The Dissertation Committee for Joseph Varughese Modayil
certifies that this is the approved version of the following dissertation:

Robot Developmental Learning of an Object Ontology Grounded in Sensorimotor Experience

Committee:

____
Benjamin Kuipers, Supervisor

____
Raymond Mooney

____
Peter Stone

____
Brian Stankiewicz

____
David Kortenkamp

Robot Developmental Learning of an Object Ontology Grounded in Sensorimotor Experience

by

Joseph Varughese Modayil, B.Sc., M.Sc.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2007

For Joanna

Acknowledgments

This work would not have been possible without the support of many individuals.

My advisor Ben Kuipers has been a source of constant support and stimulation. Without his guidance, my research would have been both less ambitious and less interesting. His relentless pursuit of clear answers to important questions in science is tempered with joviality and compassion. These qualities have shaped my approach to research.

My thesis committee has significantly influenced this work, both in my research and by the demand that lofty claims are supported by empirical evidence. Ray Mooney introduced me to the importance of evaluation in empirical AI. Brian Stankiewicz gave me an understanding of how psychologists can evaluate human performance, and this allowed me to evaluate robot performance. David Kortenkamp inspired me to think about how a robot can pay attention to events outside of a narrowly specified task with a talk at AAAI that mentioned the difficulty of having a planetary rover notice a dinosaur bone. Peter Stone encouraged me to consider the relevance of reasoning about agents and actions on physical robots.

My lab colleagues have patiently endured many discussions, thoughtfully prodded on the weak spots in my research, suggested improvements, critiqued documents, and aided in experimental setups. Special thanks go to Patrick Beeson, Aniket Murarka, Matt MacMahon, Subramanian Ramamoorthy, Jefferson Provost, Harold Chaput, Jonathan Mugan, Changhai Xu, and Shilpa Gulati.

My many friends in Austin have preserved my sanity over the years. Special thanks go out to Mikhail Bilenko, Matt Horstman, Prem Melville, Amol Nayate, and Lucas Wilcox for being great friends and roommates. Thanks are also due to Katherine Mack whose excitement in robotics sustained my faith.

Finally, thanks are due to my family for supporting me through the years. I would not have been able to come this far without their constant encouragement. Their support has been immeasurable.

JOSEPH VARUGHESE MODAYIL

The University of Texas at Austin
August 2007

Robot Developmental Learning of an Object Ontology Grounded in Sensorimotor Experience

Publication No. ______________________

Joseph Varughese Modayil, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Benjamin Kuipers

How can a robot learn to conceptualize its environment in terms of objects and actions, starting from its intrinsic “pixel-level” sensorimotor interface? Several domains in artificial intelligence (including language, planning, and logic) rely on the existence of a symbolic representation that provides objects, relations, and actions. With real robots it is difficult to ground these high-level symbolic representations, because hand-written object models and control routines are often brittle and fail to account for the complexities of the real world. In contrast, developmental psychologists describe how an infant’s naïve understanding of objects transforms with experience into an adult’s more sophisticated understanding. Can a robot’s understanding of objects develop similarly?

This thesis describes a learning process that leads to a simple and useful theory of objects, their properties, and the actions that apply to them. The robot's initial "pixel-level" experience consists of a range-sensor image stream and a background model of its immediate surroundings. The background model is an occupancy grid that explains away most of the range-sensor data using a static world assumption. To this developing robot, an "object" is a theoretical construct abduced to explain a subset of the robot's sensorimotor experience that is not explained by the background model.

This approach leads to the Object Perception and Action Learner (OPAL). OPAL starts with a simple theory of objects that is used to bootstrap more sophisticated capabilities. In the initial theory, the sensor returns explained by an object have spatial and temporal proximity. This allows the robot to individuate, track, describe, and classify objects (such as a chair or wastebasket) in a simple scene without complex prior knowledge [73]. The initial theory is used to learn a more sophisticated theory. First, the robot uses the perceptual representations described above to create structurally consistent object models that support object localization and recognition [74]. Second, the robot learns actions that support planning to achieve object-based goals [75].

The combined system extends the robot's representational capabilities to include objects and both constant and time-varying properties of objects. The robot can use constant properties such as shape to recognize objects it has previously observed. It can also use time-varying properties such as location or orientation to track objects that move. These properties can be used to represent the learned preconditions and postconditions of actions. Thus, the robot can make and execute plans to achieve object-based goals, using the pre- and post-conditions to infer the ordering constraints among actions in the plan.

The learning process and the learned representations were evaluated with metrics that support verification by both the robot and the experimenter. The robot learned object shape models that are structurally consistent to within the robot's sensor precision. The learned shape models also support accurate object classification with externally provided labels. The robot achieved goals specified in terms of object properties by planning with the learned actions, solving tasks such as facing an object, approaching an object, and moving an object to a target location. The robot completed these tasks both reliably and accurately.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

People exhibit intelligence in language, planning and reasoning. People are better than robots at these tasks because people are good at decomposing the world into weakly interacting components. In particular, people decompose the world into objects. A person observes a scene and conceptualizes it in an internally constructed ontology of objects that represents object properties, relations, and potential interactions. These internal representations can be employed to solve a variety of tasks.

Although robot perception and control can benefit from an object based model of the world, years of research have shown that constructing such models is difficult. Perception is often hard-coded or learned offline for a handful of task specific object classes. Control laws require a programmer to select a task relevant world model with state variables that are estimated from the robot's sensory experience. This approach to perception and action leads to robots that can rarely solve interesting tasks in dynamic environments. The limitations often arise not because the robot's sensors and motors are inadequate, but because the robot cannot *perceive* objects—it cannot construct stable object representations.

Object perception is a critical developmental step for a person learning to interact with the environment. Whether human or robot, an intelligent agent can use perceived objects to select appropriate actions based on past experiences. Moreover by using an object representation, a robot can specify and test hypotheses, which can facilitate autonomous learning of actions [85, 30].

In spite of its apparent utility, no system for general purpose object perception and action is currently available. This thesis will first present desirable attributes for a developmentally inspired object perception system and then describe an implementation that meets these requirements using the sensory and computational resources available on modern robots. The Object Perception and Action Learner (OPAL) is used to test the following claims. One, starting from egocentric and allocentric representations of sensory experience, a robot is able to reliably perceive objects in unstructured environments without complex prior knowledge. Two, the robot is able to create structurally consistent object models that support fast localization and recognition. Three, the representations generated by perception are sufficient for learning actions that can be used by a planner to achieve goals.

1.1 Motivating the Development of an Object Ontology

Decades of AI research have yielded techniques for learning, inference, and planning that depend on human-provided ontologies of self, space, time, objects, actions, and properties. Since robots are constructed with low-level sensor and motor interfaces that do not provide these concepts, the human robotics researcher must create the bindings between the required high-level concepts and the available low-level interfaces. This raises the developmental learning question of how a learning agent can create high-level concepts from its own low-level experience.

People have acquired their cognitive capabilities through evolution and individual development. Understanding how cognitive competence can arise in autonomous robots poses an interesting developmental learning problem at the boundary between robotics, machine learning and knowledge representation. Robotics research provides an embodied agent that uses sensorimotor interfaces to interact with the world. Machine learning research provides a collection of methods for learning to perform tasks. Knowledge representation research provides symbolic inference methods that mimic several cognitive capabilities of people. This raises the research question of whether these components can be combined and extended to create intelligent machines.

Since objects are natural for human reasoning, a great deal of cognition relies on the availability of objects. Most nouns in natural language refer to objects. Logical reasoning occurs in symbolic worlds populated with objects. Methods for motor skill acquisition and intelligent behavior in robots assume that relevant objects can be identified in the environment. Object perception transforms the robot's experience from sensorimotor impressions into persistent object representations that facilitate communication, planning and learning.

1.2 The Apparent Simplicity of Object Perception

On first consideration, object perception is simple and effortless. As people, we look around and are immediately aware of a world filled with objects. On further reflection, we realize that what we sense with our eyes, ears and skin are not the objects themselves, but rather impressions from the world that our minds are aggregating to form object models. We have no sensor that returns objects. However, we have a deep-seated tendency to perceive the world as being composed of objects, and this tendency is not easily changed.

The perception of objects occurs without conscious effort. Infants appear to perceive objects. Babies see and mimic faces from birth. Shortly after birth, chicks imprint on a nearby object and they perceive it as a relative [71]. This example suggests that extensive worldly experience is not always required for object perception. If even baby bird-brains can see objects, how difficult can object perception be? The sensory input for object perception appears to be similar for robots and animals. Visual input is represented by neuron activations for animals and pixel activations for robots and the two inputs are roughly comparable.

1.3 The Difficulty of Object Perception

Children parse their worlds into distinct objects. A child on the beach will happily perceive a wave and a beach-ball as objects. However, the beach-ball and the wave are so different that it is difficult to define what makes them both objects. The boundary of the beach-ball is clearly visible; the boundary of a wave is not. The beach-ball is nearly rigid; the wave is amorphous. The beach-ball exists for an extended duration; the wave ceases to exist when it hits the beach. The matter that comprises the beach-ball is largely fixed in time as an amount of plastic in the ball's shell. The matter that composes the wave is largely transient, since the water in the wave does not move along with the wave.

In spite of all these distinctions, the most intuitive interpretation for a child is that both the beach-ball and the wave are objects. A physical characterization of the child's intuitive notion of an object is surprisingly difficult. Our first inclination is to say that an object is a physical entity. But this does not explain the existence of the wave, whose constituent matter is constantly changing. Alternatively, we can claim that an object's existence is provided by the extent of its physical form, the boundary between where the object exists and where it does not. This does not suffice to describe a wave that does not posses a precise boundary, for the wave's crest is composed of a mixture of water and air. We can claim that although objects in the world might not be simple to define, objects are segmented from the images formed by the world projected onto our eyes, and that this segmentation into figure and ground is unambiguous. However, this interpretation fails because several optical illusions provide multiple interpretations for the "object". Figure and ground can be interchanged with minimal effort in the classic vase and faces illusion.

This leaves the explanation that objects are being formed by our minds. Initially, this is troubling. How did they get there? How does everyone appear to perceive the same objects? Why do our minds form objects? Are there objects in our heads? How can we provide object models for robots? Do roboticists have to worry about things like waves, or is most of the world sufficiently well behaved that robots can manage by believing that objects exist in the world? How can an experimenter validate that the objects perceived by the robot correspond to objects perceived by people? There are two ways to proceed given the ambiguous nature of object semantics in the physical world: to forgo object models completely, or to construct object models that explain an agent's sensorimotor experience of the world.

The first approach leads to computational methods that ignore object perception to avoid the fragility and computational burden of reliably finding object instances in an agent's sensory input. This is the approach promoted in behaviorism and reactive robotics [16], which advocate having agents that do not model the world. However, these approaches fail to provide insight into the mechanisms that generate the objects that humans manifestly use. They also fail to provide interfaces that allow humans to provide information about objects. Finally, these methods are limited by their inability to generalize from prior experiences with objects.

Approaches that build object models use different object representations depending on the task. Machine vision addresses several tasks including image segmentation (which groups each pixel in an image with neighboring pixels that lie on the same object), structure recovery (discovering the three dimensional structure of the object using a variety of information sources including motion, shading, stereo disparity, and texture), registration (finding the object model parameters that best describe an object's pose in the world), and recognition (matching an observed entity with a known class). Many algorithms can also jointly perform multiple steps (for example simultaneous classification and segmentation), which often provides better performance though at a higher computational cost.

These techniques can work well when their prerequisites are satisfied, and the prerequisites tend to be reasonable for controlled environments. Because of their reliability, many machine vision algorithms have been successfully deployed in industrial settings that provide a controlled closed-world environment where only a few rigid objects must be recognized and where the object models are provided *a priori*. These prerequisites are significantly more onerous for mobile robots that contend with worlds that are not closed (unknown environments and objects), and limited computation. Although traditional 3D structural object models can support computationally efficient object localization, existing theories provide few clues about how such complex models can acquired autonomously.

1.4 Bootstrap Learning

This work on learning object representations is part of a larger research endeavor to understand how an autonomous agent can develop an understanding of its environment. It is inspired by the observation that adults acquire their cognitive capabilities through a combination of evolution and development. It sometimes is called bootstrap learning to indicate that the robot's existing representations and capabilities must be used to bootstrap new representations and algorithms. Although this approach is relatively new and the goals are ambitious, significant progress has already been made.

The term bootstrap learning is used in [60] to describe a self-supervised learning process. [1] Initially, the robot has the ability to use its travel history for place recognition. This ability was used to provide labeled examples to train a fast place recognition algorithm that only requires a single sensor observation. This new capability allows the robot to relocalize in its map if it becomes lost, since the fast process for place recognition does not rely on the robot's having an accurate model of its past experience.

Pierce and Kuipers [85] showed how a simulated robot can take a bag of sensors and determine their spatial structure, use the perceptual spatial structure to reformulate its motor interface, and create control laws to navigate through large scale spaces. Perceptual organization was extended by Olsson and colleagues [82] to use an information metric to measure the distance between sensors. The work on learning spatial navigation is extended by Provost and colleagues. [87] to operate without requiring control templates. Research by Stronger and Stone [103] explore how an agent can autonomously learn the relationship between continuous sensors and effectors.

In a discretized environment, Drescher [30] showed how a simulated agent can generate new states and learn rules to predict the consequences of its actions. Chaput [20] extended this work by showing how this learning process can be more efficiently implemented using self organizing maps.

[1]The term is also used for the related task of semi-supervised learning [114].

1.5 Learning about Objects

This thesis describes how an agent can bootstrap useful object representations using simple theories of objects to explain the robot's experiences in unstructured environments. In contrast, most other approaches construct object models by some combination of external knowledge of specific objects of interest and learning environments that are constructed to simplify perception.

This thesis makes the assumption that two ontologies are already developed and available on the learning robot. One is egocentric, describing the world using sensors and acting in the world using motors. The second is allocentric, describing the environment using a static reference frame, and acting in the world by setting goal points, planning motor commands and performing them. The first is provided by default on engineered mobile robots, and the second is provided for certain robot configurations by localization and navigation algorithms. Both descriptions are at the level of pixels since the value of each feature provides relatively little information. These ontologies provide the input for learning an object ontology.

The goal is to understand how an agent can move from a pixel-level understanding of its sensorimotor experience to an object level understanding. The Object Perception and Action Learner (OPAL) permits robust object perception in spite of environmental clutter. The first step is to describe a process by which early perceptual learning can occur without requiring engineered environments, precise sensor modeling, or restrictive assumptions about the physical nature of the objects. Subsequent steps improve on the initial capabilities by creating structural object models and learning actions that change object properties.

The main results of this work demonstrate how a robot can start the development of an object ontology by tracking dynamic objects, forming structural models of object shapes, and learning actions for objects. The next chapter reviews relevant related work and the insights provided from prior research. The following chapter provides an overview of the object ontology, which is then described in detail in the remainder of the thesis.

Chapter 2

Related Work in Object Perception and Action

The question of how an agent can perceive and act on objects is not new. Prior research into object representations in natural and artificial systems provides both motivation and guidance. Studies of natural systems demonstrate both that multiple representations are used and that control mechanisms are involved at a low level. Studies in artificial systems demonstrate impressive performance when task appropriate prior knowledge is available. However, natural systems are usually too complex to be completely understood, and artificial systems are rarely robust in unstructured environments due to difficulties in acquiring adequate background knowledge.

This chapter surveys research on how agents perceive objects and how they interact with them. Although there are useful studies on action, particularly haptic control in humans and reinforcement learning in artificial systems, the majority of these studies focus on perception. Object perception is studied for several distinct purposes including control, recognition, communication, and inference. The variation in the goals and topics of study in different papers yields significantly different answers to the following questions.

- What constitutes an object representation for an agent?

- How are the object representations used in perception?
- What actions do the object representations afford?

Research into object perception can be broadly divided into the study of biological and artificial systems. Object perception in biological systems is studied in physiology and psychology. Researchers in physiology investigate perception in the nervous system and in the brain. Psychological research can be divided into three relevant areas: comparative psychologists study perception in animals, developmental psychologists study perception in infants, and perceptual psychologists study perception in mature humans. Object perception in artificial systems is studied by researchers in machine vision and robotics. The machine vision community has developed an array of techniques for object segmentation and recognition. Much of this work assumes that the environment can be controlled, the sensors have been modeled, and that object models have already been acquired. Robotics researchers often use task specific heuristics (blobs of the right size are legs, people can be discovered using heat sensing and color matching) to perceive objects in unconstrained environments. These models tend to sacrifice generality for robustness and simplicity.

2.1 Natural Systems

Animals, infants and adults are natural systems that perceive and act on objects. Neural representations that are directly observable in natural systems are typically far from high level cognition. However, indirect experiments are still able to shed light on many aspects of natural object representations. Two ideas recur in several studies. One is that natural noise can be robust to large amounts of systematic and random perturbations. Another is that multiple coupled object representations are used. The robustness and adaptability of these systems contrasts with the artificial systems discussed later. Finally, since objects are defined by the agent, the cognitive abilities and needs of the agent result in substantially different object representations for animals, infants, and adults.

2.1.1 Human Physiology

It is particularly interesting when the usually robust adult human visual system fails. Many cases arise from patients who exhibit unusual behavior after some mental trauma. One account [43] describes a patient after a near asphyxiation who was unable to verbally describe visual orientations, nor was she able to move her hand to match the orientation of a slot. However, she was perfectly capable of "mailing" a letter into a slot, a task that required the same perceptual and control skills. Further experiments suggest the motor control system can perceive the primary axis of an object, but does not have significantly more shape information, as the subject was only 50% accurate at "mailing" an elongated cross into its corresponding slot.

Livingstone and Hubel [67] discuss the separation of the human visual system into two cooperating systems where each system has a cortical/subcortical division, the dorsal/magnocellular system and ventral/parvocellular system. The magnocellular system handles depth and motion at high speeds. The parvocellular system works on a slower time scale and handles color and form. They separate the parallel systems through optical illusions that work with intensity contrasts but change radically under equiluminance displays (using two colors for contrast). Their examples demonstrate perceptual differences in scenarios with the same information, which supports the existence of two distinct pathways. Although the cortical/subcortical division looks convincing, other research suggests that this claim is oversimplified.

Mareschal and Johnson [69] explore the two cortical visual object processing routes. They state the dorsal route is concerned with the "where" information needed for action (location, motion, size, crude shape but no color or face information) and the ventral route is concerned with the "what" information needed for identification (color, face information, size, shape, but no location). They postulate that these two routes are integrated in the frontal lobes. Since the frontal lobes are not fully developed in four month old infants, tests on these infants should reveal distinctions in performance. The results of object occlusion studies suggest that these infants have difficulty remembering information from both routes. Moreover, objects that afford action retain location properties while other objects preserve identification properties. This is surprising, since the authors state that infants are unable to physically reach for objects at this age. This suggests the infants are remembering how other agents interact with objects.

Although vision research is the most prevalent in the literature, there is relevant research into other senses including sound, touch and smell. Research into perceptual grouping of auditory cues found Gestalt principles similar to those observed in vision [15] and has been explored computationally [25]. Research into grasping has shown that models of the backs of objects are formed for grasping, whereas vision is used to generate models of the front of objects [64]. Even smell may be subject subject to a perceptual grouping process that uses a variant of blind source separation to identify distinct objects [50].

2.1.2 Comparative Psychology

A study of object parsing in rhesus monkeys [78] compared animal responses to infant responses. The study finds that monkeys show surprise similarly to infants older than 11 months when presented with an unusual support scenario (support from above). However, they differ in surprise to support situations involving hands: infants recognize that hands provide support from an age of 6 months but adult rhesus monkeys seem to ignore hands. Both appear to parse objects using feature cues. Earlier work demonstrates the use of spatio-temporal cues for object perception in younger infants.

There are several animals that learn to use objects in complex ways. Crows create and use tools to fetch food [112]. Grey parrots can name objects and solve simple grouping and counting tasks [83]. Otters learn to use stones as anvils in the wild [48]. These examples provide sample tasks that are cognitively simpler than matching the capabilities of an adult human.

Several examples of object perception and action from [71] are described below. The examples provide strong evidence that animals perceive objects but interact with them with differing levels of sophistication.

The well-known "monkeys and bananas" logic problem is likely inspired by Köhler's work [56]. Köhler described chimpanzees stacking boxes to reach bananas suspended from a ceiling. He also filmed chimpanzees using poles to bring down suspended bananas, as well as chimpanzees who used combinations of boxes and poles to get the bananas. Other chimpanzees would place a pole on the ground, run up the pole and grab the bananas before the pole toppled. In a different setup, he showed a chimpanzee in a cage with two poles, where neither pole was long enough to reach the bananas. After failing to reach the bananas with the short poles, the chimpanzee plays with the poles, accidentally fits them together to make a long pole, and then immediately uses the extended pole to reach the bananas.

Köhler used these scenarios to claim the chimpanzees were showing signs of insight. This interpretation may be overly simplistic, since Schiller [91] observed chimpanzees stacking crates for fun. He observed that chimpanzees stack crates and climb to the top with their arms outstretched even when they have never been presented with bananas suspended from a ceiling. Still, Köhler's work and films show chimpanzees can acquire skills with objects, and can organize tools in a variety of novel ways.

Wild chimpanzees use grass and twigs as tools, using them to extract termites from mounds [110]. This skill is learned by child chimps from adult chimps. This suggests that chimpanzees must learn to discretize part of their environment into objects for use as tools, they learn new control skills with these objects, and they learn which objects can be acted on with these tools.

Herring gulls have problems distinguishing between eggs and boxes [4]. This suggests that they do not construct a structural representation for an object before performing classification. Their poor classification skills stand in contrast to birds targeted by cuckoos. Cuckoos lay their eggs in the nests of other birds, and individual cuckoos specialize in making their eggs look like those of a particular host species. Species which have been targeted by the cuckoo in the past tend to reject eggs which do not look substantially similar to their own [28].

2.1.3 Developmental Psychology

Spelke [99] has investigated the mechanisms used by infants to find objects in the environment. Infants begin individuating objects by using low level features, and they increase their inference capabilities with age. Infants use similar motion to group features into objects (the Gestalt mechanism of “common fate”) before they gain related motor skills (head control and grasping). After these motor skills are acquired, they acquire the remaining Gestalt grouping skills. A later paper [98] discusses spatio-temporal integration of edge filters into the same framework.

Bloom [10] suggests that these “Spelke objects” discovered by prelinguistic children correspond directly to those found relevant to language acquisition. He claims that the rapid vocabulary learning in young children is accounted for in part by a child’s knowledge of these objects.

Carey and Xu [18] speculate that the mechanisms used in infant vision are the same as those used in mid-level vision by mature subjects. The speculative nature of this claim reveals the current limitations in the understanding of perceptual development. The paper also points to computational models of attention and object perception, notably the MOT (multiple object trackers) model of Pylyshyn [92].

2.1.4 Perceptual Psychology

Braun [14] showed that people can track “objects” that are spatially overlapping multidimensional features that are continuously changing. Each object was a translucent diffraction pattern with three features (color, orientation, and wavelength). The experimenters overlapped two patterns and continuously varied the feature values. Observers were told to track the two objects. By the end of each trial, the two object had exchanged their original feature values. The results showed that people tend to track an object as a whole instead of as individual features.

Geisler and Diehl [40] present a summary of recent psychological experiments that use Bayesian ideal observers to perform perceptual tasks. Since different assumptions lead to different ideal observers, different theories can be compared based on how well their ideal observers match human performance. This approach provides insight into which perceptual representations are effective for capturing environmental statistics and how well these statistics predict human behavior. This work supports the claim that an agent can improve its perceptual skills by learning the statistics of its operating environment.

Cavanaugh and colleagues [19] measure the response time required to classify the motion of moving dots. Even when patterns are simple (rotating pairs) or familiar (human gaits), it appears that attention is required. Moreover, the authors suggest that the creation of links between the dots is an active part of perception and not recognized by innate "sprites". The authors claim that sprites can be used to reduce cognitive load once a dynamic representation is constructed.

Scholl, Pylyshyn and Feldman [92] explore reasonable definitions of visual objects. They use multiple object tracking experiments to study the effect of distractors on an observer's object construction process.

Kubovy and Valkenburg [59] state that definitions of "object" tend to be rooted in the visual system. However, this leaves the status of objects in other modalities unclear. They give a definition of objects based on an observer's ability to separate an object signal from the background.

2.1.5 Action

There are at least two reasons to characterize object properties for action. One is to generate a functional object description, namely to determine the resulting state from performing an action on an object. The other is to develop a model of an object's dynamics for skilled motion control.

Several studies on control in psychology examine hand motor skills. Lederman and Klatzky [64] provide an overview of a sequence of studies on active haptic perception. People use several actions to characterize and recognize objects. By manufacturing a set of physical objects that vary in shape, shape envelope, mass, hardness, and texture, the experimenters were able to determine that people select actions that efficiently identify the objects.

Another study explored the relationship between prediction and control [39]. Prediction and control are intimately related as prediction provides a forward model of a system (an action from a state leads to a next state), and control requires an inverse model (getting to a desired state from a known state requires the identification of appropriate actions). The experimenters required a subject to push a lever that has a non constant load. By measuring grip and load along the path, the experiment demonstrated that people learned to predict the required force before they learned to control the load. By careful construction of apparatus, experimenters are able to decouple actions from sensations [58]. This approach to generating control from prediction is employed by artificial systems using model-based reinforcement learning.

2.2 Artificial Systems

Object perception in artificial systems is still heavily influenced by the work of Marr [70] who advocated constructing a geometric representation of the world at each time step. The difficulties of constructing and maintaining world models at each time step led to the subsequent popularity of reactive robotics [16], which advocated using little (if any) internal representation. More recently, particle filters and Bayesian probability have been employed to simplify the process of maintaining models of the world [52]. This probabilistic approach uses knowledge of the past to propose models for the present. For this probabilistic approach, the two main tasks are to generate reasonable starting model parameters from data and to generate new models to explain data better than existing models.

In contrast with a person's perceptual robustness in arbitrary environments, machine vision studies have usually focused on generating systems with high recognition accuracy in controlled environments [36]. Many industrial object classification tasks are now computationally feasible, and system performance can be quantified and evaluated. To maximize recognition rates, machine vision systems tend to constrain both the environment and the number of potential objects. The problem of reliably perceiving an object is complicated by the many factors that affect an object's appearance including lighting, reflectance, scale, rotation and deformation.

2.2.1 Object Segmentation

Traditionally, segmentation first finds low-level features in an image and then connects these features to segment the image. Several mechanisms are known for producing reliable low-level visual features. Edge detection and kernel convolution are standard preprocessing techniques for finding interesting image features. Despite the relatively long history of this approach [70], difficulties remain with thresholding features and achieving reliable segmentation.

Sometimes, low-level features are selected to support reliable tracking. Without knowing the actual objects in a scene, many point features can be reliably tracked [96] by assuming the features arise from locally planar surfaces. With the appropriate statistical machinery, more complex entities such as splines can be used track object boundaries [52].

An alternative to tracking a single feature through multiple frames is to segment the entire image into interesting and uninteresting regions. The simplest method for this separation is to perform background subtraction when the background is static or known. If the images of the objects do not touch, then each object is a connected component. This approach to object discovery was used effectively in the VSAM project [24] to automatically discover categories of objects in an outdoor urban environment, and to classify future observations into known categories.

Moving objects can be discovered using a fixed camera [90]. First a model of the background is learned. As objects are brought into the scene, their associated pixels violate the background model and these errors can be spatially clustered to define objects. This approach allows objects to be defined without strong assumptions about object structure (e.g. rigidity, opaqueness). This approach can construct a complete object model even when the object is always partially occluded in observations. Limitations of this work include inadequate support for recognition, and the reliance on a fixed video camera.

When the background is not static, images are segmented by clustering image regions based on figural goodness [53]. Several tools are available for automatically making trade-offs between the accuracy of a description and its generality. Some approaches that have been taken include minimum description lengths [29], Bayesian techniques [55], and energy minimization [57].

Shi and Malik [95] perform image segmentation using a graph-theoretic notion of a *normalized cut*. Small portions of an image (or image sequence) are compared with their neighbors to determine similarity. This transforms the image into a graph with weights between image fragments. They introduce the criterion of a normalized cut to create a bipartition of the graph which minimizes the weight of the cut while compensating for the size of the cut. Although an exact computation of the optimal cut is NP-complete, effective approximation techniques are known. These approximation techniques are based on computing generalized eigenvectors, where the signs of the eigenvector components indicate membership. Multiple objects are found by either recursively breaking components, or by using multiple eigenvectors to generate multiple cuts. The experiments demonstrate the utility of this algorithm measuring similarity on a variety of features: motion, color, and texture. However, this method is too slow to run online.

Weiss [113] uses the apparent motion of local features as evidence for object motions. This work demonstrates how features from the overlap of two distinct objects can lead to local image motions which do not correspond to any physical entity. This model demonstrates how several optical illusions involving motion can be explained by Bayesian principles. The model encodes as prior knowledge that there is image noise and that the scene contains a few layers of slow, smooth motion. The model achieves similar degradations in behavior as humans in varying conditions. Weiss also makes the insightful observation that in the presence of noise, point and line features lie on a continuum.

Another algorithm [55] decomposes unconstrained video into a background and layers of flexible sprites. Users can click on an object to remove it from the video sequence, which in turn reveals objects that were previously occluded. This method does not need to be given structural object models, but the input does include the number of sprites and layers. The algorithm permits semi-transparent objects to be present, and it handles background motion. The method uses an approximate Bayesian approach for inference.

Nicolescu and Medioni [80] describe a method for grouping moving objects in images using 4D tensors that encode local velocity and position. They note that motion estimates near object boundaries are uncertain. The algorithm produces accurate motion segmentation within these regions by only using neighbors on the same side of the perceived boundary to influence the motion estimate. They have tested their algorithm on both natural and synthetic data.

2.2.2 Object Recognition

Object recognition is a motivating task for much of machine vision. The standard approach is to perform segmentation, identify the structure of each object, and then classify each object based on its structure. Faugeras [36] describes the geometry and algorithms required for the recognition of rigid three dimensional objects. This work is mostly concerned with object recognition and registration using stereo depth perception. The theory relies heavily on the mathematics of projective geometry and linear algebra. It provides results that are robust to noise and occlusion. However, the approach is limited to searching for small numbers of rigid objects. Moreover, this work does not address the problems of model acquisition.

Traditional approaches to recognition have focused on settings where only a few object models are available for matching, and these approaches do not scale well to recognizing many objects in unstructured environments. In contrast, Edelman's "Chorus of Prototypes" [33] can efficiently recognize objects from 100-1000 models using a limited number of comparisons by comparing an object's silhouette to those of a few prototypes. An alternative approach is to use a grammar to specify possible configurations of primitives. This has been explored with geons [8], and it has been tested with both neural net implementations and human trials.

Recent work has shown some exciting results for recognition without segmentation or known structural models. These algorithms look for constellations of visual features [37] or recurring patterns of low level features [93]. This is a reasonably challenging learning scenario, but it appears some human adults are capable of learning object prototypes without segmentation as well [13]. This is an even more interesting because only half of the adult subjects were capable of performing this task. However, it seems that much prior experience with object shapes is required to solve this task.

A major limitation of most artificial systems is their inability to learn quickly. One extension of the constellation work [66] generates model priors so that it can learn a new class from a single observation. Another extension has been to quickly learn from labeled feature sets instead of feature vectors [45].

Incorporating language capabilities into robots usually relies on object recognition. Duygulu and colleagues [32] approach object recognition as a machine translation task. For input, their algorithm takes a set of images annotated with a few words that describe the main objects in the image. They use the normalized cut algorithm to segment images into a set of regions, and they use the words to define potential labels for each region. For each region, 33 features are measured. Each featurized region is then a point in a 33-dimensional space, and these points are grouped using k-means to form blobs that represent categories. Finally, EM is used to generate probabilistic matches between blobs and words. Evaluation consists of measuring precision-recall curves for retrieval on automatically annotated images. From a vocabulary of 371 words, models were learned for 80 words.

Work on word learning [115] produced a system that learns both visual feature representations of objects and phoneme sequences to represent words. In the learning scenario, a storybook with pictures is read aloud. The algorithm relies on the reader or listener to point to the appropriate object as each word is read. Their visual representation of objects as feature vectors is not sensitive to image segmentation errors, and objects are defined using agglomerative clustering.

Campbell and Krumm [17] investigate object recognition in an intelligent room. They first capture a few images of each desired object. The object is manually outlined, and the algorithm synthesizes more views based on the assumption that the image of the object is an orthographic projection. Scale is not considered to be a problem, since the chosen objects are observed on a desk. Each image is passed through a Canny edge detector, and features are generated by extracting 7×7 sub windows around every detected edge. These features are quantized and run through a dendogram to find prototype features for the object. From the training images, each prototype feature is associated to many possible locations for the object's centroid. A Hough kernel generates potential locations for the object in a test image by first classifying the edge features in the test image and then accumulating the votes from each prototype. The experiments demonstrate that this recognition process must be integrated with color pruning for reliability. This method achieves high recognition rates at 0.7 Hz for partially occluded objects. Although tracking is not used in this work, the authors suggest the use of a tracker could reduce the number of false positives.

2.2.3 Object Based Attention and Inference

Sun and Fisher [104] describe a visual attention system that incorporates both object and feature driven competition. Two major mechanisms are present: visual salience of objects and hierarchical selectivity. The system is evaluated on both natural and synthetic scenes. They address three major concerns: (1) top-down and bottom-up processing, (2) the dependence of attention on stimulus, and (3) the dependence of attention on objects and spatial proximity. This system does not define objects, but rather it demonstrates how attention can move between objects. The authors also assume that hierarchies are provided by an earlier process. They demonstrate the shift of attention in visual scenes, but there is no clear evaluation of the results.

Chella and colleagues [21] describe a logical representation for representing the dynamics of objects in scenes. The main thrust of this work is to create a "conceptual space" which acts as an intermediary between low level subsymbolic data and high level symbolic representations. Actions and processes can be represented in this space. However, they do not describe how these conceptual spaces can be created automatically or used for learning. Since they use superquadrics to represent objects at a low level, they demonstrate intelligent reasoning and action is possible with only an approximate shape representation.

2.2.4 Learning Actions

Finney et al. [38] simulated the task of a robot learning to manipulate its environment using a deictic representation. This work presents fairly negative results, but it highlights two facts. One is that reinforcement learning, like most existing artificial learning systems, has difficulties with representations that use objects. The second is that using a deictic representation (coupling the object with its egocentric position into a world state) does not immediately solve the problem, since attention becomes a major stumbling block.

A unified approach to handling uncertainty in actions is to use a reinforcement learning framework. In model-based reinforcement learning, Atkeson and Santamaria [3] constructed both a forward and an inverse model of the results of actions on the state space. This information is used to predict, plan, and update actions. This approach allowed a control task to be learned in approximately ten trials instead of thousands of trials. The input representation consists of a couple of relevant continuous state variables (coded using CMAC), and joint actions. All the of the feedback for the system comes through the reward function, which is provided to the learning agent.

2.2.5 Robotics

Several algorithms in robotics research find and use objects in the environment. Objects are required for control, communication and interaction. In robotics, perception often uses task specific heuristics that simplify object individuation.

Horswill [51] developed a robot that perceives colored blobs in the environment and can interact with them. Although the visual system is limited to dealing with objects of a single color, these objects can be used in many ways. The robot can crash into a desired block ("Crash into the red block"). An object can be specified using its relative position ("Turn towards the blue block on the red block"). Limited natural language queries about a scene can be answered. This robot can individuate objects, act on them, and ascribe properties to them. The robot can also specify objects using language, and use relational information between objects.

Object tracking has been explored by many researchers, particularly with known object models. Work on the NavLab project [111] demonstrates the ability to track and model objects with both structural models (cars) and behavioral models (pedestrians).

Biswas and colleagues [9] find quasi-static objects in occupancy grids. They assume the environment is stationary during observation, and objects in the environment can move between visits. The robot relocalizes in the environment at the start of each visit. By comparing the grids between visits, the robot can find non-touching objects. A low pass filter is applied to remove noise on the boundary of the object. The number of models present can be estimated using a Bayesian technique. Object's shapes are found using expectation maximization and are represented using occupancy grids. The algorithm as demonstrated learned four objects. Although learning more objects causes scaling problems in their formulation, the authors claim techniques are available for overcoming this issue.

Angelov and colleagues [1] have shown how 3D range scans of people can be automatically decomposed into articulated components. Starting with segmented raw data, the algorithm uses the local distances between points to approximate the surface of the object. Using multiple examples of the same object with different articulations, the algorithm is able to find joints in the object and optimize the required number of joints using the EM algorithm.

Steels and colleagues [101] investigated how robots can acquire a lexicon to describe objects in their world. The robots can point to objects, associate sounds with objects, create object properties which may also acquire a name, and engage in word games with other agents to induce a common lexicon. This work explores how discrete, shared, and stable concepts can arise as multiple agents experience continuous, noisy worlds. To attain real-time interactions on a computationally limited platform, their method does not separate the object from the background. The vision input is run through a histogram to generate a statistic summarizing the sense impression of perceiving the object. For recognition, the histograms are classified using the nearest neighbors algorithm.

Becker and colleagues [6] have a robot that can perceive, recognize and grasp common objects. A model of each object is learned from multiple views of the object rotating on a turntable. The most discriminating edge features are stored in chains that are used later for recognition. Each known object is assumed to possess strong edge features, and these features allow the object to be perceived on backgrounds without strong edge features. The recognition process also finds the object's pose in the environment. The robot uses the pose information with predefined grasp positions to determine how to grasp the object.

There is also work on learning the motion patterns of objects. Arbuckle and colleagues [2] describe how temporal occupancy grids can separate the environment into regions with similar temporal properties. This approach does not yield objects, but it can be used as a precursor to finding dynamic objects in occupancy grids. Jenkins and Matarić [54] present an algorithm that automatically characterizes the motion pattern of an articulated figure using a nonlinear dimensionality reduction technique. The input consists of joint angles that are captured using special markers on a human. This work demonstrates how motions primitives can be mined from articulated objects whose joint angles can be reliably tracked.

2.3 Summary

Despite extensive study, there is not an adequate understanding of how agents can learn to perceive objects or learn to interact with objects. There are several studies of perception in natural agents that indicate motion is incorporated at a low level. Other studies describe how object perception is used by high-level cognitive processes. Implementations of object perception in artificial systems yield several computational models for aspects of the problem. Existing research provides many useful tools and techniques for solving subproblems in creating an object ontology. However, since these studies use different object representations, they do not provide an integrated picture of how an agent can learn about objects.

Chapter 3

An Object Ontology

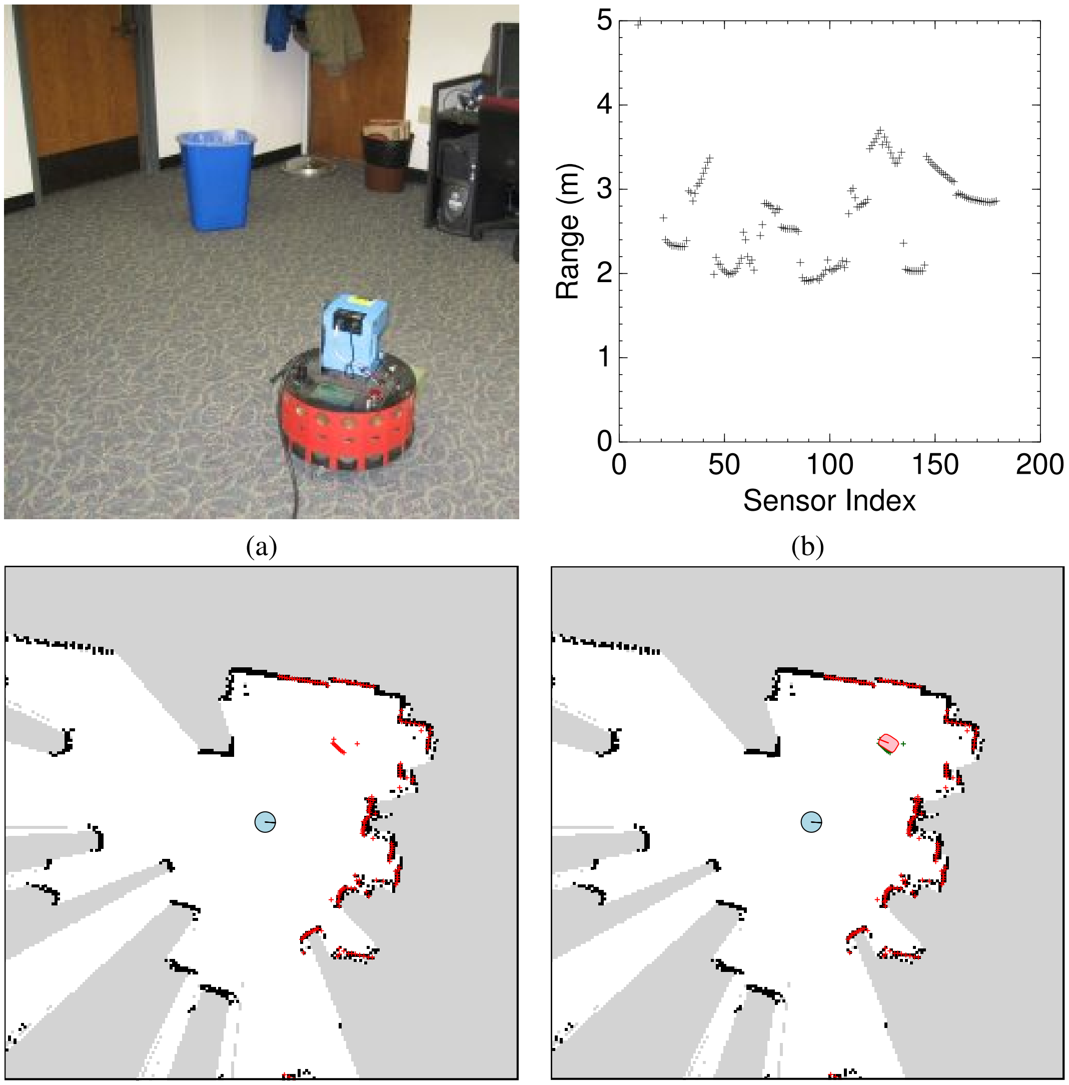
A robot's knowledge of objects is limited by its experience and by its background knowledge. This chapter introduces the components of an ontology for objects. The subsequent chapters describe how the elements in this ontology are created and learned from existing ontologies of self and space.

3.1 Platform

The experimental platform for this work is a Magellan Pro with a SICK laser rangefinder, shown in Figure 3.1(a). The robot has a non-holonomic base that is commanded with drive and turn velocities through a Player [41] interface. The laser rangefinder takes measurements of the distance to obstacles on a plane approximately 40 cm above the ground. The laser rangefinder returns one distance reading per degree, over a 180° field. Each observation is a vector of distance readings over the sensor indices Θ as shown in Figure 3.1(b),

$$z_t : \Theta \rightarrow \Re.$$

The robot also has a model of its local space that is maintained by a simultaneous localization and mapping (SLAM) algorithm [106]. Figure 3.1(c) shows a local map, m, in which the



(a) (b)

(c) (d)

Figure 3.1: (a) A scene with the learning robot observing a physical object. (b) The sensor measures distances to obstacles, with readings taken at every degree. (c) The robot creates an occupancy grid using a geometric projection of the observations from the range sensor. The occupancy grid provides a map of the static obstacles around the robot. The robot stays localized in the map by continuously updating its estimated position and heading. (d) At the end of the object learning process, the robot has a structured description of its local environment consisting of a static map, the learning robot, and a recognized object.

robot is able to stay localized. The robot has actions that allow it to move to accessible goal poses within the map.

The robot's task is to create an object ontology that enables object perception, reasoning, and interaction. Figure 3.1(d) shows the robot's description of the environment after the object ontology has been learned. The robot can then represent an object in the environment, recognize the object, and estimate an object's pose from a single observation. The robot can also interact with the object using learned actions.

3.2 Conceptual Assumptions

The learning process starts from the assumption that the robot has already acquired ontologies of self and space. Although the experimental platform uses manually provided interfaces to the robot's ontology of self and space, it is reasonable to assume that the robot can autonomously develop similar interfaces.

Research by Pierce and Kuipers [85] describes how the robot can autonomously develop an ontology of self. That work showed how an unordered collection of sensors can be spatially organized, providing a meaningful sensory index to each distance reading. That work also showed how a drive and turn motor interface can be learned for robots with a different motor interface (such as velocities for the left wheel and right wheel).

The ontology of local space has been extensively studied for robot mapping but there is not a complete theory for how the robot autonomously develop this ontology. There are two critical components, a local map and actions to move the robot to a goal. For large scale space, multiple studies [85, 87] have shown how the robot can learn control laws for navigation between places. The construction of a local map in SLAM algorithms relies on the knowledge of the transformation between the range data and Cartesian space. A similar (though not identical) transformation between sensor data and Cartesian space is provided by the Action Respecting Embedding algorithm [12]. Given this transformation, a local map can be maintained using scan matching to estimate the robot's pose. Actions are also required to move a robot to a given pose. The ontology of self provides drive and turn primitives, for which the robot can form local motion models. Local motions can chained to go to a desired pose in the absence of obstacles. When obstacles are present, motion planning algorithms can use local motion models to find a collision free path to a goal using A^* on a discretized state space or the Rapidly-exploring Random Tree (RRT) [63] algorithm on a continuous state space. In short, although a complete local space ontology has not been learned autonomously on a mobile robot, it is plausible that a local space ontology can be learned.

3.3 Prerequisites

At a low level of description, the robot's sensors produce a stream of observations as input

$$\ldots, z_{t-1}, z_t, z_{t+1} \cdots \tag{3.1}$$

and the robot generates a sequence of motor commands as output

$$\ldots, u_{t-1}, u_t, u_{t+1} \cdots \tag{3.2}$$

which move the robot through space. Note that the sensor data and motor commands do not correspond to exactly the same time t. There are typically some latencies in the system in the acquisition of sensory data z_t, in the computation of an appropriate motor command u_t, and in the time taken for the motors to achieve a steady state output.

The structure of these inputs and outputs on a robot are at the "pixel level". The input might be an array of range sensor readings or an array of pixels in an image. The output might be set-points for voltages or velocities. Programming the robot at this level of description makes it difficult to progress beyond simple reactive behaviors.

Several abstractions are used to provide more semantically meaningful interfaces to the external world. These are *perceptual filters* (which are functions of the observation stream), *classes* (each class is a set of values from a perceptual filter), and *actions* (which abstract the motor output). The perceptual filters, classes and actions form components of an *ontology* for representing the robot's knowledge of self and space.

In the robot's existing ontologies of self and space, the robot senses the rangefinder data and wheel odometry. The simultaneous localization and mapping algorithm provides perceptual filters that simultaneously generate the local map and estimate the robot's position and heading in the map. The robot can also abstract the local map into a view [61], and each view forms a class. The robot has actions to move to a desired pose in the local map.

3.4 Formalism

The object ontology is a tuple,

$$\mathcal{O} \equiv \langle\, Trackers, Filters, Classes, Actions \,\rangle \tag{3.3}$$

consisting of trackers ($Trackers$), perceptual filters ($Filters$), perceptual classes ($Classes$), and actions ($Actions$).

An *object*, considered as part of the agent's knowledge representation, is a hypothesized entity that accounts for a spatio-temporally coherent cluster of sensory experience. Note that the word "object", when used in this sense, does not refer to a thing in the external world, but to something within the agent's knowledge representation that helps it make sense of its experiences.

A *tracker* $\tau \in Trackers$ names two corresponding things. One is the active process that forms clusters from sensory experience gathered over multiple time steps. The second is the symbol in the agent's knowledge representation that represents the object (i.e., the hypothesized entity that accounts for the tracked cluster).

A *perceptual filter* $f \in Filters$ is used to estimate some variable of interest based on the robot's observation stream. A filter f is parameterized by the tracker τ to measure an *object property* $f(\tau)$ such as the object's location in the map. The *percept* $f_t(\tau)$ represents a property of τ

at time t. The percept is formed from the sensory experience of the tracker τ. Examples of simple percepts include the distance, location, and color of a particular object at a particular time. A more complex percept is the shape of an object, which can be assembled from multiple observations over time.

For a particular perceptual filter f, a *class* $\sigma_{f,q'} \in Classes$ is the set of percepts similar to a prototype percept $q' \equiv f_{t'}(\tau')$,

$$\sigma_{f,q'} = \{q \mid d(q, q') < \delta\}, \tag{3.4}$$

where d is an appropriate distance function and δ is a threshold. This is an unsupervised learning approach, but the choice of the distance function d will strongly influence the utility of the learned classes. In this thesis, the distance functions are manually specified along with the perceptual filters. For a robot equipped with vision, the robot could create prototype color classes and declare (color(τ)= 'red'). Using range sensors, the robot can create a structural model of an object's shape, allowing the robot to create a class based on prototype shapes and declare (shape(τ) = 'recycle-bin'). Figure 3.2 shows ten shape models, which are percepts obtained from the robot's sensory experience with the ten depicted objects. These individual percepts belong to ten classes, each corresponding to percepts obtained from the same real-world object on different occasions.

An *action* $\alpha \in Actions$ is specified by a description D of its effects on the object's percepts, the perceptual context C in which the action is reliable, and an associated control law H.

$$\alpha = \langle D, C, H \rangle \tag{3.5}$$

Each action is described by a qualitative change that it induces in one percept, for example causing the distance to an object to decrease. The context is a set of constraints on the percepts required for the control law to reliably have the desired effect, for example requiring that the object lies in front of the robot. A control law is a function from percepts to a motor output. The robot can use actions to achieve a goal by executing actions that reduce the difference between its current state

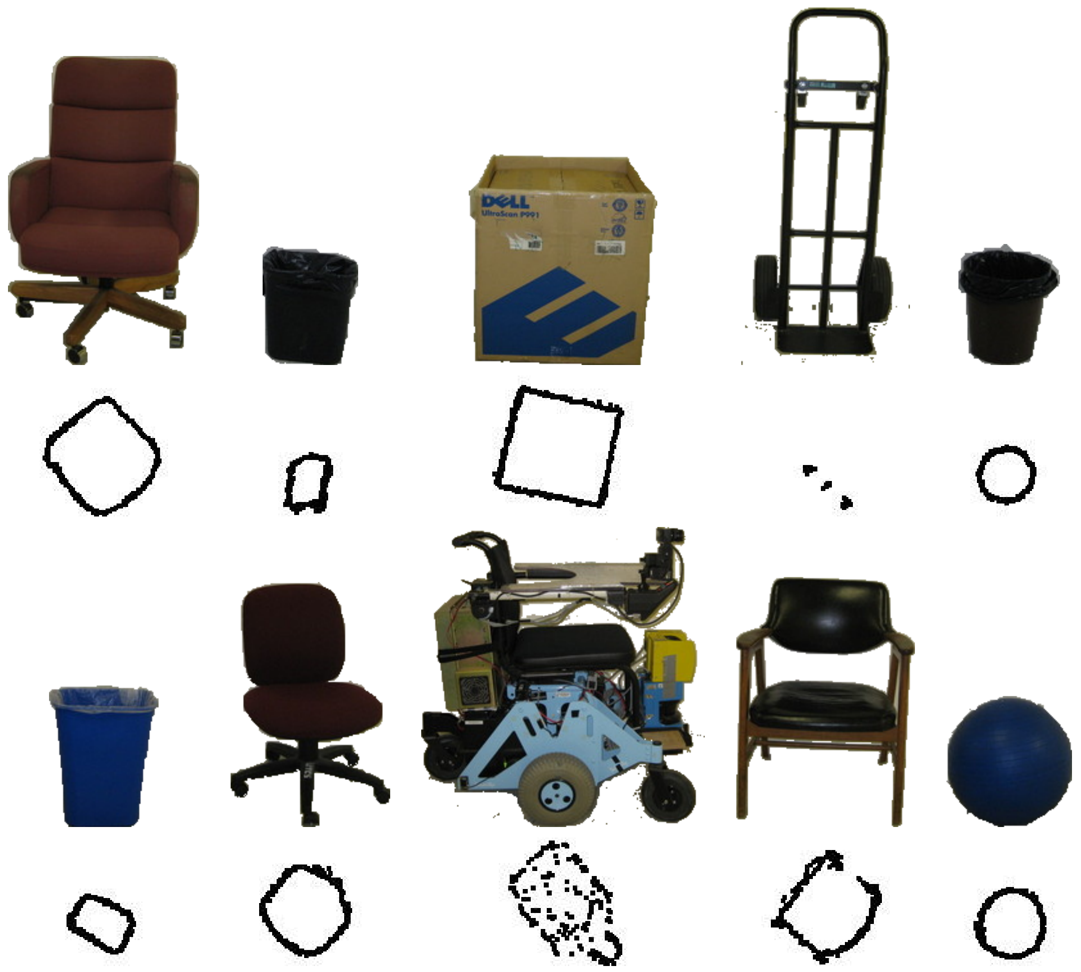


Figure 3.2: A set of physical objects with their learned shapes.

and the goal state. Actions are described in greater detail in Chapter 6.

The learned object ontology changes the robot's description of the world to contain objects as shown in Figure 3.1(d). The robot knows its self, with a shape and pose in the map. The robot represents the static world as obstacles and clear space. The robot represents objects that it can track. The robot can interact with the objects and it can recognize some objects that it has previously observed.

The above ontology differs from purely symbolic ontologies as used in symbolic planning examples. The above ontology describes the sensed environment in terms of tracked objects (τ), continuous properties of these objects as measured by perceptual filters (such as the distance to the object $\mathsf{distance}(\tau)$), perceptual classes used for recognition (such as $\mathsf{shape}(\tau)$='recycle-bin'), and actions that can be applied to an object (such as decreasing $\mathsf{distance}(\tau)$). In a STRIPS representation of the blocks world [89], objects are defined by atoms (A,B) and the objects have corresponding perceptual classes (BLOCK(A)). However, additional geometric knowledge is required to support spatial binary relations (ON(A,B)) and fully symbolic parameterized high-level actions (MOVE(B,X,Y)). In future work, the ontology of objects in continuous spaces developed in this thesis might be used to support the development of a completely symbolic object ontology.

The subsequent chapters describe the Object Perception and Action Learner (OPAL), which is a set of algorithms that creates the object ontology from the robot's experience. The trackers, classes and actions are learned, meaning that they are induced from the data. The perceptual filters are specified manually as functions of the tracker data stream. Although the perceptual filters have not been learned, they are defined using the representations available to the robot.

Chapter 4

Bootstrap Learning for Object Discovery

A significant amount of effort in robotics is focused on achieving competent behavior for a small predetermined set of tasks. While this leads to reliable behaviors, the emphasis on performance and accuracy for engineered tasks leads to robots that lack any ability to autonomously learn in the unstructured world. This chapter examines how a robot can discover unknown objects, namely how a robot can perceive, track, model and recognize novel dynamic objects in an unstructured environment.

This work is inspired by the developmental learning process, where a learning agent must autonomously construct its own internal vocabulary to describe and interact with the world. To achieve this goal, robots can not use algorithms that provide competence only for a limited set of objects and viewing circumstances. Instead, robots should begin with an inclusive definition of objects so that they achieve some level of interaction with a broad range of objects. Other work has shown how a robot can develop more sophisticated skills using statistics gathered from moving objects [88]. This work demonstrates how a basic level of competency can be acquired.

For a robot to learn about an unknown world, the robot must identify the objects in its environment, learn object properties, and learn how to classify objects. The robot's sensorimotor system provides a “pixel-level” ontology of time-varying sensor inputs and motor outputs. A substantial learning process [85] can provide the organization on the sensors along with the ability to follow control laws and define distinctive states to describe the large-scale structure of the environment. However, at the end of this learning process the robot’s ontology still does not include *objects*. Starting from lower-level ontologies that provide range sensors, incremental motion, and an occupancy grid model of the local environment, this chapter shows how an ontology of objects can be learned without external supervision.

4.1 Principles

This thesis on developmental learning of an object ontology has several motivating principles.

1. **A developing perception system should work in unstructured environments.** Animals live in unstructured environments. Perceptual systems should begin learning in unstructured environments with simple models and should generate progressively more sophisticated models. When the environmental structures become unreliable due to the presence of noise and background distractions, perception should degrade back to simpler models.

2. **Objects are cognitive structures used to explain observations of the world.** Attempts to create precise physical definitions of objects are elusive and usually unproductive. People are adept at decomposing a scene into sets of objects, but they are equally capable of changing their attention so as to perceive new objects from parts of objects or clusters of objects.

3. **A developing perception system should learn how to factor observations into representations of objects.** Factoring observations of the world into objects helps an agent generalize from past experience into the present. An agent should learn how to factor its observations since the agent’s understanding of objects evolves over time. The agent should learn representations of objects to improve object perception.

4. **Dynamic sensor readings that are spatially and temporally proximal may have a common cause.** Hypothesizing an object to explain motion is a reasonable mechanism for generating object models. There is physiological and psychological evidence that motion is perceived early in evolution and development. A mature agent's skill in static scene analysis may be bootstrapped from the knowledge attained through the observation of mobile objects.

5. **Perception can function over many computational scales.** Organisms have a wide range of sensors and computational capabilities. It is likely that perceptual capabilities of modern robots can be improved without waiting for faster processors or better sensors. Perceptual algorithms should be designed to accept additional sensors and computational resources without substantial alteration.

4.2 Learning about Objects

These principles lead to the Object Perception and Action Learner (OPAL) that can autonomously learn an ontology of objects to explain aspects of its sensor input from an unknown dynamic world. The proposed unsupervised learning method uses the properties of allocentric occupancy grids to classify individual sensor readings as static or dynamic. Dynamic readings are clustered and the clusters are tracked over time to identify objects, separating them both from the background of the environment and from the noise of unexplainable sensor readings. Once trackable clusters of sensor readings (i.e., objects) have been identified, the robot generates shape models where they are stable and consistent properties of these objects. However, the representation can tolerate, represent, and track amorphous objects as well as those that have well-defined shape. The learned ontology makes it possible for the robot to describe a cluttered dynamic world with symbolic object descriptions along with a static environment model, both models grounded in sensory experience, and learned without external supervision.

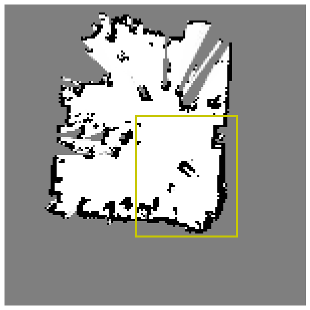
The occupancy grid representation for local space does not include the concept of "object." It assumes that the robot's environment is static, that it can be divided into locations that are empty and those that are occupied, and that the set of occupied locations has an arbitrary shape that can be detected by range sensors. A cell of an occupancy grid holds the probability that the corresponding region of the environment is occupied. Simultaneous localization and mapping (SLAM) algorithms can efficiently construct an occupancy grid map and maintain accurate localization of a mobile robot within it using range sensor data [76, 108, 34].

This work shows how a robot can acquire a working knowledge of objects from unsupervised sensorimotor experience by representing movable objects in four steps: Individuation, Tracking, Image Description, and Classification. This learning process is demonstrated using a mobile robot equipped with a laser range sensor, experiencing an indoor environment with significant amounts of dynamic change. For the work presented in this chapter, the robot was manually controlled by a joystick, and maneuvered around a room with multiple objects. As such, the robot's sensations of the world were controlled externally, but the robot is completely autonomous in building its perceptual models of the environment.

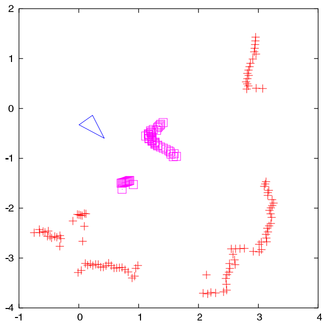
The learning process is a kind of "bootstrap learning" [60] since the algorithm combines multiple learning methods, where each step in the process is learning the prerequisites for subsequent steps. For example, the representations used by tracking are used to aid in object description.

A major motivation for this work is to understand how complex cognitive structures can autonomously develop in a learning agent. Tremendous leaps in cognitive complexity occur through evolution and during infant development, using experience acquired in unconstrained environments.

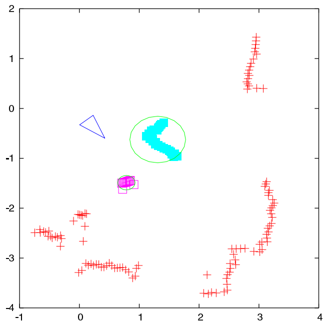
Learning in a high dimensional representation space (such as an observation stream) should be substantially more difficult than learning in a low dimensional (symbolic) representation. The premise of bootstrap learning is that an agent can apply a variety of high bias, but unsupervised learning algorithms to simple tasks (recognizing movable objects) to transform a high dimensional representation (an observation stream) into one with significantly lower dimension (a symbolic representation).



(a)



(b)



(c)

Figure 4.1: Object Individuation. (a) The occupancy grid representation of the environment generated online by a SLAM algorithm up to the current time t. The boxed region is shown in the following plots. (b) Sensor readings at time t classified as static ($+$) or dynamic ($\Box$) according to the occupancy grid cells they fall on. The robot ($\triangleright$) is in the upper-left portion of the plot, so nearby dynamic objects occlude parts of the static environment. (c) Dynamic readings are clustered and hence individuated into objects. Each of the two clusters is assigned to a tracker (circles). [*All of these figures are clearer in the color PDF than in grayscale prints.*]

4.2.1 Individuation

The occupancy grid representation embodies a static world assumption. Sense data that correspond to dynamic change in the environment are treated as noise. Fortunately, occupancy grid algorithms are quite robust to failures of the static world assumption. If changes in the environment are slow relative to repeated observation (12 Hz for the laser range-finder), changes in occupancy are quickly washed out by new observations, restoring the occupancy grid to a reasonably accurate description of the current state of the environment.

The stability of the occupancy grid in the presence of dynamic changes allows the addition of a new attribute to the occupancy grid. A grid cell is labeled *transient* if it has ever been unoccupied (i.e., the probability of occupancy falls below a threshold), and *permanent* if it has never been unoccupied.[1]

The low-resolution occupancy grid cell labeling is used to classify individual high-resolution range sensor readings. Each individual range sensor reading is labeled as *static* or

[1]To account for small localization errors, a transient cell may also require that all of its neighbors cells are unoccupied, which leaves permanent cells surrounded by a thin rim of unlabeled cells.

dynamic, depending on whether the sensor reading falls in a cell labeled as permanent or transient, respectively. Permanent grid cells and static sensor readings represent the static background environment, and the learning algorithm restricts its attention to the dynamic range sensor readings. Note that a non-moving object such as a trash bin is perceived with dynamic sensor readings if the robot *ever* observes the space in which the readings are located as unoccupied.

Next, the learning algorithm clusters the endpoints of the dynamic range sensor readings.[2] The coordinates of the endpoints x_i are represented in the fixed local frame of reference of the occupancy grid. Two endpoints are considered close if their distance is less than the threshold value δ_I:

$$close(x_i, x_j) \equiv \|x_i - x_j\| < \delta_I.$$

The individual clusters are the connected components of the $close$ relation: i.e., the equivalence classes of its transitive closure. Within a single observation frame at time t, these clusters $\{S_{i,t}\}$ are called *object snapshots*. They are the initial representation for individual objects. The process of individuation is shown in Figure 4.1. Each tracker T_k introduces a new tracker symbol τ into the robot's object ontology. Properties of the tracked object are interpreted as perceptual filters associated with the tracker. Examples of the perceptual filters include the snapshots, the center of mass of the snapshot and the radius of the snapshot.

4.2.2 Tracking

An object snapshot $S_{i,t}$ at time t has a spatial location and extent $\langle \mu_i, r_i \rangle$: its center of mass μ_i and the distance r_i from its center of mass to its farthest reading. The dissimilarity between two snapshots S_i and S_j is

$$d_S(S_i, S_j) = \|\mu_i - \mu_j\| + |r_i - r_j|.$$

This function is robust to random noise and incorporates both the observed center and radius since the snapshots of a moving, dynamic object (such as a person) will vary in both dimensions. Where

[2]Recall that the endpoints of range sensor readings, like the localization of the robot, are not limited to the resolution of the occupancy grid, but have real-valued coordinates, albeit with limited precision and accuracy.

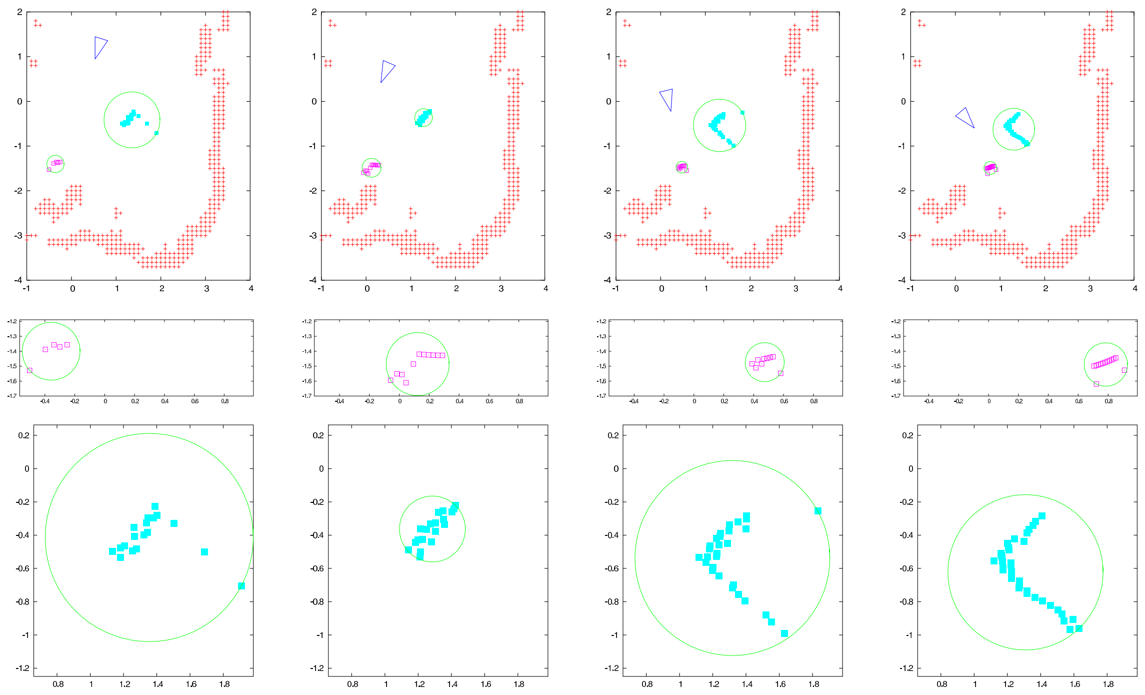


Figure 4.2: Object Tracking. The observations of an object's shape can vary greatly during tracking, regardless of whether the object has a rigid body. This figure shows a sequence of time steps prior to the scene in Figure 4.1. The actual trackers use data at much finer temporal granularity than the time-points (columns) shown. Note that the robot is moving while tracking. **Top**: The tracked dynamic objects, superimposed for reference on a low-intensity display of the permanent cells in the occupancy grid. **Middle**: A tracked pedestrian object, showing its irregular shape over time. **Bottom**: Tracked snapshots of a non-moving object (an ATRV-Jr).

the successor to time t is t', define the object snapshot S_t to have a *unique clear successor* $S'_{t'}$ if

$$d_S(S_t, S'_{t'}) < \delta_T$$

$$\forall S''_{t'} \neq S'_{t'} \quad d_S(S_t, S''_{t'}) > d_S(S_t, S'_{t'}) + \delta_R, \text{ and}$$

$$\forall S''_{t} \neq S_{t} \quad d_S(S''_t, S'_{t'}) > d_S(S_t, S'_{t'}) + \delta_R.$$

An *object tracker* is a function $T_k(t)$ whose value is an object snapshot $S_{i,t}$ at time t, such that for successive time-points t and t', $T_k(t')$ is the unique clear successor of $T_k(t)$. An object tracker T_k thus defines a collection of corresponding object snapshots extending from frame to frame in the observation stream, with at most one snapshot in each frame. The process of object tracking is depicted in Figure 4.2.

The domain of a particular object tracker ends at the time-points where the *unique clear successor* relation cannot be extended. “Object permanence”, the ability of an object tracker to tolerate breaks in the sequence of frames, is arguably a learned ability in young children [99]. The current implementation includes the ability to tolerate two missing frames in a sequence. Three missing frames terminates a tracker. New trackers are generated for large unexplained snapshots. Small snapshots without trackers are treated as noise and ignored.

Dynamic objects being tracked will converge and diverge, for example pedestrians in a crowded hallway. Object trackers will successfully track individuals over segments of their behavior, losing them when they get too close together and their readings are merged into a single snapshot. When they separate again, new trackers will be created to track the different individuals. More sophisticated methods for “object permanence” will be required to infer the identity of object trackers across such merges and splits. Following the bootstrap learning approach, the robot learns properties of objects from portions of experience when tracking is unambiguous and learning is easy.

Define these trackable clusters of dynamic sensor readings to be objects. Each tracker represents a distinct symbolic identity which is assumed to be the cause of the readings associated with it. At this stage, objects have only two properties: spatial location and temporal extent. These properties are sufficient for the trackers to guide the robot’s actions to acquire additional information about the object. For example, control laws for following, circling and avoidance are easily specified using trackers to specify the desired goals. The next step will be to acquire properties of the object instances that are stable across changes in space and time. This makes it possible to associate trackers with object classes.

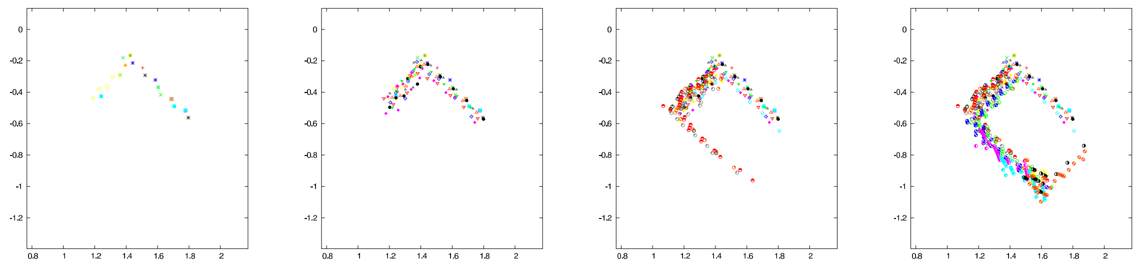


Figure 4.3: Perceptual Shape Model. This shows the incremental perceptual shape model creation for the ATRV-Jr observed in Figure 4.2. The range sensor endpoints in each snapshot are shown with different symbols. Selected snapshots combine to form a shape model.

4.2.3 Image Description

The *object snapshot* is defined to be the set of sensor readings associated with an object at a particular time. The perceptual *shape model* $M_k(t)$ for an object is a subset of the object snapshots collected over the time that the object is tracked.

$$M_k(t) \subset \{T_k(t') \,|t' \leq t\}. \tag{4.1}$$

The problem is how (and whether) the snapshots can be aggregated into a consistent, object-centered frame of reference. It is important for the robot to describe both objects with stable shapes that can be learned, and objects that are *amorphous* in the sense that they can be individuated and tracked, but their shape is beyond the capacity of the agent to describe and predict. For the robot learning agent, at its current level of sophistication, *pedestrians* are good examples of amorphous objects. In future work, the learning agent may be able to model a pedestrian as two alternately-moving legs (observed as 2D blob shapes), but for now, object snapshots of pedestrians change too much to form stable shape models.

Consider a temporarily non-moving object such as an ATRV-Jr (a mobile robot). To be individuated and tracked as an object, it must be located at a position that was unoccupied at some time, so its sensor readings are considered dynamic. Since the object does not move in the environment, tracking is quite simple. However, as the robot moves around the object, the snapshots change slowly (Figure 4.2).

The agent creates a shape model by accumulating distinctive snapshots while the object appears to be non-moving (Figure 4.3). Both tasks, detecting the lack of object motion and determining distinctiveness, are accomplished by a non-symmetric dissimilarity function d_D that compares sets of points (snapshots).

$$d_D(S_{new}, S_{old}) = \frac{1}{|S_{new}|} \sum_{s \in S_{new}} \min(1, \frac{1}{\epsilon} \min_{t \in S_{old}} \|s - t\|)$$

When successive snapshots differ by a large amount, δ_M, the agent assumes the object has moved, and discards the current shape model. Otherwise, if the current snapshot is sufficiently distinct (δ_N) from the points currently in the shape model, the new snapshot is added to the shape model. Finally, snapshots in the shape model are discarded if they are incompatible with the full set of current sensor readings.

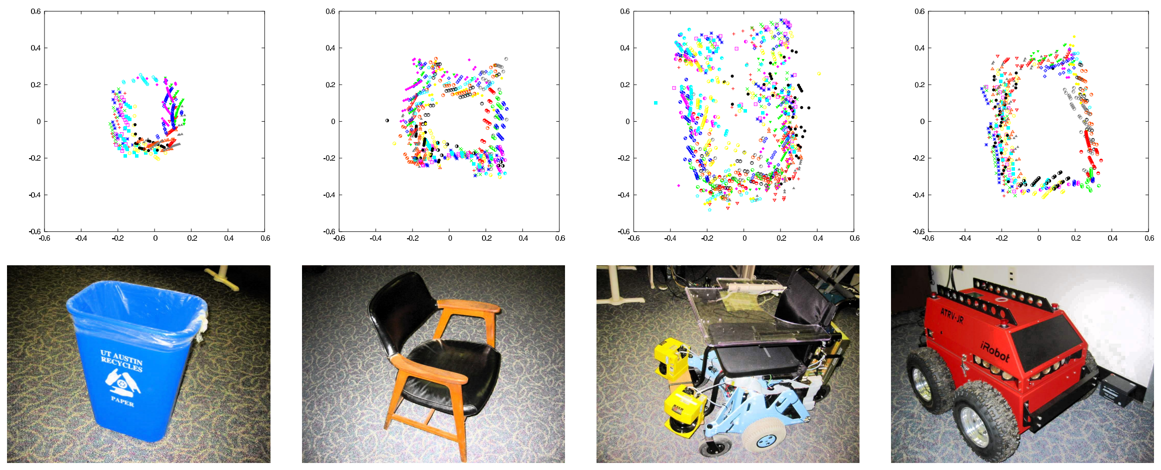
The shape model also records the directions from which the snapshots have been observed, and is considered *complete* when the full $360°$ surround has been sufficiently densely sampled.[3]

While the shape model is incomplete, it is considered "amorphous". When the shape model is complete, the agent creates a *standard shape image* for the object by placing the snapshots of the shape model into a canonical frame of reference. The snapshots are first rotated so that the primary axis of the sensor readings is aligned with the y-axis. This is accomplished by rotating the shape model to minimize the entropy of the projection onto the x-axis. Next, the shape model is translated to minimize the distance of the farthest points from the origin. (See Figure 4.4.)

4.2.4 Classification

Once an individual object has a standard shape image, the agent must classify it. Note that the learning agent is responsible for building its own classes. Moreover, since the object observations come in incrementally, the agent must add new classes incrementally. The task of adding

[3]In the current implementation, this means at least one snapshot exists in each of six $60°$ pose buckets around the object.



(a) (b) (c) (d)

Figure 4.4: Object Classification. Standard shape images and photographs for four learned object shape classes: (a) recycling bin, (b) chair, (c) robot wheelchair, and (d) an ATRV-Jr robot.

new classes incrementally is known as online clustering, and several algorithms exist [31]. For simplicity however, this clustering task is solved with a distance function.

Define the asymmetric dissimilarity function between two aligned shape images V and W by comparing their component snapshots

$$d'(V,W) = \frac{1}{|V|}\sum_{v\in V}\min_{w\in W} d_D(v,w).$$

This is used to define the symmetric distance function [4]

$$d_C(V,W) = \max(d'(V,W), d'(W,V)). \tag{4.2}$$

If the shape image of an instance is less than a threshold distance, δ_C, from multiple known prototypes, then its classification is uncertain. If there is only one known prototype within δ_C, then the shape image is placed in the prototype's class. If a shape image is more than δ_C from all prototypes, then a new shape class is formed. For example, when the shape model in Figure 4.3

[4]The "distance" function d_C is not a mathematical distance function since it does not ensure $d(x,y) = 0 \iff x = y$.

is converted into a standard shape image and compared to the known classes in Figure 4.4, the shape is recognized as an instance of the ATRV-Jr class. The prototype is then displayed, as in Figure 4.5(d).

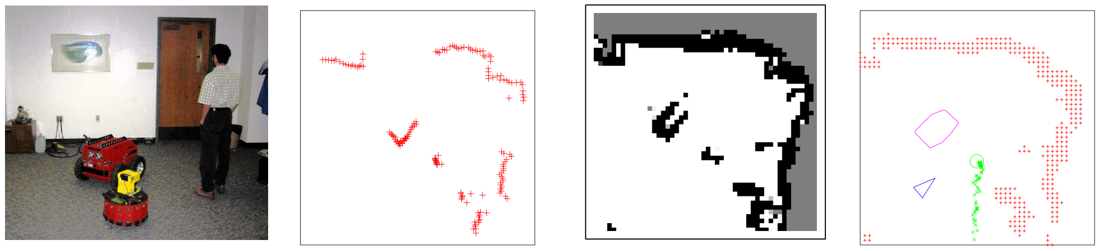
At this stage in OPAL, the robot cannot learn a shape model by observing a continuously moving object, but the robot can learn a shape model if the object stops for a short period. Once an object has been classified, the tracker retains this classification and the corresponding shape image even when perception is difficult. Furthermore, the robot can obtain individual snapshots of a moving object, and at a later stage these snapshots are useful for the classification of a moving object within an existing class hierarchy.

Even without a complete shape model, the robot can still generate a standard shape image for an object. For an incomplete image, the dissimilarity function is useful because it has the property that if $V \subset W$, then $d'(V, W) = 0$. This makes it suitable for comparing an incomplete model of an instance V with complete models that are already known. Also, this can be used to guide active perception by defining the observations that are most informative for classification.

4.3 Demonstrative Results

The above system was implemented on an iRobot Magellan Pro robot equipped with a SICK PLS laser rangefinder. The parameters mentioned in the paper had the following values: $\delta_I = 0.5m$, $\delta_T = 1.0m$, $\delta_R = 0.01m$, $\delta_M = 0.5$, $\delta_N = 0.1$, and $\delta_C = 0.33$. The system performance not very sensitive to the selected parameter values and similar results arise when the parameters are varied by twenty percent.

The implementation was tested by running the robot in the lab. An occupancy grid representation of the environment (shown in Figure 4.1(a)) is generated online in the presence of object motions. The process of individuation is displayed in the subsequent two images, first showing the classification of laser scans as static or dynamic, and then clustering the dynamic readings to form snapshots. The snapshots are associated with trackers in Figure 4.2, providing temporal extent to the object representation. The ATRV-Jr robot is not moving during this time, so a shape model is

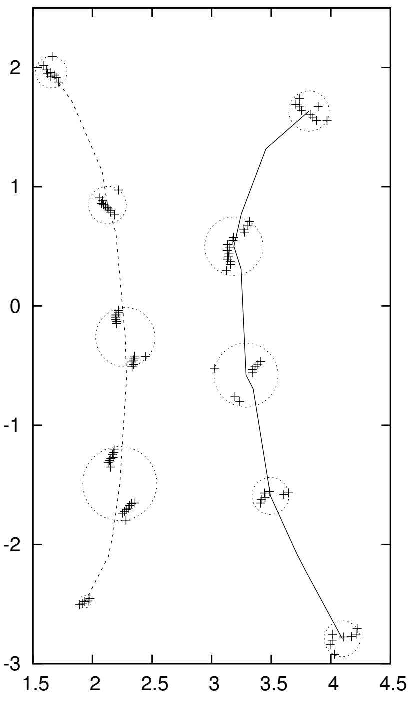


(a) (b) (c) (d)

Figure 4.5: Multiple representations of the scene in Figure 4.1. The robot observer is the small round robot in the foreground. The larger ATRV-Jr is used as a non-moving object. (a) A photograph of the scene. (b) A range scan representation of the scene. (c) An occupancy grid representation of the scene. (d) An iconic representation of the scene. This is a symbolic description of the robot's environment enabled by the learned object ontology. The location of the observing robot is indicated by a small triangle (▷). A moving object (pedestrian) of amorphous shape is shown with its trajectory. A non-moving object (ATRV-Jr) has been classified (as an instance of Figure 4.4(d)), and is shown by the convex hull of its shape model. The permanent cells in the occupancy grid are shown for reference, representing the static environment.

incrementally accumulated, as shown in Figure 4.3. When the description is sufficiently complete, the agent compares it to the objects learned earlier in the run, shown in Figure 4.4. The agent discovers that the shape model best matches that of the ATRV-Jr robot.

The agent's world description is graphically represented in Figure 4.5 along with a photo of the same scene. The result is a discretization of a natural environment into several entities which are useful for later reasoning: a coarsely represented fixed environment (walls+furniture), a localized agent (the Magellan Pro robot), an amorphous moving object (a pedestrian), and a classified known object (the ATRV-Jr). This demonstrates that object perception (individuation, tracking, description, and classification) in cluttered environments is feasible even when using limited prior knowledge and simple representations. Moreover, since the agent can autonomously generate new classes online, its ability to succinctly and accurately describe nearby objects should improve with experience.



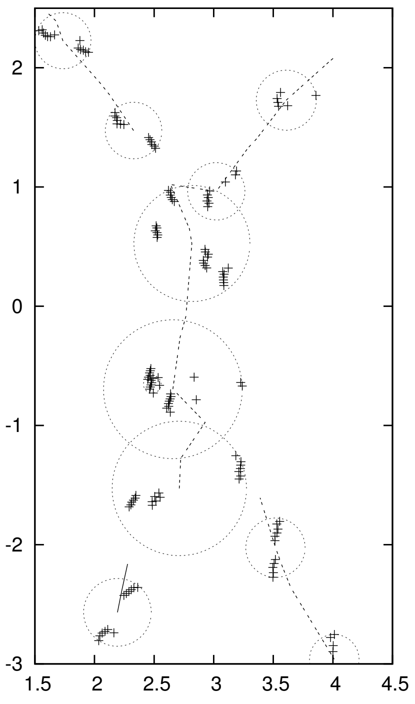


Figure 4.6: Two examples of tracking. In both examples, two people are walking, starting at the bottom of the figure and moving to the top. The plus signs show laser readings taken at one second intervals, shown in the coordinates of the local map. These readings are shown in the bounding circles of their associated snapshots. The lines indicate the path of the trackers. The units on the axes are in meters. (a) The people first approach each other and then move apart. Since the people do not come too close together, the tracking system generates exactly two trackers, one for each person. The two paths show the trajectory of the center of each tracker. (b) This example demonstrates two methods by which tracking can fail. One is the failure to track a single object in the presence of distractions. The second is the failure to notice when a tracked cluster separates into multiple distinct entities. Initially at the bottom of the figure, each person has a separate tracker. As they approach, the individuation process returns a single snapshot for both people. Since the snapshot at the merger has no clear successor (as each of the original trackers can explain the merged snapshot), the existing trackers are terminated and a new tracker is created. A similar inference occurs when the two people move apart. Object individuation creates two snapshots that must be explained. The tracking algorithm finds that one snapshot is the only clear successor for the merged tracker and a new tracker is formed for the second snapshot.

4.3.1 Tracking Limitations

Part of an agent's development are early failures in the perceptual system. One commonly discussed phenomenon in child development is the lack of object permanence, where the child fails to maintain a representation for an object when it is not perceived in the observation stream. Another type of failure is the failure to correctly track objects. The following example demonstrates both kinds of failure.

Figure 4.6 shows two examples of a scenario where two people are walking. The people start at the bottom of the figures and walk towards the top, first approaching one another and then separating. The people are observed using a SICK LMS-200 sensor at a height of 40cm above the ground plane. In the first example, the people remain sufficiently far apart so both individuation and tracking keep the two people separate. In the second example, as the people move close together, the individuation process fails to separate them. This failure in individuation causes a new tracker to be created. As the two people separate, the tracker for the pair follows one person, and a new tracker is created for the other person.

The second example demonstrates the lack of object permanence since the robot does not represent the persistence of individual objects through the merge and split of trackers. One approach to represent persistence is to create relations between trackers. Relations can be created when two trackers merge to create a new tracker, or when one tracker splits off from another tracker. An agent can reason over these relations to infer the potential location of an individual physical object that has been merged into a group.

The second example also demonstrates a limitation of the underlying tracking system. One way to improve tracking is to improve object individuation through the use of temporal information. In the current algorithm, the individuation process does not use information from prior time steps to create snapshots at the current time step. The robot could improve the individuation of a scene into object snapshots by incorporating information about the snapshots and the trackers from previous time steps.

4.4 Related Work

Other researchers have examined how object classes can be learned. Work on learning object classes in vision [66] demonstrates how a new class can be learned from a few examples. However, that work requires significant background knowledge of both generative models and priors on parameters. The background knowledge must in turn be acquired from a more intensive learning process. In contrast to their work, OPAL shows how complex background knowledge can be obtained autonomously by the robot. Related work has shown how the objects found through motion segmentation can provide background knowledge to bootstrap a static image segmentation [88].

There is a large body of literature on individuation in both psychology and computer vision. Work in developmental psychology [99] suggests that infants learn Gestalt principles of perception. Work in perceptual psychology [40] demonstrates that the natural statistics of the environment can provide sufficient training data for acquiring grouping mechanisms. The normalized cut algorithm [95] has been used for segmentation in vision in several feature dimensions.

Recent work on the Navlab project [111] has demonstrated the feasibility and value of tracking unknown objects in the environment. This work describes how a truck equipped with multiple range sensors is able to detect and track moving objects while driving down a road. The ability to track unknown moving objects is required for their goal of safe autonomous control at high speeds on urban streets. This approach is also able to recognize a few object classes. A significant difference from the work in this thesis is their inability to generate new object shape classes.

Other research has explored the construction of shape models of non-rigid objects [47]. Using a variant of the iterative closest point algorithm, the algorithm is able to merge dense three-dimensional range scans into a single coherent shape model even when the object undergoes small motions. This algorithm creates a qualitatively consistent model when a person moves their arms or head between successive scans. Because it relies on having significant amounts of data to align the scans, it is unclear that this method can be extended to handle non-rigid motion as observed by a two-dimensional range scanner.

Work by Biswas and colleagues [9] creates shape models from occupancy grids to generate new shape classes. They assume that the world is static during observation, which permits the use of a standard SLAM algorithm to capture the shape of the objects in a grid representation. The assumption that the entire environment stays static is fairly restrictive, since in many environments the objects of interest move regularly. Moreover, their algorithm uses an offline learning process. This makes the online incremental acquisition of new object shape classes difficult.

Our approach to tracking unknown objects provides the learning agent with symbols (trackers) which are grounded in the sensory experience (snapshots). Issues related to symbol grounding have also been explored in [26]. In this work they describe how anchored symbols can be used for planning and reasoning in a dynamic world.

There is extensive work on view based classification, particularly in the vision community. The "Chorus of Prototypes" [33] uses prototype silhouettes to efficiently index a space object views. Using distances between views provides an efficient mechanism for determining when a view is sufficiently distinct from previously known models.

Work on the VSAM project [24] demonstrated visual detection and tracking of objects using multiple cameras at fixed locations. Objects are detected and tracked using frame differencing and background subtraction. These objects were later classified using silhouette models of the object shapes. This work does not address the needs of mobile robotics very well, since their vision algorithms rely heavily on the cameras having fixed locations.

4.5 Summary

This chapter describes the first stage of OPAL: a method for an agent to autonomously learn properties of novel dynamic objects in a unstructured environment without complex prior knowledge. It introduces trackers for objects, creates perceptual filters for the trackers (radius, centers of mass, and shape models), and creates shape classes. This work demonstrates how a learning agent can efficiently learn representations of objects as part of a bootstrap learning process. Using this autonomously acquired vocabulary, a robot can classify some of the dynamic objects it encounters in the world. Classification is improved in the next chapter which provides an algorithm that creates structurally consistent object shape models.

Chapter 5

Learning Structural Shape Models

Mobile robots do not adequately represent the objects in their environment; this weakness hinders a robot's ability to utilize past experience. This chapter describes a simple and novel algorithm to create object shape models from range sensors. The algorithm enforces angular constraints between sensor scans of an object. These constraints are used to align the scans, creating a coherent object shape model. Consisting of scans and poses, this shape model is useful for both object recognition and localization. The structural reconstructions are accurate to within sensor precision. The structural model improves the object ontology with the introduction of more accurate object classes, and the addition of a new perceptual filter for object localization.

5.1 Introduction

The previous chapter showed that individual moving objects can be separated from the static environment. The occupancy grid representation for local space can distinguish between cells that are occupied either permanently or transiently. Sensor returns falling in transient cells are clustered in space, then tracked over time. The tracked entities acquire a shape model and can be classified. This is a form of bootstrap learning since the learning process does not rely on prior knowledge about objects.

This chapter shows how coherent models of the shapes of objects can be created. These models are used to reliably recognize objects and to localize the object in the robot's egocentric reference frame.

This chapter introduces an algorithm that defines angular constraints between multiple sensor scans of an object. These constraints are used to align the scans, creating a maximally coherent object shape model. The shape model, consisting of scans and poses, is useful for both object recognition and localization.

The problem of aligning scans occurs in different guises. Most commonly in robotics, scan-alignment is required for mapping and localization. These localization algorithms typically use some combination of scan-matching (matching points in scans to one another [46]), feature matching (Kalman filter approaches [65]), and grid based approaches (matching the points in a scan to an occupancy grid [107]). In contrast to these methods, structural shape model relies on the *angular constraints* of a scan that bound the space occupied by the object.

To clarify the distinction between using this approach and the point/feature/grid matching algorithms, consider the following thought experiment. Imagine using a range sensor to observe a leafy bush from several distinct poses in an otherwise featureless environment. Moreover, assume that no two scans observe data from the same part of the bush. This type of data from "porous" objects can cause significant problems for scan matching and feature based approaches. Probabilistic occupancy grid mapping approaches can create a fuzzy image given an appropriate likelihood function. However, the task should not be difficult to solve since the angle and the distance to the bush are tightly bounded. This is the underlying intuition behind the use of angular constraints.

5.2 Motivation for Structural Models

Finding objects in the environment is useful for interacting with the world. Using structural models of objects, it becomes simple to describe several of the interactions that a robot could learn in its environment, such as pushing a chair, hiding in a box, or putting garbage in the trash. Before learning these actions, the robot must learn to perceive and represent these objects.

Research into object representations falls broadly into three types of model: appearance-based, structural and functional. Appearance-based models represent the sensory impression of the object, storing sense impressions from particular viewpoints. Appearance-based models are particularly important in vision [33, 66]. Structural models represent the geometry of the object. A geometric shape model permits the robot to reason about space, and to find consistent configurations of objects, namely those where the objects are non-intersecting. Structural models can also support object recognition and object localization (estimating the object's pose in a robot centered reference frame). Finally, functional models capture the interactions of the object with a robot or human agent, thus yielding object affordances [102]. This chapter only develops a structural model of the object's shape.

Although work on robot mapping has generated several algorithms and data structures for representing the robot's surroundings, these techniques do not adequately address the needs of object perception. Range sensors might only receive a few readings from an object at a distance, and this data sparsity can cause SLAM techniques to fail. An object can be moved by external forces while being observed by the robot, so good motion models might not be available. Confronted with these problems, particle based localization methods can not adequately search the space of poses, and point based scan matching techniques lack sufficient data. Approaches based on point features are limited by the nearly non-existent texture in range sensor data. Faced with these weaknesses, a new scan-alignment technique is desirable for learning object shape models.

5.3 Using Angular Constraints for Scan-Alignment

Given sensor scans of an object as shown in Figure 5.1(a), the task of scan-alignment is to infer a pose for each scan to create a coherent shape model. This work makes the assumption that the robot has a range sensor that returns the distance to obstacles along different angles. Let Θ denote the angles at which the sensor takes measurements, so the distance readings of a scan $S = (r_S, \text{Supp}_S)$ are given by

$$r_S : \Theta \to \Re,$$

and the support of the object on the sensor indices is given by $Supp_S \subset \Theta$.

A *shape model* is defined as a set of observed scans (indexed in Λ) with their associated poses:

$$M = \{(S, p_S) | S \in \Lambda\}.$$

For a given pose, define $x_S(p_S)$ to be the Cartesian coordinates of the *scan endpoints* that fall on the object, namely

$$x_S(p_S) \equiv \{T(p_S, r_S(\theta), \theta) | \ \theta \in Supp_S\}$$

where the function $T(p, r, \theta)$ is a transformation of a point from polar coordinates (r, θ) with an origin at the pose p to Cartesian coordinates, namely

$$T((x, y, \theta_1), r, \theta_2) \equiv (x + r\cos(\theta_1 + \theta_2), y + r\sin(\theta_1 + \theta_2)).$$

From the sensor scan S, define the left and right limits of the object's extent (shown in Figure 5.1(b)).

$$LeftObject_S \equiv \max\{\theta \in Supp_S\}$$

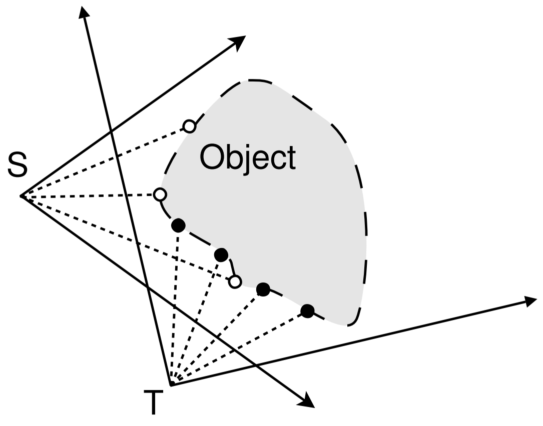
$$RightObject_S \equiv \min\{\theta \in Supp_S\}$$

$$LeftBound_S \equiv \text{successor}(LeftObject_S, \Theta)$$

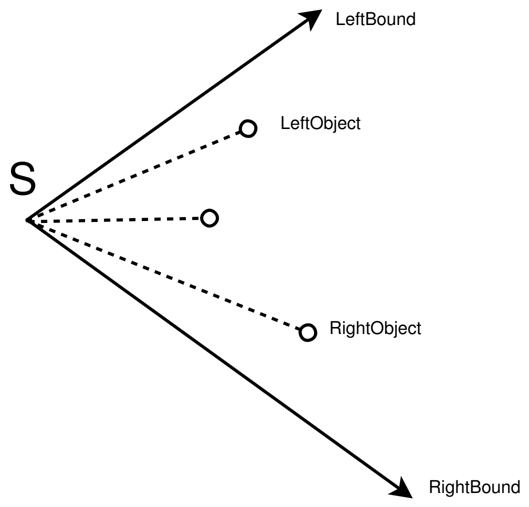
$$RightBound_S \equiv \text{predecessor}(RightObject_S, \Theta)$$

The algorithm ignores scans for which the angles $LeftBound_S$ or $RightBound_S$ do not exist. A ray is defined from the pose p at each of the above angles. These rays are denoted by $\overline{LO}_S(p), \overline{RO}_S(p), \overline{LB}_S(p)$, and $\overline{RB}_S(p)$. The scan endpoint at $LeftObject_S(p)$ is denoted $LO_S(p)$ and the endpoint along $RightObject_S(p)$ is $RO_S(p)$.

Given an object O, the bounding rays constrain the space occupied by the object. Using a pose p as the origin of a radial coordinate system, denote the angle from p to a point $q \in \Re^2$ by $\psi_p(q)$. The angular projection of each point on the object is constrained between $LeftBound_S$



(a)



(b)

Figure 5.1: (a) An object being observed from two different poses generates the sensor scans S and T. Each scan has readings from the object (points) and readings from the environment (rays). The task of scan-alignment is to infer poses for the scans when the object is unknown. (b) For each scan, define the extreme left and right angles that bound the object. Also define the extreme left and right angles at which the object is observed. The radial projection of the object on the observation pose can not extend beyond the bounding angles, and must extend to the object angles.

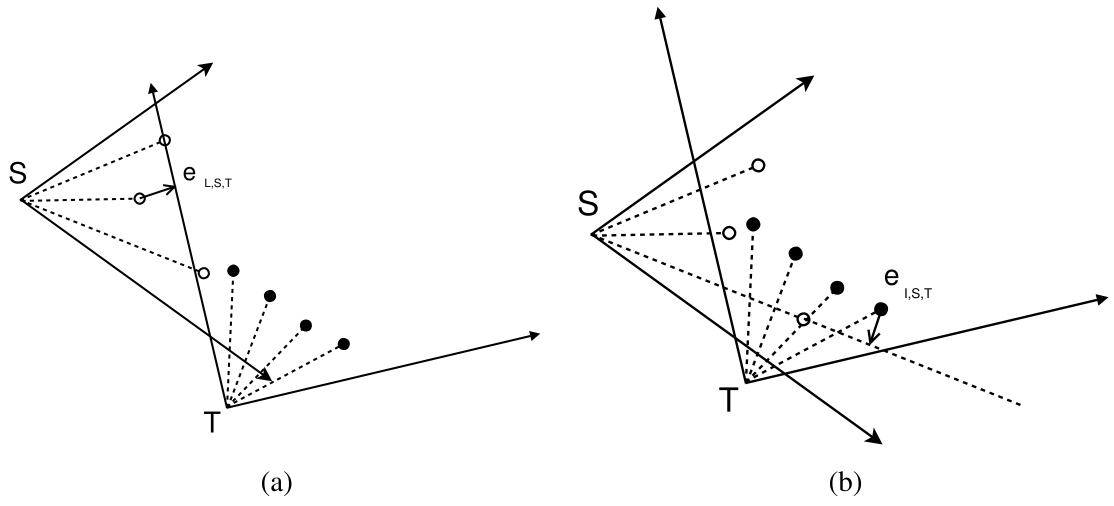


Figure 5.2: (a) Exterior constraints: Since all points on the object must lie within the constraints of T, the pose of S must change by (at least) the vector $e_{L,S,T}$ which is formally defined in Equation 5.2. (b) Interior constraints: For a convex object, at least one point from T should lie between the boundary and object rays of S. The right side of S is chosen here because p_T is on the non-object side of the right bounding angle. The interior error vector is denoted by $e_{I,S,T}$ and is defined in Equation 5.3.

and $RightBound_S$, i.e.

$$LeftBound_S > \psi_p(o) > RightBound_S,$$

for all points $o \in O$. Thus, the rays $\overline{LB}_S(p_S)$ and $\overline{RB}_S(p_S)$ *constrain* the space occupied by the object. Hence, given two scans S, T of the object taken from different poses, as shown in Figure 5.1(a), the bounding rays $\overline{LB}_T(p_T)$ and $\overline{RB}_T(p_T)$ constrain $x_S(p_S)$, the scan endpoints of S.

The above formulation is valid when there is no inaccuracy in the sensor readings. Real sensors suffer from multiple limitations including discretization, precision limits, and noise. Hence, the algorithm minimizes the violations of these constraints. First, define the set of vectors representing constraint violations of the left bound of T,

$$U \equiv \{vec(x, \overline{LB}_T(p_T))\ | x \in x_S(p_S), \psi_{p_T}(x) > LeftBound_T\} \cup \{(0,0)\} \tag{5.1}$$

where $vec(p, L)$ denotes the shortest length vector connecting the point p with the line L. Using this, define the left *exterior* error vector as the largest in U (see Figure 5.2(a))

$$e_{L,S,T}(p_S, p_T) \equiv \arg\max_{u \in U} ||u||. \tag{5.2}$$

The right exterior error vector ($e_{R,S,T}$) is defined correspondingly.

Even if all exterior error vectors are zero, the scans can be significantly misaligned. Figure 5.2(b) demonstrates how no point in x_T lies between the rays $\overline{LO}_S(p_S)$ and $\overline{LB}_S(p_S)$. Although this object shape is possible (as demonstrated in Figure 5.1(a)), it is improbable. Thus, the algorithm minimizes the distance of the point set x_T from the region between these two rays ($\overline{LO}_S(p_S)$ and $\overline{LB}_S(p_S)$). The exterior constraint vector handles points on the wrong side of $\overline{LB}_S(p_S)$ so only $\overline{LO}_S(p_S)$ remains to be considered.

This motivates defining an *interior* error vector as the minimum length translation required by p_T so that some point in $x_T(p_T)$ will fall on to the object support ray $\overline{LO}_S(p_S)$ (see Figure 5.2(b)). Assume that $\psi_{p_S}(p_T) < RightBound_S$. Then, using

$$V \equiv \{vec(q, \overline{RO}_S) \mid q \in x_T,\ \psi_{p_S}(q) > RightObject_S\,\}$$

and

$$W \equiv \{q \mid q \in x_T,\ \psi_{p_S}(q) \leq RightObject_S\},$$

define the interior error as

$$e_{I,S,T} \equiv \begin{cases} \arg\min_{v \in V} ||v|| & \text{if } W = \emptyset \\ (0,0) & \text{otherwise.} \end{cases} \tag{5.3}$$

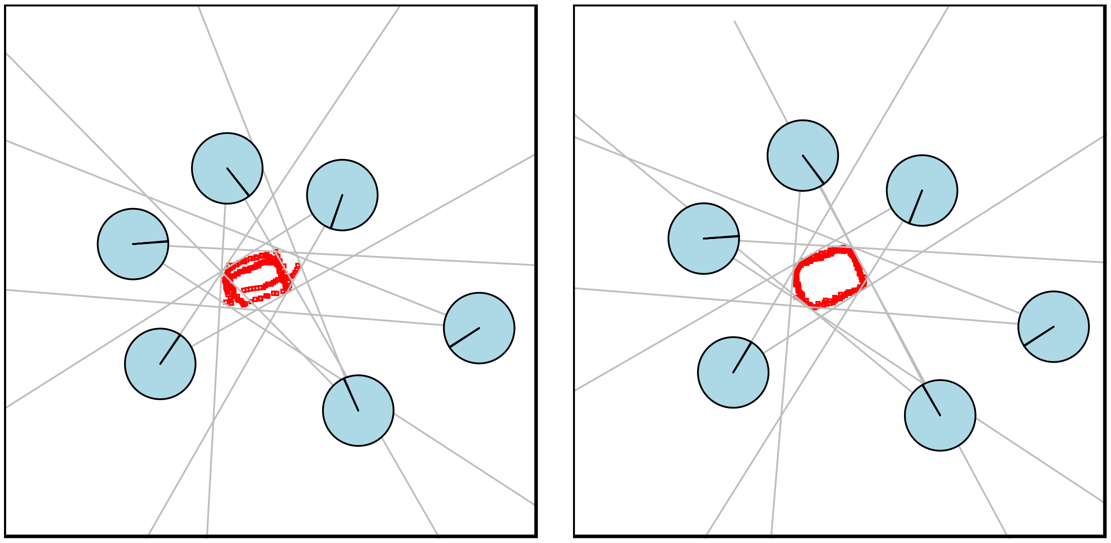


Figure 5.3: Scans from an object (a) before and (b) after scan-alignment. The two rays emanating from each robot pose indicate the left and right bounds. Scans are aligned by minimizing BV in Equation 5.4.

The above definition is used only when $\psi_{p_S}(p_T) < RightBound_S$. Otherwise, the interior error vector is defined similarly using the left boundaries.

Given all the error vectors $\{e_L, e_R, e_I\}$, scan-alignment becomes the task of finding poses such that all vectors are as close to zero as possible. This is accomplished by defining an objective function for boundary violations

$$BV(M) \equiv \sum_{S\in\Lambda}\sum_{T\in\Lambda} ||e_{L,S,T}(p_S,p_T)||^2 + ||e_{R,S,T}(p_S,p_T)||^2 + ||e_{I,S,T}(p_S,p_T)||^2, \tag{5.4}$$

and then using a numerical optimizer to minimize BV by perturbing the poses. The resulting figures are visually accurate (Figure 5.3).

5.4 Recognition and Localization

Object recognition and localization are pragmatically important for a robot. Object recognition allows a robot to define a context for its current situation, and thus generalize from past experience. Object localization allows the robot to reason about the configuration of the environment, which is necessary for planning and acting with objects in continuous worlds.

Fortunately, the tasks of object recognition and localization can be solved using the same error minimization technique used for scan-alignment. The tasks can be split up into three categories: (1) object recognition without model-alignment using properties of an object instance, (2) object recognition by finding the best shape model-alignment of the instance to the canonical model, and (3) localizing the pose of the class shape model from a single scan of the object instance (object localization).

Recognition by properties without model-alignment can significantly reduce the computational load of object recognition. Aligning shape models can be computationally expensive, particularly if model-alignment must be performed for each object shape class. Recognition is significantly simplified by computing coordinate-invariant properties of an object instance and using the properties to classify the object.

First, given a shape model M, define the *hull* as the convex hull of the object endpoints.

$$hull(M) = ConvexHull(\cup_{S \in \Lambda} x_S(p_S)).$$

The hull is efficiently computed in $O(n \log n)$ time [44] where n is the number of points. Since the hull is a polygon, computing the area, perimeter and center of mass are straightforward.

Properties of the hull, such as area and perimeter, can be used for object recognition without shape model-alignment. These properties can be used for learning names for objects (supervised learning) or for learning to form object shape classes by clustering (unsupervised learning). The utility of a property is inversely proportional to its variance. Ideally, the measurement varies less between object instances from the same class than between classes.

For some tasks, merely recognizing the object is insufficient. The robot might need to find the transformation between the class model and instance model to apply knowledge from the shape class, for example to identify the “front” of the object. Aligning the shape models also permits more discriminating recognition than is available through properties alone.

Model-alignment is performed by minimizing the BV error between the class model, M and instance model N, namely

$$f_C(M, N) \equiv \min_p BV(M \cup t(N, p)) \tag{5.5}$$

where $t(N, p)$ translates the pose of each scan in N by p. The error is minimized by hill-climbing from the robot’s initial poses. To speed convergence of the numerical optimizer, the initial value for p is selected so the center of mass of the hull of N coincides with that of M, so only the angle needs optimization.

The final task considered here is object localization. When an object is moving, the robot may know the approximate object pose based on prior observations, but may need to know the object’s pose precisely using a single scan. This is the object localization task, and it introduces new perceptual filters into the object ontology for the location ($\mathsf{location}(\tau)$) and the heading ($\mathsf{heading}(\tau)$) of the object τ. Localization is solved by minimizing f_C, where N contains the single scan and the algorithm assumes an initial estimate of the pose is available. Since N is incomplete, the optimization occurs over both the position and heading of the pose p. The accuracy of localization in heading is naturally limited for objects with rotational symmetries.

5.5 Evaluation

The robot autonomously collected observations of selected objects that can be easily confused for one another. The robot was a Magellan Pro equipped with a SICK LMS 200. The sensor returned the distance to obstacles with a one centimeter precision, over a 180 degree viewing angle, with one reading per degree. A Nelder-Mead simplex algorithm from the *scipy* library for Python was used for numerical optimization. The algorithm is initialized with the original poses provided by the robot's mapping and localization algorithm, and performs hill-climbing to find a local minimum for the objective function.

The robot collected data by circling around an object selected by the experimenter. Five objects were selected, and each object was observed five times. The following subsections describe how this data was used to evaluate model construction, recognition and localization.

5.5.1 Model Construction

The quality of the scan-alignment is good both visually and quantitatively. The scans for one object are shown in Figure 5.3, showing both the raw scan-alignment provided by the robot's map-localization, and the aligned scans. The aligned scan endpoints for all observed object instances are shown in Figure 5.4. A comparison of the BV error before and after scan-alignment is shown in graphs in Figure 5.5. Notably, the maximum external error is less than a centimeter, which is the sensor's precision.

An explanation is required for the effectiveness of a generic optimizer for scan-alignment. One caveat is that the starting poses for scan-alignment come from robot map-localization in a known environment. This yields accurate orientations, but can suffer from visible errors in translation. For fixed orientations, the objective function is a piecewise smooth quadratic function, a good property for most optimization techniques. Allowing the angle to vary implies BV is no longer quadratic in theta, but the objective function is still differentiable and the starting values for theta are near the true value.

5.5.2 Recognition

Property reliability for recognition without model-alignment was measured by computing the area and perimeter of all objects, then computing variation within and between classes. The results for property reliability are shown in Figure 5.6 and Table 5.1. Figure 5.6 graphically demonstrates that most of the objects are separable by either perimeter or area, although the garbage-can and recycle-

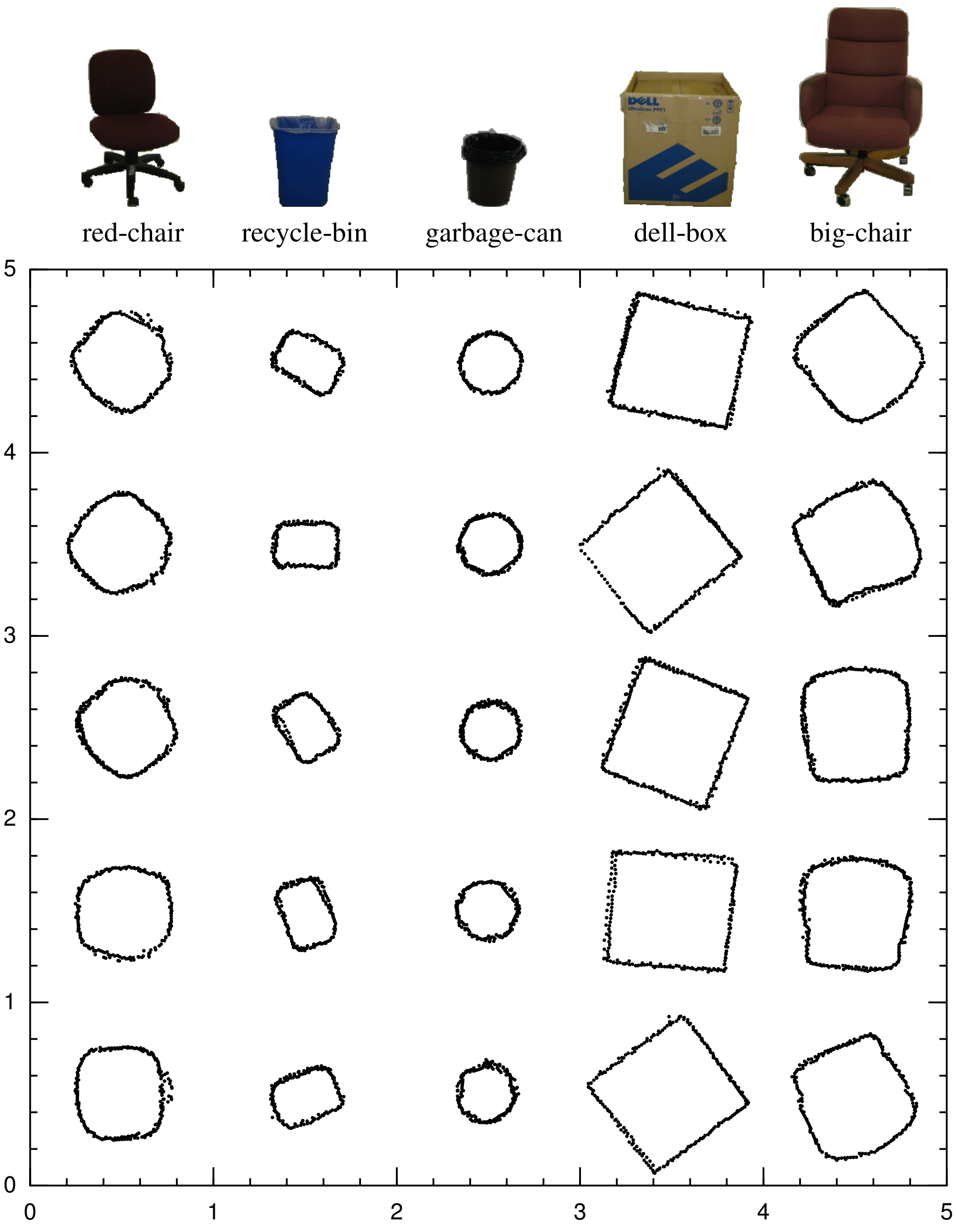
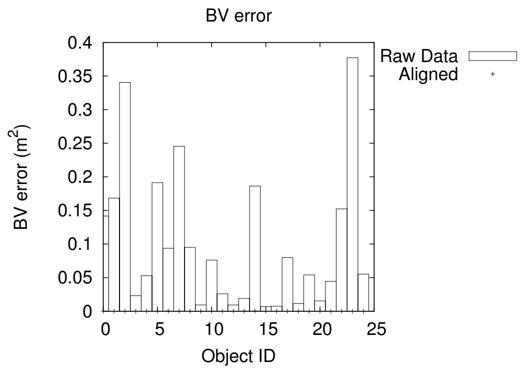


Figure 5.4: Pictures of the objects with labels are shown with aligned shape models generated from separate observations. The scale for the axes is in meters. Each object is observed by the range sensor from approximately 40 cm above the ground plane. Scans of an object are autonomously gathered by the robot. For each object observation, the endpoints from the aligned scans are shown below the object's image.



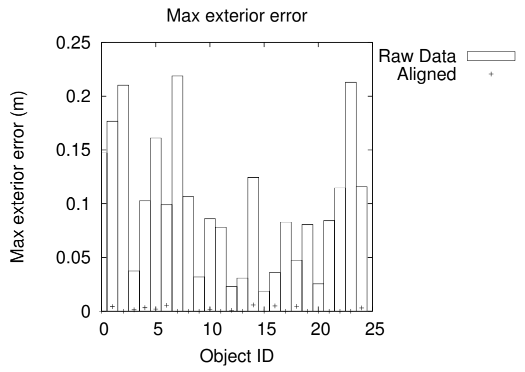


Figure 5.5: Modeling error measured by constraint violation. The graphs compare the error in the raw data with the errors after scan-alignment. The errors after scan-alignment are significantly smaller than the original data. The maximum length of an exterior error vector is less than a centimeter, indicating that the scans are aligned to within the sensor precision.

	Area	σ	Perimeter	σ
recycle-bin	932	39	113.7	2.3
big-chair	3288	42	210.4	1.3
garbage-can	889	27	106.6	1.5
red-chair	2237	44	169.9	1.6
dell-box	4257	80	252.3	1.7

Table 5.1: Property reliability for object recognition. The area (in cm^2) and perimeter (cm) along with their standard deviations. Note that the standard deviation in perimeter is small, making it a particularly useful property for recognition.

f_C error (min–max)	recycle-bin	big-chair	garbage-can	red-chair	dell-box
recycle-bin	0.001–0.004				
big-chair	1.412–2.005	0.001–0.005			
garbage-can	0.013–0.039	1.662–2.643	0.001–0.005		
red-chair	0.479–0.690	0.193–0.376	0.609–0.986	0.001–0.003	
dell-box	2.775–4.532	0.239–0.439	3.073–5.632	0.958–1.789	0.002–0.006

Table 5.2: Residual error after shape model-alignment. Models are aligned by minimizing f_C in Equation 5.5. The performance of model-alignment is measured by the range of f_C error (minimum–maximum). The f_C error can successfully discriminate between all tested objects.

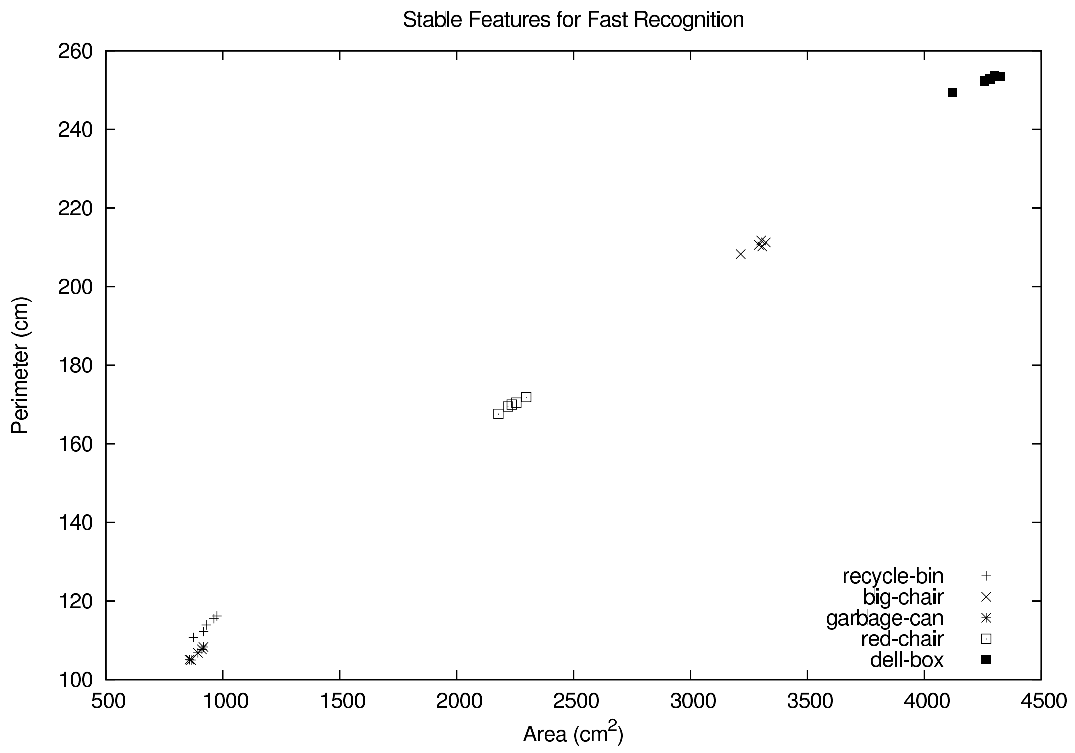


Figure 5.6: Perimeter and area are properties that can discriminate between many object instances without shape model-alignment. All classes are well separated except for recycle-bin and garbage-can. These classes can be separated after model-alignment (Table 5.2) but that is more computationally expensive.

bin are confusable. The table demonstrates that the variance in perimeter is largely independent of object size, but the variance in area is significantly larger.

Model-alignment for recognition was measured by comparing all observations of an instance of one object with all instances from a second object (dropping self comparisons). The minimum and maximum errors were measured and are shown in Table 5.2. From the table, perfect classification can be achieved using the f_C-error with a single threshold of 0.01. Notably, f_C-error does a better job than properties for discriminating between the garbage-can and the recycling bin. The use of f_C-error for recognition is limited, because f_C-error cannot distinguish between objects that have the same hull. If required, better distance metrics may be efficiently employed by minimizing f_C-error to initially align shape models.

5.5.3 Localization

Localization is evaluated in three steps. First, two instances of the same object are aligned using f_C error. Second, a single scan from the second model is perturbed by a fixed amount. Finally, the perturbed scan is localized against the first model and subtracting the resulting pose difference from the unperturbed scan provides a measure of localization error. The measured error is the difference between object pose estimates in the robot's reference frame. The errors are tabulated in Table 5.3. Relocalization error in position is typically less than 1 cm, and never greater than 3 cm. After localization, the maximum external error is less than 1 cm, indicating the final poses are consistent with the shape-model.

5.6 Classification

Two questions arise when predicting externally provided labels for a physical object using its observed shape. One question is the extent to which the robot can use an observed shape model to predict the externally provided labels. A second question is the extent to which the robot can use internally defined symbolic shape classes to predict externally provided labels.

Error	Localization		f_C	
	position	θ	BV	max(e_L, e_R)
	(m)	(rad)	(m^2)	(m)
recycle-bin	0.012	0.003	0.000	0.000
	0.003	0.091	0.000	0.008
	0.005	0.112	0.000	0.002
	0.009	0.036	0.000	0.000
big-chair	0.006	0.043	0.001	0.006
	0.005	0.034	0.001	0.006
	0.003	0.006	0.001	0.007
	0.023	0.137	0.001	0.008
garbage-can	0.005	0.006	0.000	0.005
	0.008	0.110	0.000	0.001
	0.006	0.159	0.000	0.000
	0.009	0.121	0.000	0.000
red-chair	0.009	0.010	0.000	0.000
	0.006	0.017	0.000	0.005
	0.021	0.017	0.000	0.000
	0.015	0.009	0.001	0.006
dell-box	0.011	0.290	0.001	0.006
	0.010	0.034	0.001	0.006
	0.006	0.087	0.001	0.006
	0.006	0.125	0.001	0.006

Table 5.3: Localization error for each object after a perturbation of (0.1 m, 0.1 m , 0.1 rad). One instance of the object is taken as the canonical model. From the each of the other four model-aligned instances of the object, a single scan is perturbed by (0.1 m, 0.1 m , 0.1 rad) and then the scan is localized to the model. The deviation between the pose after localization from the original pose is used to define a positional and angular error. Note that the errors in position (xy) and the exterior vectors are small (typically less than a centimeter).

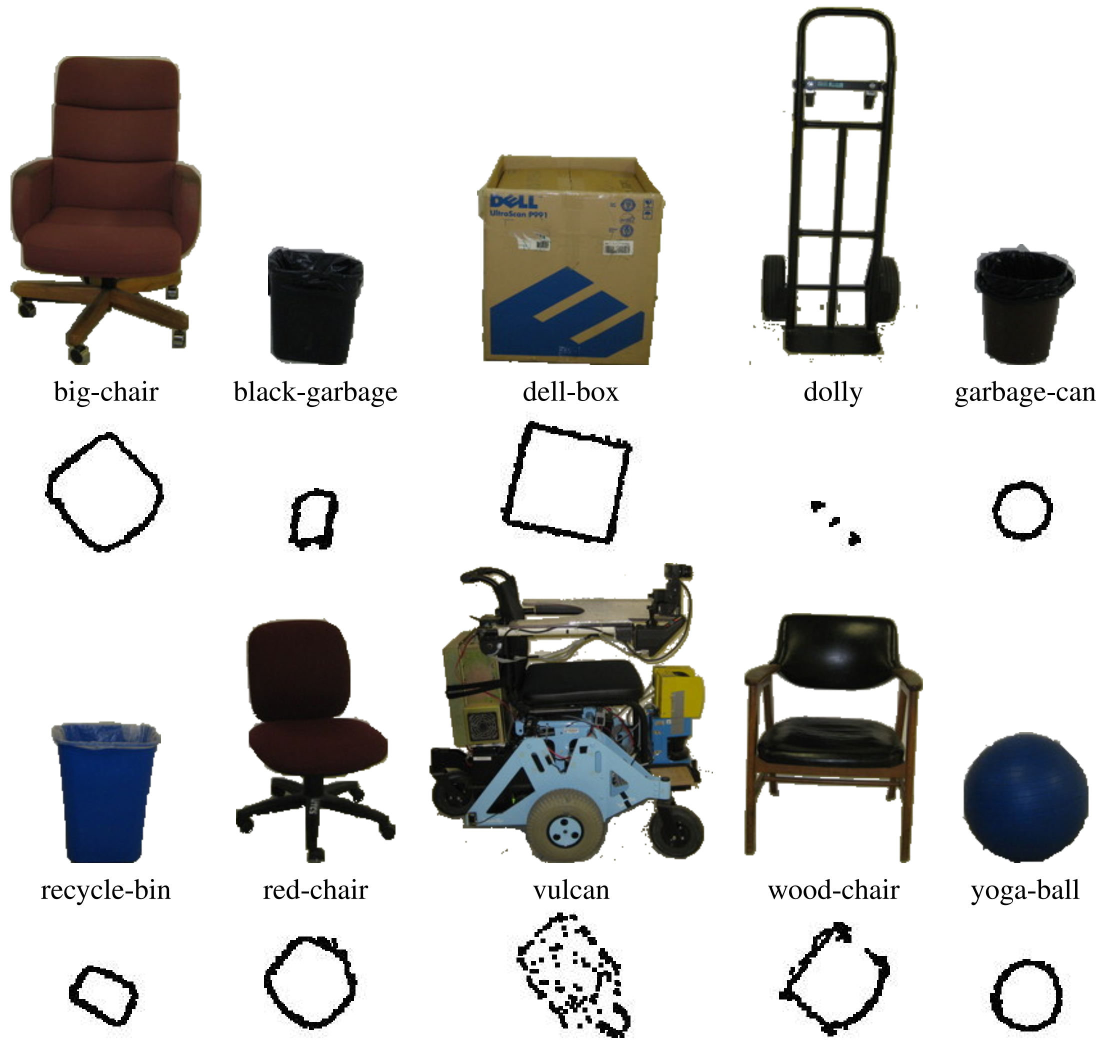


Figure 5.7: A set of physical objects with their learned shapes.

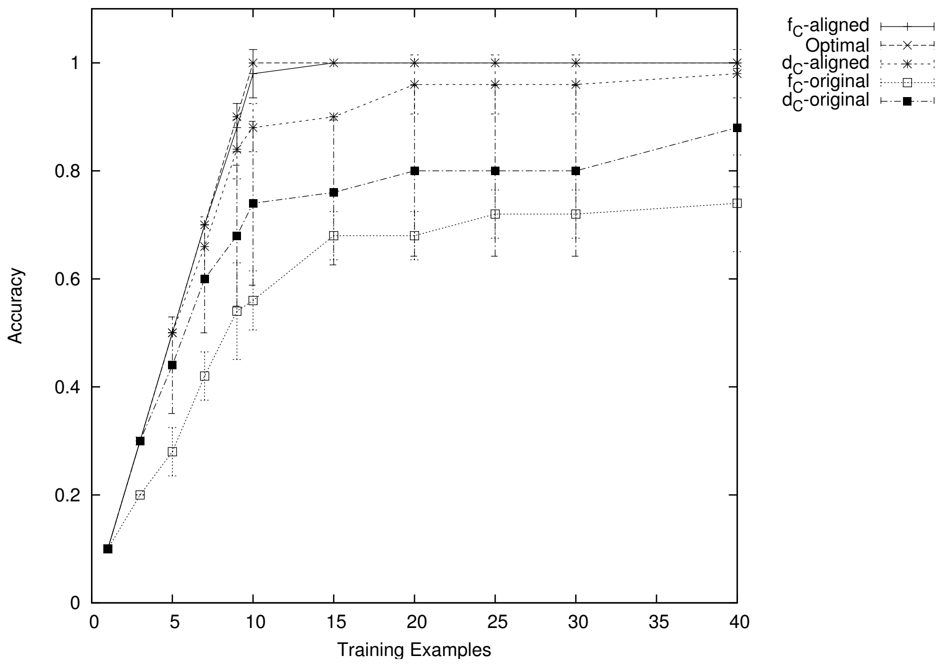


Figure 5.8: Classification accuracy using the nearest neighbor based on shape.

The data for this task comes from the ten physical objects in Figure 5.7 which are each observed on five separate occasions. Each observation contains scans of the object from multiple angles. The robot gathered the data by autonomously circling each object using an externally provided control routine. Each physical object is given a distinct label, and each observation of a physical object is tagged with the object's label. The robot's task is to predict the externally provided label using the observed shape model. The collected data is used to run classification experiments using five-fold stratified cross validation.

Multiple shape distances are tested on this classification task. The f_C function is defined in Equation 5.5 using the angular constraints between shape models M and N as

$$f_C(M,N) \equiv \min_p BV(M \cup t(N,p)).$$

The d_C function defined in section 4.2.4 also measures the distance between shapes but uses the

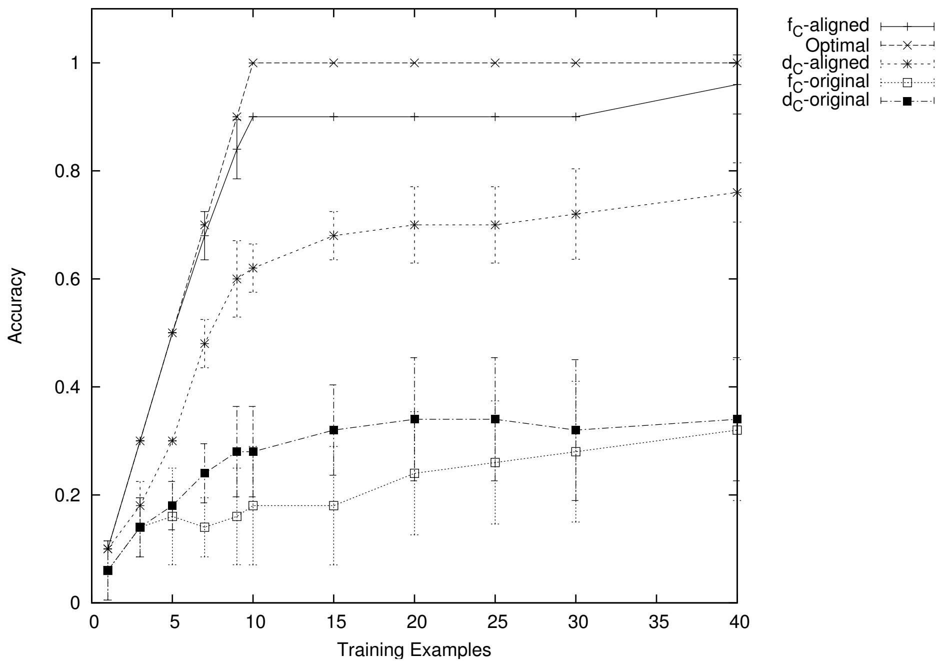


Figure 5.9: Classification accuracy using the intrinsically defined symbolic shape classes.

points in the standard shape images.[1] The distances are applied to shape models formed with both the original scans and the aligned scans (aligned with BV error minimization).

Recall from Section 3.4, that for a particular shape distance d, a shape model M is used as

[1]Recall from chapter 4 that d_D is defined as a distance between the points of two snapshots S and S' (where each snapshot is a set of points in the plane).

$$d_D(S', S) = \frac{1}{|S'|} \sum_{p \in S'} \min(1, \frac{1}{\epsilon} \min_{q \in S} ||p - q||)$$

Then d_D is used to define the distance d' between shapes V and W which are represented as sets of snapshots.

$$d'(V, W) = \frac{1}{|V|} \sum_{v \in V} \min_{w \in W} d_D(v, w).$$

Finally, d_C is defined as

$$d_C(V, W) = \max(d'(V, W), d'(W, V)).$$

a prototype for the symbolic shape class $\sigma_{f,M}$ (with a threshold distance δ).

$$\sigma_{f,M} = \{N \mid d(N, M) < \delta\}$$

When the robot observes an object with the shape N, the robot checks if the shape falls into an existing class, and if not the robot creates a new class.

The two tasks are to perform classification with nearest neighbors using (1) the shape model, and (2) the symbolic shape class of the object. For classification of an object based on its symbolic shape class, the algorithm returns the label most frequently observed with the instances associated with the symbolic shape class. As there are ten possible object labels, random guessing would achieve ten percent accuracy, while an optimal learner would achieve 100% accuracy after observing all ten objects.

The results from the first task of shape model classification are shown in Figure 5.8. Classification accuracy after shape alignment is good both with d_C and f_C. Classification accuracy is significantly worse before alignment for both distance functions (using a two-tailed paired t-test at $p = .01$). Using the aligned data, the f_C function performs better than the d_C function, but the differences are not significant on this task.

The second task is for the robot to use the internally defined shape classes to predict the externally provided labels. This task is a better method for evaluating a robot's classification accuracy in open environments as the robot encounters novel objects for which it has not formed models. The threshold δ used for defining classes was set manually using the sensor resolution of 0.01m for f_C function with aligned shape data, the resolution of the occupancy grid of 0.1m for the unaligned f_C function, and 0.33 for the d_C function (as set in Chapter 4). The results from the using symbolic shapes are shown in Figure 5.9. The results from using symbolic shape classes are worse than using shape models because generalization does not extend beyond the threshold δ for the formation of a new class. Classification accuracy is significantly better for aligned models than the unaligned models. Moreover, on the aligned data the f_C function performs significantly better than the d_C function (using a two-tailed paired t-test at $p = .01$).

From the two experiments, there are multiple conclusions. One is that classification is significantly better using shape models aligned with the BV function than with unaligned models. The f_C function is better than d_C though neither is really bad. Classification is better using the raw shape models than using the shape classes. However, using the symbolic shape class reduces the memory requirements for the robot since it does not have to remember every observation of every physical object and the symbolic shape classes still perform well (90% accuracy with the f_C function). Future work can look at forming classes with semi-supervised methods that make use of external labels or other object information.

5.7 Hierarchies of Shape

The distance functions f_C and d_C are used to induce a hierarchy between the shape models as shown in Figure 5.10. Different observations of each physical object are given distinct identifiers in this figure. The scans in individual shape models have been previously aligned using BV error minimization. The individual shape models can be considered as the nodes in a graph, and each distance function defines the length of an edge between all pairs of nodes. A shape hierarchy is created by finding the minimum spanning tree in the distance graph. The tree is displayed as a dendrogram where the horizontal axis indicates the link length at which a set of nodes is connected to its neighbors.

The groupings for f_C demonstrate that all observations of a physical object are typically clustered together with a distance near zero before joining larger clusters to include observations from other objects. This fails only for distinguishing between the recycle bin and the black garbage bin, which have nearly indistinguishable shapes.

Although the number of examples is limited, this figure provides some evidence that f_C distance between object shapes could be used to predict a human notion of an object's function. The recycle bin and black garbage bin are the closest two objects, and the garbage can is linked to these before any others. All three of the physical objects share the human notion of being waste receptacles. The three chairs are also linked together into a cluster. This "chair" cluster is joined

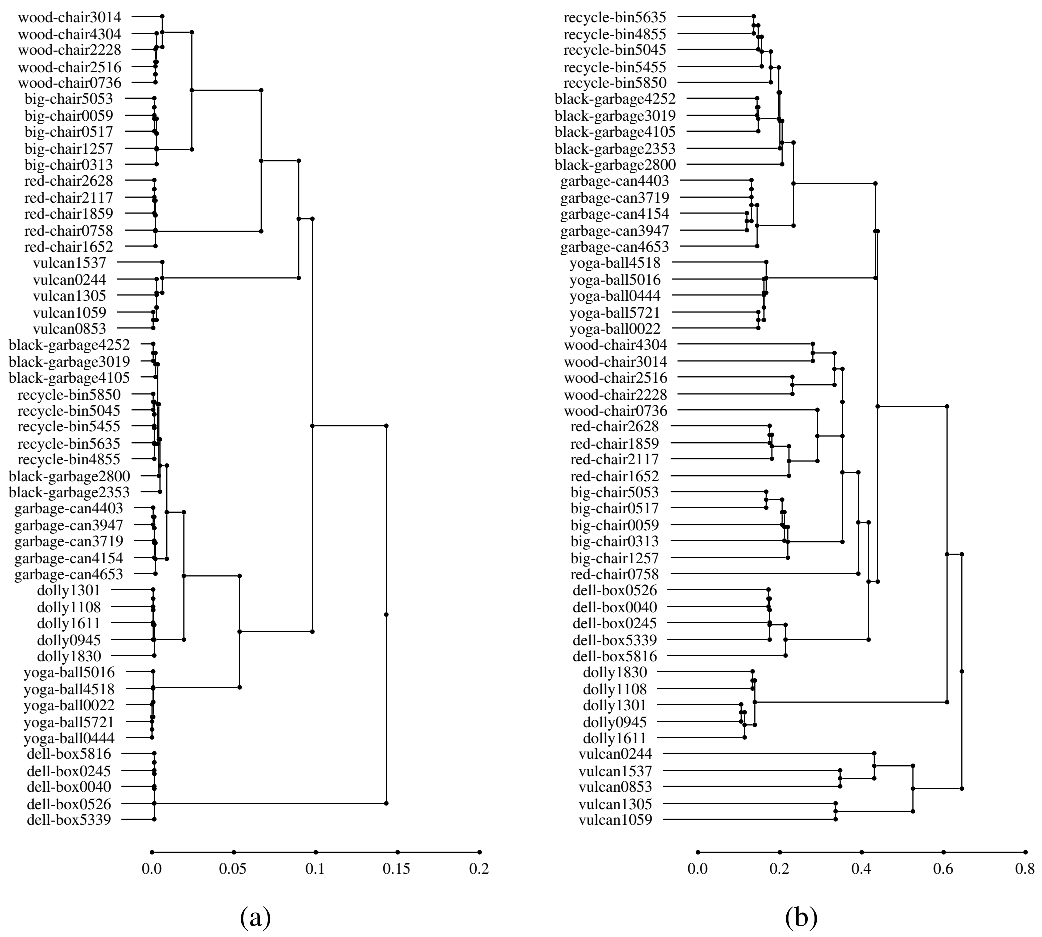


Figure 5.10: Shape hierarchies are created from multiple observations of the objects in Figure 5.7. The identifiers for the objects are included here for clarification and they are not available for the clustering algorithm. The graphs depict the minimum spanning trees formed using the distances between the observed shapes. The trees are induced by (a) f_C (the distance function using angular constraints) and by (b) d_C (the view based distance function). Both distances are computed after scan alignment using BV. The horizontal axis indicates the distance at which a branch is connected to its neighbors. In the tree formed by f_C, all observations of the same physical object are grouped near zero. However, the d_C function does not form clusters of a single physical object at a single threshold.

to another cluster with a variant of a chair (Vulcan the wheelchair) but excludes other large objects (including the yoga ball and the box).

In contrast to the f_C function, the d_C function does not form clusters with a single threshold from multiple observations of the same physical object. Without such a threshold, a robot without external supervision can not reliably detect when it is observing a previously modeled physical object. Hence, although the d_C function can be used successfully for the classification of object shapes, it does not form symbolic shape clusters as well as the f_C function.

In summary, the d_C function is less useful for forming a shape hierarchy than f_C, and it is less capable of autonomously defining shape classes that correspond to a single physical object. The f_C function provides a shape hierarchy that is able to separate most of the physical objects into distinct classes. Finally, future work may be able to predict human notions of similar object function based on the distance between object shapes as measured with the f_C function.

5.8 Related Work

This work demonstrates the use of angular constraints to solve the tasks of creating and using object shape models. This approach is accurate to the precision of the sensor. Related work on shape modeling has focused on somewhat different issues. Biswas et al [9] demonstrate how occupancy grid models of an object's shape can be constructed. Angelov et al [1] have demonstrated how articulated models can be constructed from dense 3D range data. Neither of these papers evaluate the empirical error of the resulting models for recognition and localization. Shape model algorithms in the graphics community can create dense three dimensional models of sculptures when given more computation and data [27], but do not address object recognition or localization.

The idea of using geometric constraints in 3D mapping has been explored for buildings [109]. Similar ideas for inferring object structure from silhouettes or shadows have been pursued in vision, typically without the use of range information, but with notable exceptions. Esteban and Schmitt combine stereo with silhouettes to model objects in three dimensions [35], and a similar approach is taken in [22]. Both approaches provide a solution for three dimensions but neither empirically quantifies the errors in model construction, recognition, and localization.

5.9 Summary

The primary idea in this chapter is that the underlying tasks in object shape model construction, recognition, and localization can be solved by minimizing violations of angular constraints. Scans are aligned by constraining the object data in one scan using bounds from other scans. This approach is effective as quantified by the measured residual errors. Moreover, these residual errors are of approximately the same magnitude as the sensor precision.

This chapter improves the original object ontology in multiple ways. It introduces the use of angular constraints, which improves on the original perceptual filter for shape, and thus improves the shape classes. The same constraints are used to create new perceptual filters for object localization in real-time, allowing the robot to estimate the pose of known rigid objects.

Future work can extend this approach to 3D data from stereo vision or other range data. The next chapter will use object localization to facilitate mining a robot's experience for interesting actions.

Chapter 6

Learning Actions for Objects

Previous chapters have shown how objects can be individuated from low-level sensation, and certain properties can be learned for individual objects. This chapter shows how high-level actions can be learned autonomously by searching for conditions where object properties change in predictable ways. A physical robot is used to demonstrate how high-level actions are learned from the robot's own experiences, and how the robot achieves goals by planning with the learned actions.

6.1 Introduction

People and robots use a finite set of sensor and actuator capabilities to interact with the effectively infinite state of the environment. To manage the inherent complexity, people generate high-level abstractions to simplify reasoning about a problem and planning a solution. Abstraction must also occur within a developing agent learning models of the reliable portions of its sensorimotor experience. Previous chapters have demonstrated how a robot can autonomously learn object models. Perceptual filters applied to trackers generate object properties that the robot can observe but does not yet know how to control.

This chapter describes an algorithm for a robot to autonomously learn new high-level actions to change object properties. This work extends the learned ontology of objects by learning actions that are grounded in the robot's sensorimotor experience. This learning approach is inspired by the observation that infants incrementally develop capabilities both to perceive and to act [68, 99].

Actions are created by learning the conditions under which individual object properties change reliably. The algorithm searches through a log of the robot's experience of object interactions to find constraints on perceptual contexts and motor commands that cause an object property to change reliably. The learned actions are evaluated for their utility in accomplishing goals.

6.2 Continuous Actions

Previous chapters have described how the robot can track objects and learn models of object shapes. The robot also can estimate an object's position and heading in the world, and it is possible to define additional perceptual filters to estimate the distance and angle to the object. However, the robot does not have actions that can change these properties. For example, the robot does not know how to move an object to a target location, how to approach an object, or how to turn to face an object. These behaviors can be specified using target values for object properties. The actions learned by OPAL have control laws that cause one property to change reliably. The robot can use these actions to achieve perceptually specified goals, such as changing the position of a recycling-bin to lie in the center of a room.

6.2.1 Background

Actions vary in form and function in symbolic planning and motion planning. In symbolic planning, actions are hand-crafted to achieve domain specific symbolically parameterized subgoals. At this level of description, an action from the Monkeys and Bananas problem can be `Move(Box, Bananas)` for the high level task of moving a box under a bunch of bananas. Converting this symbolic action into a continuous motion in the environment involves changing a continuous state variable (the box's position) to a target region (under the bananas). In motion planning, algorithms operate with a low dimensional continuous variable, using forward and backward chaining of motion primitives to plan a path to a target region. For this, motion planning requires forward or backward motion models. Motion planning algorithms plan over all components of the state vector simultaneously which often causes the computational complexity to rise exponentially with dimension.

Consider the task of moving an object to a target location. One method for the robot to accomplish this task is to push the object towards the target. But the robot cannot successfully push the object if the robot is not sufficiently close to the object. Hence, the robot must first approach the object. But to approach the object, the robot must be facing the object. Facing an object is a relatively simple task when the object is perceived, as it only requires turning in place.

This example illustrates several issues that arise when acting on objects. One is that accomplishing a task can be complex and involve multiple stages, which suggests that planning is useful. A second is that some tasks can be specified as a constraint on a single object property, for example that the distance from the object to the target is less than a threshold. A third is that destination locations can vary, and the action must generalize across destinations. A fourth is that the action does not depend greatly on which object is being moved, and so actions should support generalization across objects.

6.2.2 Actions for Planning

Several difficulties arise when creating descriptions of actions. The *qualification problem* is the difficulty of specifying all of the preconditions that must be fulfilled for an action to be successful. The *ramification problem* is the difficulty of specifying all consequences of performing an action. The *frame problem* is the difficulty of specifying all non-consequences of an action (all states of the world that do not change).

To guarantee the success of a plan, a planner needs to represent all the consequences and preconditions of an action. However, the real world is infinitely complex and so all consequences and preconditions of an action can not be known. The robot can still plan and achieve goals with high reliability if it knows the consequences and preconditions that are applicable in its operating environment. In particular, the robot can learn actions whose consequences and preconditions have simple descriptions.

For actions to support forward or backward chaining, the consequences of one action should help to fulfill the preconditions of other actions. Thus, goals and action consequences should have complementary forms. Since changing an object property to a desired value can take multiple time steps, it is useful for the robot to monitor the effects of executing an action to ensure progress is being made.

6.2.3 Learning

The complexity of the real world presents a daunting challenge for an autonomous agent learning new actions. The agent must reason with its finite resources about the known consequences of an action but the results of procedural routines for perception and action depend implicitly on the effectively infinite state of the world.

The robot learns actions by observing the effects of performing random motor babbling in the presence of the object. Motor babbling is a process of repeatedly performing a random motor command for a short duration. One strength of the action learning approach presented here is that the robot is able to use this goal-free experience to form actions that are used for goal directed planning. The robot performs self-supervised learning, where the observations in the training data are labeled using the qualitative changes that occur to a single object property. The learned actions are used to achieve goals by reducing the difference between the robot's current perception and the desired goal. This is demonstrated in the evaluation. Thus goals are not required while the actions are being learned but they are required for planning and execution.

Name	Definition	Dim
robot-location	robot's location in the map	2
robot-heading	robot's heading in the map	2
distance(τ)	distance from sensor to object τ	1
angle(τ)	angle from sensor to object τ	1
location(τ)	location in map of object τ	2
heading(τ)	heading in map of object τ	2

Figure 6.1: Perceptual filters used by the learning robot. The robot has previously learned actions that change its location and heading in the map. The robot learns to control the properties of the object image on its sensor array (the angle and distance to the object). The robot also learns an action to control the object position in the environment (Figure 6.2).

6.3 Action Definition

An action is defined in Equation 3.5 as

$$\alpha = \langle D, C, H \rangle$$

a tuple with a description D of the change in an object property caused by the action, a perceptual context C in which the action can reliably cause the described change, and a control law H that sends output to the motors. These components are now formally defined.

The action learning algorithm is restricted to perceptual filters f_j that are either vector-valued or failure (when the object is not perceived).

$$f_{j,t}(\tau) \in \Re^{n_j} \cup \{\perp\}. \tag{6.1}$$

In particular, the robot does not learn an action to change the shape of an object, as a shape percept is represented by a set. The change in a perceptual filter is denoted by δ,

$$\delta_{j,t} = f_{j,t+1} - f_{j,t}. \tag{6.2}$$

The description of an action's consequences,

$$D = \langle\, j, b, q_{j,b} \,\rangle,$$

consists of the name j of the perceptual filter to be controlled, the qualitative behavior b,

$$b \in \{inc, dec\} \cup \{dir[f_k] \mid f_k \in Filters\}, \tag{6.3}$$

and the quantitative effect $q_{j,b}$. Two qualitative behaviors are defined for a scalar perceptual filter: increasing (inc) and decreasing (dec). The qualitative behavior $dir[f_k]$ for a vector-valued perceptual filter f_j means that f_j changes in the direction of the vector-valued perceptual filter f_k. The quantitative effect for scalars is bounded by $\epsilon_{j,b}$,

$$q_{inc}(\delta_{j,t}) \equiv \delta_{j,t} > \epsilon_{j,inc}, \quad q_{dec}(\delta_{j,t}) \equiv -\delta_{j,t} > \epsilon_{j,dec}. \tag{6.4}$$

The quantitative effect for vectors is also bounded by ϵ'_j,

$$q_{dir[f_k]}(\delta_{j,t}) \equiv ||\delta_{j,t}|| > \epsilon_{j,dir[f_k]} \wedge \frac{\langle \delta_{j,t}, f_{k,t} \rangle}{||\delta_{j,t}|| \cdot ||f_{k,t}||} > 1 - \epsilon'_j. \tag{6.5}$$

The context of an action is represented as a conjunction of inequality constraints on scalar perceptual filters:

$$(fRc) \text{ where } f \in Filters,\ R \in \{\leq, \geq\},\ c \in \Re. \tag{6.6}$$

Finally, the control law of an action is a function H from the perceptual filters to a motor output. In this work, the control laws are restricted to constant functions.

6.4 Learning Algorithm

The learning algorithm is initially given a log of observations of object interactions. The log consists of percept values and motor outputs gathered at every time step. The learning algorithm uses the log to search for actions that reliably induce a qualitative change in the observed values for each perceptual filter.

The learning algorithm takes multiple steps to find the components of the action as defined in Equation 3.5. The learning algorithm starts with a perceptual filter f_j and a qualitative behavior b. First, a threshold ϵ is selected from the observed values of δ to complete the description. Next, the quantitative effect is used to search for constraints on the perceptual context and motor output that reliably induce the desired behavior. Finally, the constraints are used to define a perceptual context and a control law. The learning process is initially described for scalar perceptual filters, and then the modifications for vector perceptual filters are described.

For each scalar perceptual filter with a qualitative behavior, a threshold ϵ is chosen by running a Parzen window with a Gaussian kernel over the observations of δ for $b = inc$ ($-\delta$ for $b = dec$). The threshold ϵ is set to the first local minimum above zero if it exists, otherwise it is set to a value one standard deviation from the mean. This completes the description D of the action.

The threshold ϵ is used to define $q_{j,b}$, and $q_{j,b}$ is used to label the examples of the desired behavior in the training data. The learning algorithm uses the labeled examples to search for constraints on the percepts and motor outputs. The constraints are represented by the intersection of axis aligned half-spaces, specified as inequalities (as in Equation 6.6) over the scalar perceptual filters (f_k) and the components of the motor vector ($\pi_k(u)$).

Given the log of babbling experience as a sequence of observations (z_t) and motor output (u_t), the action learning algorithm attempts to find perceptual constraints C and motor constraints M that support the following rule:

$$z_t \in C \wedge u_t \in M \implies q_{j,b}(\delta_{j,t}) \wedge z_{t+1} \in C. \tag{6.7}$$

This equation states that if at one time step the robot's perception satisfies the perceptual constraints ($z_t \in C$) and the robot sent a motor command that satisfies the motor constraints ($u_t \in M$) then at the next time step the object property of interest changed in the desired manner ($q_{j,b}(\delta_{j,t})$) and the robot's perception still satisfies the perceptual constraints ($z_{t+1} \in C$). This constraint on the observation of the next time step is included to encourage the discovery of repeatable actions: actions that can be used on multiple consecutive time steps.

The learning algorithm is capable of only creating actions whose context can be be expressed as a conjunction of perceptual constraints. This limitation simplifies not only the learning process, but also the difficulties of planning. The absence of disjunctions in the context allows backwards chaining without the need to consider multiple options for satisfying the context of a desired action.

To find the perceptual constraints C and motor constraints M, define a set of measures for the precision (μ_0), recall (μ_1), and repeatability (μ_2). The utility function U is their geometric mean. These functions are defined using empirical probabilities (Pr) as observed in the training data.

$$\begin{aligned} Pr(a|b) =& \ \tfrac{Pr(a,b)}{Pr(b)} = \tfrac{\#\{t|a_t \wedge b_t\}}{\#\{t|b_t\}} \\ \mu_0 =& \ Pr(q_{j,b}(\delta_{j,t}) \mid z_t \in C \wedge u_t \in M) \\ \mu_1 =& \ Pr(z_t \in C \wedge u_t \in M \mid q_{j,b}(\delta_{j,t})) \\ \mu_2 =& \ Pr(z_{t+1} \in C \mid z_t \in C \wedge u_t \in M) \\ U =& \ (\mu_0 \mu_1 \mu_2)^{\frac{1}{3}} \end{aligned} \tag{6.8}$$

The components of this utility function are each contributing an important weight. The precision measures the probability that a positive classification implies a positive label. The recall measures the probability that a positive label implies a positive classification. The repeatability measures the probability that if the robot's motor command meets the motor constraints and the robot's percept meets the perceptual constraints, then the robot perception at the next time step will also meet the perceptual constraints (and thus can repeat the motor command).

The learning algorithm creates a constraint set that is initially empty, and greedily adds constraints. At each iteration, the algorithm finds the best new constraint to add for each variable (searching over both the perceptual filters and the motor outputs). For each variable, the algorithm sorts the observed values of the variable. For each value, the algorithm tests the utility of setting adding a constraint at that value. The algorithm selects the proposed constraint that yields the greatest increase in utility when added to the current constraint set. The process terminates when adding a constraint provides no significant improvement to utility. The newly generated action is discarded if the final utility measure is low.

For vector percepts the above algorithm requires a few modifications. First, the threshold for finding ϵ is set using a Parzen window over $||\delta||$. Second, the function $q_{j,b}$ in the search for C and M is given by $q = q_{j,mag}(\delta) = ||\delta|| > \epsilon_{j,mag}$. Third, the constant ϵ'_j is set after the constraints C and M are learned. The algorithm generates a potential threshold α for every observed value (from Equation 6.5).

$$\alpha = \frac{\langle \delta_{j,t}, f_{k,t} \rangle}{||\delta_{j,t}|| \cdot ||f_{k,t}||}. \tag{6.9}$$

In this form, α acts as an analog version of the precision, as it quantifies the amount by which the change lies in the direction of f_k. The algorithm selects the value of α that maximizes the product of α and the recall. Finally α is used to define the threshold $\epsilon'_j = 1 - \alpha$.

The learned constraints are split between the perceptual constraints C and the motor constraints M. The perceptual constraints C become the context of the action. A constant control law is defined from M.

$$H(z_t) = m = \arg \min_{u_t \in M} ||u_t|| \tag{6.10}$$

The constant control law is enhanced in two ways. The first is to account for perceptual latencies by predicting the current value of the percept. The second is to scale the motor output by the minimum effect ϵ, when the robot wants to change a percept to a goal value g.

$$s(g, \tau) = \min(1, ||E[f_{k,t}(\tau)] - g||/\epsilon) \tag{6.11}$$

$$H(z_t) = s(g, \tau) \cdot m \tag{6.12}$$

Putting the learned components together creates the new action $\alpha = \langle D, C, H \rangle$.

6.5 Training Scenario

Several perceptual filters were used for learning and they are listed in Table 6.1. Filters for the robot's location and heading come from the robot's ontology of space. The robot's heading is represented as a unit vector. The remaining filters are computed for each tracker. The support of a tracker, $\text{Supp}_t(\tau) \subset \Theta$, is given by the sensor indices of the points in the snapshots, and is represented as a subset of the sensor indices Θ. The support is used to define the angle and distance to the object.

$$\mathsf{angle}_t(\tau) = \text{mean}\{i \mid i \in \text{Supp}_t(\tau)\} \tag{6.13}$$

$$\mathsf{distance}_t(\tau) = \min\{z_t(i) \mid i \in \text{Supp}_t(\tau)\} \tag{6.14}$$

Object localization, described in the previous chapter, provides the position and heading of the object.

The robot was physically modified for this experiment by the addition of a small foam bumper to the front of the robot. The bumper reduced the impact of the object collisions on the robot's body. The bumper also kept the objects further away from the range sensor which limited the amount of the environment obscured by the object. This was necessary as the localization algorithm requires an adequate view of the static environment to keep the robot localized in the map.

The robot created a log of observations for training data. The robot gathered observations by randomly selecting a motor command and executing it for a fixed duration. The motor commands for drive and turn (linear and angular velocities) were selected from the following set.

$$\{-0.2, 0.0, 0.2\}m/s \times \{-0.4, 0.0, 0.4\}rad/s$$

The data was gathered in different environmental configurations, where the experimenter changed the environment between trials. The experimenter ensured the robot could see the training object (the recycle-bin) at the start of every trial. Approximately ten minutes of training data were gathered, at approximately one observation per second.

Running the learning algorithm generated several useful actions that are shown in Figure 6.2. These actions can be thought of as simple affordances of the object; actions which the robot currently assumes will always work. However, the action for pushing an object can fail for heavy objects. As an extension to the current work, the robot can use these actions to create new perceptual filters that predict which objects are not pushable.

The learning algorithm was not able to learn an action to control every object property. In particular, it was not able to learn a useful action for changing the heading of the object. The robot's physical configuration makes it is difficult for the robot to continuously push and turn the object. When the robot pushes the object off-center, the object tends to slide off the bumper. Because of the lack of examples of successfully turning an object, the learning algorithm was unable to create a reliable action for the task.

6.6 Planning with Goals

Part of the value of the object ontology lies in providing a language for representing goals. The high-level task given by "Place a recycle bin in the center of the room" can be represented as a goal state with a tracker whose shape corresponds to a recycle bin and whose location is in the center of the room. The robot can set goals and measure its progress towards achieving them. The learned actions are used by a planner to achieve goals by sequentially reducing differences between the robot's current percepts and the goal.

To achieve goals, the planner uses backward chaining with the constraints provided in an action's context. The planner creates reactive plans to change a percept to a desired value. Attempting to sequentially satisfy the constraints in an action's context can fail when more than one precondition is not satisfied.

In this condition, the robot simulates observations from possible robot poses to find a pose from which the perceptual context is satisfied. The algorithm creates an error vector, with one dimension per constraint. The value of each component of the vector is the amount by which the

Action I		
	Description	$\langle$ angle(τ) , inc , $\delta > 12$ $\rangle$
	Context	$\emptyset$
	Control Law	(0.0 m/s , -0.4 rad/s)
	Utility	73 %
Action II		
	Description	$\langle$ angle(τ) , dec, $-\delta > 12$ $\rangle$
	Context	$\emptyset$
	Control Law	(0.0 m/s , 0.4 rad/s)
	Utility	74 %
Action III		
	Description	$\langle$ distance(τ) , inc , $\delta > .19$ $\rangle$
	Context	$\emptyset$
	Control Law	(-0.2 m/s , 0.0 rad/s)
	Utility	69 %
Action IV		
	Description	$\langle$ distance(τ), dec, -$\delta > .19$ $\rangle$
	Context	(distance(τ) ≥ 0.43) $\wedge$ (angle(τ) ≤ 132) $\wedge$ (angle(τ) ≥ 69)
	Control Law	(0.2 m/s , 0.0 rad/s)
	Utility	61 %
Action V		
	Description	$\langle$ location(τ), dir[robot-heading], $\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context	(distance(τ) ≤ 0.22) $\wedge$ (angle(τ) ≥ 68) $\wedge$ (angle(τ) ≤ 103)
	Control Law	(0.2 m/s , 0.0 rad/s)
	Utility	65 %

Figure 6.2: The above actions were learned by the robot from its observations of the effects of motor babbling. These actions cause changes in (I,II) the angle to the object (by turning), (III,IV) the distance to the object (by driving), and (V) the location of the object in the map (by pushing). The context of an action represents the precondition for the action to be successfully executed for a single time step.

Behavior	Goal	Distance	Accuracy	Time (s)
Face	$\mid \mathsf{angle}(\tau)$ -90 $\mid < 10$	45 °	7.4° $(\sigma = 3)$	4.38 $(\sigma = .51)$
Approach	$\mathsf{distance}(\tau) < 1$	1.8 m	0.04 m $(\sigma = 0.02)$	21.1 $(\sigma = 5.7)$
Move	$\mid\mid \mathsf{location}(\tau)$ - (3,2) $\mid\mid < 0.15$	2.0 m	0.09 m $(\sigma = 0.04)$	258 $(\sigma = 138)$

Table 6.1: The robot used the learned actions to perform three tasks: facing the object, approaching the object and moving the object. The columns indicate the initial distance to the goal, the final distance from the goal and elapsed time. Ten runs were performed for each task, and the results are shown with the standard deviations. Accuracy was measured by the final distance to the target value as measured by the robot. All trials succeeded with the robot accurately achieving its goals. The time to task completion has a high variance since the robot keeps trying until it succeeds.

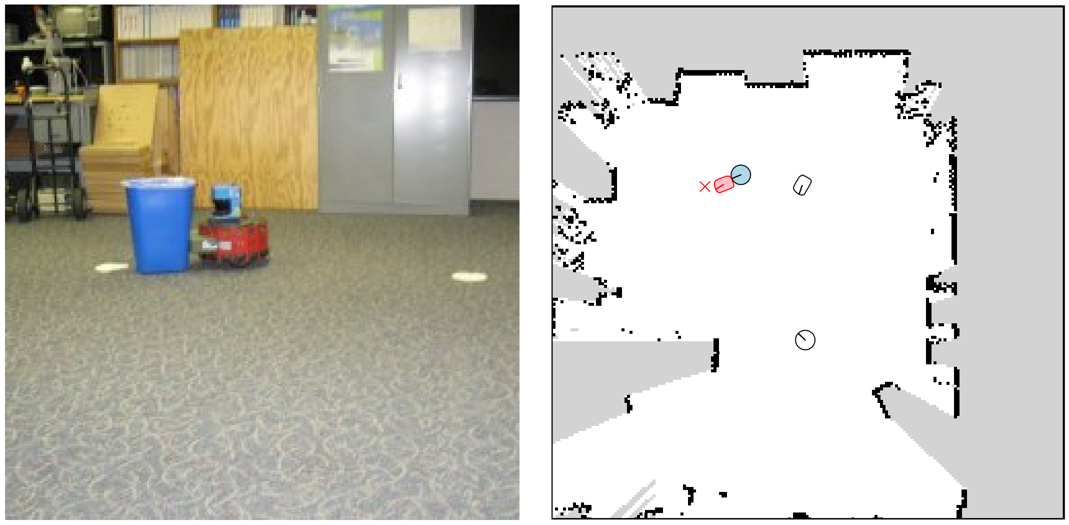
corresponding constraint is violated. The algorithm hill-climbs on the error vector length in the space of robot poses to find a pose with a zero length error vector. A procedure for simulating observations from different poses was provided externally to the learning agent. The robot moves to this pose and then attempts to execute the desired action.

6.7 Evaluation

The learned actions were evaluated on three tasks: facing the object, approaching the object and moving the object to a location. These tasks were represented by setting goal values for the $\mathsf{angle}(\tau)$, $\mathsf{distance}(\tau)$ and $\mathsf{location}(\tau)$ properties respectively.

The starting state for the three tasks was approximately the same (the object was placed at different orientations), and is shown in Figure 6.3. The robot starts at the origin of the map with an orientation of 45° from the object. The object starts three meters away from the robot at the map coordinates (3,0), and the location target for the object is at (3,2) which is two meters away from the object's starting position. The desired final states for the tasks are to have the object in front of the robot ($|\mathsf{angle} - 90| < 10$), to have the robot near the object ($\mathsf{distance} \leq 1.0$), and to have the object at the goal location in the figure ($||\mathsf{location}(\tau) - (3,2)|| < .15$ m).

Ten runs were performed for each task. The same final state is used in each run to reduce task variation between runs. The experimenter physically verified successful task completion for



(a) (b)

Figure 6.3: (a) The robot pushes a recycling bin towards a goal location. (b) The shaded shapes show the robot's percepts for itself and the object. The starting poses of the robot and the object are shown unshaded, and the goal location for the object is indicated by ×.

Behavior	Goal	Distance	Time (s)	Optimal
Face	$\lvert$ angle(τ) -90 $\rvert < 10$	45 ° ($\sigma = 3$)	4.38 ($\sigma = .51$)	4.0
Approach	distance(τ) < 1	1.8 m	21.1 ($\sigma = 5.7$)	11
Move	$\lVert$ location(τ) - (3,2) $\rVert < 0.15$	2.0 m	258 ($\sigma = 138$)	27

Table 6.2: The robot used the learned actions to perform three tasks: facing the object, approaching the object and moving the object. The columns indicate the distance to the goal, the elapsed time, and a lower bound on the optimal time.

each run, and the robot measured its performance for each run. The results in Table 6.1 show that the robot is able to achieve these goals reliably and accurately. Figure 6.3 shows an example of the robot pushing the object to a goal.

6.7.1 Optimal Performance for Action Tasks

The robot used learned actions to complete multiple tasks. Lower bounds on the optimal time required to complete the tasks are computed by adding the latency in the motor system with the travel time required by the robot at maximum velocity. The robot's maximum command velocities are 0.2 m/s and 0.4 rad/s, and the latency is 2 seconds. Thus for the Face task, the minimum time is

$$2s + \frac{\pi}{4}rad/(0.4rad/s) \approx 4.0s.$$

For the Approach task, the robot must move 1.8 meters so the minimum time is

$$2s + 1.8m/(.2m/s) \approx 11s.$$

For the Move task, the robot first must approach an object that is three meters away and then push it for two meters so the minimum time is

$$2s + 3m/(.2m/s) + 2m/(.2m/s) \approx 27s.$$

The robot's performance does not approach the lower bound except for the Face task, but the robot's performance is within a factor of two on the Approach task and a factor of ten on the Move task. These times indicate that the robot is accomplishing the tasks efficiently.

The performance times exceed the lower bounds for the Approach task since the robot does not turn to directly face the object before approaching it. Rather, the robot only attempts to satisfy the angular constraints for approaching the object and it can satisfy the angular constraints even when turned slightly away from the object. In this condition, the robot makes progress for multiple time steps, but eventually it will fail to satisfy the angular constraints. A better control law would provide a closed loop correction [86] on the angular constraints to improve the robot's ability to approach the object.

For the Move task, a significant amount of time is spent by the simulation routine to find a pose from which multiple perceptual constraints are simultaneously satisfied. The simulation routine is externally provided to the robot and provides a function from the robot and object poses to sensor observations. This function could be learned autonomously by the robot using a large number of training examples. Although this function could be learned, learning the function is uninteresting for evaluating the performance of the learned actions and so it has been provided manually in this work.

The robot uses the simulation to find a pose from which it can simultaneously satisfy multiple perceptual constraints. The mean time spent in simulation on a single run is 113 seconds with a standard deviation of $\sigma = 73$ seconds. In addition to the time taken by simulation, the robot occasionally needs to reposition itself. While pushing the object, the robot can move out of alignment with the object and the goal. The robot must move to reposition itself in this situation, and this takes additional time. The number of reposition events varied between runs, and this contributed to the large variance in the time required on the Move task. The amount of repositioning required could be reduced by using a closed loop control law.

6.7.2 Alternate Utility Functions for Learning Actions

Utility functions other than U (as defined in Equation 6.8) can be used for learning actions. Several functions were tried and found to be lacking. The following tables show the actions that are learned from the data using several utility functions. Except for the U function, all utility functions generate at least one action that is qualitatively inadequate.

The observation log from the robot's motor babbling is used to learn actions. The effects for each action are defined from the observation log as in the original action learning experiment. An example from the observation log is given a positive label if the observed effect between time steps meets the threshold of quantitative change $q_{j,b}(\delta_t)$. Using this labelled data, the utility functions are used to learn perceptual and motor constraints. An example is classified as positive if it meets the perceptual and motor constraints in C and M (namely $z_t \in C \wedge u_t \in M$).

The utility functions are defined using the number of examples with a given label and classification.

	Positive Label	Negative Label
Classified Positive	true positive	false positive
Classified Negative	false negative	true negative

The counts for these examples are used to define the $precision$ and $recall$.

$$precision = \text{true positive}/(\text{true positive} + \text{false positive })$$

$$recall = \text{true positive}/(\text{true positive} + \text{false negative})$$

The $repeatability$ is defined to be the fraction of the examples that satisfy the perceptual constraints at the next time step that are classified as true for the current time step.

$$repeatability = \frac{\#\{t \mid z_{t+1} \in C \wedge z_t \in C \wedge u_t \in M\}}{\#\{t \mid z_t \in C \wedge u_t \in M\}}$$

The tested utility functions are listed below. The U utility function is introduced in this thesis. The U without repeatability function is used to measure the contribution of repeatability to the learning process. The F-measure is commonly used in information retrieval [5]. The accuracy function [72] is typically used in machine learning. The weighted accuracy function is a variant of the accuracy function which provides equal weight to the positive and negative examples, and thus corrects for the skewed distribution towards negatively labelled training data.

Utility Function	Description
U	$(precision \cdot recall \cdot repeatability)^{\frac{1}{3}}$
U w/o repeatability	$(precision \cdot recall)^{\frac{1}{2}}$
F-measure	$2 \cdot (precision^{-1} + recall^{-1})^{-1}$
Accuracy	$\frac{\text{true positive+true negative}}{\text{number of examples}}$
Weighted accuracy	$\frac{1}{2}(\frac{\text{true positive}}{\text{true positive+false negative}} + \frac{\text{true negative}}{\text{true negative+false positive}})$

Errors in learned actions consist of qualitative inadequacies. The inadequacies are not known to the robot but are known to an external experimenter. One kind of error is the failure to find relevant perceptual and motor constraints, a failure that is most visible in the actions learned with accuracy as the utility function. A second failure mode is the inclusion of irrelevant constraints into an action's context. To test for the second kind of failure, features for the robot and object

Utility Function	Action I	Action II	Action III	Action IV	Action V	Details
U	Y	Y	Y	Y	Y	Table 6.4
U w/o repeatability	N	Y	Y	N	N	Table 6.5
F-measure	N	Y	N	N	N	Table 6.6
Accuracy	N	N	N	N	N	Table 6.7
Weighted Accuracy	Y	Y	Y	Y	N	Table 6.8

Table 6.3: The table shows which actions are qualitatively adequate, namely the actions include required constraints and exclude irrelevant constraints. Learned actions that are qualitatively adequate are labelled with Y and others are labelled N. The weighted accuracy and F-measure perform well, but fail on at least one action.

heading were included as the scalar features robot-theta and object-theta. These features measure the angle in radians (between $[-\pi, \pi]$) of the robot and object in the reference frame of the map, and both of these features are irrelevant for the all of the learned tasks.

Table 6.3 summarizes the qualitative adequacy of the learned actions. The U utility function is qualitatively adequate for all actions. Removing the repeatability component from U causes the learned actions to include more irrelevant features in the perceptual constraints. This problem also affects both the F-measure and weighted accuracy. The standard accuracy function fails to find correct actions in all cases, because for every desired action there are many more negative examples than positive examples.

6.8 Extensions

The action learning algorithm has been tested on a physical system with real data, but it has not been tested on a great breadth of problems. The algorithm can be improved in many ways. A simple extension is to learn non-constant closed-loop control laws. Another extension is to learn actions to recover from perceptual failures (when the perceptual filter returns $\perp$). Directional control can be extended to handle cases where the direction of change is a function of percepts instead of being parallel to a single percept. This approach requires the simulation of observations from different states to resolve conflicting preconditions. This simulation function could be learned instead of

Action I		
	Effect:	$\langle$ angle,inc, $\delta > 12.30$ $\rangle$
	Context:	
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	73 %
	Adequate:	Yes
Action II		
	Effect:	$\langle$ angle,dec,- $\delta > 12.30$ $\rangle$
	Context:	
	Control:	(0.0 m/s, 0.4 rad/s)
	Utility:	74 %
	Adequate:	Yes
Action III		
	Effect:	$\langle$ distance,inc, $\delta > 0.19$ $\rangle$
	Context:	
	Control:	(-0.2 m/s, 0.0 rad/s)
	Utility:	70 %
	Adequate:	Yes
Action IV		
	Effect:	$\langle$ distance,dec,- $\delta > 0.19$ $\rangle$
	Context:	distance$\geq 0.430 \wedge$ angle$\geq 68 \wedge$ angle≤ 132
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	61 %
	Adequate:	Yes
Action V		
	Effect:	$\langle$ location,dir[robot-heading],$\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context:	distance$\leq 0.220 \wedge$ angle$\leq 103 \wedge$ angle≥ 68
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	65 %
	Adequate:	Yes

Table 6.4: Actions learned using the U utility function.

Action I		
	Effect:	$\langle$ angle,inc, $\delta > 12.30$ $\rangle$
	Context:	angle$\leq 130 \wedge$ object-theta$\geq$-0.923 $\wedge$ angle$\geq 31 \wedge$ robot-theta$\geq$-2.866
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	71 %
	Adequate:	No: includes constraint on an irrelevant variable
Action II		
	Effect:	$\langle$ angle,dec,- $\delta > 12.30$ $\rangle$
	Context:	angle≥ 25
	Control:	(0.0 m/s, 0.4 rad/s)
	Utility:	66 %
	Adequate:	Yes
Action III		
	Effect:	$\langle$ distance,inc, $\delta > 0.19$ $\rangle$
	Context:	angle$\leq 122 \wedge$ angle$\geq 53 \wedge$ distance≥ 0.360
	Control:	(-0.2 m/s, 0.0 rad/s)
	Utility:	65 %
	Adequate:	Yes
Action IV		
	Effect:	$\langle$ distance,dec,- $\delta > 0.19$ $\rangle$
	Context:	distance$\geq 0.430 \wedge$ angle$\leq 132 \wedge$ angle$\geq 68 \wedge$ robot-theta≤ 2.924
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	59 %
	Adequate:	No: includes constraint on an irrelevant variable
Action V		
	Effect:	$\langle$ location,dir[robot-heading],$\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context:	distance$\leq 0.220 \wedge$ angle$\leq 103 \wedge$ angle$\geq 68 \wedge$ robot-theta$\geq$-2.483
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	66 %
	Adequate:	No: includes constraint on an irrelevant variable

Table 6.5: Actions learned using the U without repeatability utility function.

Action I		
	Effect:	$\langle$ angle,inc, $\delta > 12.30$ $\rangle$
	Context:	angle$\leq 130 \wedge$ object-theta$\geq$-0.923 $\wedge$ angle$\geq 31 \wedge$ robot-theta$\geq$-2.866
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	70 %
	Adequate:	No: includes constraint on an irrelevant variable
Action II		
	Effect:	$\langle$ angle,dec,- $\delta > 12.30$ $\rangle$
	Context:	angle$\geq 25 \wedge$ distance≥ 0.320
	Control:	(0.0 m/s, 0.4 rad/s)
	Utility:	67 %
	Adequate:	Yes
Action III		
	Effect:	$\langle$ distance,inc, $\delta > 0.19$ $\rangle$
	Context:	angle$\leq 122 \wedge$ angle$\geq 64 \wedge$ object-theta$\leq 2.624 \wedge$ object-theta$\geq$-0.921
	Control:	(-0.2 m/s, 0.0 rad/s)
	Utility:	69 %
	Adequate:	No: includes constraint on an irrelevant variable
Action IV		
	Effect:	$\langle$ distance,dec,- $\delta > 0.19$ $\rangle$
	Context:	distance$\geq 0.430 \wedge$ angle$\leq 130 \wedge$ angle$\geq 68 \wedge$ object-theta$\geq$-2.474 $\wedge$ robot-theta$\leq 2.924 \wedge$ robot-theta$\geq$-2.688
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	60 %
	Adequate:	No: includes constraint on an irrelevant variable
Action V		
	Effect:	$\langle$ location,dir[robot-heading],$\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context:	distance$\leq 0.220 \wedge$ angle$\leq 103 \wedge$ angle$\geq 68 \wedge$ robot-theta$\geq$-2.483
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	65 %
	Adequate:	No: includes constraint on an irrelevant variable

Table 6.6: Actions learned using the F-measure utility function.

Action I		
	Effect:	$\langle$ angle,inc, $\delta > 12.30$ $\rangle$
	Context:	object-theta≥ 2.444
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	92 %
	Adequate:	No: includes constraint on an irrelevant variable
Action II		
	Effect:	$\langle$ angle,dec,- $\delta > 12.30$ $\rangle$
	Context:	angle≥ 174
	Control:	(0.0 m/s, 0.0 rad/s)
	Utility:	86 %
	Adequate:	No: no motion
Action III		
	Effect:	$\langle$ distance,inc, $\delta > 0.19$ $\rangle$
	Context:	
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	94 %
	Adequate:	No: turning to increase distance
Action IV		
	Effect:	$\langle$ distance,dec,- $\delta > 0.19$ $\rangle$
	Context:	
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	94 %
	Adequate:	No: turning
Action V		
	Effect:	$\langle$ location,dir[robot-heading],$\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context:	object-theta$\leq$-3.042
	Control:	(0.0 m/s, 0.0 rad/s)
	Utility:	92 %
	Adequate:	No: no motion

Table 6.7: Actions learned using the accuracy utility function.

Action I		
	Effect:	$\langle$ angle,inc, $\delta > 12.30$ $\rangle$
	Context:	
	Control:	(0.0 m/s, -0.4 rad/s)
	Utility:	82 %
	Adequate:	Yes
Action II		
	Effect:	$\langle$ angle,dec,- $\delta > 12.30$ $\rangle$
	Context:	
	Control:	(0.0 m/s, 0.4 rad/s)
	Utility:	78 %
	Adequate:	Yes
Action III		
	Effect:	$\langle$ distance,inc, $\delta > 0.19$ $\rangle$
	Context:	
	Control:	(-0.2 m/s, 0.0 rad/s)
	Utility:	85 %
	Adequate:	Yes
Action IV		
	Effect:	$\langle$ distance,dec,- $\delta > 0.19$ $\rangle$
	Context:	distance$\geq 0.430 \wedge$ angle$\leq 134 \wedge$ angle≥ 68
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	86 %
	Adequate:	Yes
Action V		
	Effect:	$\langle$ location,dir[robot-heading],$\|\|\delta\|\| > .21 \wedge \epsilon' = .13$ $\rangle$
	Context:	distance≤ 0.230
	Control:	(0.2 m/s, 0.0 rad/s)
	Utility:	87 %
	Adequate:	No: Missing constraints on angle

Table 6.8: Actions learned using the weighted accuracy utility function.

being externally provided. A final extension is to move beyond motor babbling to active learning, where the robot actively gathers data to test partially learned actions.

6.9 Related Work

Work in psychology has explored the development of object representations in children. Spelke [99] has studied the perceptual development in children as it progresses from using motion as a indicator of object unity to using other perceptual cues. Mandler [68] has explored how concepts might form in more general conditions. Bloom [11] has studied how objects and concepts are used to quickly learn a language.

Previous work in developmental robotics [85, 84, 23] has shown how the structure of an agent's sensory and motor systems can be learned. A key method in this process is the projection of high-dimensional observations into a low-dimensional space. Further advances include Isomap [105] which identifies manifolds in the data, and the use of information distance [82].

Other researchers have also studied how a robot can learn object representations with actions. Stoytchev [102] has a robot learning the affordances of simple tools using a single continuous variable to represent the state. Other work [49] demonstrates how a robot can learn the preconditions for actions. Natale [79] has shown how motor babbling with a robot arm can be used to learn how to move objects. These approaches use stationary perception, whereas the work in this thesis demonstrates the use of mobile perception.

Related work has also explored object recognition and action but not in conjunction. Work on object recognition has used mapping techniques [9], and image-based models [66].

Actions have been learned in simulated symbolic domains [94, 42, 7, 81, 116]. These approaches provide methods for learning actions when an object model is already available, but do not address how continuous actions can be learned on a mobile robot.

6.10 Summary

This chapter adds learned actions to the object ontology. Actions are learned to reliably change scalar and vector perceptual filters. Using these actions, the robot is able to plan and achieve goals.

OPAL shows how a robot starting with ontologies of sensorimotor organization and space can construct object representations. Multiple representations are acquired, which can be interpreted as perceptual, structural and functional object models. The object snapshots and trackers form perceptual representations. The shape model provides a structural representation. A functional model is available through the learned action of pushing: this action could be used to define the pushing affordance of an object. By integrating these different aspects of objects, the learned representations support perception, geometric inference, and goal-directed planning.

These representations are part of an ontology of objects grounded in the robot's sensorimotor experience. The learned ontology creates trackers for individual objects, forms percepts from observations, forms classes to generalize from past experience, and learns actions to change the perceived properties of an object. Using this ontology, the physical robot is able to recognize objects and plan with learned actions to achieve goals. The learned ontology is simple and lays the foundation for learning more complex object models. In future work, the learned object representations can be used to extend a robot's ability to understand and interact with its environment.

Chapter 7

Contributions and Future Directions

This thesis describes how a robot can learn an ontology of objects, bootstrapping from ontologies of its self and space. The Object Perception and Action Learner (OPAL) generates trackers to represent individual objects, uses perceptual filters to measure object properties, creates perceptual classes to recognize objects, and learns actions that change object properties. OPAL permits the robot to succinctly describe its current situation, to generalize from past experience, to have goals specified using objects, and to plan with learned actions to achieve these goals.

Description of the object ontology

The learned object ontology contains object trackers, perceptual filters, classes, and actions. This ontology provides counterparts to the individual objects, properties, and actions used in traditional symbolic planning and inference. Individuals (atoms) are represented by trackers, object properties are provided by classes, and both continuous and symbolic actions support planning by backward chaining from goals.

Learning the object ontology

The thesis describes algorithms for the robot to create trackers, perceptual classes and actions. The thesis defines several perceptual filters for objects, but does not provide an algorithm for automatically generating perceptual filters. Methods for constructive induction, and dimensionality reduction can generate new perceptual filters automatically [85, 87].

Evaluated the learned object ontology

The thesis has evaluated the learned object representations. The robot is able to recognize several objects from the lab, using the limited sensing ability of a laser rangefinder. The experiments have shown how a robot can learn perceptual, structural and functional representations for objects. These representations permit the robot to specify goals and achieve them.

OPAL can be extended in many ways. Some fall within the scope of a learned object ontology, such as learning improvements in perception through statistical inference and extensions to more complex sensorimotor modalities including vision and articulated robot bodies. Another extension is to study how a robot can learn integrated ontologies of self, agency, space, and objects to create a complete developing robot that can learn through its interactions with a teacher that the robot perceives as a socially contingent agent. A third extension is to study how the learned ontologies of objects and actions can be used by a robot to learn language through interactions with people.

Bootstrapping statistical perceptual inference

In the completed work, the robot does not improve its ability to individuate and track objects. However, the existing system provides labeled data that can be used to train better perceptual filters for individuation and tracking. One paper [88] has shown that it is possible to learn to segment people in static images using training data acquired through motion segmentation.

Additional modalities

This thesis describes how a robot with a laser rangefinder can learn about objects. Although many of the same processes can work for vision, additional perceptual filters will likely be required to create a useful object ontology. Robot hands allow actions such as grasping to become physically possible and potentially learnable. Using additional modalities, it becomes easier to study how perceptual filters can be autonomously generated. One exciting avenue is to explore how the perceptual filters for the object ontology (and those of the lower levels) can be learned automatically using techniques such as SODA [87].

Foundations for cognition

Knowledge of objects is one of the domains of common-sense knowledge, and is of comparable importance to knowledge of self, space, and other agents. Work on how these other competencies can arise have been developed by other researchers (self [85, 84, 79, 30], space [106, 62], and agents [77]). Although many studies use robots equipped with range-sensors, other sensorimotor modalities have also been explored (vision, sound, touch). If these methods can be integrated in one platform, this should create a rich representation for the robot's experience. Using such a representation, the robot might be able to learn tasks using methods such as supervised learning, reinforcement learning, and programming by demonstration.

Language learning

Several researchers have proposed methods [10, 97, 100, 51, 115, 101] by which an agent can learn a language, or at least words in a language. Much of this language learning research assumes that the learning agent has already discretized its sensory experience into objects, with continuous or discrete object properties. By providing an object ontology, OPAL can be used to support language learning on a physical robot.

Adding Relations and Functions to the ontology

Although not discussed in this work, functions and relations play an important role in an ontology. Spatial relations can be used to represent the relative locations of two objects. Object properties already provide unary relations for objects. There is also the need for functions to provide mappings between the object representations and the sensor and map representations. These functions already exist implicitly in the simulation scans used in planning. However, methods for the robot to learn relations without supervision is still an open research question.

OPAL provides a foundation for a robot to learn about objects. It provides basic mechanisms for learning, and is extensible to support further developmental learning. This is one step towards the goal of achieving human level cognition in an autonomous robot.

Bibliography

[1] D. Angelov, D. Koller, H.-C. Pang, P. Srinivasan, and S. Thrun. Recovering articulated object models from 3D range data. In *Proceedings of the Uncertainty in Artificial Intelligence Conference (UAI)*, 2004.

[2] D. Arbuckle, A. Howard, and M. J. Matarić. Temporal occupancy grids: a method for classifying spatio-temporal properties of the environment. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, pages 409–414, 2002.

[3] C. G. Atkeson and J. C. Santamaria. A comparison of direct and model-based reinforcement learning. In *International Conference on Robotics and Automation*, pages 3557–3564, 1997.

[4] G. P. Baerends and J. P. Kruijt. Stimulus selection. In R. A. Hinde and J. Stevenson-Hinde, editors, *Constraints on Learning: Limitations and Predispositions*, pages 23–50. Academic Press, London, 1973.

[5] R. Baeza-Yates and B. Riebeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999.

[6] M. Becker, E. Kefalea, E. Maël, C. von der Malsburg, M. Pagel, J. Triesch, J. C. Vorbrüggen, R. P. Würtz, and S. Zadel. GripSee: A gesture-controlled robot for object perception and manipulation. *Autonomous Robots*, 6(2):203–221, 1999.

[7] S. Benson. Inductive learning of reactive action models. In *Int. Conf. on Machine Learning*, pages 47–54, 1995.

[8] I. Biederman. *An Invitation to Cognitive Science*, volume Visual Cognition, chapter 4, pages 121–165. MIT Press, second edition, 1995.

[9] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, pages 1014–1019, 2002.

[10] P. Bloom. *How Children learn the meaning of words*. The MIT Press, 2000.

[11] P. Bloom. Précis of How Children Learn the Meanings of Words. *The Behavioral and Brain Sciences*, 24(6):1095–1103, 2001.

[12] M. Bowling, D. Wilkinson, and A. Ghodsi. Subjective mapping. In *Proc. 21st National Conf. on Artificial Intelligence (AAAI-2006)*, pages 1569–1572, 2006.

[13] M. J. Brady and D. Kersten. Bootstrapped learning of novel objects. *Journal of Vision*, 3:413–422, 2003.

[14] J. Braun. Intimate attention. *Nature*, 408(6809):154–155, Nov 9 2000.

[15] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of sound*. The MIT Press, Cambridge, Massachusetts, 1990.

[16] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[17] R. Campbell and J. Krumm. Object recognition for an intelligent room. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 691–697, July 2000.

[18] S. Carey and F. Xu. Infants' knowledge of objects: beyond object files and object tracking. *Cognition*, 80(1–2):179–213, June 2001.

[19] P. Cavanagh, A. T. Labiancaa, and I. M. Thornton. Attention-based visual routines: sprites. *Cognition*, 80(1–2):47–60, June 2001.

[20] H. Chaput. *The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots*. PhD thesis, Dept. of Computing Sciences, University of Texas at Austin, 2004.

[21] A. Chella, M. Frixione, and S. Gaglio. Understanding dynamic scenes. *Artificial Intelligence*, 123(1–2):89–132, 2000.

[22] G. K. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time: A 3D reconstruction algorithm combing shape-from-silhouette with stereo. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2003.

[23] Y. Choe and N. H. Smith. Motion-based autonomous grounding: Inferring external world properties from encoded internal sensory states alone. In *Proc. Nat. Conf. Artificial Intelligence (AAAI-2006)*, 2006.

[24] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, The Robotics Institute, Carnegie Mellon University, 2000.

[25] M. Cooke. *Modelling auditory processing and organisation*. Cambridge University Press, 1994.

[26] S. Coradeschi and A. Saffiotti. Perceptual anchoring of symbols for action. In *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 407–412, 2001.

[27] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. 23rd Annual Conf. on Computer graphics and interactive techniques*, 1996.

[28] N. Davies and M. de L. Brooke. An experimental study of coevolution between the cuckoo *cuculus canorus* and its hosts. Part I. host egg discrimination. *J. of Anim. Ecol.*, 59:207–224, 1989.

[29] R. H. Davies, C. J. Twining, T. F. Cootes, J. C. Waterton, and C. J. Taylor. 3D statistical shape models using direct optimisation of description length. In *Computer Vision - ECCV 2002 Part III*, number 2352 in LNCS, pages 65–81, 2002.

[30] G. L. Drescher. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, 1991.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, second edition, 2001.

[32] P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Computer Vision - ECCV 2002 Part IV*, number 2353 in LNCS, pages 97–112, 2002.

[33] S. Edelman. *Representation and recognition in vision*. MIT Press, Cambridge, MA, 1999.

[34] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 1135–1142. Morgan Kaufmann, 2003.

[35] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, pages 367–392, 2004.

[36] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.

[37] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2003.

[38] S. Finney, N. H. Gardiol, L. P. Kaelbling, and T. Oates. The thing that we tried didn't work very well: Deictic representation in reinforcement learning. In *ICML-2002 Workshop on Development of Representations*, July 2002.

[39] J. R. Flanagan, P. Vetter, R. S. Johansson, and D. M. Wolpert. Prediction precedes control in motor learning. *Current Biology*, 13:146–150, Jan 2003.

[40] W. S. Geisler and R. L. Diehl. A Bayesian approach to the evolution of perceptual and cognitive systems. *Cognitive Science*, 27(3):379–402, May-June 2003.

[41] B. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, June 2003.

[42] Y. Gil. *Acquiring Domain Knowledge for Planning by Experimentation*. PhD thesis, Carnegie Mellon University, 1992.

[43] M. A. Goodale. *Visual Cognition*, volume 2 of *An Invitation to Cognitive Science*, chapter 5: The Cortical Organization of Visual Perception and Visuomotor Control. MIT Press, 1995.

[44] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[45] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760, Apr 2007.

[46] J.-S. Gutmann and C. Schlegel. AMOS: comparison of scan matching approaches for self-localization in indoor environments. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, pages 61–67. IEEE Computer Society Press, 1996.

[47] D. Hähnel, S. Thrun, and W. Burgard. An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 915–920, 2003.

[48] K. Hall and G. B. Schaller. Tool-using behavior of the california sea otter. *Journal of Mammalogy*, 45(2):287–298, May 1964.

[49] S. Hart, R. Grupen, and D. Jensen. A relational representation for procedural task knowledge. In *Proc. 20th National Conf. on Artificial Intelligence (AAAI-2005)*, 2005.

[50] J. J. Hopfield. Olfactory computation and object perception. *Proc. Natl. Acad. Sci. USA*, 88:6462–6466, 1991.

[51] I. Horswill. Tagged behavior-based architectures: Integrating cognition with embodied activity. *IEEE Intelligent Systems*, 16(5):30–38, September/October 2001.

[52] M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[53] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, Sep 1999.

[54] O. C. Jenkins and M. J. Matarić. Deriving actions and behavior primitives from human motion data. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, pages 2551–2556, 2002.

[55] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *Proc. IEEE Computer Vision and Pattern Recognition*, volume I, pages 199–206, 2001.

[56] W. Köhler. *The mentality of apes*. Routledge and Kegan Paul, London, 1925.

[57] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Computer Vision - ECCV 2002 Part III*, number 2352 in LNCS, pages 65–81, 2002.

[58] K. P. Körding and D. M. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427:244–247, 2004.

[59] M. Kubovy and D. V. Valkenburg. Auditory and visual objects. *Cognition*, 80(1–2):97–126, June 2001.

[60] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. 18th National Conf. on Artificial Intelligence (AAAI-2002)*, pages 174–180. AAAI/MIT Press, 2002.

[61] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *IEEE Int. Conf. on Robotics & Automation (ICRA-04)*, 2004.

[62] B. J. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[63] S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[64] S. J. Lederman and R. L. Klatzky. The intelligent hand: An experimental approach to human object recognition and implications for robotics and "AI". *AI Magazine*, 15(1), 1994.

[65] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, 1992.

[66] F.-F. Li, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 1134–1141, 2003.

[67] M. Livingstone and D. Hubel. Segregation of form, color, movement and depth: Anatomy, physiology, and perception. *Science*, 240(4853):740–749, May 6 1988.

[68] J. Mandler. *The Foundations of Mind: Origins of Conceptual Thought*. Oxford University Press, 2004.

[69] D. Mareschal and M. H. Johnson. The "what" and "where" of object representations in infancy. *Cognition*, 88(3):259–276, 2003.

[70] D. Marr. *Vision*. W. H. Freeman, San Francisco, 1982.

[71] D. McFarland. *Animal Behaviour*. Longman Scientific and Technical, second edition, 1993.

[72] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, 1997.

[73] J. Modayil and B. Kuipers. Bootstrap learning for object discovery. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, pages 742–747, 2004.

[74] J. Modayil and B. Kuipers. Autonomous shape model learning for object localization and recognition. In *IEEE International Conference on Robotics and Automation*, pages 2991–2996, 2006.

[75] J. Modayil and B. Kuipers. Autonomous development of a grounded object ontology by a learning robot. In *Proc. 22nd Conf. on Artificial Intelligence (AAAI-2007)*, 2007.

[76] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[77] J. R. Movellan. An Infomax controller for real time detectin fo social contingency. In *Proc. of the 4th IEEE Int. Conf. on Development and Learning*, pages 19–24, 2005.

[78] Y. Munakata, L. R. Stantos, E. S. Spelke, M. D. Hauser, and R. C. O'Reilly. Visual representations in the wild: How rhesus monkeys parse objects. *Journal of Cognitive Neuroscience*, 13(1):44–58, 2001.

[79] L. Natale. *Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping*. PhD thesis, LIRA-Lab, DIST, University of Genoa, Italy, 2004.

[80] M. Nicolescu and G. Medioni. Perceptual grouping from motion cues using tensor voting in 4-D. In *Computer Vision - ECCV 2002 Part III*, number 2352 in LNCS, pages 423–437, 2002.

[81] T. Oates and P. Cohen. Searching for planning operators with context-dependent and probabilistic effects. In *AAAI96*, pages 863–868, 1996.

[82] L. Olsson, C. Nehaniv, and D. Polani. From unknown sensors and actuators to actions grounded in sensorimotor perceptions. *Connection Science*, 18(2):121–144, June 2006.

[83] I. M. Pepperberg. *The Alex studies : cognitive and communicative abilities of grey parrots*. Harvard University Press, 1999.

[84] D. Philipona, J. K. O'Regan, and J.-P. Nadal. Is there something out there? Inferring space from sensorimotor dependencies. *Neural Computation*, 15:2029–2049, 2003.

[85] D. M. Pierce and B. J. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92:169–227, 1997.

[86] J. Provost. *Reinforcement Learning in High-Diameter Environments (to appear)*. PhD thesis, University of Texas at Austin, 2007.

[87] J. Provost, B. Kuipers, and R. Miikkulanien. Developing navigation behavior through self-organizing distinctive state abstraction. *Connection Science*, 18(2):159–172, 2006.

[88] M. G. Ross and L. P. Kaelbling. Learning static object segmentation from motion segmentation. In *Proc. 20th National Conf. on Artificial Intelligence (AAAI-2005)*, 2005.

[89] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2002.

[90] B. Sanders, R. Nelson, and R. Sukthankar. The OD theory of TOD: The use and limits of temporal information for object discovery. In *Proc. 18th National Conf. on Artificial Intelligence (AAAI-2002)*, pages 777–784, 2002.

[91] P. Schiller. Innate constituents of complex responses in primates. *Psychol. Rev.*, 59:177–191, 1952.

[92] B. J. Scholl, Z. W. Pylyshyn, and J. Feldman. What is a visual object? Evidence from target merging in multiple object tracking. *Cognition*, 80(1–2):159–177, June 2001.

[93] L. Shams and C. von der Malsburg. Acquisition of visual shape primitives. *Vision Research*, 42:2105–2122, 2002.

[94] W.-M. Shen and H. A. Simon. Rule creation and rule learning through environment exploration. In *IJCAI89*, pages 675–680, 1989.

[95] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[96] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[97] J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90, 2001.

[98] W. Smith, S. Johnson, and E. Spelke. Motion and edge sensitivity in perception of object unity. *Cognitive Psychology*, 46(1):31–64, 2002.

[99] E. S. Spelke. Principles of object perception. *Cognitive Science*, 14:29–56, 1990.

[100] M. Steedman. Formalizing affordance. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, pages 834–839, Fairfax, VA, August 2002.

[101] L. Steels and F. Kaplan. AIBO's first words : The social learning of language and meaning. *Evolution of Communication*, 4(1), 2001.

[102] A. Stoytchev. Behavior-grounded representation of tool affordances. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3071–3076, 2005.

[103] D. Stronger and P. Stone. Towards autonomous sensor and motor model induction on a mobile robot. *Connection Science*, 18(2):97–119, 2006.

[104] Y. Sun and R. Fisher. Object-based visual attention for computer vision. *Artificial Intelligence*, 146(1):77–123, 2003.

[105] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[106] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[107] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1–3):29–53, 1998.

[108] S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distribution. In *Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000)*, pages 859–865. AAAI Press/The MIT Press, 2000.

[109] R. Triebel and W. Burgard. Improving simultaneous localization and mapping in 3D using global constraints. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005.

[110] J. van Lawick-Goodall. Tool-using in primates and other vertebrates. In D. S. Lehrman, R. Hinde, and E. Shaw, editors, *Advances in the Study of Behavior*, volume 3. Academic Press, New York, 1970.

[111] C.-C. Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *IEEE International Conference on Robotics and Automation*, pages 842–849, 2003.

[112] A. A. S. Weir, J. Chappell, and A. Kacelnik. Shaping of hooks in new Caledonian crows. *Science*, 297(5583):981, Aug 9 2002.

[113] Y. Weiss. *Bayesian motion estimation and segmentation*. PhD thesis, MIT, May 1998.

[114] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[115] C. Yu, D. H. Ballard, and R. N. Aslin. The role of embodied intention in early lexical acquisition. In *Proc. 25th Annual Meeting of the Cognitive Science Society*, 2003.

[116] L. Zettlemoyer, H. Pasula, and L. P. Kaelbling. Learning planning rules in noisy stochastic worlds. In *Proc. 20th National Conf. on Artificial Intelligence (AAAI-2005)*, pages 911–918, 2005.

Vita

Joseph Modayil was born in Edmonton, Canada. He received a B.Sc. and M.Sc. in Mathematics at the University of Alberta. He then moved to the warmer climes of Austin to pursue more practical research on the cognitive development of intelligent robotics under the supervision of Ben Kuipers. He is currently pursuing snow, which he expects to find in Rochester.

Permanent Address: 4505 Avenue A
Austin, TX, 78751

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1]LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.