) = t_{in}$ and $type(\tau_i) = t_{out}$), $\mathbf{s}_{cur}$ is the current state, and $\mathbf{s}_{des}$ is the desired next state along the route to a rewarding state under policy π. Policy ψ can be extended to multi-objective control laws by finding the set of procedural parameterizations that preserve the desired state transition. This technique is reminiscent of the graph homomorphism technique presented in Platt (Platt, 2006). It is also an extension of previous work where relational models were used to capture procedural task knowledge (Hart et al., 2005).

Examining the procedural context of a particular control basis program also supports inferences regarding when reward is *unlikely* to occur for any resource allocation. This likelihood is captured by the probability of achieving reward for a given policy and context, $Pr(reward|\pi, f)$. Consider a case in which Dexter explores REACH-TOUCH, but no object is present in the reachable workspace: this probability should be sufficiently low. In such a situation, the schema should report that it is not in a region of its state-space where its goals can be met, and evaluate to the undefined "$-$" condition. It can then inform any higher-level program that is using it hierarchically that it is unlikely to be able to achieve its goals.

The procedure for how a control basis program can be generalized into a schema with procedural contingency plans—called ASSIMILATE()—is provided by Algorithm 2. It is similar to ACCOMMODATE() except that, instead of using Q-Learning to learn an action-value function Φ, it estimates a policy ψ and various probability distributions that capture procedural information. ASSIMILATE() takes as input the action set $\mathcal{A}$, a policy π that maps states in the state space $\mathcal{S}$ formed by the actions in $\mathcal{A}$ to those actions, a set of resources Ω that can be used to re-parameterize the actions suggested

Algorithm 2 ASSIMILATE($\mathcal{A}$, π, Ω, m, T)

```
1: Let $\mathcal{A}'$ be the set of non-composite actions in $\mathcal{A}$
2: Let $\mathcal{S}$ be the state space formed from the predicates of the actions in $\mathcal{A}'$
3: $n \leftarrow |\mathcal{A}'|$, $\epsilon = 0.2$
4: for $i = 1$ to $m$ do
5:     $k \leftarrow 0$
6:     $reward \leftarrow$ false
7:     while $(reward = \text{false}) \wedge (k < T)$ do
8:         observe features $f \in \mathcal{F}$
9:         $\mathbf{s}^k \leftarrow (p_1^k \ldots p_n^k)$
10:        $a \leftarrow declarative(\pi(\mathbf{s}^k))$
11:        $\omega \leftarrow \psi(a, f)$ (via $\epsilon$-greedy selection), where $\omega \in \Omega$
12:        allocate $a$ with $\omega$ to form control action $c$
13:        repeat
14:            execute $c$ for one iteration
15:            $k \leftarrow k + 1$
16:            $\mathbf{s}^k \leftarrow (p_1^k \ldots p_n^k)$
17:        until $\mathbf{s}^k \neq \mathbf{s}^{k-1}$
18:        update $Pr(\mathbf{s}^{k-1}, \mathbf{s}^k | c, a, f)$
19:        evaluate $r^k$ according to Equation 4.4
20:        if $r^k > 0$ then
21:            $reward \leftarrow$ true
22:            update $Pr(\tau | \sigma, reward)$ for all rewarding control actions $c(\phi, \sigma, \tau)$,
23:        end if
24:    end while
25:    update $Pr(reward | \pi, f)$
26: end for
```

by π, the number of learning episodes m to be performed, and a "timeout" parameter T that limits the number of state transitions for each episode (set to 100 in the following experiments). Line 18 updates the probability distribution used to evaluate ψ as seen in Equation 5.4. Line 25 updates the probability of achieving reward for the given policy and the observed run-time context $f \in \mathcal{F}$.

5.3.1 ReachTouch Procedural Artifacts

In the last chapter, Dexter acquired a REACHTOUCH program using ACCOMMODATE() in a constrained setting. This program provides a policy for the robot to uncover TOUCH affordances in its environment using its right hand. The REACH-

Touch strategy, however, can be applied equally well to uncover left-handed or bimanual Touch affordances. In this section, we construct a simulation illustrating how Dexter can adapt a ReachTouch program through a process of assimilation into a schema that has procedural contingency plans for handedness and knowledge concerning when objects are out of reach. We demonstrate how the proposed technique for generalization improves performance over control basis programs where these techniques are not employed. We conclude by comparing the simulated results to the results of learning experiments performed on the real robot. We present three approaches for learning handedness.

5.3.1.1 Approaches for Learning Handedness

Flat Learning Approach: A "flat" baseline learning experiment that does not employ any techniques for generalization was first performed in simulation. During learning, objects of diameter either 50 *cm* or 10 *cm* were presented to the robot in a variety of positions and with a variety of velocities. In half of the training episodes a ball of the larger diameter was placed in front of the robot. In the other half of the episodes, a smaller ball was presented, placed in a stationary position to the left or right sides of the robot or on one side moving with a velocity of 0.05 m/s in the direction of the opposite hand. The larger objects require a bimanual strategy to successfully track reference contact signals. The smaller and moving objects require policies that consider handedness and anticipatory reaches. The object was occasionally presented outside of the robot's initial field of view, requiring the use of the SearchTrack program.

In this learning experiment, separate control actions for reaching and touching with left-, right-, or two-handed options were provided as separate explorable actions. To reduce the possible combinatoric action space for this learner, we prevented composite control combinations between the actions, resulting in the action set

$$\begin{aligned}\mathcal{A}^1_{rt} &= \{\textsc{SearchTrack}, \textsc{Reach}(\textsc{LeftArm}), \textsc{Touch}(\textsc{LeftHand}),\\ &\quad \textsc{Reach}(\textsc{RightArm}), \textsc{Touch}(\textsc{RightHand}),\\ &\quad \textsc{Reach}(\textsc{BothArms}), \textsc{Touch}(\textsc{BothHands})\}.\end{aligned}$$

State predicates were provided in the state description capturing the binary "locale" and "scale" of the observed object $p_{locale} \in \{\texttt{left}, \texttt{right}\}$ and $p_{scale} \in \{\texttt{small}, \texttt{large}\}$, respectively, along with a characterization of its direction of movement $p_{vel} \in \{\texttt{left}, \texttt{right}, \texttt{stationary}\}$. These state predicates provide a simplified (discrete) state representation that gives the flat learner an advantage over agents that learn with real-valued data. The corresponding state space for this learning agent is $\mathcal{S}^1_{rt}$ where states $\mathbf{s} \in \mathcal{S}^1_{rt}$ are evaluated such that $\mathbf{s} = (p_{st} \quad p_{reach(left)} \quad p_{touch(left)} \quad p_{reach(right)} \quad p_{touch(right)} \quad p_{reach(both)} \quad p_{touch(both)} \quad p_{vel} \quad p_{locale} \quad p_{scale})$. Dexter explored this state and action space to learn a ReachTouch policy using the Accommodate() procedure (with the augmented state represenetation). One hundred trials of 200 learning episodes were conducted in this learning experiment. Each episode terminated when reward was received from the affordance discovery motivator by the quiescence of *any* of the Touch controllers.

Concurrent Learning Approach: A second learning experiment was performed in simulation using the techniques for generalization and abstraction proposed in this chapter. During this learning experiment, the learning agent used the affordance discovery reward function to *concurrently* acquire declarative and procedural policies. The declarative policy was learned using Q-learning with state space $\mathcal{S}^2_{rt}$ where states $\mathbf{s} \in \mathcal{S}^2_{rt}$ are evaluated as $\mathbf{s} = (p_{st} \; p_{reach} \; p_{touch})$ and with action set

$$\mathcal{A}^2_{rt} = \{\textsc{SearchTrack}, \textsc{Reach}, \textsc{Touch}, \textsc{Reach} \triangleleft \textsc{Touch}, \textsc{Touch} \triangleleft \textsc{Reach}\}.$$

A procedural policy was learned in the form of Equation 5.4 based on a joint probability distribution estimating $Pr(\mathbf{s}_{cur}, \mathbf{s}_{des} | c_i, a, f)$ with feature set $\boldsymbol{f}_{rt} = [\mathbf{x}_{obj}, \dot{\mathbf{x}}_{ob}, \upsilon_{obj}]$, where $\mathbf{x}_{obj}$ is the Cartesian position of the simulated object in the robot's coordinate system, $\dot{\mathbf{x}}_{obj}$ is its velocity, and υ_{obj} is its spatial volume (i.e., its scale). In simulation these quantities were known precisely and observed at the beginning of each episode. When a declarative action was chosen at each new state transition, the procedural policy inferred the best parameterization for that action using ϵ-greedy exploration ($\epsilon = 0.2$) based on its experience in the learning trial up until that point. This experiment was repeated for 100 trials of 200 episodes in the same training context as the flat learning agent (large and small objects; moving and stationary). I hypothesize that the techniques for abstraction and generalization will give this learning agent an advantage over the flat learning agent because it has a more compact declarative state/action space that it can generalize across run-time contexts.

Staged Generalization Approach: In the third experiment, we simulate the performance of a 2-staged learning approach. Dexter first undergoes a period of accommodation in which the robot learns a declarative policy π in a constrained training situation using the ACCOMMODATE() procedure. During a subsequent period of assimilation (via ASSIMILATE()), the robot learns a procedural policy ψ to adapt its original program into new, more complex situations. In the accommodation stage, only (smaller) objects with a diameter of approximately 10 *cm* were placed in a stationary location on the robot's right-hand side. During this stage, the robot could reach to the object by moving its right-arm and the right-hand motor variables $\{\mathbf{x}_{r,arm},\ \boldsymbol{\theta}_{r,hand}\} \in \Omega_\tau$. The declarative policy was learned using the ACCOMMODATE() procedure with action set

$$\begin{aligned}\mathcal{A}^3_{rt} \;=\; & \{\textsc{SearchTrack}, \textsc{Reach}(\textsc{RightArm}), \textsc{Touch}(\textsc{RightHand}), \\ & \textsc{Reach}(\textsc{RightArm}) \lhd \textsc{Touch}(\textsc{RightHand}), \\ & \textsc{Touch}(\textsc{RightHand}) \lhd \textsc{Reach}(\textsc{RightArm})\}\end{aligned}$$

and states $\mathbf{s} \in \mathcal{S}_{rt}^3$ where $\mathbf{s} = (p_{st}\ p_{reach(right)}\ p_{touch(right)})$. The robot is rewarded when the environment affords touch actions, captured by the affordance discovery reward function when the Touch(RightHand) action quiesces. Each episode terminated when intrinsic reward was received.

After 25 training episodes of declarative learning (the accommodation stage), the assimilation process began. During this stage, the simulated context was expanded to the complexity of the other two experiments, and the robot was allowed to perform Reach and Touch actions with the left-arm and the left-hand motor variables $\Omega = \{\mathbf{x}_{l,arm},\ \boldsymbol{\theta}_{l,hand}\} \in \Omega_\tau$. The typing specifications of the original policy were preserved and the robot explored left-, right-, and bimanual-reach actions. During these episodes, the robot learned a procedural policy based on a joint probability distribution estimated over the feature set $\boldsymbol{f}_{rt}$. The learning agent used this policy to parameterize the abstract actions of the declarative policy at run-time based on its previous experience in the trial. This experiment was repeated for 10 trials of 200 learning episodes. Twenty-five episodes were executed using the Accommodate() procedure to estimate a policy π. This policy was then used in 175 episodes of the Assimilate() procedure with the additional resources in Ω. I hypothesize that this learning approach will outperform both of the other learning approaches. This learning agent will be able to bootstrap learning in the complex environments by first learning in simple situations.

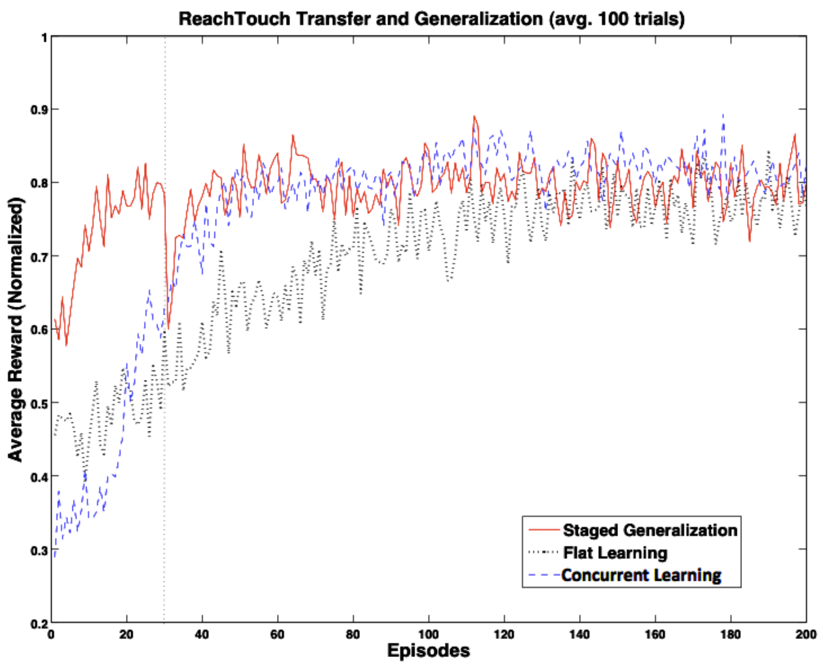
Real Robot Learning Example: A final set of learning experiments were performed on the real robot and the results were compared to those of the staged learning experiment performed in simulation. During this experiment, objects with highly-saturated hues were presented to the robot. As in the simulated staged learning experiment, a declarative policy was learned by the Accommodate() procedure with states $\mathbf{s} \in \mathcal{S}_{rt}$ where $\mathbf{s} = (p_{st}\ p_{reach(right)}\ p_{touch(right)})$ and actions

$$\begin{aligned}\mathcal{A}_{rt} \;=\; \{&\textsc{SearchTrack}, \textsc{Reach}(\textsc{RightArm}), \textsc{Touch}(\textsc{RightHand}),\\ &\textsc{Reach}(\textsc{RightArm}) \triangleleft \textsc{Touch}(\textsc{RightHand}),\\ &\textsc{Touch}(\textsc{RightHand}) \triangleleft \textsc{Reach}(\textsc{RightArm})\},\end{aligned}$$

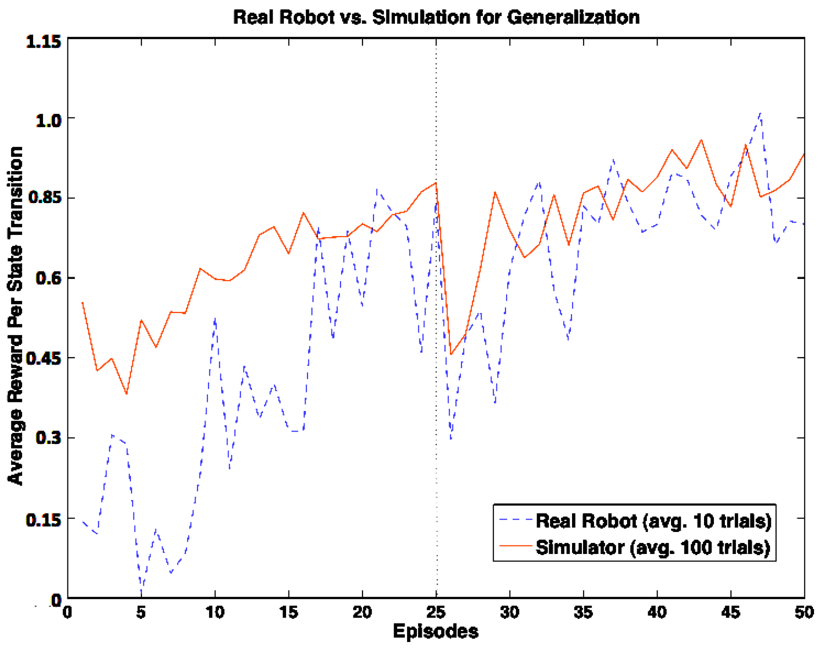
receiving reward according to the affordance discovery reward function when the TOUCH(RIGHTHAND) controller quiesced. In these experiments, the feature vector $\boldsymbol{f}_{rt}$ was computed by estimating the position, velocity, and scale of a highly-saturated visual feature on the robot's left and right camera frames, ($\boldsymbol{\gamma}^{l}_{sat,10}$,$\boldsymbol{\gamma}^{r}_{sat,10}$). In the assimilation stage (in which the context was enriched), the C4.5 decision-tree learner (Quinlan, 1993) was used to estimate the procedural policy (Equation 5.4) suggesting how the REACH and TOUCH policies should be parameterized based on $\boldsymbol{f}_{rt}$. This algorithm is advantageous because it is simple, fast, and provides intuitive results that are easily interpretable by a human. In this experiment, 10 trials of 50 learning episodes were performed (25 episodes of ACCOMMODATE() followed by 25 episodes of ASSIMILATE()).

5.3.1.2 Comparison of Approaches

Figure 5.8(a) shows the average reward per state transition for the three simulated REACHTOUCH experiments. The learning curves are averaged over the 100 simulated trials and normalized to their optimal average reward per state transition values. Normalization results in an expected asymptotic reward equal to 0.8 for each learning agent because ϵ-greedy exploration strategy is used with $\epsilon = 0.2$. Normalization was necessary for comparing results because the flat and factorable optimal policies requred a different number of state transitions to touch the object and performance is judged by how long each of the learning approaches take to stabilize to the asymptotic level of average reward.



(a)



(b)

Figure 5.8. Panel (a) shows the average reward per state transtion over 100 trials for the REACHTOUCH experiments performed in simulation. Panel (b) shows the performance of generalized learning on the robot in (dashed-blue) and in simulation (solid-red).

The flat learner (shown in light, dotted gray in Figure 5.8(a)) takes about 140 episodes to stabilize—a long time compared to the other two learning agents—even with the simplified, discrete state representation and the limited action set with no composite control actions. If continous state information had been incorporated instead, we hypothesize that its performance would have further degraded compared to the other learners. The dashed blue line shows the performance of the concurrent learner that learns the declarative and procedural policies at the same time. While this learner acquires its policies, it is often challenged by the fact that it can not discern whether lack of reward is due to having poor declarative organization or to a poor procedural policy. As hypothesized, however, the concurrent learner outperforms the flat learner, asymptotically converging after about 50 learning episodes (on average). I argue this is because it uses an abstract state/action space that reduces the explorable search space.

The solid red line in Figure 5.8(a) shows the staged generalization experiment where the declarative policy is learned first in a contrained accommodation stage. This agent outperforms both other agents, as hypothesized. At episode 25 the training context was unconstrained and the assimilation stage begins. After an initial sharp drop at the beginning of this stage (lasting about 10 episodes), performance rapidly improves as the procedural policy is learned with contingencies taking into account the position, velocity, and volume of the simulated object. During these episodes, if the intial resource allocation did not work, the actions were re-allocated by the stochastic procedural policy until the desired (declarative) state transition was achieved. This experiment shows how a policy learned in a constrained setting can assimilate new contexts quickly with only a temporary decrease in performance. The declarative policy that this agent learned in the accommodation stage provides structure to its exploration in the assimilation stage with the richer environmental context. The

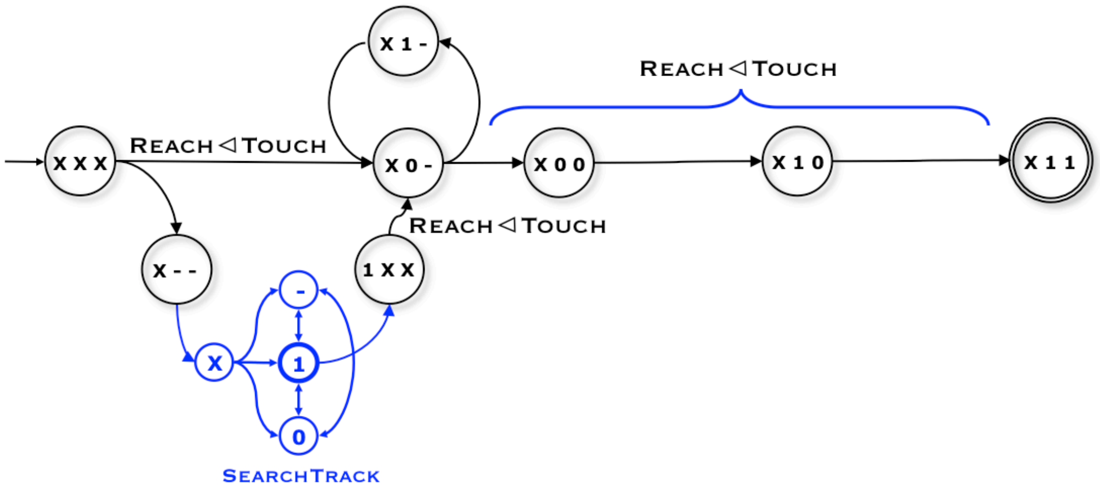


Figure 5.9. The transition diagram for the policy learned for ReachTouch. $\mathbf{s} = (p_{st}\ p_{reach}\ p_{touch})$, where p_{st} is the predicate value of the SearchTrack program.

result is that more reward is achievable quicker, and performance suffers less as the context changes.

During the declarative learning portion of the staged learning experiment, the simulated robot learned a simple policy to get reward. During the assimilation stage, the policy was adapted slightly as seen by the transition diagram in Figure 5.9. This diagram is almost identical to the one shown in Figure 4.10, except that it shows a state transition from (X1−) back to (X0−). This transition occurred because sometimes, when the Reach action failed to produce a tactile response, the procedural policy stochastically re-parameterized the action with a different resource. If a tactile response was still not forthcoming, it returned to state (X1−), otherwise the state transitioned to (X00). This loop in the learned policy effectively allows the simulated robot to "re-try" its actions with different resources if the observed transition dynamics do not lead to reward. As the robot incorporates more observations to inform its procedural policy, this process will eventually allow the robot to find the "correct" resource allocation (if there is one) for the current context and transition to state (X11) where it receives reward.

Figure 5.8(b) shows how the experiment conducted on the real Dexter (averaged over 10 trials) compared to the simulation results for the staged learner (averaged over 100 trials). The real robot lags in performance compared to the simulated agent in the first 15-20 learning episodes, but manages to learn the same policy illustrated by Figure 5.9 by the end of the accommodation stage (25 episodes). For these experiments, an object distinguished by highly-saturated features was presented within reach of the right hand for the first 25 episodes and the robot was able to learn the initial, declarative policy. After this point, the period of assimilation began in the less-constrained training context.

Figure 5.10 shows the decision tree learned using the C4.5 algorithm after one of the training trials on the robot. Other trials produced similar results. This tree indicates that if the ball is large (i.e., it has appreciable volume), then a 2-handed reach should be used. Moreover, small moving objects indicate that the robot should reach with the arm that anticipates the movement, otherwise, the object would move out of the workspace of the hand chosen. Stationary objects indicate the use of the hand on the same side as the object. This policy reflects clear common sense knowledge about handedness, scale, and velocity concerning one- and two-hand ReachTouch options.

The experiments performed on the real robot demonstrate how the proposed staged learning technique can lead to solutions in relatively short periods of time. Each episode for ReachTouch takes about 30 seconds to 1 minute (depending on how stable the policy is). This resulted in a total trial time (50 episodes) of about an hour. Extrapolating from this, Figure 5.8(a) suggests that a flat learning approach performed on the real robot would take about three or four hours, even though the training situation is still relatively simple. It should be clear that as more proce-

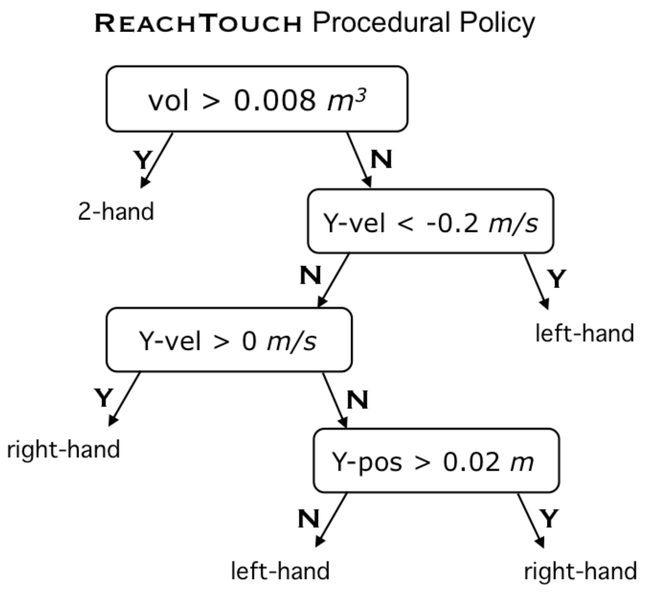


Figure 5.10. This decision tree shows the resulting procedural policy for choosing which arm to allocate for reaching based on object volume, position, and velocity.

dural categories are introduced, it would quickly become intractable in comparison. The staged learning approach, however, overcomes this scaling issue by incrementally extending the structure—or *scaffolding*—acquired in previous stages to new contexts.

5.3.1.3 Learning "Out of Reach"

Dexter performed an additional 50 learning episodes in which objects of various sizes were placed on the table in front of the robot, half of the time outside the robot's reach. This new learning stage was conducted for the robot to acquire common sense knowledge concerning when features do not afford a controlled Touch response no matter which resources are allocated because they are "out of reach" and too far away to be touched. The C4.5 algorithm was used to learn a decision tree based on the feature vector $\boldsymbol{f}_{rt}$ concerning what contexts lead to reward using the ReachTouch policy and which do not. This tree reflects the probability distribution $Pr(reward|\pi, f)$ updated in Line 25 of the Assimilate() procedure.

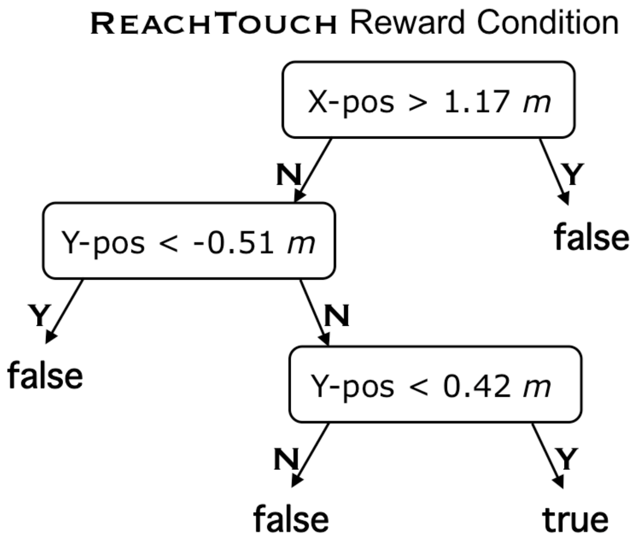


Figure 5.11. This decision tree shows conditions under which ReachTouch is likely to achieve reward. It captures when objects are out of the robot's work space in terms of their position.

The resulting decision tree is shown in Figure 5.11. We see that for objects placed in x locations greater than 1.17 m in front of the robot or too far over to the robot's side in the y-direction, the program is not likely to achieve reward. This tree represents more knowledge about the run-time context of the robot that can be used to inform future control decisions.

5.3.2 Hierarchical Generalizations

In the framework presented in this document, control programs are assembled because they afford controllable interactions with the world. However, each subgoal in a program can posture the robot to discover even more kinds of affordances. These programs supply pre-conditions for many other controllable interactions—*orienting* the robot for further types of interactions. Thus SearchTrack makes triangulation possible and greatly improves the probability of generating ReachTouch awards. Morevover, Touch, TactileProbe, ReachTouch, and BimanualTouch all provide different contexts in which Dexter can receive reward for the quiescence of a

TOUCH controller. Thus, schema induce categories that can be used in higher-level schema to accomplish desired subgoals. As a result, generalizations over hierarchical schema are possible in a policy where entire programs—explored as indivisible temporally extended actions—are chosen based on the run-time context.

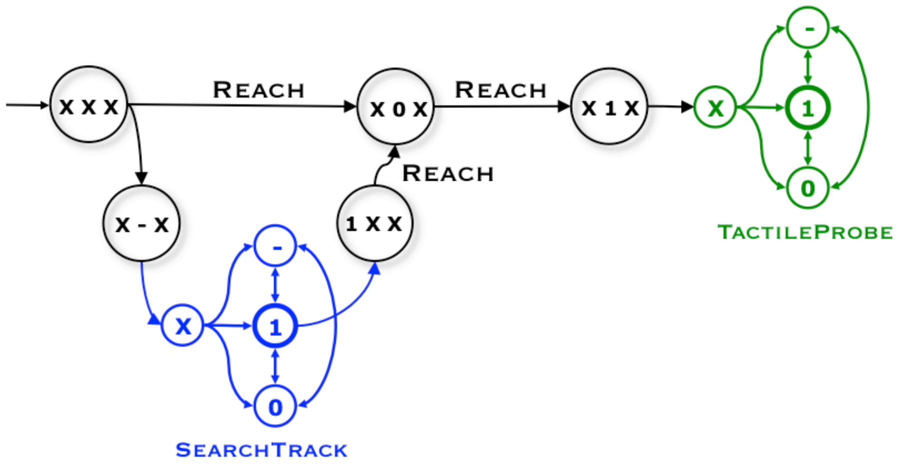


Figure 5.12. A possible re-parameterization of the REACHTOUCH schema utilizing the TACTILEPROBE program to achieve reward in place of the TOUCH primitive. This schema has states $\mathbf{s} \in \mathcal{S}'_{rt}$ where $\mathbf{s} = (p_{st}\ p_{reach}\ p_{tp})$, and p_{st} and p_{tp} are the state values of the SEARCHTRACK and TACTILEPROBE programs, respectively.

For illustrative purposes, consider a re-parameterized REACHTOUCH schema where the TOUCH action is replaced by the TACTILEPROBE schema as seen in Figure 5.12. Both ultimately achieve the same rewarding objective (quiescence of TOUCH), but each works in different situations. Using TOUCH in Figure 5.12 relies on REACH to bring the robot's tactile sensors into contact with the object. This, however, is often not the case, say when the object is small. By substituting TACTILEPROBE for TOUCH the robot can search for contact within the entire reachable workspace of the hand. This strategy is likely to work in more cases than just the primitive TOUCH action.

5.4 Discussion

In this chapter, a computational framework is proposed in which a robot can acquire programs in simple contexts and later generalize them incrementally to address more complex situations. This framework extends computational models of sensorimotor schema beyond those already in the literature. The mechanisms developed are compatible with Piaget's concept of accommodation and assimilation where existing behavior conforms to new situations and expands an organism's capabilities.

The ability to re-apply behavior in different contexts provides a means by which interaction with a relatively small segment of the environment can induce a greater potential for categorical distinctions that prove useful in other contexts. For example, one object may be red-trackable, touchable, and liftable (via BimanualTouch), while another might be blue-trackable, but not touchable or liftable, and so on. In the next chapter, we will investigate how long-term memory structures based on patterns of such categorical distinctions can be assembled, explored, and stored in memory structures called *catalogs* of control affordances.

CHAPTER 6
WORLD MODELING

In the previous few chapters, a framework was proposed in which a robot can autonomously assemble hierarchical control strategies out of its sensory and motor subsystems. Under this framework, a robot can learn behavioral schema that it can use to model its world and its ability to interact with it. Psychological theories for interpreting the world in terms of the ability to engage it were initially proposed by J. J. Gibson in his "theory of affordances," (Gibson, 1977) and developed further by his wife Eleanor (Gibson, 2000). A coffee mug, for example, can be interpreted as a collection of affordances such as "graspable" and "liftable." Chairs, despite a variety of physical forms, all share the affordance of being "sit-onable" by human beings.

Functional representations of knowledge—like Gibsonian affordances—provide a convenient and natural way of organizing cognitive structures in embodied agents that take actions, such as humans or robot manipulators. Furthermore, they alleviate much of the difficulty of interpreting objects in terms of their elusive Platonic "essence" because they provide a simple means of verification. To test whether an object holds a particular affordance (e.g., "graspable") the organism need only try taking the requisite action (grasping) and observe the result.

In this chapter, we introduce a mechanism to construct patterns of co-occurring affordances that define entities in the world. This mechanism allows a robot to explore the conditions in which behavioral schema are likely to provide reward from the affordance discovery motivator until it achieves sufficient confidence in its internal models. In other words, until it *habituates*. The result is a complementary process to the mechanisms for acquiring the behavior in the first place that can guide a robot towards an enhanced understanding of how it can interact with its environment. We demonstrate how this technique can be used to assemble "catalogs" of affordances for objects in the robot's environment.

6.1 Modeling Affordances

In the last chapter, we examined how a robot can accommodate new programs in simple training situations that can subsequently assimilate new environmental contexts. The robot achieved this by finding decision boundaries in the input signals to its control actions and determining how to engage different resources in different situations. Formally, the robot learned distributions of the form $Pr(reward|f, a_i)$ capturing the likelihood of achieving reward for action $a_i \in \mathcal{A}$ after observing the run-time environmental context $f \in \mathcal{F}$. We chose to look exclusively at the values of the feedback signals, $\sigma \subseteq \Omega_\sigma$, used in the control action set of the schema in order to restrict a possible infinite feature space to a tractable collection of variables.

Examining the feature set $\mathcal{F}$ allows a robot to use its experience to build models of contexts that are likely to lead to positive reward if a given program is run. We believe such models best capture the likely affordance of a particular action at any given time. We therefore call such models "affordance models" defined as the probability

$$Pr(f|reward, a_i). \tag{6.1}$$

In this document, we will estimate affordance models as multi-dimensional Gaussians over features $f \in \mathcal{F}$.

In the rest of this chapter we examine two methods for intrinsically motivating a robot to take actions until it is confident in its affordance models. Confidence is measured as a decrease in the change in variance of the models as experience is gathered. We expect this reward function to be non-stationary in that it should provide diminishing returns for taking the same action repeatedly because its corresponding affordance model will grow more robust. The result is a process of habituation that modulates the reward gained by the affordance discovery motivator based on the familiarity of the run-time context (c.f., (Singh et al., 2004a)).

The modulated reward function is defined as the affordance discovery motivator (Equation 4.4) multiplied by the habituation metric, such that

$$r^k = \sum_i \left(h_i^k r_i^k \right), \tag{6.2}$$

where the habituation metric h_i^k is defined as

$$h_i^k \;=\; |\Sigma_i^k - \Sigma_i^{k-1}| - \rho_i, \tag{6.3}$$

where Σ_i^k is the variance of the distribution $Pr(f|reward, a_i)$, $reward$ is a boolean variable, a_i is an action (possibly a schema), and ρ_i is a small positive *cost* for performing that action. In general, this cost could be behavior specific, proportional to the amount of energy expended during execution, but the experiments in this document all used a fixed cost per action for simplicity. In cases where the affordance model is a multi-dimensional Gaussian distribution, the variance is computed by summing all of the elements of the co-variance matrix. The habituation metric evaluates the information gained for taking action a_i based on how it affects the robot's "confidence" in the corresponding distribution.

We assume that the variance of a given affordance model converges as more experience is gathered. Therefore, we expect a robot using reinforcement learning with this reward function will build stable affordance models over time. Moreover, we expect the robot to initially get large amounts of reward for engaging novel contexts, habituating quickly to those contexts that are relatively common. We also expect the robot to spend more time exploring more variable contexts and to react strongly to contexts that change in "surprising" and unexpected ways.

6.2 Catalogs

How can a robot collect clusters of affordances describing entities (i.e., objects) it encounters regularly in its environment? Such knowledge is necessary for a robot to perform dexterous motor skills in an open and unstructured environment.

We attempt to build object models by cataloging systems of controllable events afforded by the run-time environment. We will define a *catalog* as a collection of likely affordances. The construction of a catalog begins by configuring a control circuit referenced to stimuli in the environment and evaluating whether the current environment affords quiescence in the resulting controller. Clusters of associated affordances model objects, groups of objects, or larger contexts that influence the probability of controllable events. As a result, catalogs are knowledge structures more abstract than "objects" in that they might contain affordances relating to multi-body relationships like stacks and assemblies. In this chapter, we examine how to build catalog models for features relating to a single, real-world object that the robot is exposed to. In the next chapter, we will investigate how catalogs can incorporate multi-object affordances.

6.2.1 A Probabilistic Method for Exploring Catalogs

Catalogs of affordances can be explored using the n-armed bandit formalism for maximizing return and credit-assignment (Sutton & Barto, 1998). Consider a learning agent that receives a real-valued reward, $r \in \mathbb{R}$ (possibly stochastic), as a result of taking n possible actions. The goal of the agent is to maximize return by repeatedly taking actions and receiving reward. As it gains more experience, the agent updates its estimates of how much reward is likely to be received per action in order to inform its future selections. If $Q(a)$ represents the expected reward for taking action $a \in \mathcal{A}$, then the agent selects the best action at each time-step a^* such that $a^* = \text{argmax}_a Q(a)$. If $Q(\cdot)$ is known *a priori* this process is trivial. If it is not, it can be estimated from experience by the update rule:

$$Q^{k+1}(a) \quad \leftarrow \quad Q^k(a) + \alpha\big(r^k - Q^k(a)\big) \tag{6.4}$$

where k is the time-step, α is a positive constant step-size, and r^k is the reward received after taking action a. At each decision point, an action a is selected—either greedily or with some amount of exploration—and tested. One exploration technique (called softmax exploration) selects each action according to the Boltzmann distribution, with probability

$$\frac{e^{Q(a)/\tau}}{\sum_{a' \in \mathcal{A}} e^{Q(a')/\tau}} \tag{6.5}$$

where τ is a temperature parameter with initial value $\tau^0 = 1$ and decay rate 0.99. The n-armed bandit problem is a simple form of the reinforcement learning formalization in which there is one state and n actions.

To explore an affordance catalog, the robot selects a feature, $\sigma_i \in \Omega_\sigma$, and instantiates an n-armed bandit problem with the action set containing all the control basis schema parameterized by σ_i, such that $\mathcal{A} = \{$SearchTrack(σ_i), ReachTouch(σ_i), BimanualTouch(σ_i), VisualInspect(σ_i,σ_j)[1]$\}$. Using the reward function provided in Equation 6.2 in conjunction with this action set, the n-armed bandit formulation allows the robot to explore and estimate its affordance models. The procedure for catalog exploration for a set of actions $\mathcal{A}$ is shown in Algorithm 3. Line 22 shows

[1]Note that the VisualInspect action takes an additional feature, $\sigma_j \in \Omega_\sigma$. Many possible additional features may exist for a given object, and thus many Track affordances uncovered through VisualInspect may exist for it. The robot updates its catalog for the object with affordances relating to each of these additional features.

Algorithm 3 ExploreCatalog($\mathcal{A}$)

```
1: $k \leftarrow 0$
2: $\epsilon \leftarrow$ a small positive constant
3: $\rho \leftarrow$ a small positive cost
4: $\forall\,(a_i \in \mathcal{A})$, initialize $\Sigma_i^0 = 1$ and $Q^0(a_i) = \mathbf{0}$
5: repeat
6:    $reward \leftarrow$ false
7:    $a \leftarrow \text{argmax}_{a'} Q^k(a')$ (via softmax exploration)
8:    observe features $f \in \mathcal{F}$
9:    execute $a$
10:   $k \leftarrow k + 1$
11:   if $a$ is rewarding by Equation 4.4 then
12:      $reward \leftarrow$ true
13:      update $Pr(f|reward, a)$
14:      observe variance $\Sigma^k$ of $Pr(f|reward = \text{true}, a)$
15:      evaluate $r^k$ by Equation 6.2
16:   else
17:      $r^k \leftarrow -\rho$
18:   end if
19:   $Q^k(a) \leftarrow Q^{k-1}(a) + \alpha\big(r^k - Q^{k-1}(a)\big)$
20:   update $Pr(reward|f, a)$
21:   update $Pr(f|a)$
22: until $\forall\,(a_i \in \mathcal{A})\,\big(|\Sigma_i^k - \Sigma_i^{k-1}| < \epsilon\big)$
```

how the ExploreCatalog() procedure is run until all of the affordances models for an action set $\mathcal{A}$ habituate.

This set of actions will allow the robot to test simple visual- and force-domain affordances. SearchTrack tests if a feature is trackable, ReachTouch tests if the corresponding Cartesian location is touchable. BimanualTouch provides a simple way of checking if touchable objects are also liftable (if it is, the robot is likely to uncover an additional touch affordance with its other hand), and VisualInspect augments the catalog with additionally trackable visual features (via Track). In general, the visual feature set could grow across all channels of hue, saturation, or intensity (or even other features like texture). In the following demonstrations, only other hues will be considered for simplicity.

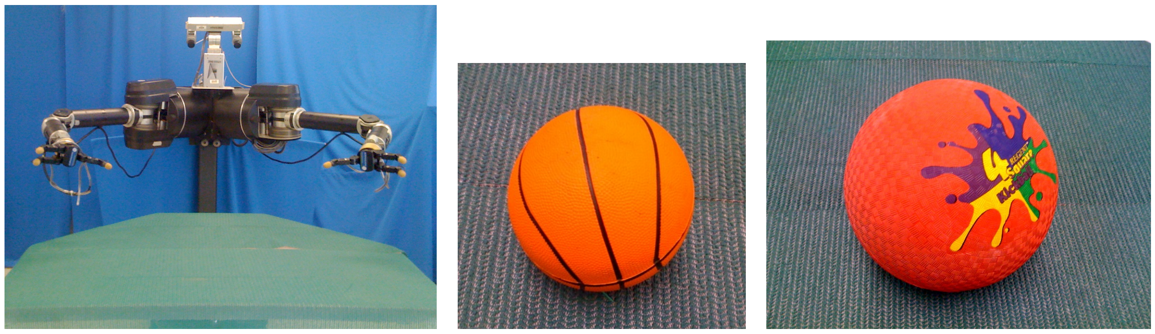


Figure 6.1. Dexter's first three objects: (a) a large green table, (b) a small basketball, and (c) a red ball with colored splotches on it.

It should be noted that when the robot performs each of its schematic actions while exploring affordance catalogs, it can apply all of the declarative and procedural knowledge it has previously learned by the methods discussed in Chapters 4 & 5. For example, to evaluate Touch-ability, the robot will use its right or left hand (or both) as appropriate to maximize the likelihood of getting a controllable tactile response. This capability exploits the dexterity of the robot to evaluate affordances in many different ways depending on the character of the object and other facets of the run-time context.

6.2.2 Demonstration: Learning Three Catalogs

We now provide three examples of the robot Dexter exploring affordance catalogs using the ExploreCatalog() procedure. In the first example, we place a large green table in front of the robot. In the second, we place a small orange basketball on that table. In the third, we place a larger red ball with small blue and yellow features on the table. These three objects are shown in Figure 6.1. Performance is evaluated both on Dexter and in a simulation environment that generates feature values according to models learned in the real robot trials.

6.2.2.1 Experimental Setup

At the beginning of the first demonstration, the robot begins by searching over 10 hue-space channels for a detectable TRACK affordance via SEARCHTRACK and discovers a large green "blob" corresponding to the table the field of view. It then tests whether this "blob" affords other TRACK and/or TOUCH actions via REACH-TOUCH, BIMANUALTOUCH, or VISUALINSPECT. This latter action will discover any additional visual affordances via inspection. A NOOP action with no cost ($\rho = 0$) is included in its action set to provide a choice to take when all models have converged. This set is defined as $\mathcal{A}_1 = \{$NOOP, SEARCHTRACK(GREEN), REACH-TOUCH(GREEN), BIMANUALTOUCH(GREEN), VISUALINSPECT(GREEN,$\cdot$)$\}$. The VI-SUALINSPECT action will be parameterized at run time; at the point when the second visual tracking controller is about to run. At this point, the robot searches its observable hue-channels for additional (non-green) features. For every channel that provides a response, an adding an additional VISUALINSPECT action to the action set $\mathcal{A}_1$.

As in Chapter 5, we define the feature spaces to be the first order dynamics of the input signals to each schema's control actions. The feature space for the TRACK affordance models uncovered by the SEARCHTRACK and VISUALINSPECT schema is a five dimensional vector, $\boldsymbol{f}_v = [\boldsymbol{\gamma}^l_{hue,i}\ \dot{\boldsymbol{\gamma}}^l_{hue,i}\ \upsilon_\gamma]$, containing the heading on the robot's left camera image to hue-feature i, $\boldsymbol{\gamma}^l_{hue,i}$ (for SEARCHTRACK, this feature relates to heading towards green features ($i = 5$); for the VISUALINSPECT, this feature relates to the heading towards the additional feature of appropriate channel), its area υ_γ, and its velocity, $\dot{\boldsymbol{\gamma}}^l_{hue,i}$. The feature space for the TOUCH affordance models uncovered by the REACHTOUCH and BIMANUALTOUCH schema is a seven-dimensional vector, $\boldsymbol{f}_x = [\mathbf{x}_{hue,i}\ \dot{\mathbf{x}}_{hue,i}\ \upsilon_x]$, containing the Cartesian location, $\mathbf{x}_{hue,i} \in \mathbb{R}^3$, of the table found by triangulating the view of the hue-space pixel regions in both of the robot's camera images, that region's estimated 3D volume, υ_x, and its velocity, $\dot{\mathbf{x}}_{hue,i}$. We define the affordance models for each of these behaviors to be multi-dimensional

Gaussians over these corresponding feature spaces. The models are initialized with zero mean and unit variance.

Instead of exploring each catalog until all models habituate (Line 22 of the ExploreCatalog() procedure), each trial in the following examples consisted of 100 actions. This was done to gather a consistent number of data samples for each trial for analysis purposes. In these experiments, $\rho = 0.05$, for all schema. The change in variance after each action is normalized to the maximum value observed during the trial across all affordance models in the catalog to provide a reward signal between 0 and 1. If the action does not successfully complete (it does not achieve reward from the affordance discovery motivator), no information is gained, and r^k is evaluated as pure cost. Using this approach, we expect the robot to take actions according to its confidence in the corresponding affordance model. When the expected information gain for taking an action becomes sufficiently low, it should favor testing other affordances in the action set. Eventually, we would expect the robot to select the NoOp action over its other actions as their cost eventually outweighs the expected information gain, and NoOp has no cost. When all of the models have habituated, the robot will have a relatively confident assessment of which affordances can be associated with the initial feature σ_i.

In the second and third examples, we place the basketball and the red ball, respectively, on the table and the robot follows a similar process. The action sets $\mathcal{A}_2$ and $\mathcal{A}_3$ for these respective examples are similar to $\mathcal{A}_1$, except that they use headings and positions toward orange and red pixel regions, measured on visual hue-channels 3 and 0, respectively, in place of the headings toward green pixels measured on hue-channel 5.

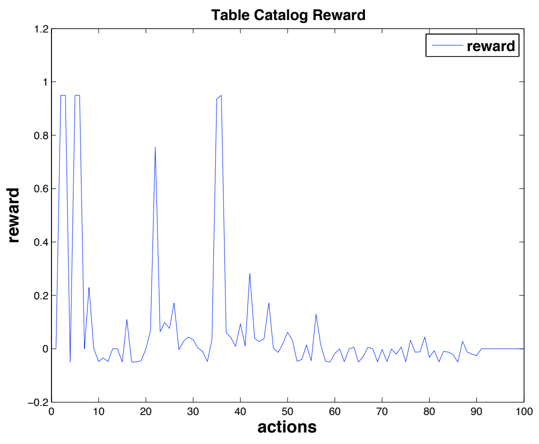
After a single trial of 100 training actions for each of these examples, 25 addition simulated trials were performed using the estimated affordance models of the form

$Pr(f|reward, a_i)$ learned on the robot to generate simulated samples. We now discuss of the results of both the real and simulated examples.

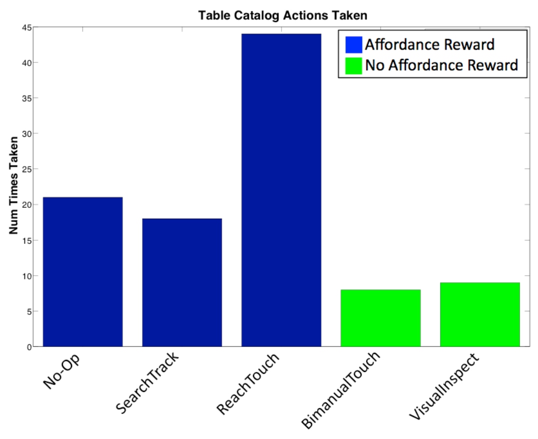
6.2.2.2 Results

Figure 6.2 shows the results of exploring the green table affordances. Figure 6.2(a) shows the reward after each action for the real robot trial. We see a number of large spikes during the first fifty actions, before the reward levels out to approximately 0 for the remainder of the trial Figure 6.2(b) shows the number of times each of the actions in $\mathcal{A}_1$ were selected during this trial. The number of successful (rewarding) actions are shown in blue, the unsuccessful (not rewarding) actions are shown in green. This figure shows that SEACHTRACK and REACHTOUCH were rewarding 100% of the time they were selected. The BIMANUALTOUCH and VISUALINSPECT actions were never rewarding because the table, although consistently TRACK-able and TOUCH-able, is not transferable between the robot's hands (it is large and the robot could not grab it with REACHTOUCH), and it has no additional hue-space visual features that are TRACK-able. Over the course of the 100 actions, the robot selected the SEARCHTRACK action 18 times, the REACHTOUCH action 44 times, and the BIMANUALTOUCH and VISUALINSPECT actions only 8 and 9 times. These latter actions were selected relatively few times because they resulted in pure cost, and no information. The NOOP action was selected 21 times, presumably after the affordance models for the other four actions habituated.

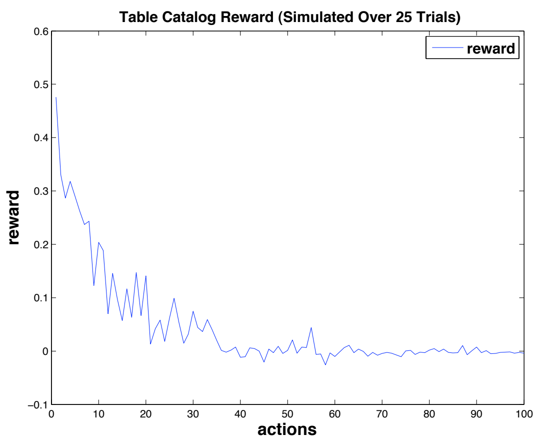
Using the affordance models and likelihood of reward for each behavior learned in the real robot trial to generate samples, 25 additional trials were performed in simulation. The average reward over these trials in Figure 6.2(c). This plot more clearly shows that, on average, the reward for interacting with the table goes down to a negligible amount after about 40-50 actions as the affordance models habituate. Figure 6.2(d) shows the change in variance for the SEARCHTRACK and REACHTOUCH



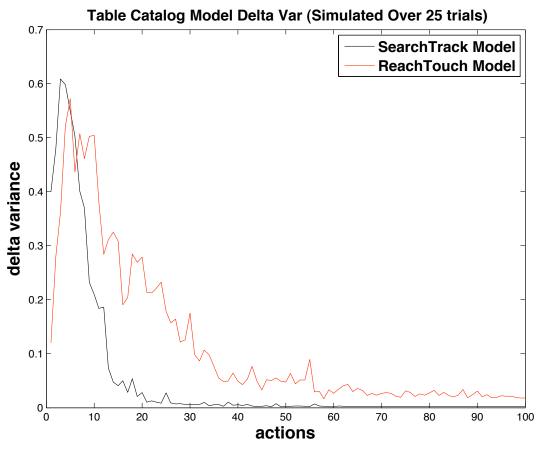
(a)



(b)

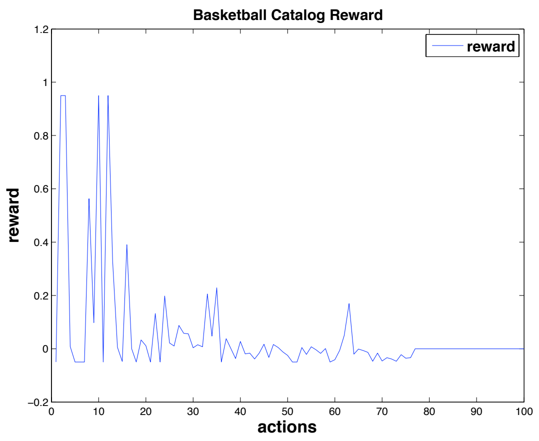


(c)

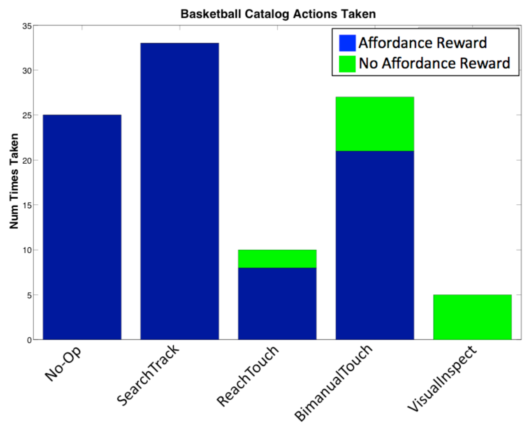


(d)

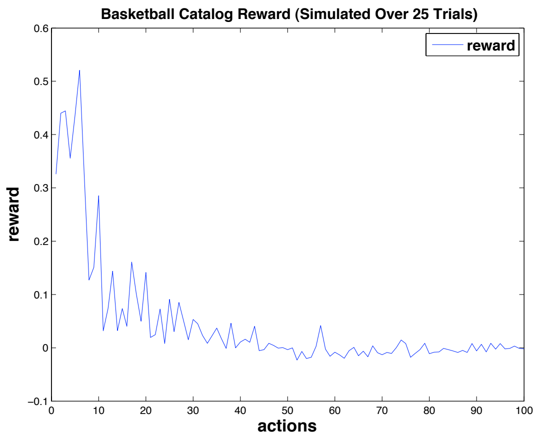
Figure 6.2. These plots show the results of exploring the affordance catalog for the green table placed in front of Dexter. Plot (a) shows the reward received after each action for the real robot trial, while chart (b) shows the number of times each action was taken during this trial. Successful actions (in which affordance discovery reward was achieved) are shown in dark blue, unsuccessful actions are shown in light green. Plot (c) shows the average reward for the simulated trials, more clearly showing the overall habituation. Plot (d) shows the actual change in variance, averaged over the simulated trials, for the SearchTrack and ReachTouch affordance models.



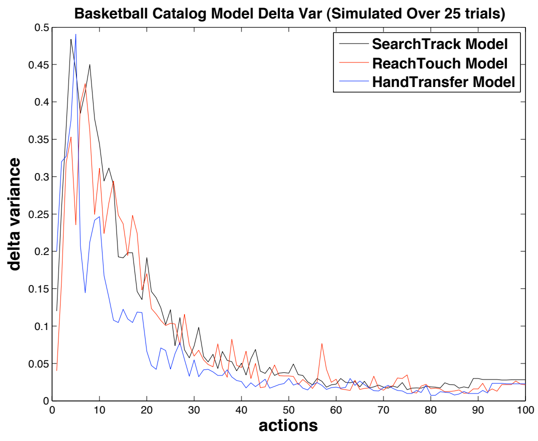
(a)



(b)

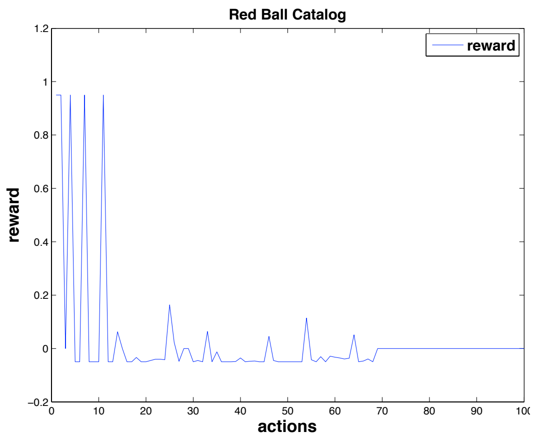


(c)

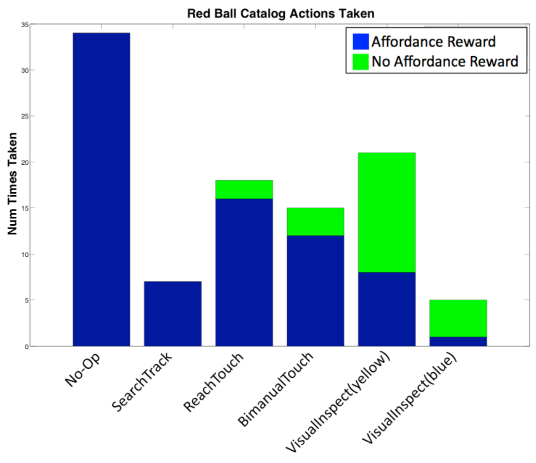


(d)

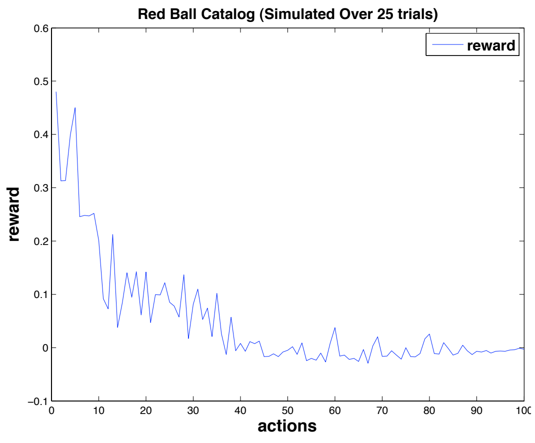
Figure 6.3. These figures show the results of exploring the affordance catalog for the basketball placed on the table in front of Dexter. Plot (a) shows the reward received after each action for the real robot trial, while plot (b) shows the number of times each action was taken during this trial. Plot (c) shows the average reward for the simulated trials, more clearly showing the overall habituation. Plot (d) shows the actual change in variance, averaged over the simulated trials, for the SearchTrack, ReachTouch, and BimanualTouch affordance models.



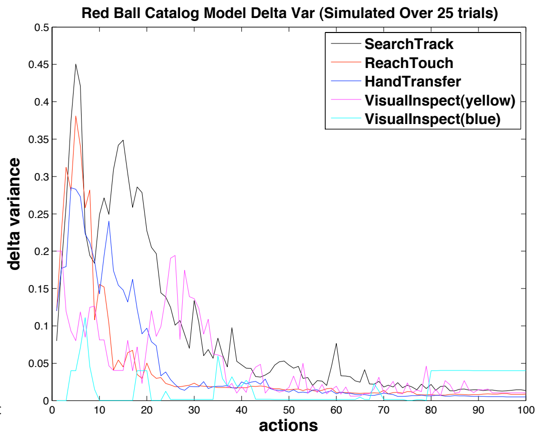
(a)



(b)



(c)



(d)

Figure 6.4. These figures show the results of exploring the affordance catalog for the red ball placed on the table in front of Dexter. Plot (a) shows the reward received after each action for the real robot trial, while plot (b) shows the number of times each action was taken during this trial. Plot (c) shows the average reward for the simulated trials, more clearly showing the overall habituation. Plot (d) shows the actual change in variance averaged over the simulated trials for the affordance models built by employing the SearchTrack, ReachTouch, BimanualTouch and two VisualInspect behavior.

affordance models averaged over the 25 simulated trials as actions are taken. It shows that it took fewer actions to habituate on the SEARCHTRACK model (about 20) than on the REACHTOUCH model (about 50-60).

Figure 6.3 shows the results of exploring the affordances of the small orange basketball. Figure 6.3(a) shows the reward received after each action was taken during the real robot trial. We see large spikes early in the trial, but the reward levels out to approximately 0 after about 40 actions with only occasional small spikes thereafter. Figure 6.3(b) shows the number of times each of the actions in $\mathcal{A}_2$ were selected during the trial. This figure shows that SEARCHTRACK was deterministic in its success, but REACHTOUCH occasionally failed (about 10% of the time). This occurred when the robot accidently knocked the ball off the table or the ball was placed out of the robot's reach. Unlike the table, however, the basketball afforded a number of rewarding BIMANUALTOUCH interactions (succeeding about 75% of the time). The robot would fail on this action if it accidently knocked the ball off the table or dropped it before the second TOUCH controller could converge. The basketball, like the table, however, had no additional features TRACK-able via the VISUALINSPECT schema.

Figures 6.3(c) and 6.3(d) show the results of the 25 simulated trials in which samples were generated according to the models acquired in the real robot trial. As with the table, we see the clear overall habituation in the average reward plot, as well as the habituation for the three affordance models in the delta-variance plot.

Figure 6.4 shows results of exploring the affordances of the red ball. The reward received after each action was taken during the real robot trial is shown in Figure 6.4(a). Figure 6.4(b) shows the number of times each of the actions in $\mathcal{A}_3$ were selected during this trial. Unlike the table and basketball demonstrations, the red ball provides two additional TRACK-able affordances via the VISUALINSPECT schema, corresponding to the colored patches on its surface. Because these colors only appeared when the robot grabbed the object in particular orientation, these features do not deterministi-

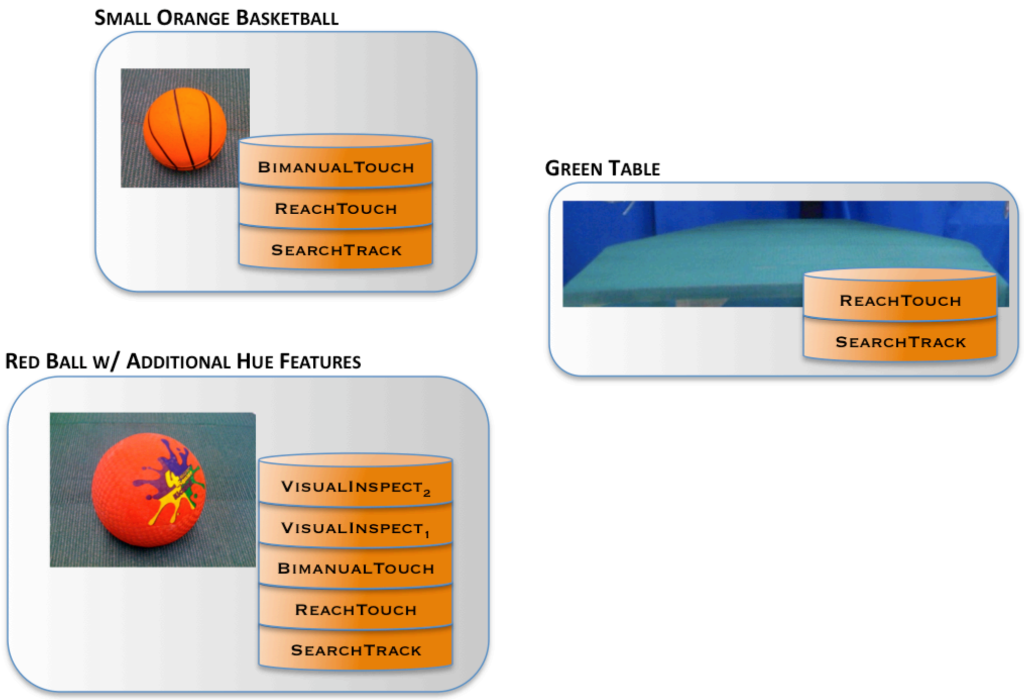


Figure 6.5. Iconic representations of the three catalogs Dexter learned for the table, the red ball, and the small basketball. Each affordance associated with the catalog is shown stacked in orange.

cally appear. In fact, Figure 6.4(b) shows that both additional features led to reward less than half of the time that VisualNspect was invoked. Figures 6.4(c) and 6.4(d) show the results for the corresponding simulated experiment.

The above demonstrations show how Dexter can sample purely visual features in its environment and build affordance-based catalog models with respect to those features and its behavioral capabilities. The robot can explore each of these affordances until it gains some confidence in its internal cognitive models. Figure 6.5 shows an iconic representation of the robot's three catalog models at the end of the robot's exploration stages. The affordances are represented in layers of capabilities associated together into primitive "catagories" that the robot can use to interpret its world. We also see, as in the case of the red ball, that multiple procedural instantiations of a program—here the VisualInspect behavior—can be associated in the same catalog because they orient the robot to uncovering a variety of related affordances (e.g., Track-ability). In the next chapter, we will examine how multi-object affordances can be added to these catalog models relating, for example, the orange ball to the table through a "placeable" affordance. Before we move on, however, we present a second methodology for estimating affordance models in which their dynamics are represented explicitly in a state description that can be explored using reinforcement learning and the reward function presented in Equation 6.2.

6.3 Model Exploration Programs

We now define a class of MDPs called *model exploration programs*. These MDPs have action set $\mathcal{A}$. State is evaluated as an $|\mathcal{A}|$-dimensional vector that captures the confidence of each of the actions' corresponding affordance model. Reward is evaluated according to Equation 6.2. The status of each affordance model is defined by a 4-valued predicate as follows:

$$p(\dot{\Sigma}_i) = \begin{cases} \text{X} & : \; Pr(f|reward, a_i) \text{ is unknown} \\ - & : \; Pr(f|reward, a_i) \text{ has undefined features } f \\ 0 & : \; |\dot{\Sigma}_i| > \epsilon_m \\ 1 & : \; |\dot{\Sigma}_i| \leq \epsilon_m, \end{cases} \tag{6.6}$$

where ϵ_m is a small, positive constant, Σ_i is the variance of model $Pr(f|reward, a_i)$, and $\dot{\Sigma}_i$ is its rate of change in model variance as experience is gathered.

Before the robot has experience with action a_i, its affordance model will be unknown, and $p(\dot{\Sigma}_i)$ = "X". If the environment does not afford a behavior at a given time the corresponding predicate value will be undefined, and $p(\dot{\Sigma}_i)$ = "−". This occurs when the relevant input signals to the action are not present, and the dynamic status of the behavior can not be evaluated. In all other cases, the predicate value for an action is either 0 or 1 depending on the quiescence of the affordance model. It is worth noting the similarity of this state representation to the state representation in control basis programs; one captures the dynamics of a robot's actions, the other captures the changing properties of its internal knowledge structures. These cognitive models, however, are based entirely on experience and are not guaranteed to converge as asymptotically stable navigation functions do. Nevertheless, estimating the quiescence of affordance models provide a useful tool for describing the "state" of the robot's knowledge structures that can bias the robot's exploratory actions to areas of its world that it is uncertain about.

6.3.1 Demonstration: Exploring Affordance Models

To provide a demonstration of how model exploration programs may be used in practice, we present three simple examples in which Dexter engages objects in its workspace. Because it is our goal in this section to examine the performance of model exploration programs, small state/action spaces are provided for each program by the programmer *a priori*.

In each of the following three demonstrations, the robot uses a model exploration program to guide its behavior in estimating multi-dimensional Gaussian affordances models for each of the program's actions. For each demonstration, a single trial was performed on Dexter, and twenty-four additional trials (for a total of twenty-five) were performed using the robot simulator. The simulation experiments used the probabilistic models learned in the real trial (the likelihood of success for each action given the observed context, and the estimated affordance model for each action) to generate realistic samples. The empirical results presented are averaged over all twenty-five trials. This approach was taken to provide a reasonable approximation of what the average of many real robot trials would look like.

6.3.1.1 Experimental Setup

In all three experiments, a model exploration program is instantiated and explored using Q-learning with ϵ-greedy exploration ($\epsilon = 0.05$) and an action penalty of $\rho = 0.05$. A small exploration constant was chosen to inject some stochasticity into the action selection process even though exploration arises naturally from the non-stationary aspects of the habituation metric. At the beginning of each trial, the robot has no experience and thus the state of its models are unknown ("X"). As the robot performs actions it receives reward by reducing the change in variance of the corresponding affordance models. The action becomes unrewarding when the change of the variance drops below $\epsilon_m = 0.1$ between successive executions (i.e., the model quiesces). Each model is updated after the execution of the corresponding action if reward from the affordance discovery motivator is received. The models are not updated when the action does not succeed, accumulating no reward from the affordance discovery motivator, only cost. Performance is analyzed over 25 trials (1 real, 24 simulated) of 100 actions each. The change in variance after each action was normalized to the maximum value observed during each trial across all affordance models in the program. We describe each of the three experiments next.

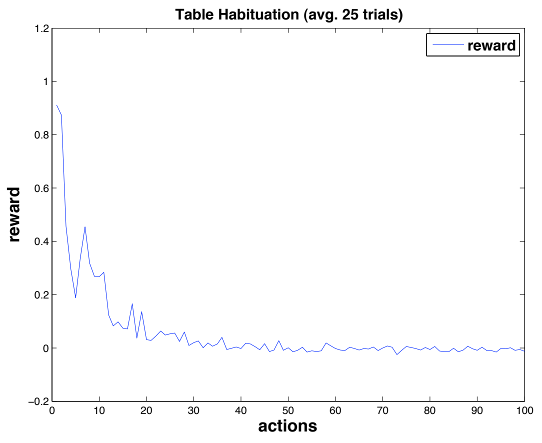
Dexter's Table: In the first example, Dexter's first catalog object—the green table illustrated in Figure 3.2—is placed in the robot's workspace. The robot explores the table affordances and the NoOp action, such that $\mathcal{A}_1 = \{$NoOp, SearchTrack(green), ReachTouch(green)$\}$ and creates a model exploration program that assess these affordances.

Three Moving Objects: In the second example, three different colored objects (a red, a yellow, and a blue ball) are placed on the green table in front of the robot. The robot samples the set of possible new Touch affordances that can be parameterized by the colors of these objects to build a model exploration program with action set $\mathcal{A}_2 = \{$NoOp, ReachTouch(red), ReachTouch(yellow), ReachTouch(blue)} and corresponding state space. During the trial, the objects are placed in random locations on the table. After each action, the positions of the red, yellow, and blue balls on the table are moved to locations that differ from their original position with (approximate) variances of 0.01 m, 0.05 m, and 0.25 m, respectively. We hypothesize that the robot will explore each object in proportion to the variance—it will engage the blue object more than the yellow or red object, and the yellow object more than the red.

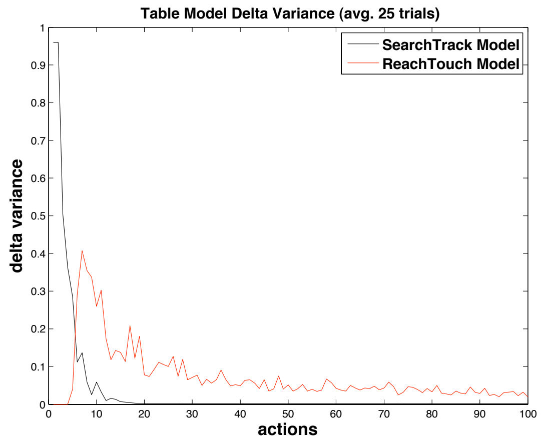
Novel & Surprising Objects: In the third example, a two-predicate model exploration program is instantiated in which Dexter explores the action set $\mathcal{A}_3 = \{$NoOp, ReachTouch(red), ReachTouch(blue)$\}$ and their corresponding affordances. For the first 25 actions, a red ball is placed in locations on the table. After these actions, a blue ball is placed on the table. After 25 additional actions are taken, the position of the red ball is moved to a different location on the table 25 centimeters from its original location. In this example, we hypothesize two results: 1) Dexter will engage the novel blue object after it has habituated the affordances of the red object because it will produce more reward, and 2) Dexter will exhibit a form of "surprise" by re-engaging the habituated red object when it observes variations in its estimated model.

6.3.1.2 Results

Figure 6.6 shows the performance of the model exploration program for the table experiment. We see in Figure 6.6(a) the reward received after each action is taken (averaged over the 25 trials). This figure shows that after 20-30 actions, the robot receives negligible reward for taking either of these two actions. Figure 6.6(b) shows the change in variance for each of the affordance models in the schema. We can clearly see the switch that occurs after about five or six actions when the change in



(a)



(b)

Figure 6.6. Plot (a) shows the reward (averaged over all 25 trials—real and simulated) received after each action for the model exploration program that tracks and touches the green table. Reward becomes negligible after 20-30 actions. Plot (b) shows the change in model variance for the two affordance models used in the schema.

variance of the Touch affordance model starts to outweigh the habituating Track affordance model.

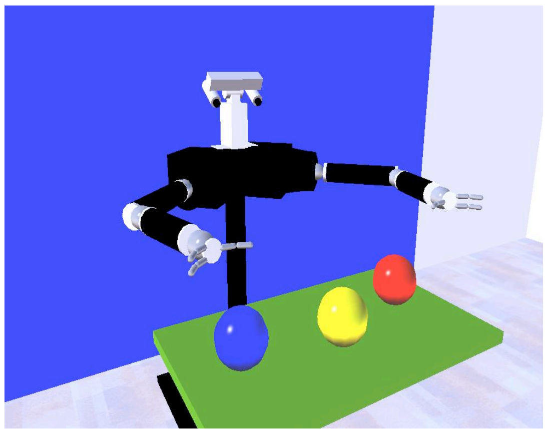
Figure 6.7 shows the performance of the model exploration program for the three moving objects experiment. In these experiments, it takes the robot about 30-40 actions to habituate on its three affordance models, as seen in Figure 6.7(b). As expected, we see in Figure 6.7(c) that it takes each of the three models an amount of time to quiesce that grows with the variance in which the balls are placed. Figure 6.7(d) shows the average number of actions taken to explore each of the colored objects before model quiescence, averaged over the 25 trials. It clearly shows that it takes increasingly more actions to quiesce on objects with more variable locales. On average about 7 actions are taken before the robot habituates on the red object, about 18 for the yellow object, and about 27 for the blue object.

The final experiment shows that model exploration programs also yield behavior that is typically described as reacting to "novelty" and "surprise." Figure 6.8 shows the reward and variance plots for this experiment. The robot quickly habituates to the red object and attends to the novel blue object when it appears after 25 actions. Exploration directed at this affordance habituates after about 25 more actions. Also, when the position of red object changes, it surprises the robot, temporarily creating more uncertainty in the affordances for ReachTouch(red), causing the robot to re-engage that object, and to receive reward until it habituates once again.

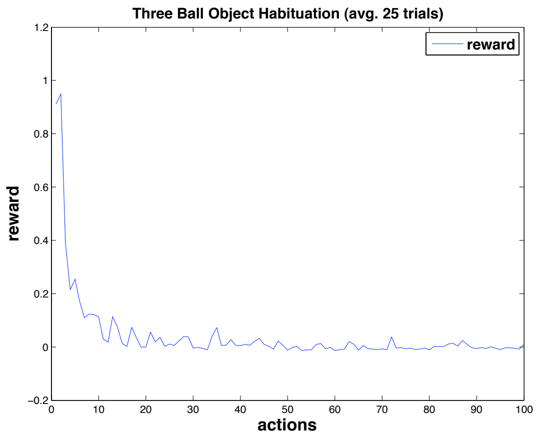
6.4 Discussion

In this chapter, we demonstrated how the control basis framework supports intrinsically motivated affordance modeling. As a robot builds grounded behavioral programs according to the affordance discovery motivator, it can explore the environmental conditions under which these behaviors succeed. We provided three demonstrations in which Dexter explored affordance catalogs until it was confident in its internal models. These catalogs provide a categorical means by which a robot can interpret its environment entirely in terms of its ability to interact with it. We also provided three demonstrations in which a robot explored programs that utilize a state representation capturing the dynamic state of its internal models. Both exploration methodologies utilize the same reward function to assign credit to an estimate of the robot's confidence.

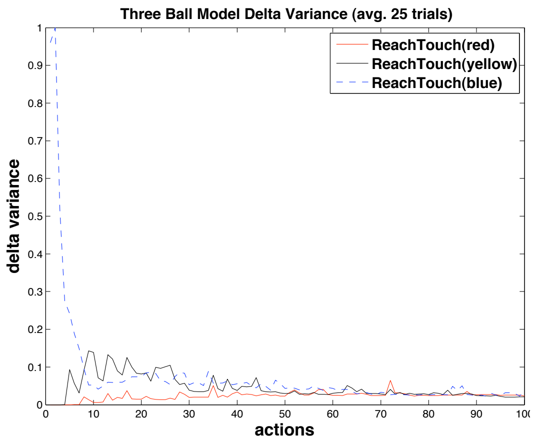
Although the empirical demonstrations in this chapter were performed in simple contexts with simple objects, we expect the methodology to apply in more sophisticated situations as well. In the next chapter, Dexter undergoes an additional learning stage in which it acquires a PickAndPlace behavior. This behavior provides a new means for the robot to interpret features it encounters and, as we will demonstrate, can ground multi-object affordances such as "stackability" and "insertability."



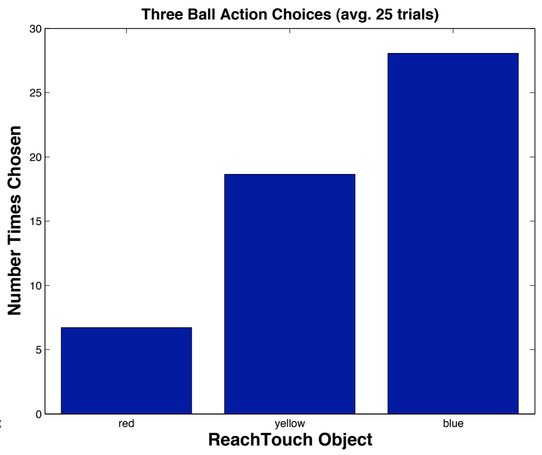
(a)



(b)

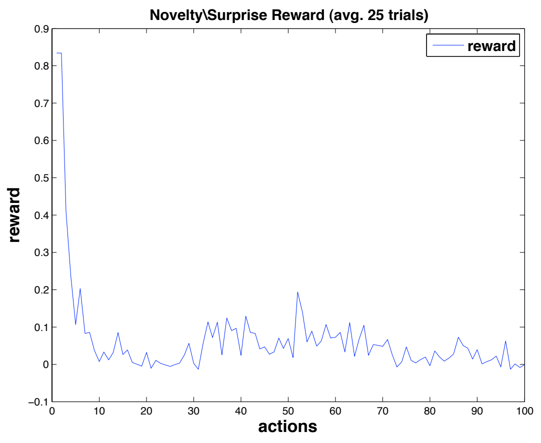


(c)

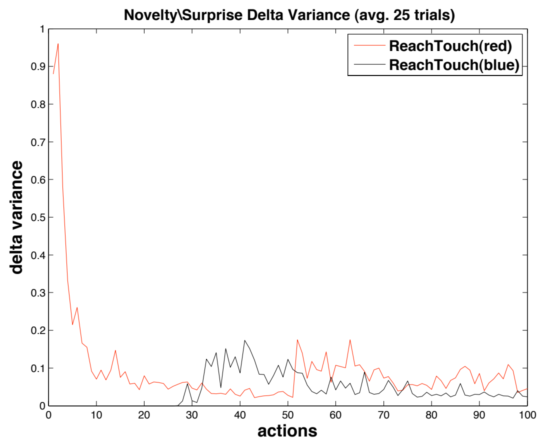


(d)

Figure 6.7. Frame (a) shows a screenshot of the simulation environment for the three ball example. Plot (b) shows the reward (averaged over all 25 trials—real and simulated) received after each action for the model exploration program that reaches out and grabs the three colored balls. Plot (c) shows the change in model variance for the three affordances used in the program. The chart in (d) shows the number of times each action is taken before the corresponding affordances quiesce.



(a)



(b)

Figure 6.8. Plot (a) shows the reward (averaged over all 25 trials—real and simulated) received after each action for the model exploration program for the third experiment in which the second object only becomes available after 25 actions, and the first object changes location after 50 actions. Plot (b) shows the change in model variance for the two affordance models used in the schema.

CHAPTER 7
MULTI-BODY RELATIONSHIPS

Up to this point, we have examined how a robot can acquire behavioral knowledge by engaging combinations of hierarchical control circuits. This framework was demonstrated in applications where a robot developed skills with which to visually search for objects that it could then touch, grab, transfer between its hands, and inspect more closely. This chapter proposes a process by which a robot can interact with and manipulate *multiple* objects in a controlled manner. We will first examine how a robot can bring two objects into contact with each other, and then discuss how it can explore the visual- and force-domain affordances that arise when it does. During this exploration, the robot can apply all of its prior behavioral knowledge—learned over the course of its development—to ground new categorical descriptions using the techniques introduced in the last chapter. We conclude with a discussion of how this approach results in models of simple two-body relations that can form the basis of a variety of further intrinsically motivated exploratory behavior.

The ability of a robot to bring two objects into contact with each other in a controlled manner has been studied extensively in the literature. Typical approaches divide the task into pieces that can be planned independently with constraints designed to meet global objectives (Lozano-Pérez, 1981; Brooks, 1983; Lozano-Pérez et al., 1989). These systems are difficult to employ because they require highly calibrated devices and fully observable environments. Furthermore, these approaches formulate the task as a search problem that has little to do with the robot's prior experience in similar tasks.

7.1 PickAndPlace

In this section, we discuss a developmental stage that was constructed for Dexter to acquire and generalize a skill called PickAndPlace. This skill provides a principled means of bringing the features of one catalog into contact with the features of another and affords the opportunity to model how pairs of catalogs create controllable visual- and force-domain relationships that are associated with stable multi-body relationships. Through a process of accommodation and assimilation, Dexter acquires this behavior in a simple training context and then generalizes it to new situations that require more sophisticated strategies for force-domain interactions.

7.1.1 Declarative Structure

During the accommodation stage of learning PickAndPlace, we designate a small state/action space that leverages existing knowledge efficiently and provides a simple new learning context to make the desired declarative structure conspicuous. To do this, we provide control basis actions to the robot to construct a new three-predicate state space. The first predicate describes the status of the ReachTouch program that, as we have seen, finds, reaches to, and grabs some simple objects. We will use a ReachTouch referenced exclusively to headings toward highly-saturated regions on the camera's image plane (as in the previous declarative learning episodes). ReachTouch incorporates common sense procedural knowledge regarding handedness (as presented in Chapter 5).

The other two predicates provide the dynamic state of the following two controllers:

Transport moves the position of an object held in Dexter's right hand, $\mathbf{x}_{obj} \in \mathbb{R}^3$, to a goal location, $\mathbf{x}_g$, by following the gradient of a harmonic potential field, $-\nabla\phi_h$. This path avoids collisions with other features detectable in Ω_σ. The goal is defined by the Cartesian centroid of a particular hue, saturation, or intensity pixel region present in the visual feedback from both of Dexter's camera images. In the training context, a possible reference for this goal is defined by the centroid location of the *green* hue-channel, $\mathbf{x}_{hue,3}$, corresponding to the experimental table (the rest of the table is designated as an obstacle). Because harmonic functions are not navigation functions (they are not admissible) their gradient can not be used directly as a control signal. Instead, we compute a small relative displacement in the direction of the gradient that is used as a bounded-input signal to the virtual spring potential function, ϕ_s, with a $\mathbf{0}$ reference. This displacement is proportional to the value of the harmonic potential and is defined as:

$$\Delta\mathbf{x} \triangleq -\frac{\nabla\phi_h}{||\nabla\phi_h||}\phi_h. \tag{7.1}$$

This simple strategy yields reasonable performance in a variety of situations. $\Delta\mathbf{x}$ is bounded in the range $[0,1]$ and will decrease as the robot reaches the goal. The TRANSPORT controller moves to minimize this relative displacement by providing commands to the robot's right arm. It is defined as:

$$\text{TRANSPORT} \triangleq c(\phi_s, (\Delta\mathbf{x}, \mathbf{0}), \boldsymbol{\theta}_{r,arm}), \tag{7.2}$$

resulting in joint displacements for the right arm such that

$$\begin{aligned} \Delta\boldsymbol{\theta}_{r,arm} &= -\left(\frac{\partial\phi_s(\mathbf{0},\Delta\mathbf{x})}{\partial\boldsymbol{\theta}_{r,arm}}\right)^{\#}\phi_s(\mathbf{0},\Delta\mathbf{x}) && (7.3)\\ &= \left(\Delta\mathbf{x}^T\mathbf{J}_{r,arm}\right)^{\#}\left(\frac{1}{2}\Delta\mathbf{x}^T\Delta\mathbf{x}\right) && (7.4)\\ &= \frac{1}{2}\left(\mathbf{J}^{\#}_{r,arm}\Delta\mathbf{x}\right), && (7.5)\end{aligned}$$

where $\mathbf{J}_{r,arm}$ is the manipulator position Jacobian for the right arm. TRANSPORT provides a control input based on feedback signals derived from stereo triangulation equations and a harmonic path-plan, not direct feedback signals in $\Omega_{\sigma(env)}$, and is thus not rewarding by the affordance discovery motivator. Because this controller is designed to transport an object that a robot holds in its hand, we will designate it as "undefined" if no object is held.

Place is designed to detect and control a reaction force of $2N$ in the opposite direction of gravity (i.e., in the positive z world frame direction) between a grasped object and another object. Place creates a virtual spring in the force domain to control the net force felt the three fingertip load cells of the robot's right hand, $\mathbf{f}_{r,net}$. To reduce the error $\boldsymbol{\epsilon}= (\mathbf{f}_{ref} - \mathbf{f}_{r,net})$, Place moves the joint angles of the robot's right arm:

$$\textsc{Place} \triangleq c(\phi_s, (\mathbf{f}_{ref}, \mathbf{f}_{r,net}), \mathbf{x}_{r,arm}). \tag{7.6}$$

Place computes control outputs, such that

$$\begin{aligned}
\Delta\mathbf{x}_{r,arm} &= -\left(\frac{\partial\phi_s(\mathbf{f}_{ref}, \mathbf{f}_{r,net})}{\partial\mathbf{x}_{r,arm}}\right)^{\#} \phi_s(\mathbf{f}_{ref}, \mathbf{f}_{r,net}) &(7.7)\\
&= -\left(\frac{\partial\phi_s(\mathbf{f}_{ref}, \mathbf{f}_{net})}{\partial\mathbf{f}_{net}}\frac{\partial\mathbf{f}_{net}}{\partial\mathbf{x}_{r,arm}}\right)^{\#}\left(\frac{1}{2}(\mathbf{f}_{ref} - \mathbf{f}_{net})^T(\mathbf{f}_{ref} - \mathbf{f}_{net})\right) &(7.8)\\
&= \left(\boldsymbol{\epsilon}^T\frac{\partial\mathbf{f}_{net}}{\partial\mathbf{x}_{r,arm}}\right)^{\#}\left(\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right) = \frac{1}{2}\boldsymbol{\epsilon}. &(7.9)
\end{aligned}$$

Because this control action uses the feedback signal $\mathbf{f}_{r,net} \in \Omega_{\sigma(env)}$, it is rewarding by the affordance discovery motivator. Place can easily be re-parameterized to apply to Dexter's left- and two-handed situations.

To accommodate PickAndPlace, consider the action set $\mathcal{A}_{pp} = \{$ReachTouch, Transport, Place, Transport$\triangleleft$Place, Place$\triangleleft$Transport$\}$ and the state set $\mathcal{S}_{pp}$ where state vectors $\mathbf{s} \in \mathcal{S}_{pp}$ are defined as $\mathbf{s} = (p_{rt}\ p_{transport}\ p_{place})$ and p_{rt} is the state of the entire ReachTouch program.

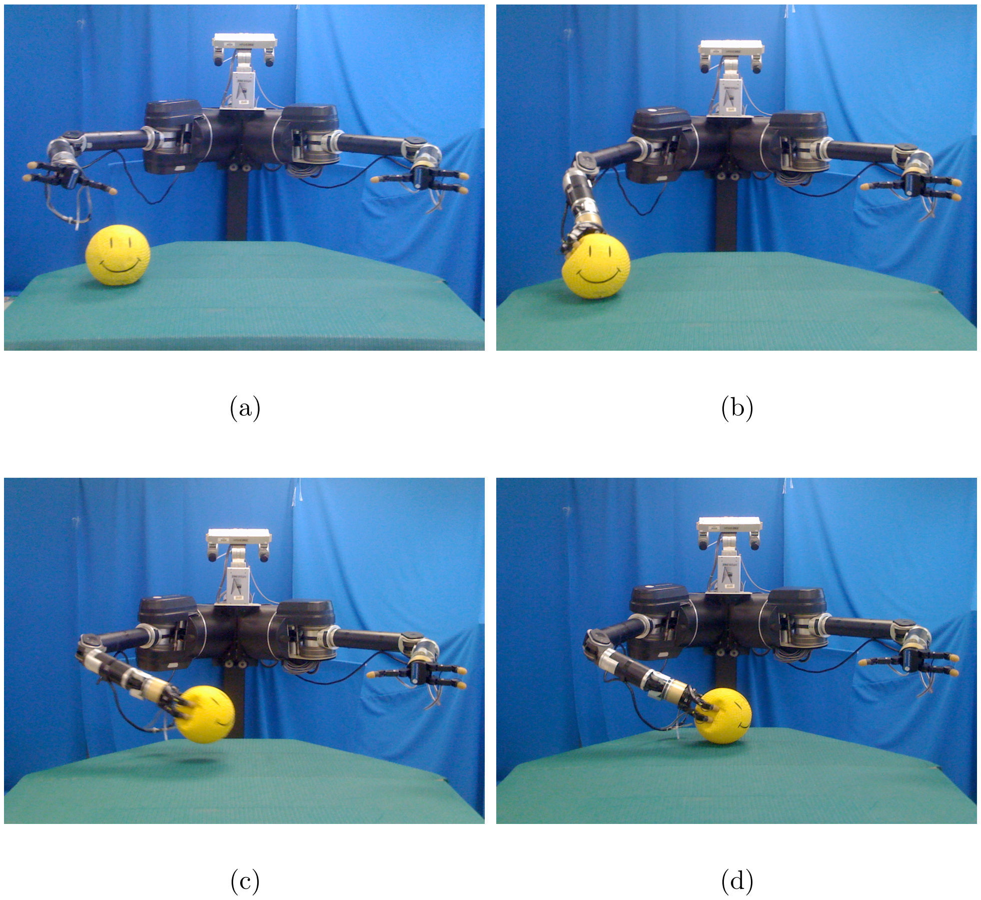


Figure 7.1. The sequence of actions Dexter learned to accomplish the PickAndPlace task. The robot begins by (a) finding a highly-saturated hue in the video stream, (b) reaching to it and performing a multi-fingered ReachTouch, (c) transporting it to the location of the green hue feature (the table), and (d), detecting the resulting reference load when the object comes into contact with the table.

During the accommodation stage, Dexter explores this state and action space using the Accommodate() procedure to find rewarding affordances—in this case, when the environment affords quiescence in the Place controller. Ten trials of 30 learning episodes were performed on the robot. During these learning episodes, the experimenter presents objects with highly-saturated visual hues on the right side of Dexter's table, as seen in Figure 7.1(a).

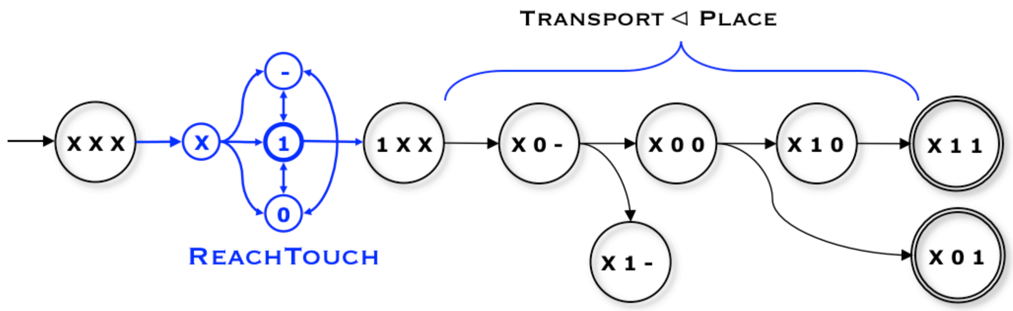


Figure 7.2. The declarative PICKANDPLACE transitions under the policy learned by Dexter after 30 training episodes. Transitions are shown if they occurred with a probability greater than 20%. The robot begins by invoking the integrated REACHTOUCH behavior, accomplishing the "pick" part of the task, and then running the composite action TRANSPORT ◁ PLACE that brings the object into contact with the goal, controlling the resulting reaction forces.

Figure 7.2 shows the resulting transition diagram for the policy learned after a typical trial. After reaching to and grabbing the object (using REACHTOUCH), the robot learned to transport it to a location on the table by means of the TRANSPORT controller. The intrinsic reward is achieved when the reaction load from the supporting surface is detected and controlled with PLACE. This sequence is illustrated in Figures 7.1(b)-7.1(d). The average reward per time step for each episode (averaged over the 10 trials) is shown in Figure 7.3. The average reward approaches 0.15 asymptotically.

7.1.2 Generalizing PickAndPlace

Although it was acquired using only the robot's right arm and with green and highly-saturated objects, the PICKANDPLACE program can be abstracted according to the methods described in Chapter 5 to apply to different contexts. In fact, the behavioral description of this skill is not complete until the robot also discovers how to recognize when it works (i.e., when it successfully leads to reward) and when it does not, and how the run-time situation influences how the abstract policy should be

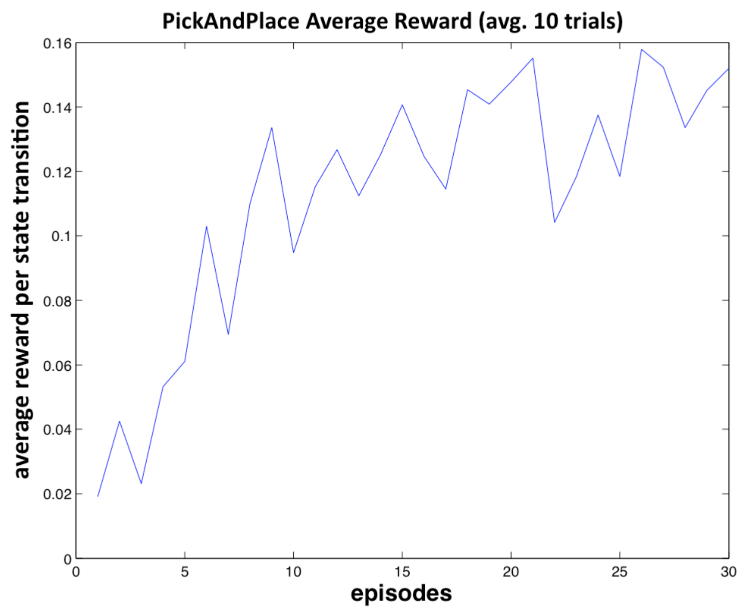


Figure 7.3. This plot shows the average reward per state transition during the accommodation phase of PickAndPlace. The results are averaged over 10 trials of 30 learning episodes each. An ϵ-greedy exploration strategy was used in these experiments with ϵ=0.2.

implemented. In a new learning stage, we therefore expanded the contexts in which Dexter applied PickAndPlace by placing a ball with highly-saturated hues in unconstrained initial locations on the table and by exploring more places to transport it. During training, Dexter learned to assimilate these situations into its PickAndPlace schema. "Place" goals were visually designated by a blue-colored hue spot that was placed in various locations on the table. In this expanded context, there are situations in which the right-handed policy does not yield reward.

The robot uses the C4.5 decision tree learner (Chapter 5) to learn a procedural policy according to the Assimilate() procedure over the joint signal space that arises from the union of the "pick" and the "place" locations. This signal space is defined as $\boldsymbol{f}_{pp} = [\mathbf{x}_{obj}\ \mathbf{x}_{goal}]$, where $\mathbf{x}_{obj}$ is the Cartesian location of the "pick" goal and $\mathbf{x}_{goal}$ is the Cartesian location of the "place" goal.

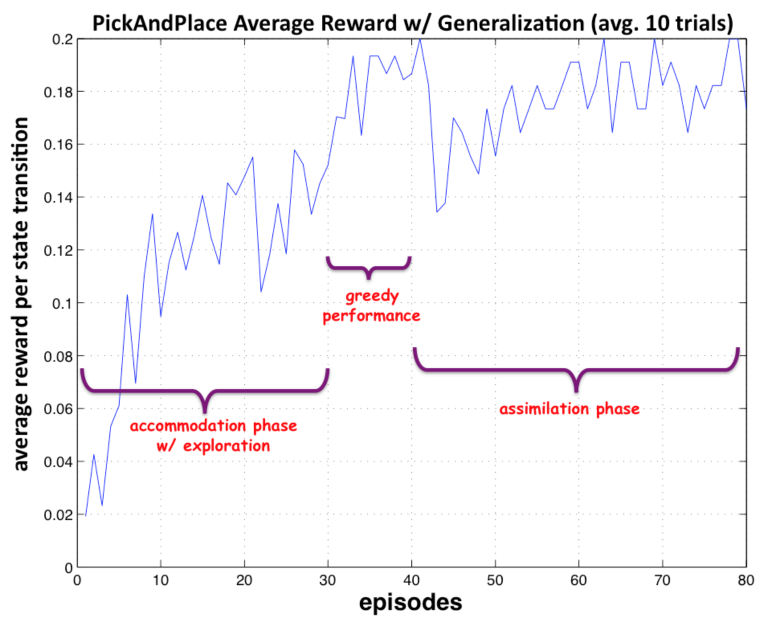


Figure 7.4. The average reward per state transition of the PickAndPlace behavior over all 80 episodes (averaged over 10 trials).

Ten training trials of 40 episodes each were conducted on Dexter. The average reward per state transition during these is episodes is plotted in Figure 7.4 (averaged over the trials). This plot shows the average reward for the accommodation stage (episodes 1-30) where the robot learned the declarative policy, with an additional 10 episodes in which exploration was turned off to reveal the greedy behavior according to the policy pictured in Figure 7.2. Starting at episode 41, the procedural context was challenged as described above, and the robot explored options to assimilate PickAndPlace handedness preferences. We see that after an initial dip in performance (episodes 41-60), the robot was able to re-gain the level of average reward per time step it achieved in the simpler training context (episodes 31-40).

The resulting procedural policy is shown in Figure 7.5. This decision tree distinguishes cases when the object and the goal are sufficiently far away from each other in the robot's y- (lateral) direction. In this case, it should invoke BimanualTouch in place of ReachTouch. Because both these schema reward the same type

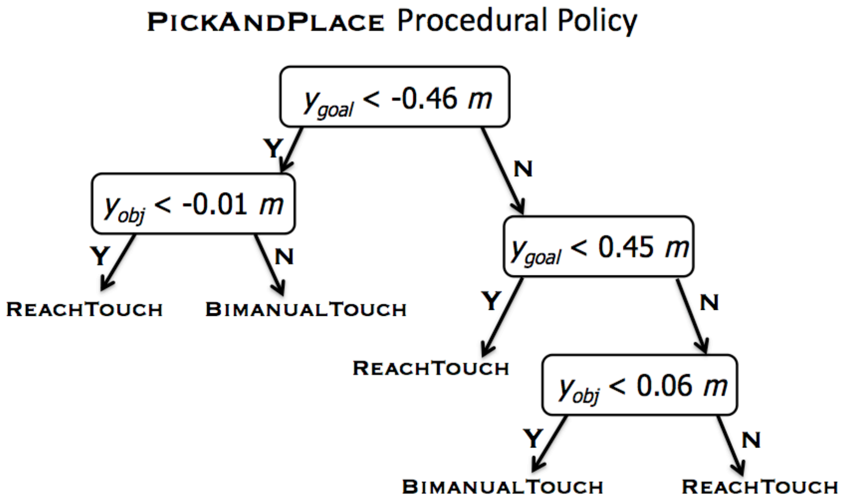
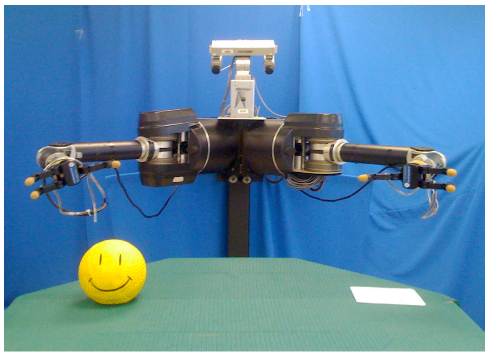


Figure 7.5. The PickAndPlace procedural policy.

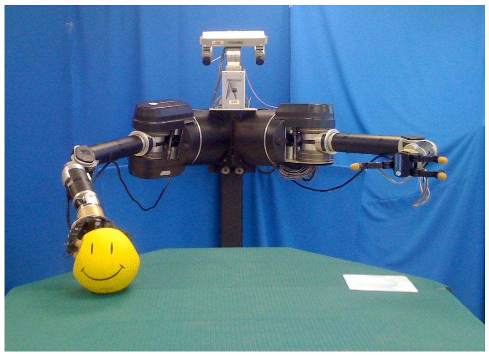
of affordance (Touch-ability), they both provide valid re-parameterizations of the original declarative policy. In the current training context, the BimanualTouch accomplishes a form of "hand-transfer" by picking up the object and bringing it into contact with the robot's other hand to achieve a grab. The resulting behavior in these situations is illustrated in the sequence of images shown in Figures 7.6(a)-7.6(d). The result is that the robot can accomplish a larger variety of PickAndPlace tasks by leveraging a large amount of prior behavioral knowledge.

7.1.3 PickAndPlace Affordances

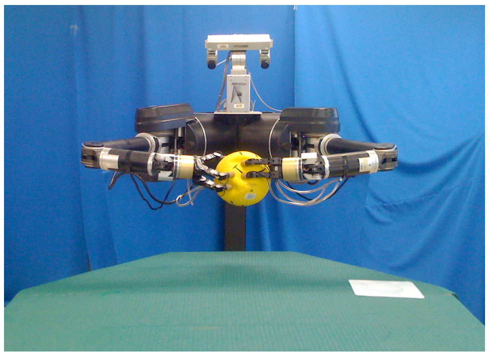
As a robot develops new behavioral structures such as PickAndPlace, it gains additional means of interpreting feedback from the environment. In Section 6.2, Dexter learned affordance catalogs for the green table and for two different balls it could pick up and examine. Now that the robot is endowed with a new PickAndPlace schema, a new affordance can be tested for membership in these catalog models. The new PickAndPlace is inherently compelling to Dexter by the affordance discovery reward because it relates objects and places and serves as the basis for a new (unhab-



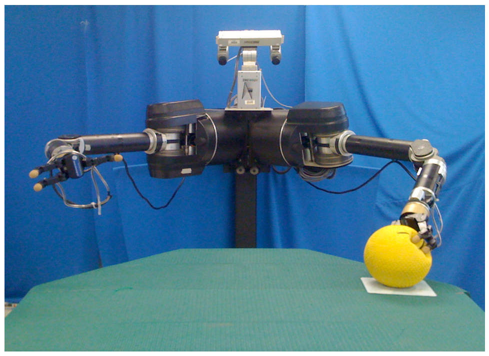
(a)



(b)



(c)



(d)

Figure 7.6. A procedural adaptation for the PICKANDPLACE schema. In (a) we see that the (blue-colored) goal is placed far to the robot's left, while the object is on its right. (b) and (c) show the robot using BIMANUALTOUCH for picking up the object and passing it between its hands, so that it can be brought to this far-away goal location as seen in (d).

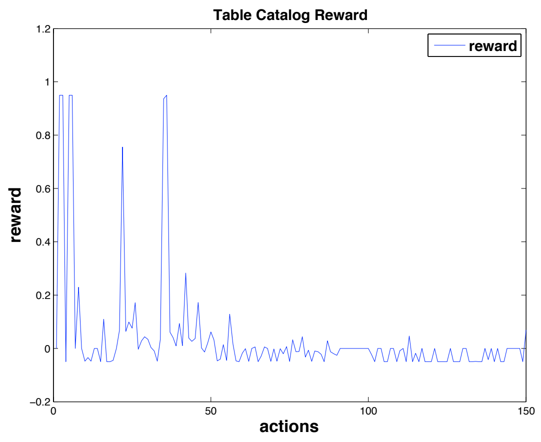
ituated) categorical distinction in the world; it potentially influences every existing catalog that the robot has so far constructed. Furthermore, since "pick" catalogs and "place" catalogs refer to two distinct environmental entities, this affordance can be viewed as belonging to a hierarchical "meta-catalog" that affords PickAndPlace between two participating catalogs.

Let us examine how the PickAndPlace affordance can be added to the catalog of a "pick" goal. Determining whether a new affordance can be added to a catalog is trivial. The robot adds the candidate action, a_i, to the action set explored using the n-armed banded learning formulation discussed in Section 6.2.1, with $Q^0(a_i) = 0$. The augmented action set is then explored using the ExploreCatalog() procedure. If all other affordances in the catalog have habituated then the robot's attention will be re-engaged, focusing on this new action. To demonstrate this process, the PickAndPlace action is added to each of the three catalogs in Figure 6.5. The robot samples its feedback signals for a "place" goal at run-time, adding the corresponding affordance to the catalog if it results in reward from the affordance discovery motivator. We now present the results of adding PickAndPlace to the three catalogs Dexter acquired in Section 6.2.2. As in the initial demonstration, we performed one trial on the real robot, then used the resulting probabilistic models to run an additional 25 trials in simulation. Each trial contained 50 action executions.

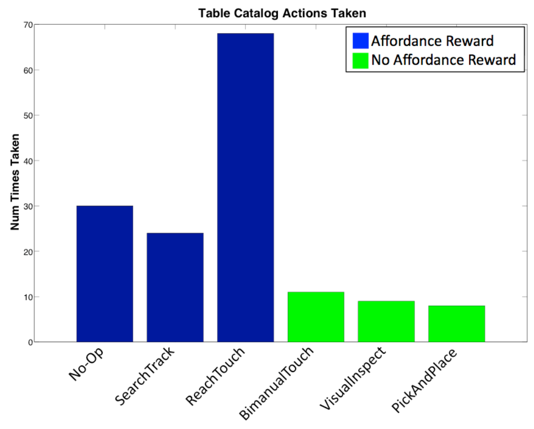
Figure 7.7 shows the reward received for each action as the robot explores the table catalog. Figures 7.7(a) and 7.7(b) show the results for the 150 actions performed on the real robot. The first 100 actions show the habituation for the initial affordances before PickAndPlace is added. The next 50 actions show the results after the new PickAndPlace action was added to the action set. We see that no additional reward was achieved when this action was added because it never succeeded (as seen in Figure 7.7(b)). The robot cannot pick up the table nor can it place it at another locaton. One interesting artifact worth noting is the temporary small dip in average reward shown in Figure 7.7(c) that occurs when the new PickAndPlace action is added to the action set. At this point, the cost of exploring PickAndPlace (to no avail) causes a penalty every time it is invoked. As there is no new information gained from its execution, the robot must learn that it is not worth the cost. As it does, the robot regains the same level of performance (0 average reward) as from before the new behavior was added.

Unlike the table catalog, the small basketball and the red ball both do afford PickAndPlace in some situations. When this new behavior was added to its action set, the robot sampled available Place parameterizations to build new affordances to add to each objects' catalogs. Figure 7.8 shows how the robot explores this new PickAndPlace behavior, placing the orange basketball on the green table. In Figure 7.8(a), we see the reward per action for the real robot trial. The first 100 actions show the habituation for the initial affordances before PickAndPlace is added. When this new action is introduced at action 101, the robot is rewarded for exploring this new affordance and extending its catalog. We see in Figure 7.8(b), that of the 22 times the robot performed this action it succeeded about 90% of the time and received reward from the affordance discovery motivator. The rest of the time, the robot dropped the object before the Place action could succeed. Figure 7.8(c) shows the average reward for the simulated trials, more clearly demonstrating how the PickAndPlace affordance habituates during the 50 additional actions taken by the robot.

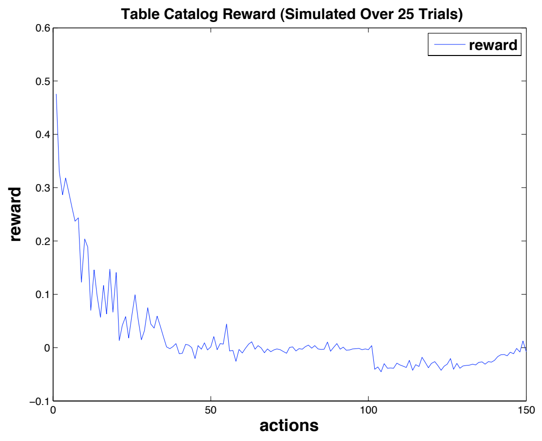
Similar results for when the PickAndPlace action was added to the red ball catalog are shown in Figure 7.9. The graphs show the results for both the real trial and the 25 simulated trials using models learned on the robot. As with the basketball, the robot was able to complete this action using the spatial location of the green table as the Place goal.



(a)

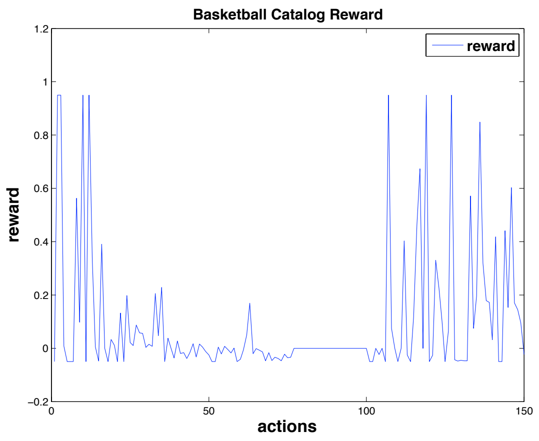


(b)

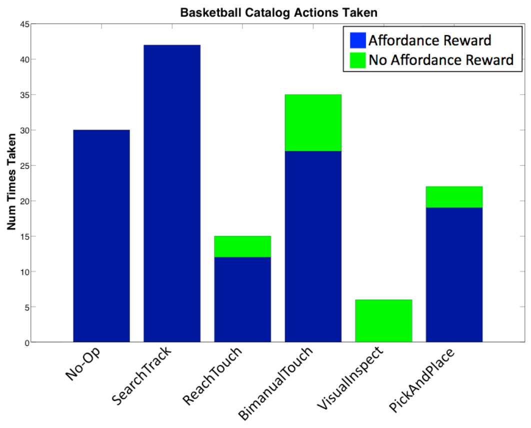


(c)

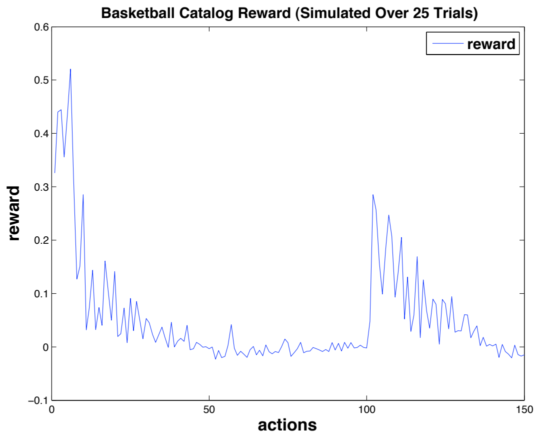
Figure 7.7. Exploring the PickAndPlace behavior in the table catalog. Frame (a) shows the reward per action over a sequence of 150 actions for a single trial on Dexter. The first 100 actions show the reward for constructing the catalog before PickAndPlace is added. The next 50 actions show the results after the new PickAndPlace action is added to the action set. Frame (b) shows the number of times each action was taken during the trial—blue parts bars indicate the number of times the behavior received reward from the affordance discovery motivator, green bars indicate that no reward was received. PickAndPlace did not achieve reward for the table. Frame (c) shows the average reward per action for the 25 simulated trials of the same experimental setup



(a)

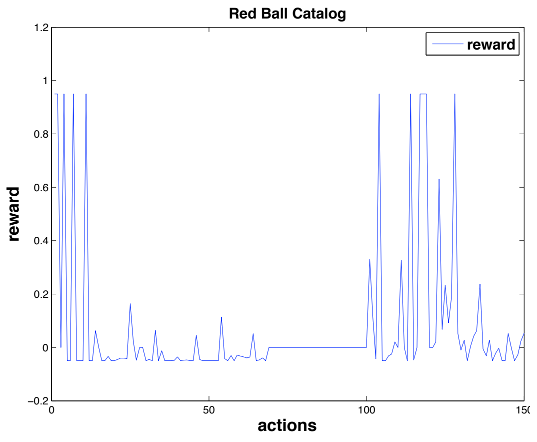


(b)

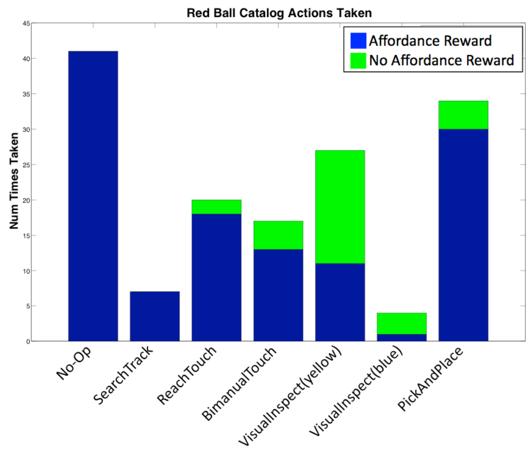


(c)

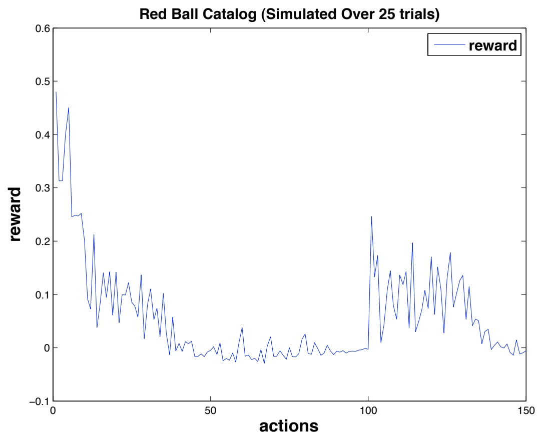
Figure 7.8. The results of adding PickAndPlace to the catalog for the small orange basketball. Frame (a) shows the reward per action over the total 150 actions engaging the basketball on the real robot. The PickAndPlace action was introduced after the first 100 actions. Frame (b) shows the number of times each action was taken, and the proportion of the actions that received reward from the affordance discovery motivator. PickAndPlace resulted in reward about 90% of the time. Frame (c) shows the average reward for the 25 simulated trials.



(a)



(b)



(c)

Figure 7.9. The results of adding PickAndPlace to the catalog describing the red ball. Frame (a) shows the average reward per action over the total 150 actions engaging the basketball on the real robot. The PickAndPlace action was introduced after the first 100 actions. Frame (b) shows the number of times each action was taken, and the proportion of times that action resulted in reward from the affordance discovery motivator. PickAndPlace resulted in reward about 90% of the time. Frame (c) shows the average reward for the 25 simulated trials.

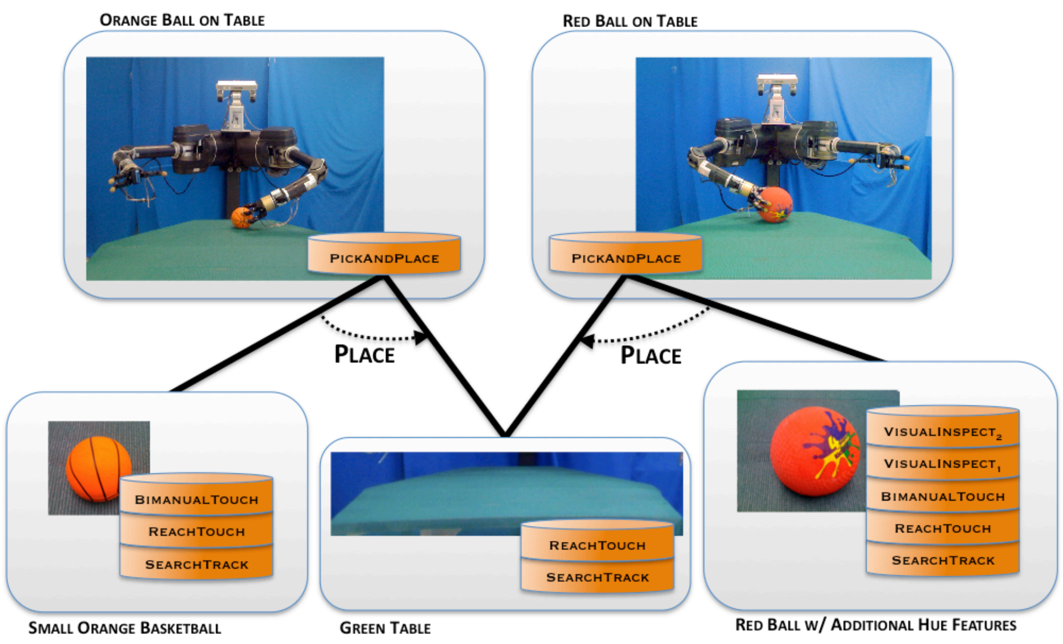


Figure 7.10. An iconic view of two "meta-catalogs" created when catalogs are brought into contact via PickAndPlace

By exploring PickAndPlace, Dexter discovered that the two balls reliably afford being placed on the green table. The result is a basic understanding of how two entities' catalogs are related. Figure 7.10 shows an iconic representation of how hierarchical "meta-catalogs" can be constructed when individual catalogs are brought into contact with each other. In the remaining sections of this chapter, we will examine how meta-catalogs provide a robot with the opportunity for long-term behavioral exploration in both the visual- and the force-domains.

7.2 Affordances Describing Environmental Impedance

In the last section, it was demonstrated how visual features can form the inputs to PickAndPlace actions to construct "meta-catalogs." Although it should be obvious that "placeability" has only a cursory relationship with the place goal being *green*, these experiments provided an appropriate initial training context for Dexter to bootstrap more advanced understanding during subsequent exploration. Affordances of multi-object interactions are more robustly encoded in the reaction forces that occur when the robot brings entities into contact with one another. In fact, tasks like mechanical assembly are grounded in such force-domain affordances.

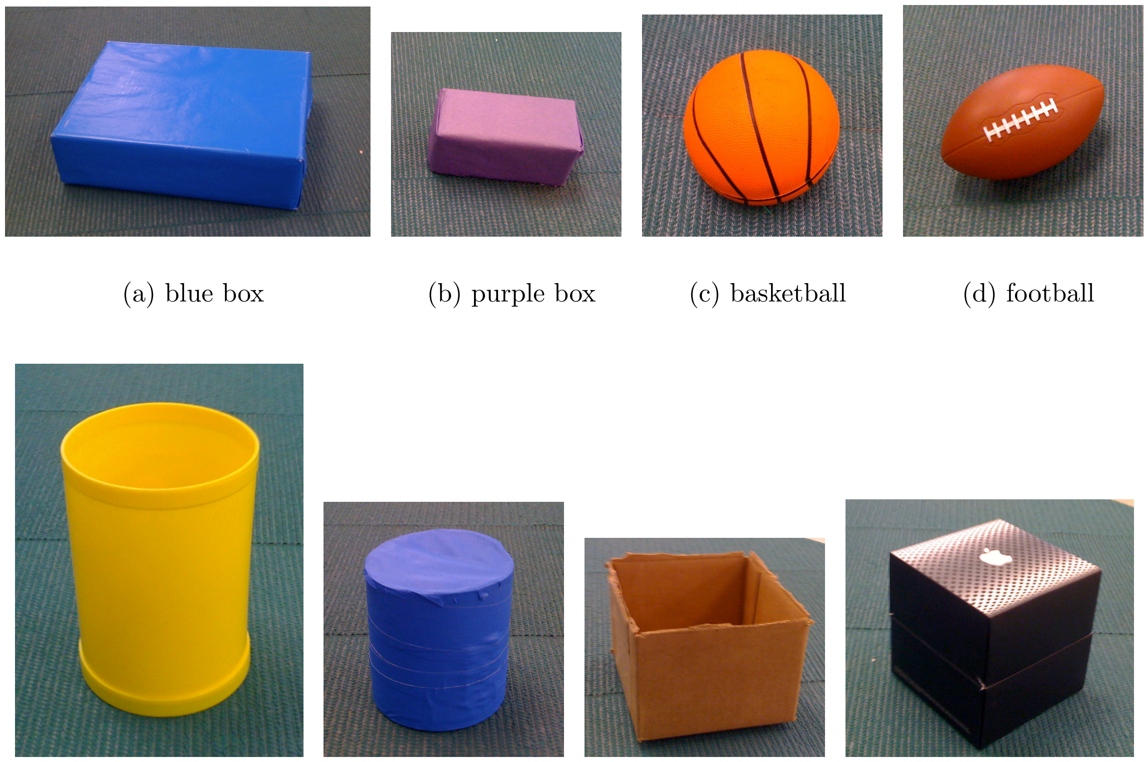
An appropriate characterization of multi-object interaction forces occurs through an examination of environmental impedance. Impedance relates forces to velocities in a system and can be used to represent mechanical assembly relations between two objects. It is defined as the ratio between the force impeding an input spatial displacement (or velocity). For a test i, an impedance $\mathbf{Z}_i$ can be measured by applying a displacement $\Delta\mathbf{x}_i$ and observing the reaction force $\mathbf{f}_i$ such that:

$$\mathbf{Z}_i = \frac{\mathbf{f}_i}{\Delta\mathbf{x}_i}. \tag{7.10}$$

In the control basis, environmental impedance provides useful feedback signals when a robot is interacting with its environment. A robot can uncover impedances by testing force tracking affordances in different directions, and observing the resulting spatial displacements. By examining the impedances in certain directions for meta-catalogs (object pairs), the robot can build the background knowledge for performing characteristic assembly operations like stacking and inserting.

7.2.1 Experimental Setup for Impedance Observations

Experiments were performed in which Dexter explored the impedances of meta-catalogs constructed when some of the objects seen in Figure 7.11 are brought into contact with each other via PickAndPlace. Six object combinations were tested. When contact was made between the objects, the robot sequentially ran 12 force-tracking control actions, each one parameterized by a 2 N reference along each axis of the robot's coordinate frame or a 2 N/m about each axis of the that frame (3 axes, 2 directions along each axis, 2 directions about each axis). If the reference force was



(a) blue box (b) purple box (c) basketball (d) football

(e) yellow cylinder (f) blue cylinder (g) brown box (h) black box

Figure 7.11. Objects explored for impedance sensing.

tracked in either the positive or negative direction of each axis test, the affordance was recorded in the corresponding meta-catalog. These six test dimensions are labeled $\{dPx, dPy, dPz, dRx, dRy, dRz\}$. A rought estimate of the impedance along each direction was also recorded by dividing the net movement during the action by the magnitude of the maximum force that was felt. In general, of course, the observed impedances will depend on the relative coordinate frames of the two objects being tested. The objects and object-relationships presented in this experiment, however, are sufficiently simple and the process of coordinate frame alignment is ignored. Despite this limitation, however, interesting results are observed. Each experiment was

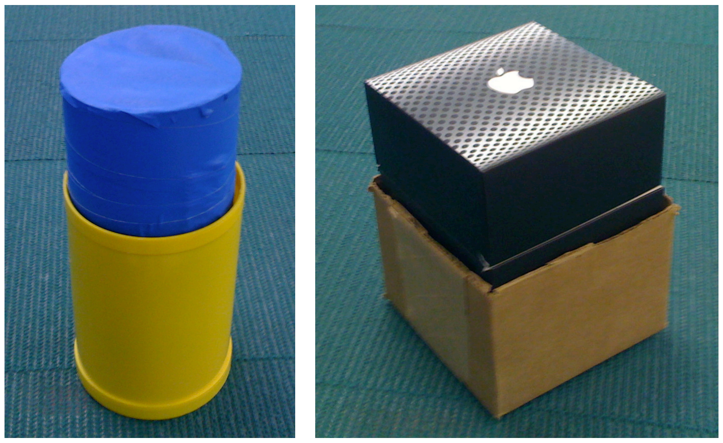

(a) (b)

Figure 7.12. Frame (a) shows the cylindrical relationship formed between the yellow and blue cylinders. Frame (b) shows the prismatic relationship formed between the brown and black boxes.

performed five times for each of the object pairs tested. The results were averaged over these five trials.

7.2.2 Detected Impedances

The average likelihood of each force-tracking affordance along the six dimensions is shown in Figure 7.13 for each of the six meta-catalogs tested. The average estimated impedance for each direction of each meta-catalog is shown in Figure 7.14. These estimates are normalized over each trial. We see that when the purple box is placed on top of the blue box, the robot is able to track a reference force in the z-direction as well as movements about the x- and y-directions every trial. This is to be expected because the object from which these catalogs derive "fit" together by way of a plane-on-plane relationship. In this case, lateral movements result in low impedance, but rotations about those axes are relatively stiff and can be tracked in the force domain. Likewise, rotation about the intersection contact normal produces low directional impedance, but translation along the inward normal of the surface has high impedance. During one of the trials, the robot was able to control the movement about the surface normal by exploiting frictional forces. This was the case when the purple object was rotated about the perceived contact normal on top of the blue object.

When the orange basketball was placed on top of the blue box, the robot was reliably able to control the force maintaining contact between the objects (the z-direction), as well as along the y-translational and x-rotational directions (resulting in approximately the same movement), due to a strong frictional force in those directions. The estimated impedance profile shows greater response in these three directions. During one of the five trials, the robot was also able to control the strong frictional force as it rotated the ball about the z-translational direction.

The meta-catalogs representing the purple box on the orange ball and the football on the orange ball resulted in no reliable force-tracking affordances except in the z-direction at the contact point. This is to be expected because, in both cases, the top object either rotated about the ball or rolled the ball with it during the test movement. The impedance profiles, likewise, also shows large magnitudes only along this z-direction.

When the blue cylinder was placed inside the yellow cylinder (resulting in the insertion relationship seen in Figure 7.12(a)), we expect to observe a cylindrical relationship to be formed in which forces in the lateral (x and y) directions and rotations about those directions create resulting reactionary forces that can be controlled, while movement along the z-direction and rotation about the same result in no such reactionary forces. We see that, 80-100% of the time, this is in fact the result that was observed by the robot both in the affordance likelihood and the estimated impedance profiles.

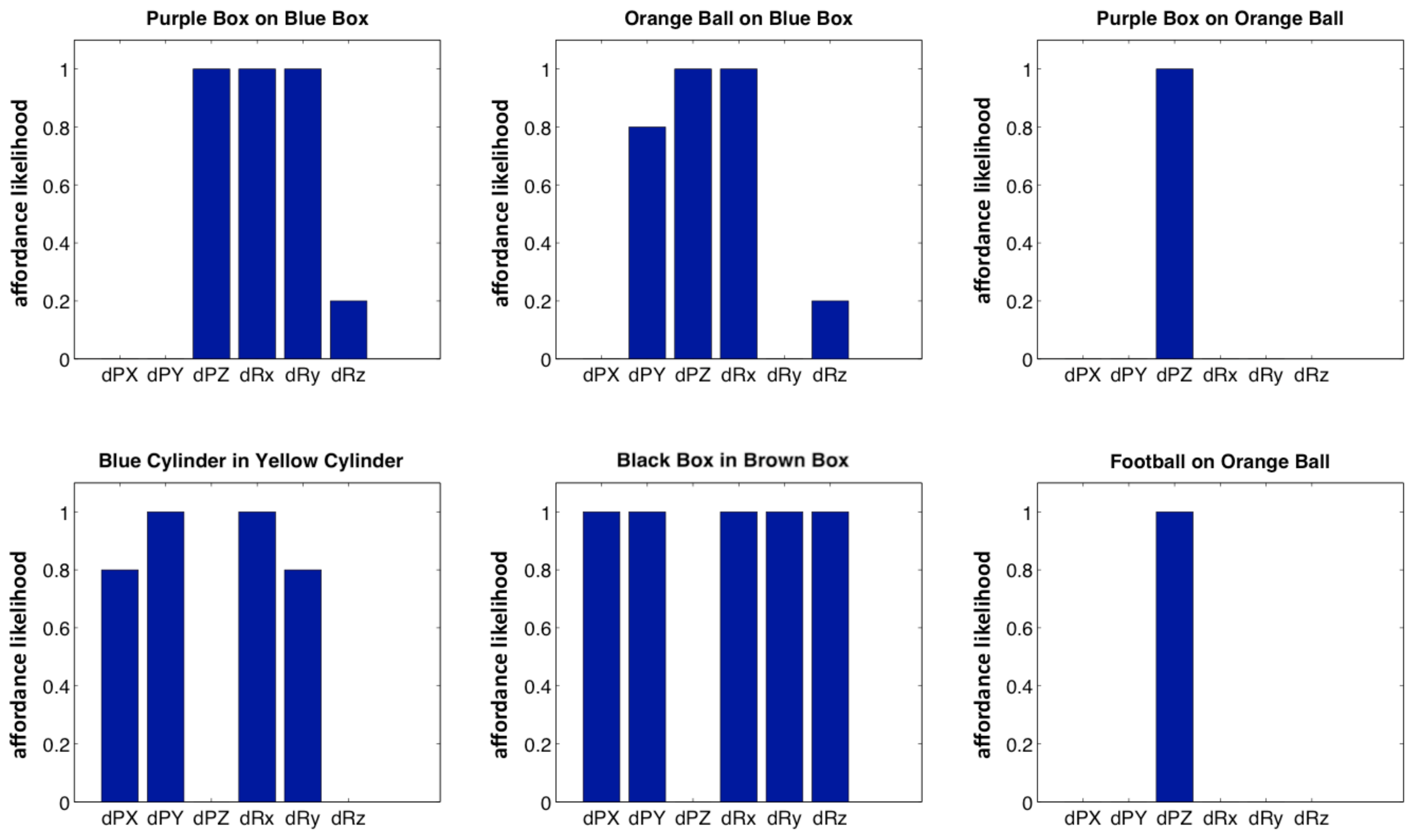


Figure 7.13. The average likelihood of the force-tracking affordance along each dimension for the six meta-catalogs.

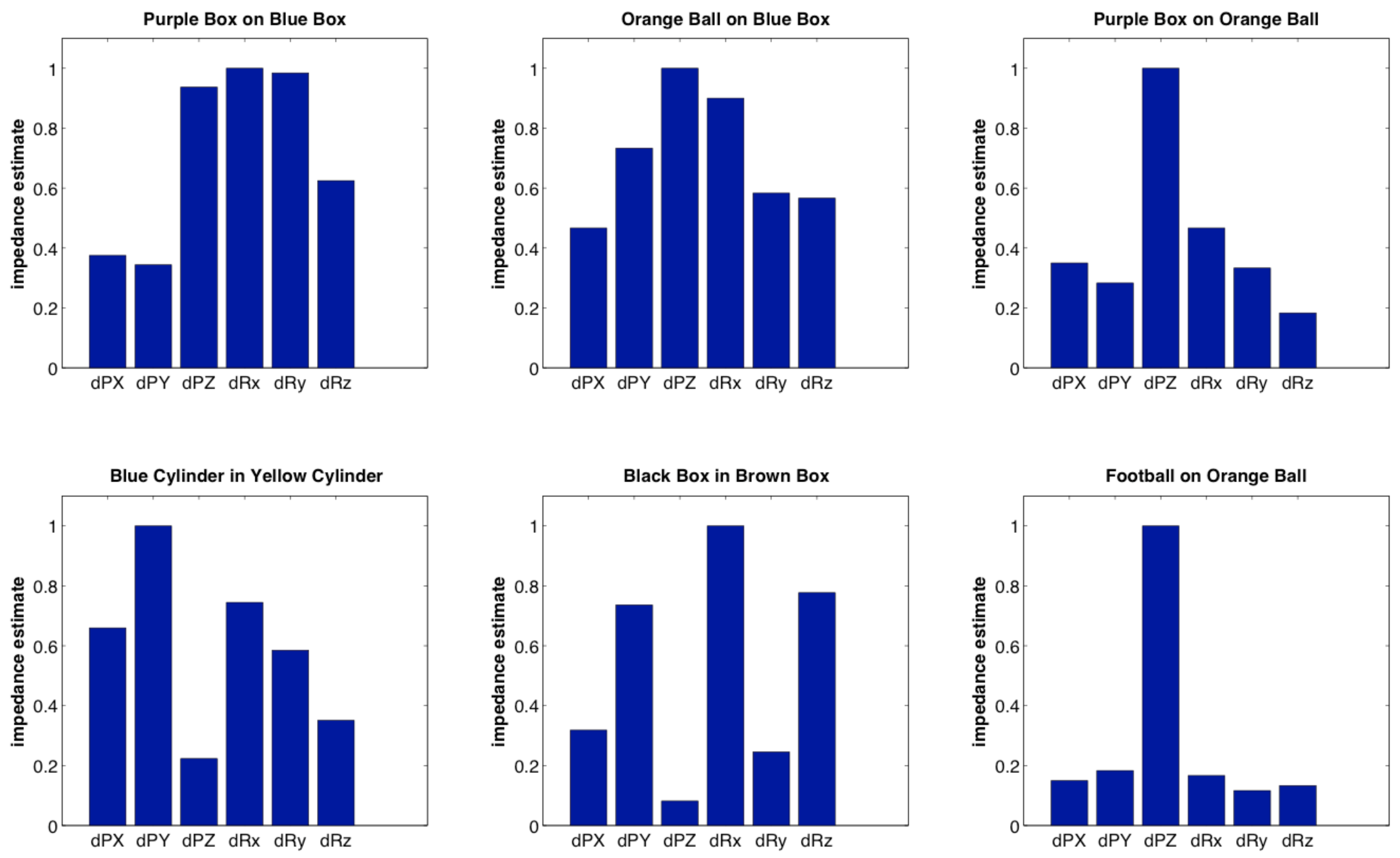


Figure 7.14. The estimated impedances (averaged over five trials) for the six meta-catalogs. The estimates are normalized according to the maximum estimate per trial.

Similarly, when the black box was placed inside the cardboard box (resulting in the insertion relationship seen in Figure 7.12(b)), we expect to observe a prismatic relationship to be formed in which forces in all directions can be controlled except along the insertion (z) axis . The robot reliably observed this to be the case in all five of its trials.

7.2.3 Discussion

The above experiments have provided the first steps toward examining how a robot can explore the force-domain affordances between two objects when they are brought into contact with each other into a meta-catalog. Predictable impedance relations, such as those in Figure 7.13, can provide a means of identifying mechanical relationships between objects and representing the categories of multi-object assemblies. The demonstrations, however, were limited by their reliance on measuring impedances in a fixed world coordinate frame. This limitation will make accurate assembly recognition more challenging. In general, it will be necessary to consider how impedance observations can be transformed into a unique 2-body coordinate frame that more appropriately represents the relative force-domain invariants.

Despite this limitation, the pilot data presented in this section suggests that the proposed techniques for affordance discovery can transform simple rules like "red objects go on blue objects" through experience into assembly concepts in the force-domain that describe how two objects "fit" together to create plane-on-plane, prismatic, cylindrical, revolute, or spherical surface relationships. One of these relationships, the "plane-on-plane" relationship, provides a reasonable estimation of what constitutes a "stack." In the remainder of this chapter, we will explore how a developing robot can ground stacking relationships in both the visual- and the force-domain affordance models that a robot can explored.

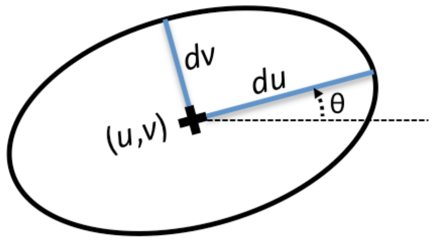
7.3 Meta-Catalog Exploration

It has been shown how meta-catalogs allow a robot to construct hierarchical catalogs out of the multi-modal feedback signals it can control. The ability to create catalogs autonomously through interaction yields a means of performing long-term behavioral exploration and world modeling. When the intrinsic drive to discover new affordances in meta-catalogs habituates, the robot can search for new features that efficiently encode useful affordances for more complex entities. As a result, the robot begins by exploring clusters of simple affordances and, with sufficient training, constructs hierarchies of multi-body catalogs. I contend that mechanisms like these can be used to acquire expertise at recognizing and exploring experimental interactions with the world and ultimately contribute to a deep understanding of manual skills.

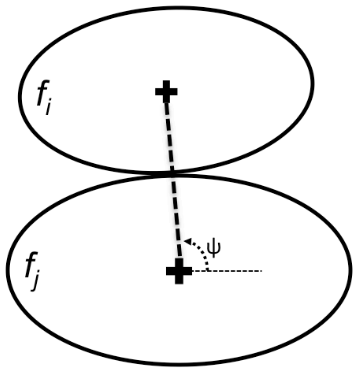
To demonstrate, we use the techniques and behavior developed thus far to create meta-catalogs from a recursive basis for catalog exploration. Experiments are presented in which Dexter searches for new visual affordances that arise when it constructs meta-catalogs via PickAndPlace. Visual affordances in meta-catalogs occur when the robot tracks *constellations* of features sampled from the constituent parts. Stable constellations are the visual associations of the force-domain invariants that arise when two objects "fit" together in a mechanical assembly such as a "stack" or an "insertion."

7.3.1 Demonstration: Learning Visual Meta-Catalog Affordances

In this experiment, Dexter explores PickAndPlace actions to build meta-catalogs and then searches for constellations of trackable features (via SearchTrack) with which to "recognize" these relations. If the resulting constellations are stable after the robot lets the object go (i.e., it does not immediately fall apart), the robot receives reward from the affordance discovery motivator for discovering a new Track affordance.



(a)



(b)

Figure 7.15. Frame (a) shows the feature descriptor for an oriented visual blob invariant f. Frame (b) shows the descriptor for a feature constellation ${}^cf$ consisting of two primitive feature descriptors f_i and f_j.

Although many representations for visual feature constellations exist, a simple choice that is adequate for these experiments consists of a combination of two oriented visual blob invariants and their relative scales. If we represent an anisotropic blob (Figure 7.15(a)) as:

$$f = [u, v, du, dv, \theta]$$

where (u, v) is the pixel coordinate of the center of the region of interest, du and dv are the axes of the region, and θ is its orientation, then a compound feature can be defined as:

$${}^cf = \left[\frac{du_i}{du_j}, \frac{dv_i}{dv_j}, \psi\right]$$

where (du_i, dv_i) and (du_j, dv_j) are the axes of the primitive features f_i and f_j, respectively, and ψ is their relative orientation, as illustrated in Figure 7.15(b).

To demonstrate how Dexter can test for the affordances of visual constellations, we provided a training context in which we place various combinations of the basketball, the football, the small purple box, and the large blue box (Figure 7.11) on the table in front of the robot. The robot tests all combinations of these objects except for

Stack Combination	**Success**
Purple Box on Blue Box	100%
Basketball on Blue Box	20%
Football on Blue Box	80%
Basketball on Purple Box	0%
Basketball on Football	0%
Football on Purple Box	40%
Football on Basketball	0%

Table 7.1. 2-Way Stack Results

Stack Combination	**Success**
Basketball on Purple Box on Blue Box	0%
Football on Purple Box on Blue Box	40%
Purple Box on Basketball on Blue Box	0%
Purple Box on Football on Blue Box	0%

Table 7.2. 3-Way Stack Results

combinations that require the robot to pick up the blue box (which it has difficulty doing). Feature constellations are constructed from hue-space pixel regions corresponding to the appearance of each of these objects on the robot's left camera image. The presence of Track-able visual constellations, in conjunction with characteristic impedance profiles (as described in Section 7.2), induce a "stackable" category that the robot can explore hierarchically. In particular, with this set of objects there are a number of 2- and 3-way potential stacks the robot can test. To provide an empirical analysis, each combination is tested five times.

7.3.2 Experimentally Determined Stackability Affordances

Table 7.1 shows the percentage of times each constellation is Track-able after the completion of the PickAndPlace action. We see that the purple box and the football stack reliably on top of the large blue box (100% and 80% of the time, respectively), and created a stable Track-able visual constellation. Occasionally,

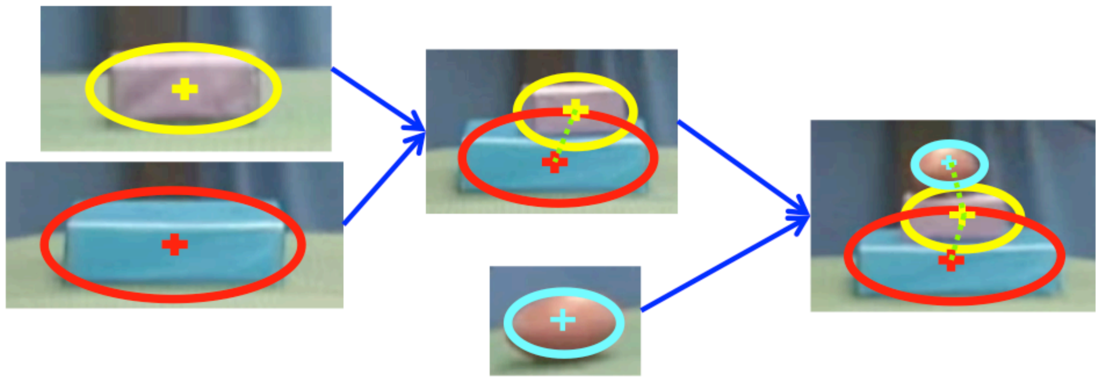


Figure 7.16. The growth of a 3-way stack consisting of the football on top of the purple box on top of the blue box. This stack occurred successfully two out of the five times it was built.

the basketball stays on top of the blue box (it rolled away four out of the five trials), and for two of the trials, the football stacked on top of the purple box. The rest of the two-way stacks did not succeed in any of the trials, and thus did not create any TRACK-able constellations. We see in Table 7.2, only one 3-way stack occurred in any of the trials, and only with a success rate of 40% (two out of the five trials). This was the football on the purple box on the blue box. All other combinations did not create a TRACK-able constellation that was stable after the robot removed its hand. The stack construct gave rise to the resulting 3-way TRACK-able constellation is illustrated in Figure 7.16.

Figure 7.17 shows the hierarchy of meta-catalogs and their respective affordances for the blue box, purple box, and football. At the lowest level of this hierarchy, we see the catalogs for the purple and blue boxes. The purple box affords receiving reward by means of SEARCHTRACK, REACHTOUCH, and BIMANUALTOUCH skills. The blue box affords only SEARCHTRACK and REACHTOUCH, because the robot was not able to grab the object (it was two narrow and wide). These two catalogs induce a meta-catalog via PICKANDPLACE that provides a visual constellation the robot can track via SEARCHTRACK This meta-catalog can also serve as a place goal for the football to induce a second-tier meta-catalog capturing the three-way stack.

Figure 7.17 shows one “slice” of Dexter’s world model consisting of the set of objects provided in the training contexts. It should be noted that other affordances between these objects and other objects also exist (e.g., all of these objects also afford PICKANDPLACE with the green table, among other things), but are not included in this illustration for clarity.

7.4 Discussion

In this chapter, new learning stages were demonstrated on Dexter in which the robot learned new affordances pertaining to the trackability of contact forces between bodies and the visual appearance of multi-object interactions. The robot was able to explore these affordances to learn new categorical distinctions revolving around a very simple form of multi-object assembly—a “stack.” The robot explored a set of objects, discovering which subsets created stable 2-and 3-way stacks, that in turn, facilitate further behavioral exploration.

Affordances pertaining to multi-object assemblies are represented as knowledge structures in the robot called “meta-catalogs.” These meta-catalogs capture both spatial and force-domain characteristics of how bodies relate to one another and allow a robot to construct new features it can explore (or “play” with) when the robot gets bored with existing catalogs. Long-term learning thus becomes a recursive form of catalog exploration in which the robot can continually find new ways to “interpret” the world. Meta-catalogs bring together many of the techniques for intrinsically motivated behavior learning and knowledge organization presented in this thesis.

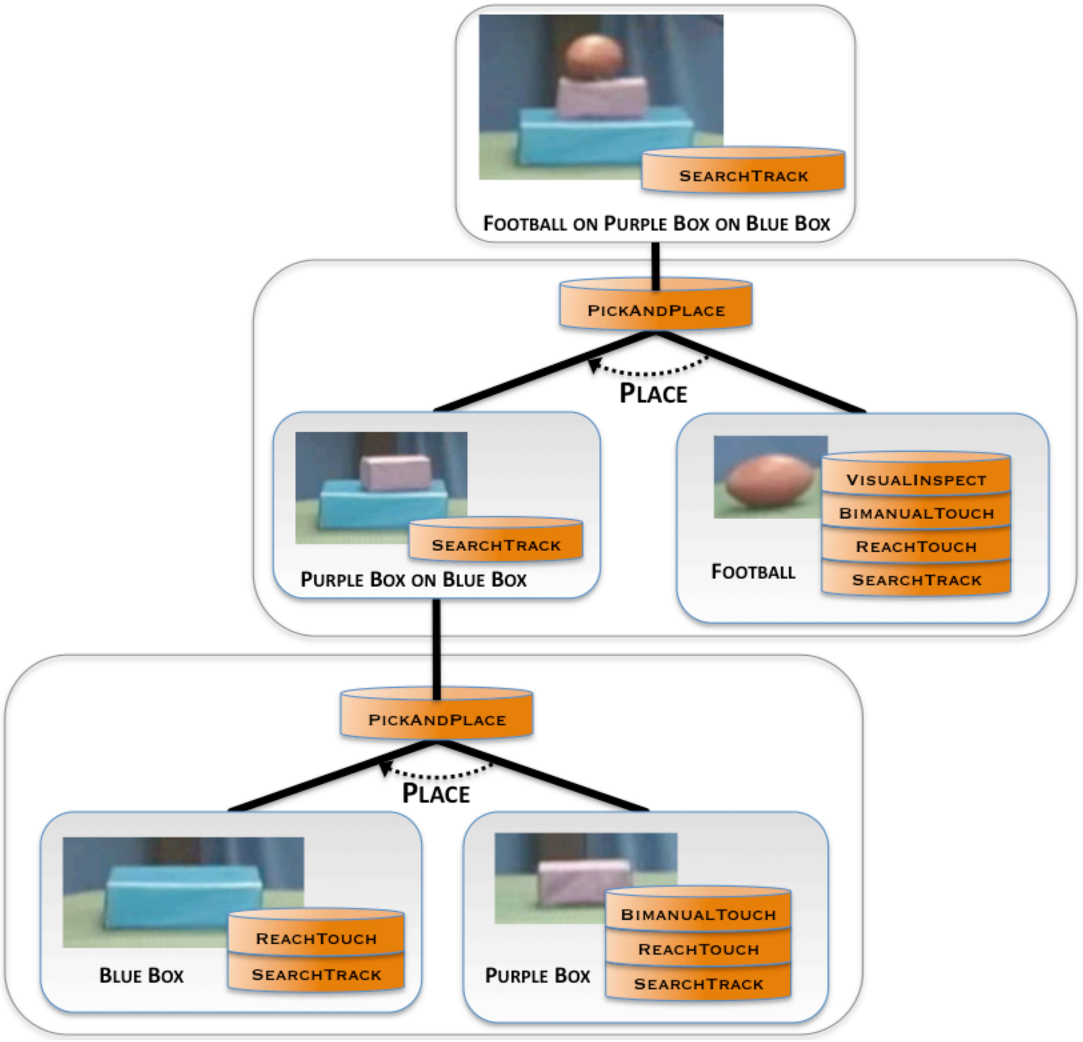


Figure 7.17. The hierarchical affordance catalog created between the blue box, the purple box, and the football. The figure shows the affordances of each object, as well as the affordances of the constellation of features created when they are stacked on top of each other through the PICKANDPLACE schema.

CHAPTER 8
CONCLUSIONS

In the first three chapters of this dissertation, I argued that manipulation was an appropriate domain to study knowledge representation, organization, and intrinsic motivation in robot systems. I argued that a dynamical systems approach to development provides a natural and biologically relevant means of representing action, while also allowing a system to get the most out of its physical structure. In the proposed framework, control circuits composed of sensorimotor resources and domain general objectives are fluidly organized into dexterous strategies for interacting with unmodelled worlds. In the next two chapters, I presented a developmental approach for acquiring hierarchical programs in this framework, demonstrating its efficacy through an extended example in which the robot Dexter learned strategies for interacting with various objects. Under this approach, the robot is "taught" programs in simple learning situations that are later generalized to meet the demands of more challenging contexts. Chapter 6 introduced how environmental affordances can be collected into memory structures called "catalogs." Chapter 7 demonstrated how catalogs can ground multi-object assembly structures such as object "stacks," while also providing a basis for long-term behavioral exploration via "meta-catalogs." In this chapter, I will briefly discuss the main contributions of the document, examine some possible future areas of related research, and provide some insights gained in the course of this work.

8.1 Contributions

Ultimately, it will be very difficult for developers to program or demonstrate all of the behavior a robot will need to behave competently in real-world scenarios. Not only is it unlikely that a programmer will be able to foresee all of the possible situations a robot might find itself in, it will be hard for developers to "get inside the skin" of any robot with a different morphology or with sensory feedback signals different from those of the programmer. Due to these challenges, it is imperative that robot programmers instead endow robots with the ability to learn and adapt to new, unforeseen situations autonomously and on their *own terms*.

In this thesis, I have argued for a developmental approach to robot programming that requires minimal human developer input, and allows a robot to judge the efficacy of its actions in terms of their run-time dynamics. Furthermore, I argue that any robot designed to perform tasks in the natural environment must perform effectively in both the spatial and the force domains. Most work in the community, however, focuses primarily on learning behavior in the spatial domain. This is true, even for work pertaining to primarily force-domain tasks such as grasping (Saxena et al., 2007; Kraft et al., 2008; Goldfeder et al., 2009). Similarly, approaches for teaching robots from demonstration have treated behavior not as "intentional" actions with desirable effects, but rather as joint-level trajectories that can be generalized from repeated examples (Schaal, 1999; Billard & Mataric, 2001; Jenkins & Mataric, 2002; Schaal et al., 2003; Kober & Peters, 2009). These approaches, I argue, will not result in dexterous solutions because they are designed to specifically ignore the most important aspects of the tasks they are designed to solve.

In contrast, the work presented in this dissertation allows a robot to learn when to appropriately and dexterously employ spatial and force domain feedback to find controllable aspects of interaction in the world. The resulting knowledge learned by this framework, as demonstrated by the experimental work on Dexter, is not engineered for any specific task. In particular, this dissertation has made a number of contributions to the state-of-the-art of specific areas in robot learning:

- **Intrinsic Motivation:** An intrinsic motivation function is presented that drives the exploration of a robot to acquire robust hierarchical strategies for manipulation by means of an inherent drive towards *controllability*. This "affordance discovery" reward is unique in the intrinsic reward literature in that it pushes the robot to interact with and model its environment, focusing on those areas where it discovers stable, closed-loop interactions.

- **Knowledge Representation:** In the proposed framework, representation begins at embodiment by providing a robot with the ability to connect its motor units to its sensory channels to achieve goal-driven closed-loop behavior. Furthermore, it was demonstrated how a number of prerogatives—or *intentions*—native to the robot, but independent of any specific task, can be assembled into goal-directed behavior that allows a robot to get the most out of its sensory and motor systems.

- **Knowledge Organization:** Behavior is factored into declarative and procedural components that facilitate generalization, hierarchical re-use, and the ability to find dexterous solutions to accomplish desired goals. Knowledge is organized into hierarchical structures for both generalized behavior (schema) and object modeling (catalogs). These structures provide an ontological means of capturing what actions a robot can perform and how it can use those actions to interpret its world.

- **Multi-Object Assembly:** Several experimental demonstrations were presented that illustrated a grounded approach to learning about the multi-object relationships that form the basis for assembly. Assembly relationships were acquired in terms of both visual and force domain affordances. These affordances inform the creation of new compound structures (meta-catalogs) that a robot can explore in a combinatoric way.

8.2 Future Directions

Although an extended demonstration was provided on the robot Dexter in a manner to make certain points clear, the framework presented in this dissertation is much more general and can be applied to any robot to address many domains of manipulation. However, many related issues, some of which are described next, require more investigation.

New Skills: Dexter was taught dexterous solutions to many manipulation tasks regarding single and multi-object tasks. What further skills might the robot learn? One possibility could consider previous research by Coelho (2001) and Platt (2006), in which these authors examined control basis approaches for achieving force-closure grasp conditions on objects. Interestingly, these formulations meet the criteria to be rewarding by the affordance discovery motivator because they move to regulate net forces and moments perceived on a robot's fingertips. How could a GRASP schema, then, be learned and employed to provide a more robust solution for grabbing objects then the REACHTOUCH schema? What additional schema could be learned on a bimanual robot such as Dexter? How could the existing schema be generalized to handle more complex situations? Can the PICKANDPLACE schema be generalized to allow a robot to perform more general assembly tasks like INSERT?

New Robots: How could the proposed techniques for developmental programming be applied to different robots with different morphologies such as the uBot-5 mobile manipulator (Kuindersma et al., 2009). Mobile manipulators face an increased amount of flexibility due to their ability to navigate around their environments. They can seek out desirable contexts and provide an increased ability to interact with both its world and other agents (such as humans or other robots). For the uBot-5, areas for developmental learning include the ability to learn and transition between modes of mobility (crawling, "knuckle-walking," balancing, etc.) that can be formulated as control tasks in the control basis. How can such modes of navigation be learned in a manner similar to how an infant learns to walk? Other questions include how behavior learned on one robot may be transfered to other robots with slightly different morphologies. Could the schema learned on Dexter be transferred to the uBot-5? Because the declarative structure of schema represent abstract goals, it is interesting to consider if transferring a schema to a new robot means only that it must learn its own procedural policy while maintaining the original declarative structure. Such an approach could bootstrap learning on the new robot to allow for new behavior to be acquired more quickly.

Long-Term Training: In teaching the robot Dexter the sequence of hierarchical programs demonstrated in this thesis, both the environmental contexts and the available resources the robot could explore was strictly controlled by the human teacher. This was done for two main reasons: (1) to show how having a human "in-the-loop" can help a robot learn skills more quickly, and (2) because leaving Dexter on and running for extended periods of time is infeasible for practical reasons. However, the framework proposed supports a more "hands-off" approach in which the robot can sample subsets of its resources to explore more autonomously with less guidance. The discrete representation of resource combinatorics in the control basis make this a feasible and tractable solution for robots that can operate for more hours with less supervision. Would a robot that was "always on" be able to learn a sequence of hierarchical skills by means of the control basis and the affordance discovery reward function if given enough time? What challenges would arise when less supervision is involved during each stage of learning?

Additional Multi-Modal Features: The behavior taught to Dexter in the experimental sections intentionally showed how to exploit very simple sensory features to learn increasingly complex manipulation skills. However, could more robust procedural strategies be acquired if the robot is allowed to use richer features such as N-jets (Koenderink, 1984; Piater, 2001), SIFT-descriptors (Lowe, 2004), or scale-space invariants (Lindeberg, 1994)? One immediate possible form of improvement would be the investigation of such texture operators to find assembly affordances. For example, corner detectors might be useful for classifying which objects stack on other objects (horizontal surfaces with four co-planar corners are a good indication). These operators, although typically applied to visual images, could easily be applied to other domains as well (e.g., tactile, auditory, etc.). How could these multi-modal texture operators enrich a robot's capabilities for manipulation?

Affordances used for Planning: The affordance catalogs presented in Chapters 6 and 7, provide a useful means of capturing how features might lead to desirable behavioral outcomes. Can these catalogs be used as forward models to predict sequences of outcomes? In other words, can these catalogs provide the inputs to higher-level planners that can be used to perform complex tasks such as assembling mechanical structures? Recent work using Object Action Complexes (OACs) for affordance modeling has suggested that this is possible (Uğur et al., 2009). This is worth investigating using the affordance models (catalogs) presented in this document.

8.3 Insights and Discussion

This document provides a novel perspective with regards to programming robots that are meant to operate in unstructured human environments. I advocate a framework in which a robot can assemble its sensory and motor resources to achieve goals that might be used to accomplish tasks if the circumstances are ripe. A number of insights were gained in the course of examining the proposed framework:

- Situated representations for state, action, and reward—as presented in Chapters 3 and 4—provide an appealing substrate for a robot to learn strategies that can be used in many tasks. This is a *bottom up* perspective that allows a robot to see how it can use its resources to solve new tasks, rather than a *top down* perspective in which a programmer fits a robot to a task that the robot may not have been designed for. The "trick" in such approach is in providing a robot with a number of task-independent objectives that have more to do with how best it can use its body, and a number of resources it can use to accomplish those objectives. I argue that such an approach will be absolutely necessary for robots learning over the long-term because it will be impossible to foresee all of the functions that a robot may need to perform during its extended "lifetime."

- A large number of tasks can be accomplished in which the perceptual capabilities of a robot are fundamentally and exclusively grounded in a robot's ability to interact with its environment. This idea underlies any control theoretic approach to robot programming, but has been demonstrated here in increasing layers of abstraction that allow a robot to achieve primitive levels of behavioral reasoning and planning (c.f., the learned procedural strategies for ReachTouch and PickAndPlace). This approach also suggests that a robot will learn the most compact representations for knowledge that it is able to, not trying to fit human specified features into its model that might be unhelpful or misguided. If the robot needs a particular feature to solve a task, it will recruit that feature in a closed-loop strategy, and ignore all other features it has no use for (i.e., those features that do not increase the likelihood of reward).

- The behavioral schema the robot Dexter learned were all compact because the proposed representation allows a robot to use programs hierarchically. None of the programs Dexter learned in its learning stages consisted of more than two or three actions, and thus no state vector was ever larger than two or three predicates long. Despite, such limitations, Dexter was able to learn effective strategies for a large amount of behavior, including primitive strategies for multi-object assembly. This suggests that the proposed approach is reasonably *scalable*, and will not lead to intractable state/action spaces as the complexity of new learning problems increase. The complexity, in many ways, is pushed to the procedural learning components that allow a robot to generalize its programs into contingency-handling schema. However, procedural learning strategies can be incremental, incorporating new features only if existing strategies are not sufficient. I believe this is an attractive approach to handling complexity because it leads to simple strategies quickly, but allows for adaptive readjustment over the long-term as the robot gains more experience.

- Adhering to formal typing constraints and uniform structures for assembling actions and states out of a robot's combinatorics, makes it possible to write a general software programming interface for the control basis we have called the Control Basis API (CBAPI) (Hart et al., 2009). The CBAPI requires a programmer to supply only the interface (or driver) to a robot resource along with a configuration file specifying device-specific parameters, and is given in return the ability to use that resource in formal, type-constrained control policies. Furthermore, such a formal specification allows machine learning algorithms, such as Q-learning, to explore the combinatorics of control basis state/action spaces according to the techniques for behavioral programming laid out in Chapters 4 and 5.

Taken together, the ideas presented in this dissertation provide some promising new areas of research for programming robots meant to perform “life-long” learning in unstructured, human environments. At their core, these ideas advocate an embodied, self-motivated approach to learning robust, closed-loop behavior that a robot can adapt to meet the demands of its world. I argue that the experiments presented herein demonstrate a principled means of constructing directed behavior from undirected means—something any next generation adaptive robot will have to consider as it exceeds its factory-equipped capabilities to help humans in both everyday or highly-specialized tasks.

BIBLIOGRAPHY

Akishita, S., Kawamura, S., & Hayashi, K. (1990). Laplace potential for moving obstacle avoidance and approach for a mobile robot. *1990 Japan-USA Symposium on Flexible Automation, A Pacific Rim Conference* (pp. 139–142).

Angeles, J., & Lopez-Cajun, C. (1992). Kinematic isotropy and conditioning index of serial robotic manipulators. *The International Journal of Robotics Research*, *11*, 560–571.

Arbib, M. (2003). Schema theory. *The Handbook of Brain Theory and Neural Computation*. Cambridge, MA: MIT Press.

Arkin, R. C. (1998). *Behavior-based robotics*. MIT Press.

Asada, H., & Granito, J. C. (1985). Kinematic and static characterization of wrist joints and their optimal design. *International IEEE Conference on Robotics and Automation*. St. Louis, MO.

Asada, M., MacDorman, K., Ishiguro, H., & Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, *37*, 185–193.

Asadi, M., Papudesi, V. N., & Huber, M. (2006). Learning skill and representation hierarchies for effective control knowledge transfer. *ICML 2005 Workshop on Structural Knowledge Transfer for Machine Learning*. Pittsburgh, PA.

Ballard, D. (1991). Animate vision. *Artificial Intelligence*, *48*, 57–86.

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

Bellman, R. (1961). *Adaptive control processes: A guided tour*. Princeton, NJ: Princeton University Press.

Berlyne, D. E. (1960). *Conflict, arousal, and curiosity*. McGraw-Hill.

Berlyne, D. E. (1965). *Structure and direction in thinking*. New York, NY: John Wiley and Sons, Inc.

Bernstein, N. (1996). On dexterity and its development. In M. Latash and M. Turvey (Eds.), *Dexterity and its development*. Mahwah, N.J.: Laurence Erlbaum Associates Inc.

Berthouze, L., Bakker, P., & Kuniyoshi, Y. (1996). Learning of oculo-motor control: A prelude to robotic imitation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Billard, A., & Mataric, M. (2001). Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, *37*, 145–160.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: MIT Press.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*, 14–23.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, *47*, 139–159.

Brooks, R. A. (1983). Planning collision free motions for pick and place operations. Memo 725. MIT-AI Lab, MIT.

Burden, R., Faires, J., & Reynolds, A. (1972). *Numerical analysis.* Boston,MA: Prindle, Weber and Schmidt.

Canny, J. (1988). *The complexity of robot motion planning.* Cambridge, MA: MIT Press.

Chamero, A. (2003). An outline of a theory of affordances. *Ecological Psychology*, *15*, 181–195.

Chiacchio, P. (2001). A new dynamic manipulability ellipsoid for redundant manipulators. *Robotica*, *18*, 381–387.

Chiacchio, P., S.Chiaverini, L.Sciavicco, & B.Siciliano (1992). Influence of gravity on the manipulability ellipsoid for robot arms. *Journal of Dynamic Systems, Measurement, and Control*, *114*, 723–727.

Chiu, S. (1987). Control of redundant manipulators for task compatability. *International IEEE Conference on Robotics and Automation.* Raleigh, NC.

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

Coelho, J., Piater, J., & Grupen, R. (2000). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *First IEEE-RAS International Conference on Humanoid Robots.* Cambridge, MA.

Coelho, J. A. (2001). *Multifingered grasping: Grasp reflexes and control context.* Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Cohen, P. R., Chang, Y., & Morrison, C. T. (2007). Learning and transferring action schemas. *Proceedings of the 2007 International Joint Conference on Artificial Intelligence*. Hyderabad, India.

Cohen, P. R., Oates, T., Adams, N., & Beal, C. (2001). Robot baby 2001. Invited talk at the Twelfth International Conference on Algorithmic Learning Theory.

Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Connolly, C., Burns, J., & Weiss, R. (1990). Path planning using laplace's equation. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*.

Connolly, C., & Grupen, R. (1994). *Nonholonomic path planning using harmonic functions* (Technical Report 94-50). University of Massachusetts, Amherst.

Craig, J. (2004). *Introduction to robotics: Mechanics and control*. New Jersey: Prentice Hall. 3rd edition.

Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience*. New York, NY: Harper Perennial.

Csikszentmihalyi, M. (1996). *Creativity- flow and the psychology of discovery and invention*. New York, NY: Harper Perennial.

Davis, E. (1990). *Representations of commonsense knowledge*. San Mateo, CA: Morgan Kaufmann.

Davis, R., Shrobe, H., & Szolovits, P. (1993). What is a knowledge representation? *AI Magazine*, *14*, 17–33.

Dayan, P., & Belleine, W. (2002). Reward, motivation and reinforcement learning. *Neuron*, *36*, 285–298.

Detry, R., Popovic, M., Touati, Y., Baseski, E., Krüger, N., & Piater, J. (2009). Autonomous learning of object-specific grasp affordance densities. *ICRA 2009 Workshop Approaches to Sensorimotor Learning on Humanoid Robots*. Kobe, Japan.

Digney, B. (1998). Learning hierarchical control structure from multiple tasks and changing environments. *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*.

Doğar, M. R., Çakmak, M., Uğur, E., & Şahin, E. (2007). From primitive behaviors to goal-directed behavior using affordances. *Proceedings of the 2007 IEEE International Conference on Intelligent Robotics and Systems (IROS)*. San Diego, CA.

Drescher, G. (1991). *Made-up minds: A constructionist approach to artificial intelligence*. Cambridge, MA: MIT Press.

Edsinger, A., & Kemp, C. C. (2006). What can i control? a framework for robot self-discovery. *Proceedings of the 6th International Workshop on Epigenetic Robotics.*

Federov, V. (1972). *Theory of optimal experiment.* New York, N.Y.: Academic Press.

Festinger, L. (1957). *A theory of cognitive dissonance.* Evanston, Row, Peterson.

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

Gandolfo, F., Sandini, G., & Bizzi, E. (1996). A field-based approach to visuo-motor coordination. *Proceedings of Workshop on Sensorimotor Coordination: Amphibians, Models, and Comparative Studies.* Sedona, Arizona.

Geib, C., Mourão, K., Petrick, R., Pugeault, N., Steedman, M., Krüger, N., & Wörgötter, F. (2006). Object action complexes as an interface for planning and robot control. *Workshop 'Toward Cognitive Humanoid Robots' at IEEE-RAS International Conference on Humanoid Robots.* Genoa, Italy.

Gibson, E. (2000). Perceptual learning in development: Some basic concepts. *Ecological Psychology*, *12*, 295–302.

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

Gizster, F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993). Convergent force fields organized in the frogs spinal cord. *The Journal of Neuroscience*, *13*, 467–491.

Goldfeder, C., , Ciocarlie, M., Dang, H., & Allen, P. (2009). The columbia grasp database. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA).* Kobe, Japan.

Gomez, G. (2004). Simulating development in a real robot. *Proceedings of the 4th International Workshop on Epigenetic Robotics.*

Grupen, R., & Souccar, K. (1993). Manipulability-based spatial isotropy: A kinematic reflex. *Proceedings of the International Workshop on Mechatronical Computer Systems for Perception and Action* (pp. 157–163).

Hager, G., Hutchinson, S., & Corke, P. (1996). A tutorial on visual servo control. *Tuioriul TT3, IEEE International. Conference on Robotics and Automution.* Minneapolis, MN.

Harlow, H. (1950). Learning and satiation of response in intrinsically motivated complex puzzle performances by monkeys. *Journal of Comparative and Psychological Psychology*, *43*, 289–294.

Hart, S., & Grupen, R. (2007). Natural task decomposition with intrinsic potential fields. *Proceedings of the 2007 International Conference on Intelligent Robots and Systems (IROS)*. San Diego, California.

Hart, S., Grupen, R., & Jensen, D. (2005). A relational representation for procedural task knowledge. *Proceedings of the 2005 American Association for Artificial Intelligence (AAAI) Conference*. Pittsburgh, Pennsylvania.

Hart, S., Sen, S., & Grupen, R. (2008a). Generalization and transfer in robot control. *8th International Conference on Epigenetic Robotics (Epirob08)*. University of Sussex, Brighton, UK.

Hart, S., Sen, S., & Grupen, R. (2008b). Intrinsically motivated hierarchical manipulation. *Proceedings of the 2008 IEEE Conference on Robots and Automation (ICRA)*. Pasadena, California.

Hart, S., Sen, S., Ou, S., & Grupen, R. (2009). The control basis api - a layered software architecture for autonomous robot learning. *2009 Workshop on Software Development and Integration in Robotics (SDIR) at the IEEE Conference on Robots and Automation (ICRA)*. Kobe, Japan.

Hayes, P. J. (1978). The naive physics manifesto. In D. Michie (Ed.), *Expert systems in the micro-electronic age*. Edinburgh University Press.

Henderson, T., & Shilcrat, E. (1984). Logical sensor systems. *Journal of Robotic Systems*, *1*, 169–193.

Herrmann, J., Pawelzik, K., & Geisel, T. (2000). Learning predictive representations. *Neurocomputing*, *32-33*, 785–791.

Hobbs, J., & Moore, R. (Eds.). (1985). *Formal theories of the commonsense world*. Noorwood, NJ: Ablex.

Horvitz, J. (2000). Mesolimbocortical and nigrostriatal dopamine responses to salient non-reward events. *Neuroscience*, *96*, 651–656.

Houk, J. C., Davis, J., & Beiser, D. (Eds.). (1995). *A model of how the basal ganglia generates and uses neural signals that predict reinforcement*. Cambridge, MA: MIT Press.

Huang, X., & Weng, J. (2002). Novelty and reinforcement learning in the value system of developmental robots. *Proceedings of the 2nd International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.*

Huber, M. (2000). *A hybrid architecture for adaptive robot control*. Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Huber, M., & Grupen, R. (1996). *A hybrid discrete dynamic systems approach to robot control* (Technical Report 96-43). Department of Computer Science, University of Massachusetts Amherst.

Huber, M., & Grupen, R. (1997). Learning to coordinate controllers - reinforcement learning on a control basis. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI).* Nagoya, JP: IJCAI.

Hull, C. (1943). *Principles of behavior: An introduction to behavior theory.* New York, NY: Appleton-Century-Croft.

Hunt, H. (1965). Intrinsic motivation and its role in psychological development. *Nebraska Symposium on Motivation*, *13*, 189–282.

Jenkins, O., & Mataric, M. (2002). Deriving action and behavior primitives from human motion data. *International Conference on Intelligent Robots and Systems.*

Johnson, M., & Lakoff, G. (1980). *Metaphors we live by.* Chicago, Illonois: University of Chicago Press.

Kagan, J. (1972). Motives and development. *Journal of Personality and Social Psychology*, *22*, 51–66.

Kakade, S., & Dayan, P. (2002). Dopamine: Generalization and bonuses. *Neural Networks*, *15*, 549–559.

Kaplan, F., & Hafner, V. V. (2005). Mapping the space of skills: An approach for comparing embodied sensorimotor organization. *Proceedings of the 4th IEEE International Conference on Development and Learning.*

Kaplan, F., & Oudeyer, P. (2003). Motivational principles for visual know-how development. *Prodeedings of the 3rd International Conference on Epigenetic Robotics.* Edinburgh, Scotland.

Kavraki, L. E., Svesktka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation*, *12*, 566–580.

Kemp, C. C., & Edsinger, A. (2006). What can i control?: The development of visual categories for a robots body and the world that it inuences. *Proceedings of the 5th IEEE International Conference on Development and Learning (ICDL-06), Special Session on Autonomous Mental Development..*

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, *5*.

Kober, J., & Peters, J. (2009). Learning motor primitives for robotics. *International IEEE Conference on Robotics and Automation (ICRA).* Kobe, Japan.

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

Koenderink, J. (1984). The structure of images. *Biological Cybernetics*, *50*, 363–370.

Konidaris, G., & Barto, A. (2007). Building portable options: Skill transfer in reinforcement learning. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 895–900).

Kraft, D., Pugeault, N., Baseski, E., Popović, M., Kragic, D., Kalkan, S., Wörgötter, F., & Krüger, N. (2008). Birth of the object: Detection of objectness and extraction of shape through object action complexes. *Proceedings of the 2008 International Conference on Cognitive Systems*. Karlsruhe, Germany.

Krogh, B. (1984). A generalized potential field approach to obstacle avoidance control. *Proceedings of the SME Conference on Robotics Research: The Next Five Years and Beyond*. Bethlehem, PA.

Krüger, N., Piater, J., Wörgötter, F., Geib, C., Petrick, R., Steedman, M., Ude, A., Asfour, T., Kraft, D., Omrcen, D., Hommel, B., Agostino, A., Kragic, D., Eklundh, J., Kruger, V., & Dillmann, R. (2009). A formal definition of object action complexes and examples at different levels of the process hierarchy. http://www.paco-plus.org.

Kuindersma, S., Hannigan, E., Ruiken, D., & Grupen, R. (2009). Dexterous mobility with the ubot-5 mobile manipulator. *Proceedings of the 14th International Conference on Advanced Robotics (ICAR)*. Munich, Germany.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 191–233.

Kupiers, B., Beeson, P., Modayil, J., & Provost, J. (2005). Bootstrap learning of foundational representations. *Developmental Robotics, AAAI Spring Symposium Series*.

Latombe, J. (1991). *Robot motion planning*. Norwell, MA: Kluwer Academic Publishers.

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning.* (Technical Report 98-11). Computer Science Deptartment, Iowa State University.

Lee, M. H., & Meng, Q. (2005). Psychologically inspired sensory-motor development in early robot learning. *International Journal of Advanced Robotics Systems*, *2*, 325–333.

Lenat, D. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, *38*, 33–38.

Lindeberg, T. (1994). *Scale-space theory in computer vision*. Dordrecht, Netherlands: Kluwar Academic Publishers.

Littman, M. L., Sutton, R., & Singh, S. (2002). Predictive representations of state. *In Advances In Neural Information Processing Systems (NIPS) 14* (pp. 1555–1561). MIT Press.

Lörken, C., & Hertzberg, J. (2008). Grounding planning operators by affordances. *Proceedings of the 2008 International Conference on Cognitive Systems*. Karlsruhe, Germany.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Lozano-Pérez, T. (1981). Automatic planning of manipulation transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, *11*, 681–698.

Lozano-Pérez, T., Jones, J. L., Mazer, E., & O'Donnell, P. A. (1989). Task-level planning of pick-and-place robot motions. *Computer*, *22*, 21–29.

Lungarella, M., Metta, G., Pfeifer, R., & Sandini, G. (2003). Developmental robotics: A survey. *Connection Science*, *15*, 151–190.

Marshall, J., Blank, D., & Meeden, L. (2004). An emergent framework for self-motivation in developmental robotics. *Proceedings of the 4th IEEE International Conference on Development and Learning*.

McGovern, A. (2002). *Autonomous discovery of temporal abstractions from interaction with an environmentt*. Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in variable-reward hierarchical reinforcement learning. *Workshop on Transfer Learning at Neural Information Processing Systems*. Cornvallis, Oregon.

Metta, G. (2000). *Babybot: A study into sensorimotor development*. Doctoral dissertation, LIRA-Lab (DIST).

Microsoft Co. (2008). Microsoft robotics developers studio. http://msdn.microsoft.com/en-us/robotics/.

Minsky, M. (1974). A framework for representing knowledge. Memo 306. MIT-AI Lab, MIT.

Modayil, J., & Kupiers, B. (2007). Autonomous development of a grounded object ontology by a learning robot. *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*.

Montgomery, K. (1954). The role of exploratory drive in learning. *Journal of Comparative and Psychological Psychology*, *47*, 60–64.

Muchisky, M., Gershkoff-Stowe, L., Cole, E., & Thelen, E. (1996). The epigenetic landscape revisted: A dynamic interpretation. *Advances in Infancy Research*, *10*, 121–159.

Mugan, J., & Kuipers, B. (2007). Learning distinctions and rules in a continuous world through active exploration. *7th International Conference on Epigenetic Robotics (Epirob07).*

Mugan, J., & Kuipers, B. (2008). Towards the application of reinforcement learning to undirected developmental learning. *8th International Conference on Epigenetic Robotics (Epirob08).*

Mussa-Ivaldi, F. A., & Bizzi, E. (2000). Motor learning through the combination of primitives. *Philosophical Transactions of the Royal Society of London*, *355*, 1755–1769.

Mussa-Ivaldi, F. A., Gizster, F., & Bizzi, E. (1994). Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of the Sciences, USA*, *91*, 7534–7538.

Nakamura, Y. (1991). *Advanced robotics: Redundancy and optimization.* Addison-Wesley.

Natale, L. (2004). *Linking action to perception in a humanoid robot: A developmental approach to grasping.* Doctoral dissertation, LIRA-Lab, DIST, University of Genoa.

Newell, A., & Simon, H. A. (1961). GPS: A program that simulates human thought. In *Lernende automaten.* Munich, Oldenbourg KG: MIT Press.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 139–158.

Oates, T. (2001). *Grounding knowledge in sensors: Unsupervised learning for language and planning.* Doctoral dissertation, University of Massachusetts Amherst.

Ostroff, J. S., & Wonham, W. M. (1985). A temporal logic approach to real time control. *24th IEEE Conference on Decision and Control*, *24*, 656–657.

Oudeyer, P., & Kaplan, F. (2008). How can we define intrinsic motivation? *8th International Conference on Epigenetic Robotics (Epirob08).*

Oudeyer, P., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computing*, *11*, 265–286.

Oudeyer, P., Kaplan, F., Hafner, V. V., & Whyte, A. (2005). The playground experiment: Task-independent development of a curious robot. *Proceedings of the AAAI Spring Symposium on Developmental Robotics.*

Papudesi, V. N., & Huber, M. (2006). Learning behaviorally grounded state representations for reinforcement learning agents. *Prodeedings of the 6th International Conference on Epigenetic Robotics.* Paris, France.

Pfeifer, R. (2002). Robots as cognitive tools. *International Journal of Cognition and Technology*, *1*, 125–143.

Piaget, J. (1952). *The origins of intelligence in childhood.* International Universities Press.

Piater, J. H. (2001). *Visual feature learning.* Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Piater, J. H., & Grupen, R. A. (2000). Constructive feature learning and the development of visual expertise. *Proceedings of the Seventeenth International Conference on Machine Learning.* Stanford, CA.

Platt, R. (2006). *Learning and generalizing control based grasping and manipulation skills.* Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Platt, R., Fagg, A. H., & Grupen, R. (2002). Nullspace composition of control laws for grasping. *International Conference on Intelligent Robots and Systems (IROS).* Laussane, Switzerland: IEEE/RSJ.

Power, T. G. (2005). *Play and exploration in children and animals.* New Jersey: Lawrence Erlbaum Associates.

Quinlan, J. (1993). *C4.5: Programs for machine learning.* San Mateo, CA.: Morgan Kaufmann Publishers.

Ranjbaran, F., Angeles, J., & Kecskemethy, A. (1996). On the kinematic conditioning of robotic manipulators. *International IEEE Conference on Robotics and Automation.* Minneapolis, MN.

Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning.* Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Reed, P., Mitchell, C., & Nokes, T. (1996). Intrinsic reinforcing properties of putatively neutral stimuli in an instrumental two-lever discrimination task. *Animal Learning and Behavior*, *24*, 38–45.

Reeke, G., Sporns, O., & Edelman, G. (1990). Synthetic neural modeling: the 'darwin' series of recognition automata. *Proceedings of the IEEE* (pp. 1498–1530).

Rimon, E., & Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, *8*, 501–518.

Rohanimanesh, K., & Mahadevan, S. (2005). Coarticulation: An approach for generating concurrent plans in markov decision processes. *Proceedings of the 22nd International Conference on Machine Learning (ICML-2005).* Bonn, Germany.

Rome, E., Hertzberg, J., & Dorffner, G. (Eds.). (2006). *Towards affordance-based robot control.* Berlin: Springer.

Rosenstein, M., & Grupen, R. (2002). Velocity-dependent dynamic manipulability. *International IEEE Conference on Robotics and Automation.* Washington, DC.

Roy, D. (1999). *Learning words from sights and sounds: A computational model.* Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Şahin, E., Çakmak, M., Doğar, M., Uğur, E., & Üçoluk, G. (2007). To afford of not to afford: A formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, *4*, 447–472.

Salisbury, J., & Craig, J. (1982). Articulated hands: Force control and kinematic issues. *The International Journal of Robotics Research*, *1*, 4–17.

Sandini, G., Metta, G., & Konczak, J. (1997). Human sensori-motor development and artificial systems. *Proceedings of AIR & IHAS.* Japan.

Saunders, R. (2002). *Curious design agents and artificial creativity: A synthetic approach to the study of creative behavior.* Doctoral dissertation, University of Sydney.

Saxena, A., Driemeyer, J., & Ng, A. (2007). Robotics grasping of novel objects using vision. *International Journal of Robotics Research*, *27*, 157–173.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences.*

Schaal, S., Ijspeert, A., & Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions: Biological Sciences.*

Schöner, G., & Thelen, E. (2006). Using dynamic field theory to rethink infant habituation. *Psychological Review*, *113*, 273–299.

Schmidhuber, J. (1991a). Curious model-building control systems. *Proceedings of the International Joint Conference on Neural Networks.*

Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredome in model-building neural controllers. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.*

Schmidhuber, J., & Storck, J. (1993). *Reinforcement driven information acquisition in nondeterministic environments* (Technical Report). Fakultat fur Informatik, Technische Universit at Munchen.

Schultz, W., & Dayan, P. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593–1599.

Şimşek, Ö., & Barto, A. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proceedings of the 21st International Conference on Machine Learning.*

Sinapov, J., Wiemer, M., & Stoytchev, A. (2009). Interactive learning of the acoustic properties of household objects. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA).* Kobe, Japan.

Singh, P. (2001). The public acquisition of commonsense knowledge. http://www.openmind.org/commonsense/pack.html, 2001. The Open Mind Commonsense project.

Singh, S., Barto, A., & Chentanez, N. (2004a). Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems (NIPS).*

Singh, S., James, M. R., & Rudary, M. R. (2004b). Predictive state representations: a new theory for modeling dynamical systems. *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence* (pp. 512–519). Arlington, Virginia, United States: AUAI Press.

Slack, J. (2002). Conrad hal waddington: the last renaissance biologist? *Nature Reviews Genetics*, *3*, 889–895.

Spencer, J. P., Clearfield, M., Corbetta, D., Ulrich, B., Buchanan, P., & Schoner, G. (2006). Moving toward a grand theory of development: In memory of Esther Thelen. *Child Development*, *77*, 1521–1538.

Sporns, O. (2003). Embodied cognition. *The Handbook of Brain Theory and Neural Computation.* Cambridge, MA: MIT Press.

Stark, M., Lies, P., Zillich, M., Wyatt, J., & Schiele, B. (2008). Functional object class detection based on learned affordance cues. *Sixth International Conference on Computer Vision Systems, Vision for Cognitive Systems.* Santorini, Greece.

Steels, L., & Vogt, P. (1997). Grounding adaptive language games in robotic agents. *Proceedings of the 4th European Conference on Artificial Life.*

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

Sutton, R., & Barto, A. (1998). *Reinforcement learning.* Cambridge, Massachusetts: MIT Press.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Sweeney, J. D., Brunette, T., Yang, Y., & Grupen, R. A. (2002). Coordinated teams of reactive mobile platforms. *Proceedings of the 2002 IEEE Conference on Robotics and Automation*. Washington, D.C.

Sweeney, J. D., & Grupen, R. (2007). A model of shared grasp affordances from demonstration. *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*. Pittsburgh, PA.

Thelen, E., & Bates, E. (2003). Connectionism and dynamic systems: Are they really different? *Developmental Science*, *6*, 378–391.

Thelen, E., & Smith, L. B. (1994). *A dynamic systems approach to the development of cognition and action*. MIT Press.

Thrun, S. (1995). Exploration in active learning. *The Handbook of Brain Theory and Neural Computation*. Cambridge, MA: MIT Press.

Uğur, E., Oztop, E., & Şahin, E. (2009). Learning object affordances for planning. *ICRA 2009 Workshop Approaches to Sensorimotor Learning on Humanoid Robots*. Kobe, Japan.

Uppala, J., Karuppiah, D., Brewer, M., Ravela, C., & Grupen, R. (2002). On viewpoint control. *International IEEE Conference on Robotics and Automation*. Washington, D.C.

Waddington, C. (1943). Organisers and genes. *American Midland Naturalist*, *30*, 811–812.

Watkins, D., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Weng, J. (2002). A theory for mentally developing robots. *Proceedings of the 2nd International Conference on Developmental Learning*.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., & Thelen, E. (2001). Autonomous mental development by robots and animals. *Science*, *291*, 599–600.

White, R. (1959). Motivation reconsidered: The concept of competence. *Psychological Review*, *66*, 297–333.

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: A hierarchical bayesian approach. *Proceedings of the 2007 International Joint Conference on Machine Learning*. Cornvallis, Oregon.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, *9*, 625–636.

Winograd, T. (1971). *Procedures as a representation for data in a computer program for understanding natural language* (Technical Report TR-235). AI Lab, MIT, Cambridge, MA.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, *4*, 3–9.