A Robotic System for Learning Visually-Driven Grasp Planning (Dissertation Proposal)

Marcos Salganicoff
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Recommended Citation


This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/489
For more information, please contact repository@pobox.upenn.edu.
A Robotic System for Learning Visually-Driven Grasp Planning (Dissertation Proposal)

Abstract
We use findings in machine learning, developmental psychology, and neurophysiology to guide a robotic learning system's level of representation both for actions and for percepts. Visually-driven grasping is chosen as the experimental task since it has general applicability and it has been extensively researched from several perspectives. An implementation of a robotic system with a gripper, compliant instrumented wrist, arm and vision is used to test these ideas. Several sensorimotor primitives (vision segmentation and manipulatory reflexes) are implemented in this system and may be thought of as the "innate" perceptual and motor abilities of the system.

Applying empirical learning techniques to real situations brings up such important issues as observation sparsity in high-dimensional spaces, arbitrary underlying functional forms of the reinforcement distribution and robustness to noise in exemplars. The well-established technique of non-parametric projection pursuit regression (PPR) is used to accomplish reinforcement learning by searching for projections of high-dimensional data sets that capture task invariants.

We also pursue the following problem: how can we use human expertise and insight into grasping to train a system to select both appropriate hand preshapes and approaches for a wide variety of objects, and then have it verify and refine its skills through trial and error. To accomplish this learning we propose a new class of Density Adaptive reinforcement learning algorithms. These algorithms use statistical tests to identify possibly "interesting" regions of the attribute space in which the dynamics of the task change. They automatically concentrate the building of high resolution descriptions of the reinforcement in those areas, and build low resolution representations in regions that are either not populated in the given task or are highly uniform in outcome.

Additionally, the use of any learning process generally implies failures along the way. Therefore, the mechanics of the untrained robotic system must be able to tolerate mistakes during learning and not damage itself. We address this by the use of an instrumented, compliant robot wrist that controls impact forces.

Comments
A Robotic System for Learning Visually-Driven Grasp Planning

MS-CIS-92-27
GRASP LAB 308

Marcos Salganicoff

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

March 1992
A Robotic System for Learning Visually-Driven Grasp Planning

Marcos Salganicoff

March 22, 1992

Abstract

We use findings in machine learning, developmental psychology, and neurophysiology to guide a robotic learning system’s level of representation both for actions and for percepts. Visually-driven grasping is chosen as the experimental task since it has general applicability and it has been extensively researched from several perspectives. An implementation of a robotic system with a gripper, compliant instrumented wrist, arm and vision is used to test these ideas. Several sensorimotor primitives (vision segmentation and manipulatory reflexes) are implemented in this system and may be thought of as the “innate” perceptual and motor abilities of the system.

Applying empirical learning techniques to real situations brings up such important issues as observation sparsity in high-dimensional spaces, arbitrary underlying functional forms of the reinforcement distribution and robustness to noise in exemplars. The well-established technique of non-parametric projection pursuit regression (PPR) is used to accomplish reinforcement learning by searching for projections of high-dimensional data sets that capture task invariants.

We also pursue the following problem: how can we use human expertise and insight into grasping to train a system to select both appropriate hand preshapes and approaches for a wide variety of objects, and then have it verify and refine its skills through trial and error. To accomplish this learning we propose a new class of Density Adaptive reinforcement learning algorithms. These algorithms use statistical tests to identify possibly “interesting” regions of the attribute space in which the dynamics of the task change. They automatically concentrate the building of high resolution descriptions of the reinforcement in those areas, and build low resolution representations in regions that are either not populated in the given task or are highly uniform in outcome.

Additionally, the use of any learning process generally implies failures along the way. Therefore, the mechanics of the untrained robotic system must be able to tolerate mistakes during learning and not damage itself. We address this by the use of an instrumented, compliant robot wrist that controls impact forces.
## Contents

1 Introduction .......................................................... 1
   1.1 Visually-Guided Grasping ........................................ 1
   1.2 Learning Reactive Grasp Planning Strategies .................. 1
   1.3 Research Issues .................................................. 3
   1.4 Guide to the Proposal ............................................ 3

2 Developmental Theories of Human and Primate Visually-Guided Grasping 7
   2.1 Innate vs. Learned Knowledge .................................. 7
   2.2 Anatomical Substrate For Voluntary Reaching Behavior ....... 7
      2.2.1 Architecture of Motor Cortex as It Relates to Motor Behavior ... 8
      2.2.2 Descending Control of Reflex Pathways: Hierarchies of Control ... 9
      2.2.3 Proprioceptive Inputs to the Motor Cortex ................ 9
      2.2.4 Distal and Proximal Control of Musculature ................ 9
   2.3 Development of Basic Reaching and Grasping Behavior ... 10
   2.4 Visual Determinants in Early Reaching ......................... 10
   2.5 Control Modes for Reaching .................................... 11
   2.6 Developmental Time Course of Reaching ....................... 11
      2.6.1 Early Reaching Behaviors ................................ 11
      2.6.2 Progression of Control Modes ............................ 12
      2.6.3 Early Proprioceptive Control ............................. 12
      2.6.4 The Intermediate Visual-Visual Level .................... 13
      2.6.5 The Onset of Preshaping and Wrist Orienting Behaviors ... 13
      2.6.6 The Return of Kinesthetic Control and Onset Independent Manual Control .......... 14
      2.6.7 Reafference, Self Consistency and Its Role in the Development of Reaching .......... 14
   2.7 Conclusions ...................................................... 15

3 Related Work in Machine Learning and Robotics 17
   3.1 What is Machine Learning? ..................................... 17
   3.2 Types of Learning Processes ................................... 17
   3.3 Empirical-Model Building ...................................... 19
   3.4 Concept Learning ................................................ 19
   3.5 State-Based Learners .......................................... 21
      3.5.1 Building Models of the World ............................. 22
   3.6 Modelling with Statistical Regression ........................ 23
      3.6.2 Building Models of Reinforcement ......................... 25
   3.7 Induction of Decision Trees .................................. 26
   3.8 Learning by Instruction ...................................... 28
   3.9 Rule-Based Grasp Selection ................................... 28
   3.10 Explanation-Based-Learning Approaches ...................... 29
   3.11 Model-Based Recognition .................................... 29
   3.12 Classification-Based Approaches .............................. 30
   3.13 Neuromorphic Architectures for Sensorimotor Learning ...... 30
6 Contributions

A The Capture Probability of the Penn Hand

B Shape Extraction Hardware and Software
   B.1 Mobile Laser Ranger .................................. 101
   B.2 Vision Processing ........................................ 103
   B.3 The Instrumented Compliant Wrist ..................... 104
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A graph specifying the sequence of states and transitions rules $T_{ij}$ for an object retrieval task.</td>
</tr>
<tr>
<td>2</td>
<td>The sensorimotor perception action mapping function.</td>
</tr>
<tr>
<td>3</td>
<td>A schematic of a hypothetical simplified task for learning hand position selection based on object position. $O_x$ refers to the position of the object along the x-axis and $H_x$ refers to the position of the hand.</td>
</tr>
<tr>
<td>4</td>
<td>(a) A hypothetical raw scatter plot of reinforcement values obtained by executing an action with different perceptual states of affairs. Each point in the plot has a reinforcement mass associated with it determined by the degree of success of the action. (b) A hypothetical contour plot of a non-parametric regression fit to measured data that attempts to smooth and predict the reinforcement mass density over the entire domain of input parameters.</td>
</tr>
<tr>
<td>5</td>
<td>(a) The quadtree ($2^2$-tree) reduction of the regression surface. (b) Execution-time indexing and retrieval of domain knowledge after learning. The perceptual parameter, the observed location of the object is mapped via an associative lookup to an allowable interval that determines where the robotic hand is to be placed.</td>
</tr>
<tr>
<td>6</td>
<td>Binary Tree for indexing perceptions and actions. The tree is of non-uniform depth, determined by the distribution of reinforcement in the parameter space. Each node has a set of intervals associated that index the leaves beneath it.</td>
</tr>
<tr>
<td>7</td>
<td>Adaptive $2^k$ Generation. (a) Shows a closed unit circle function, whose output is 1 everywhere inside and 0 outside, along with the resulting quadtree $2^2$-tree generated. (b) Shows the points where the function was evaluated during the course of the tree creation. The process is uncertainty seeking, evaluating the function with higher density of samples along the decision boundary, and less so in highly uniform outcome areas.</td>
</tr>
<tr>
<td>8</td>
<td>A sample fit on a four dimensional Gaussian function in the $x_2, y_2$ space. (a) The underlying function plotted with $x_1 = .26$ and $y_1 = .1$. (b) The resulting fit using LOESS locally-weighted regression. Here $f = .5$ says that half of the observations are used in the local query estimate. (c) The same function plotted at $x_1 = .6$ and $y_1 = .6$. (d) The resulting fit using projection pursuit regress (SMART). It can be seen that the projection pursuit method performs much better than the locally weighted approach. Values of $f$ were varied widely in testing the LOESS fit with no appreciable increase in performance. The data input consisted of 400 example points in the unit cube derived from 20 uniformly distributed sampling regions with 20 random points in each region. Each sampling region has .2 dimension.</td>
</tr>
<tr>
<td>9</td>
<td>Schematic of the planar object retrieval task from the top camera view. Here, the object frame $(O_x, O_y)$, the hand frame $(H_x, H_y)$, and the no contact/contact $(R = 0, 1)$ reinforcement values are stored for each trial grasp.</td>
</tr>
<tr>
<td>10</td>
<td>The three phases of approach, contact and reflex initiation.</td>
</tr>
</tbody>
</table>
The initiation of the grasp reflex. A relative cartesian deflection or equivalent-angle axis rotation of magnitude above the given threshold value is detected by the wrist and causes the arm to stop. The hand then immediately closes.

The Experimental System. The PUMA560, Penn Hand and Wrist are controlled and coordinated using a MicroVAXII with shared memory. The MicroVAXII sends commands via a serial link to a high-level controller which interprets commands and servos the hand configuration to a desired force or position. The CCD camera output is digitized on the MicroVAXII and processed on the SUN4/260 via an Ethernet connection. The only real-time sensitive component of the system is the connection between the wrist and arm which occurs via the shared memory connection.

Histogram of centered data from 303 trial grasps in the workspace. It depicts the raw data centered at the perceived centroid of the target object and is the proportion of success to total trials summed over 50mm regions of the workspace. Notice that since the hand orientation was fixed for all trials, the histogram outlines the shape of the hand in the spherical grasp configuration.

Resulting classification based on $E(R \mid O_x, O_y, H_x, H_y)$ for objects positioned at (a) $(O_x, O_y) = (-600 \text{ mm}, 0 \text{ mm})$, (b) $(O_x, O_y) = (800 \text{ mm}, 600 \text{ mm})$, (c) $(O_x, O_y) = (400 \text{ mm}, -800 \text{ mm})$ and (d) $(O_x, O_y) = (-1000 \text{ mm}, -800 \text{ mm})$ relative to the robot base. Each density plot in the $H_x, H_y$ space represents the resulting classification where white represents an expected tactile stimulus and black represents a miss in robot-centered coordinates. From this, it can be seen that a correct decision rule for placing the hand has been induced since the spot tracks the object location. The results generalized to all positions in the plane.

(a) The theoretical greater than .5 contact probability region for the tactile stimulus, (b) The learned region for an object at (0mm,0mm) and (c) The learned region for an object at (300mm,200mm). Units are in millimeters. The graphs are in object-centered coordinates.

System Architecture

The world coordinate description (translation and rotation) of the superquadric along with the object centered description and effect of the shape parameters.

The x and y axes are free in case (a) and constrained in case (b).

Generating the basic-level action categories.

The basic-level categories.

The binding functions for each basic-level category and their arguments. Each action binders maintains a function that binds the observed object extents to its free variables that are listed within its description.

The parameters for the approach depth (side view).

The parameters for the approach height (side view).
The process of selecting a grasp. The superquadric based vision system determines that the object is in the $z_{up}$ pose. The category recognition level then selects which of the six possible grasp/approach directions are feasible based on the category learning. The action parameters, e.g. the width of the gripper preshape, are then bound based on the corresponding learned binding functions and the perceived object extents.

The $k$-$D$-tree tessellation of a data distribution. The distribution consists of 1000 points. Of those, 400 are Gaussian distributed, centered at $(25,25)$ $\sigma = 10$, 400 from the same distribution centered at $(75,75)$ and 200 are uniformly distributed. The leaf size of the $k$-$D$-tree was set to 10. It can be seen that the size of the leaves decreases as the local density of samples increases, yielding an adaptive resolution property.

The upper ($p_+$) and lower bound ($p_-$) probability estimates for three fixed outcome ratios and confidence bound $\alpha = .5$. The three ratios plotted are, 0, .5, and 1. for n observation. The upper part of each component is $p_+$ and the lower part $p_-$. It can be seen that the estimates approach their asymptotic values rapidly as a function of sample size $n$.

The selective forgetting mechanism. Each observation has a weight associated with it. If it is one of the $m$ nearest-neighbors, then its weight is updated by $w_{k,n+1} = \gamma_k w_{k,n}$. When its weight decrease below a cutoff value, the observation is deleted from the learning database. The decay rate is a function of $\tau$ which is the forgetting rate. The smaller $\tau$, the fewer subsequent observations necessary in the neighborhood of an observation before it is deleted.

The influence function for decaying observation in the neighborhood of previous observations. The $\gamma_k$, is used as a forgetting coefficient for the $k$th nearest neighbor to the new observation which is centered at 0. It is a function of the scale parameter $d^2_X$ at which $\gamma$ is unity, $\tau$ which is the forgetting rate, and the distance from this $k$ nearest neighbor to the new point. The scale parameter is adapted to the local density of points.

The proposed experimental system. The overhead mounted camera identifies the approximate object location, the scanner then generates a depth map of the object. Vision processing is done by the SUN Sparc/IPX. The compliant wrist, hand and robot are also directly coupled into the Sparc/IPX for rapid interaction.

The three representative regions used to compute the capture probabilities. 

(a) The radial probability of contact for the Penn hand with a 7cm diameter object, (b) the density plot of the same function. 

Object and Hand Frames

A merged range image.

The reduced superquadric representation of the objects.
List of Tables

1  The proposed sequence of tasks and representations. $H_x, H_y$ are the $x, y$ position of the hand in the plane, relative the robot coordinate frame. $O_x, O_y$ are the object positions, relative to the image frame. $H_\theta$ and $O_\theta$ are the orientations of the major axis of the hand and object respectively . . . . . . . 41
2  The Adaptive $2^k$-tree Creation Algorithm ........................................ 50
3  The Flow of the Learning Algorithm .................................................. 56
4  The Categorical Vision-Guided Grasping Problem ............................... 70
5  Symbolic Action Attributes and Associated Legal Attribute Values........ 76
6  Symbolic Object Description Attributes and Associated Attribute Values. . 76
7  The scales chosen for generating objects, legal objects consist of any com-
    bination where $a_x \leq a_y \leq a_z$. The scale factors represent the length factor
    of each axis relative to the absolute size size, which is chosen at 1,2 and 4
    inches for a total of 30 possible objects. ........................................ 82
8  The Density Adaptive Reinforcement Learning (DARLING) Algorithm . . 91
9  Adaptive $2^k$-tree Construction Algorithm ....................................... 91
1 Introduction

1.1 Visually-Guided Grasping

We study the acquisition of visually-driven grasping strategies by robots using both supervised and unsupervised learning methods. Visually-driven grasping entails the selection of hand approach directions and preshapes that will allow us to effectively grasp an object using visual information. This problem is of fundamental importance since it is a prerequisite for a wide spectrum of tasks that humans routinely perform, many of which we are interested in having robots perform as well. This includes everything from assembly and disassembly of complex mechanical systems to picking up litter to the recovery of spent uranium fuel rods on the bottom of a defunct reactor vessel.

The visually-guided grasp planning problem can be defined as follows: given the position, orientation and shape of a target object, select some set of feasible actions for the hand/arm system that are sufficient to pick up the object, or determine that there are no feasible actions for that situation. The actions are described in terms of posture and a pose for the hand/arm [Liu et al., 1989]. The posture is generally defined in terms of preshape that determines how many fingers are involved in the grasp and the aperture, or distance between the fingertips. The pose is determined in terms the location where the fingers will contact the object and the orientation of the hand for approaching the object. This problem is different from the control problem for grasping which involves selecting appropriate control actions for maintaining a stable grasp based on sensory information once the object has been contacted by the hand.

1.2 Learning Reactive Grasp Planning Strategies

We propose that there are two computational processes that determine how to prehend an object based on visual input. The first one is action oriented (or reactive) while the second is deliberative and involves high-level reasoning. The action-oriented approach is attempted first, and upon sufficient number of failures, the deliberative one is invoked to attempt reasoning in order to generate new approaches to the task. In this work, we will concentrate on developing the action-oriented approach.

The overall choice for the action-oriented approach is based in part on the observations of a number of workers in reactive planning [Schoppers, 1987; Firby, 1987; Chapman and
Agre, 1986; Agre and Chapman, 1987] and more recently in reinforcement learning approaches [Whitehead, 1989; Whitehead and Ballard, 1991; Kaelbling, 1990; Lin, 1992] that strive to make reactive planning systems more adaptive. In particular, it has been noted that there are essentially two types of planning at work in our everyday interactions with the world [Agre and Chapman, 1987]. The first type of planning is highly localized in terms of space and time, and involves making split-second decisions based on this local information. The other type of planning is more deliberative and involves representation of objects or phenomenon that may not be immediately observable, and therefore involves searching through the space of potential worlds, which is a much more time-consuming endeavor. The deliberative approach is therefore less likely to be timely in a rapidly changing situation.

Consider a “deliberative” methodology that must generate grasps for a given tool by reasoning about forces and possible grasps. We find that a significant amount of information about an object’s geometry, friction characteristics, mass distribution and the hand characteristics are necessary for use with a well developed theory of grasping. For example, Cutkosky [Cutkosky, 1989] has cited numerous measures that have been developed in the literature such as compliances, connectivity, force closure, form closure, grasp isotropy, internal forces, manipulability, resistance to slipping and stability. In general, using the these measures (or domain theories) can be quite computationally intensive and it is not clear which of the criteria forwarded is relevant for the object being grasped, especially if we take the usage of the object in the task into account [Cutkosky, 1989].

On the other hand, it is probably safe to say that we generally do not compute such a detailed model ourselves when we pick up the majority of objects we encounter. If the task is one that has been practiced many times before, then reactive strategies can be rapidly invoked by being “looked up” in an associative memory to determine which grasps can be used based on the results on previous interactions in the world. Such an approach is also useful since it provides a rapid computational approach for the bulk of everyday situations, most of which have relatively moderate constraints. If these previous experiences are rapidly indexable based on object shape and context, then grasp preshapes and approaches can be generated quickly with practically no effortful cognitive processing. This is seen to be the case after a skilled task has been practiced to the point that it is mastered and has very low reaction time [Fitts, 1964]. Therefore, rather than take a complex model-based approach to grasping, we choose to focus on the possibilities afforded by a reactive grasp generator
that learns by experience and instruction.

1.3 Research Issues

In this thesis we will be interested in several issues relating to learning the visually guided grasping task. The issues we investigate are:

1. The construction of robot learning systems that can learn to grasp based either in a supervised or unsupervised manner by attempting grasping tasks and accommodating their knowledge as necessary to function successfully within their own mechanical and perceptual limitations.

2. The development of reinforcement learning algorithms that can work in high-dimensional spaces with relatively few exemplars. These algorithms support the research by providing computational models for learning.

3. What are feasible mechanisms for forgetting exemplars in order to make the learning adaptive to changes in the behavior of the environment.

4. What might be the developmental mechanisms at work in human and primate infants that are used to bootstrap and facilitate learning in a high-dimensional space that constitutes the perceptual and motor world? From this follows immediately the question of what is innate and what is learned in these biological systems and how this changes during development. Answering these questions can give us some insights into how we can structure robotic learning to increase its effectiveness.

5. What are appropriate innate developmental schedules and task requirement progression that make learning favorable, and how might we automate the transition to different stages in the progression.

6. What is the appropriate set of perception and action representations?

1.4 Guide to the Proposal

This proposal tackles each of these issues in turn. Section 2 gives an overview of salient literature in developmental psychology and neurophysiology relating to motor control and development in visually-guided grasping. This section provides the background about learning
schedules and mechanisms that are pertinent for the development of a competent reactive-grasping system. While the research results reviewed are not, for the most part, directly applicable to robotic learning systems, they do provide evidence for the gradual refinement in representation of perception and action that motivates the approach of "perceptual affordances" for learning taken in section 4.

Section 3 gives an overview of machine learning techniques and previous work in systems for acquiring world models in robotic applications. A general definition of different types of machine learning is given along with a summary of various approaches to learning in robotic domains that provide a foundation for the approaches taken in sections 4 and 5. Several critical issues relating to learning in continuous domains are also discussed.

Section 4 discusses the learning approach for the first problem of sensorimotor bootstrapping the visually-guided grasping task using a new reinforcement learning algorithm and perception/action binding representation. This formulation for reinforcement learning is novel in several respects. Reinforcement learning in real-valued domain is formulated as a multivariate non-parametric regression problem. The algorithm uses projection pursuit regression [Friedman and Stuetzle, 1981] to work in parameter spaces using far fewer learning samples than would be necessary with other learning algorithms. A new action-map representation is developed to allow the regression function's peaks to be rapidly indexed and used to link percepts to actions for execution. Initial results of this approach are discussed and a set of additional experiments are proposed. The experimental sequence embodies principles from section 2 along with Gibson's [Gibson, 1969] theory of incremental development and affordances.

Section 5 describes a two-level execution architecture for generating feasible grasps and a combination supervised and unsupervised learning regime. The architecture moves up to a higher level of abstraction than section 4 by building on the approach of non-parametric regression. User specified basic-level action categories for preshape are defined at a high level. Parameter binding functions for the selected action category determine the actual instance values for the abstract actions based on the geometry and pose of the object. The learning approach is a combination of supervised instruction to generate classification and binding functions followed by unsupervised verification and adaptation mechanisms that specialize these initial functions to the limitations of the robotic hardware. A new density adaptive k-D tree-based reinforcement learning algorithm is proposed for learning
to recognize category membership based on input real-valued perceptual attributes. The density-adaptive approach automatically controls the resolution of the description of the distribution of task success and failure based on local sample density and estimates of the non-determinism in that region. The non-parametric regression techniques employed in section 4 are used for learning the parameter-binding functions for each corresponding category.

Finally, section 6 focuses on the significant contributions of the proposed work, both terms of new learning algorithms and paradigms and to the field of task-based robotic learning.
2 Developmental Theories of Human and Primate Visually-Guided Grasping

2.1 Innate vs. Learned Knowledge

Exploring what is innate and what can be learned in the context of sensorimotor development, especially as it relates to visually-driven grasping, enables us to understand much about how to structure the machine perception/action learning process.

An important question is to what level must we give a-priori structure to the task in order for it to be learnable. This is the age-old philosophical question of nativism, or what is innate, and empiricism, which looks for a mechanism by which the innate abilities are used by the infant to learn from its environment. Both give important insights for both human and machine learning, and evidence seems to support a synthesis of these two points of view.

2.2 Anatomical Substrate For Voluntary Reaching Behavior

The ability to use the hand for manipulation is a hallmark of the human. It is this ability that allows us to interact with, combine, and form raw materials in the environment into tools and end-products that we need. It is not surprising that, as such, the neural system that controls the motor system in a normal adult is a highly developed and hierarchical system, able to react quickly to sensory events during the course of interaction. Since, ultimately, the neural system limits the effectiveness of the musco-skeletal apparatus, it is worthwhile to describe some of its pertinent organizational attributes for motor control.

Voluntary reaching is structured into functionally distinct subphases that may be studied in an independent fashion. The research community seems to have agreed that grasping an object involves several phases [Jeannerod, 1988]. First, the target object must be identified. Secondarily, the spatial coordinates of the object relative the torso frame of the agent must be reliably determined since they describe the position and orientation of the object. Thirdly, the necessary trajectory to achieve this goal state must be formulated and generated on the fly and/or indexed from memory. The process of visually-guided grasping can be broken into the transport and the manipulation phase [Jeannerod, 1988]. The transport phase moves the hand close to the target object and is followed by the manipulation phase where mechanical interaction with the object occurs. Finally, the temporal sequencing and
correct activation levels to control the muscles in order for the given trajectory must be computed and invoked in a timely and accurate manner and the hand must be preshaped appropriately. In general, the velocity profile for the transport phase is bell-shaped and coordinated with the opening of the hand, so that the fingertip distance reaches its maximum just as the hand nears the object [Jeannerod, 1988].

2.2.1 Architecture of Motor Cortex as It Relates to Motor Behavior

The architecture of the motor cortex is extremely intricate and only partially understood at this time. Many areas of the brain are involved the generation of voluntary motor behavior. There is an hierarchical organization, from the neocortex to the basal ganglia to the cerebellum and finally to the spinal cord. Each one of these performs a specific function in the control loop that ultimately allows the complicated coordination and interplay between the phases of the reach and grasp.

The most well known of these areas is the primary motor cortex, or area 4. This area has long been known to elicit motor behavior in animals and in humans. Jackson [Jackson, 1931] first noticed that certain epileptic seizures began with activation of certain muscles and spread topographically from those regions, and from this was able to infer that a localized area of the brain was involved with the control of motor activity and that it would be topographically organized. This was followed up by several electrophysiological studies, culminating in Penfield’s [Penfield and Rasmussen, 1950] extensive electrophysiological mapping studies of the motor cortex in human subjects prior to neurosurgical procedures. Penfield’s work confirmed that the representation of muscles in the body is represented in an orderly topographic fashion in the cortex. Subsequently, topographic representations have been found to be prevalent in somatosensory, visual and auditory.

Recent investigations into the detailed architecture of the primary motor cortex have shown several interesting properties. This first property is that the cortex is organized into radial arrays of sub-units, named cortical afferent zones, that control individual muscles [Asanuma, 1967]. Stimulation of some zones produces sustained contraction in a given muscle, while stimulation of other zones may produce inhibition in these same muscles. Interestingly, this columnar type of architecture is similar to the organization of the somatosensory cortical area that is also organized into columnar receptive fields which topographically map certain areas of skin, as well as the visual system. Stimulation to a single
unit in the motor cortex may influence several muscles due to collateral axons indirectly projecting to other muscles. A given muscle may also be represented multiple times in the motor cortex; there is not a one to one mapping between muscles and efferent zones.

2.2.2 Descending Control of Reflex Pathways: Hierarchies of Control

Lundberg [Lundberg, 1979] has shown that descending motor connections use the same interneurons that have projections that enhance or inhibit spinal cord low-level reflexes. This property is very useful since it permits conscious goal-directed motor Behavior to mediate low-level reflexes, depending on the task. This hierarchical scheme allows the control of complicated automatic motor programs to occur in a timely fashion, since reflexes have a short reaction-time compared to conscious decisions. Examples of this are seen in locomotion and other quasi-voluntary skills that are learned. Quasi-voluntary skills are initiated voluntarily, but do not require conscious-level supervision. If they did, they would be unstable because of the long round-trip latencies involved in transmitting the somatosensory information from the joint and muscle information sensors to the neocortex and then back to the muscles. Examples of these type of skills include walking, stereotyped grasping, the playing of musical instruments, and typing at a high skill level.

2.2.3 Proprioceptive Inputs to the Motor Cortex

Neurons in the motor cortex also receive inputs as to the status of various actuators as well as cutaneous sensors. Some neurons respond to tactile stimuli, some to joint rotations and others to muscle stretch. The receptive fields of these units exhibit “local sign” [Asanuma and Sakata, 1973]; their input generally comes from muscles they project to, or from cutaneous regions close to the muscle controlled by those neurons. The computational significance of this is not completely understood. In general, local sign is such that an undesirable somatosensory stimulus, such as from a pain receptor, maps to a set of motor columns that stimulate a muscle groups that retract the limb from the noxious stimulus.

2.2.4 Distal and Proximal Control of Musculature

An extremely important division of labor that occurs in the neural architecture is the parceling of motor control into two anatomical structures, the proximal and distal systems. The proximal system controls the posture of the organism and maintains it a region of operation
that facilitates the finer motions necessary to perform more dexterous manipulation. The dexterous manipulation is controlled by the distal system.

Cortical ablation studies by Lawrence and Kuypers [Lawrence and Kuypers, 1968a; Lawrence and Kuypers, 1968b] have shown that distal movements of the hands, such as individual digit movements, are driven mostly by higher-level cortical systems and the lateral pathways, whereas proximal control such as control of movement of the upper arms and torso, is mediated mostly by the brainstem and the ventro-medial pathways.

Anatomical studies have underscored that anatomical growth is an component of manipulation development. Lawrence and Hopkins [Lawrence and Hopkins, 1972] note that the physiological development of these pathways in the rhesus monkey is in step with the emergence of dexterous hand behavior. The ventro-medial and lateral pathways do not differentiate from the rest of descending motor pathways until about 8 months of age, at which point the monkey begins to use the index finger and opposing thumb to grab food morsels. Before this point grasping is in a crude and stereotyped fashion, with all fingers opening and closing in unison. Therefore, any practice in grasping to exercise fine control of the fingers is fruitless, since no system is in place to profit from such interactions.

2.3 Development of Basic Reaching and Grasping Behavior

The sensorimotor development process of the human infants provides an unsurpassed example of a system that rapidly learns to adapt to an unfamiliar environment. This stage also allows for learning processes occurring in the human to be observed in a more unfettered way since thought processes are directly and physically manifested. In later stages of learning, higher levels of abstraction are involved and a more sophisticated fabric of background cognitive abilities are in place, which may confound the observations [Drescher, 1989].

There is a large body of literature describing and analyzing the developmental course of grasping skills in humans throughout early childhood. Excellent detailed longitudinal descriptions may be found in Von Hofsten [Hofsten, 1986] and Diamond [Diamond, 1990] and this synopsis follows their analysis for the most part.

2.4 Visual Determinants in Early Reaching

The grating acuity of very young infants is approximately 1 cycle/degree, whereas the spatial frequency acuities in normal adults is on the order of 30/60 cycles per degree of visual field
This increase in visual acuity seems to be highly correlated with the development of neuroanatomical structures that receive input from the foveal areas of the retina. The ability to process binocular disparity develops between three and five months of age [Held et al., 1980]. This relatively high-resolution metrical information becomes available just at the onset of tuning between visual and motor maps. This is another case where anatomical development seems to limit the development of the motor behavior.

2.5 Control Modes for Reaching

Grasping and reaching in mature infants and adults seems to use two distinct modes of control of arm movements [Hofsten, 1986]. They are visual-visual and visual-kinesthetic. In visual-visual reaching, the perceived hand-object distance is used to progressively decrease the distance between the object and hand. White et al. [White et al., 1964] observed that infants tended to repeatedly fixate on the hand and target during a given grasp attempt. In visual-kinesthetic reaching the direction and distance from the body-centered frame is used along with proprioceptive inputs to control the movement. This type of reach is generally called the ballistic reach, because it is essentially an open-loop motion conforming to the strict definition of a motor program.

2.6 Developmental Time Course of Reaching

Von Hofsten et al. [Hofsten, 1982; Hofsten, 1986; Hofsten and Fazel-Zandy, 1984] have performed numerous experiments that serve to document the longitudinal development of motor-behavioral components in infant grasping. The work seems to lead to several conclusions. First, the visually initiated pre-reach is innate in the infant [Hofsten, 1982] and is observable as early as five days after birth. Secondly, there is evidence for a progression from visual-proprioceptive control to visual-visual and then back to visual-proprioceptive control at the end of initial development.

2.6.1 Early Reaching Behaviors

Von Hofsten [Hofsten, 1982] demonstrated the early onset of visual grasping in a study in which infants were seated in a reclining chair and a moving multi-colored yarn ball was suspended within their reach. Two orthogonally-mounted cameras were used to videotape the infant’s activities. This permitted the three-dimensional location of the infant’s hand,
target and visual fixation to be tracked during the course of a given reaching trial. The results showed reaching behavior was driven by visual fixation. The direction of reach was keyed exclusively to the object's location and not the direction of the head. Bower [Bower et al., 1970] has demonstrated reaching behavior in very young infants as well.

2.6.2 Progression of Control Modes

Now, we focus on evidence supporting the progression from visual-proprioceptive control of grasping to visual-visual and then back to visual-proprioceptive.

2.6.3 Early Proprioceptive Control

Both Bower [Bower, 1982] and Von Hofsten [Hofsten, 1982] seem to provide evidence that early hand motions are visuo-kinesthetically controlled. Von Hofsten notes [Hofsten, 1982] that in general, infants less than two months of age seem to open their hands before or very early during the initiation of the ballistic phase of the reach. Additionally, preshape occurs independent of whether the reach is visually initiated or not. This implies the hand opening behavior is not independent from reaching at that age.

Bower's research seems to concur with this view [Bower, 1982]. Bower investigated the delay in grasping after the ballistic arm movement that brings the hand close to the desired object. In newborns this delay averaged about 400 msec, whereas in twenty week old infants, the delay was closer to 800 msec. Bower argues that this is due to the development of sub-units of behavior. The act of reaching an object begins to be decomposed into the transport and manipulation phases characteristic of mature reaching. In newborn infants the whole act of reaching is "atomized" and the necessary delineation of motor behavior components is missing. Grasp does not exist as a separate process from reach and they are forced to occur coincidentally.

Another important factor in early reaching is disinhibition of reflexes. Diamond [Diamond, 1990] has shown that reflexes play an important role in early reaching, but this lack of inhibition may be disruptive during later stages. Diamond cites evidence by Twitchell [Twitchell, 1970] that indicates that the traction (grasp) reflex can disrupt a ballistic trajectory if a non-target object is accidently contacted during the approach phase during ages 5-7 months. This unintended contact leads to a reflexive grasp or the triggering of the avoidance reaction where the infant reflexively pulls back the hand. In either case, the attempt leads to a fail-
ure. This undesirable tendency begins to decrease after 7 months which correlates closely with the growth of neocortical structures that begin to inhibit the reflexes [Diamond, 1990]. This is yet another example of neural-growth determining motor development.

This avoidance reflex view is further substantiated by Bower's [Bower, 1982] analysis of botched grasp attempts by newborns. Upon failure, the entire grasp attempted is repeated, rather than trying to correct the attempt while it is underway. The reattempt consists of removing the hand from the field of view and restarting, rather than correcting only the failed component.

2.6.4 The Intermediate Visual-Visual Level

Bower [Bower, 1982] has shown that in twenty week old infants, the process of the visual servo is in place, and if the infant does not initially succeed in grasping, then visually-guided corrections occurs within the attempt.

According to Von Hofsten [Hofsten, 1984] at around 2 months of age the hand and arm begin to develop independent control mechanisms. The hand is now closed during the approach and sometimes erroneously before contact occurs. This is in contrast to the previous “atomic” reach. After this brief period, the hand begins to open again, but only during visual fixation on the target. At the age of four months, the two phases of ballistic and visually guided approach are poorly integrated [Hofsten and Lindhagen, 1979]. Both of components tend to be of about equal duration. This period of heavy dependence on visual feedback lasts from about 2 months to 6 months at which time on kinesthetic feedback begins to predominate once again [Hofsten, 1986].

2.6.5 The Onset of Preshaping and Wrist Orienting Behaviors

Up until 3 1/2 months the infants manual contact with the target object rarely leads to successful grasping [Hofsten and Lindhagen, 1979]. However, shortly thereafter, at 4 1/2 months, grasping becomes highly effective, and targets are grasped with good reliability.

Much knowledge about object properties can be demonstrated by infants very early in development. Visually controlled adjustments of hand orientation and preshape have been observed by Fazel-Zandy [Hoftsten and Fazel-Zandy, 1984] at an age of 18 weeks. Fazel-Zandy presented infants with vertical and horizontal rod grasp targets; orientation of the hand during the last 540 msec of the approach to the object was measured. Even
at the age of 18 weeks, correct orientation relative to the target was observed although
the adjustments were often not completely followed through. The ability increased rapidly
thereafter. Thus it is reasonable to assume that the coordination of ballistic approach,
preshape and orientation is available early on, but must be calibrated via experience.

2.6.6 The Return of Kinesthetic Control and Onset Independent Manual Control

After four months, the ballistic phase rapidly begins to subsume more and more of the
reaching process. At six months, most reaches consist of two movements, consisting of
the ballistic transport and manipulation phases. As infants grow older, they become less
dependent on the visual following of the hand and reaching becomes much more automatic.
When the hand is obstructed during the reach it has less of an impact than in younger
infants [Bushnell, 1985]. For smaller targets where fine manipulation is necessary, the lack
of development of the neocortex becomes the limiting factor, and it is notable that fine
grasping with individual digits is not attained until approximately nine months of age,
which is consistent with the observations of section 2.2.4 on neural development.

The use of open-loop control of movement would imply the involvement and maturation
of neocerebellar and basal ganglia structures is the brain. The poor positioning accuracy
and instability of positional control of patients with cerebellar deficits supports the role of
the cerebellum in fine tuning of open loop learned movements. Patients with diseases of the
basal ganglia are sometimes unable to use feed-forward control at all and must revert to
visually guided positioning of limbs, where they use a series of small movements to gradually
approach the desired position. They overshoot since they rely on higher cortical centers to
compute error and send the compensating commands. These higher round trip and decision
times involved invariably leads to overshooting the goal.

2.6.7 Reafference, Self Consistency and Its Role in the Development of Reaching

Jean Piaget [Piaget, 1952] first proposed the concept of the "circular reaction" in which
intentional movements leads to a percept (either proprioceptive or exteroceptive) which is
processed by the organism and gradually by interaction with the environment, a representa-
tion and model of the external world is built up.
Piaget introduced the concept of the circular reaction loop as a process whereby successful motor strategies can be formed. Operationally, this loop consists of two components: the generation of random motor behaviors (i.e. babbling) whose outcomes are sampled with visual, kinesthetic and auditory perceptual systems and the other component consisting of the generation of a map between the motor activations and the resulting percepts. This correlation map, when fully defined permits the association and indexing of the appropriate sequence of motor activations for percepts.

Held [Held and Bauer, 1970] couches the idea of the circular reaction in more concrete terms, namely in process of reafference of intentional movements. In a series of experiments, he has shown that without active exploration and observation of the environment, the mapping between percepts and motor activation is improperly created. This is shown convincingly in early in development as well as later in life when visual remapping is forced by the use of prismatic goggles. Thus, a reasonable prerequisite of any learning robotic system is that it must be able to observe itself operating in the environment. Another important ramification of the reafference process is that absolute calibration of robots may unnecessary, since all that is needed is relative agreement between the different coordinate systems for perception and action. For example, by using a visual servo it is possible to open the fingers to the appropriate separation for grasping by moving the hand next to the object and matching its width. Fine tuning occurs when the configuration of the arm actuators is recalled upon the next presentation of the same object, and the associated interfinger separation is indexed.

2.7 Conclusions

We have seen that the development of grasping proceeds in various stages, and although there is some disagreement about the onset times of the various stages, several basic truths are evident. The first is that visually driven grasping is an instinctual reflex, based on the fact that it has been observed in infants as young as 2 weeks [Bower et al., 1970]. Additionally, upon looking at the structure of the motor system, we see it to be a multi-tiered system, with an important set of reflexes that are present from birth, such as the traction (grasping) and avoidance reflexes [Twitchell, 1970].

We also see that two major types of motor control, visual-visual and visual-kinesthetic are present during development and their different roles in calibration to the world. In
early development, infants seem to have an all-or-none approach to grasping where if they fail in the grasping attempt, no intra-trial corrections are taken and instead the entire hand is withdrawn and the trial begins again. Later on, within-trial corrections seem to predominate using the visual-visual control mode.

Self-observation seems to be critical in the calibration of the perceptual world with the action space. The intentionality of actions necessary for calibration implies a mechanism of attention which allows the relevant observations to be identified and the appropriate stimulus-action maps to be built.

Neural growth appears to play an important role in the development of sensorimotor behavior relating to distal control for fine manipulation and also in the creation of inhibition of reflexes after they have served to bootstrap the visual-manual loop. Additionally, growth of the visual channel’s capabilities create affordances that may provide information to allow control of distal manipulations as well.

The relevant question for robotic systems is how does one bootstrap to the point where one may efficiently gather evidence? Learning requires positive evidence as well as negative evidence, and innate abilities provide bootstrapping to the point where the system may gather some positive evidence. Given an unguided random search of the state space, convergence to successful outcomes by coordinating behaviors would take an inordinately long time. Therefore, prescriptive (declarative) knowledge must play a significant role in the development of skills, or innately coded mechanisms must be in force at given points during development.
3 Related Work in Machine Learning and Robotics

3.1 What is Machine Learning?

Machine learning can broadly be defined as the study of algorithms that create new knowledge in systems [Michalski, 1989]. This knowledge can be represented either symbolically or numerically. Symbolic encoding include a variety of means, from decision trees [Quinlan, 1986] to classification rules. Numerical encodings of knowledge are usually in the form of connection weights in neuromorphic architectures, nearest neighbor rules for memory-based learning, or coefficients that describe discriminating hyperplanes. Kaelbling [Kaelbling, 1990] has pointed out a disparity between symbolic and numeric learning methods: statistical learning methods tend to be robust with respect to noise in examples but are difficult to interpret, while symbolic learning methods tend to have easy to interpret explicit representations but tend to be quite brittle to noise.

More recent work in machine learning has attempted to bridge this gap [Aha et al., 1991; Schlimmer, 1987; Fisher, 1987; Quinlan, 1986; Kaelbling, 1990] by incorporating statistical measures into the creation of symbolic category descriptors and also by modeling actions and their outcomes in a stochastic fashion either by building world models [Mel, 1991; Drescher, 1989; Christiansen et al., 1991] or by generating models of expected reinforcement [Sutton, 1988; Watkins, 1989; Whitehead, 1989; Kaelbling, 1990].

3.2 Types of Learning Processes

Carbonell et al. [Carbonell et al., 1983] has categorized machine learning algorithms into the following classes according to the strategies used and how much inference is performed:

**Rote Learning:** No inference is performed by the learner. Knowledge is directly implanted, e.g., learning by programming.

**Learning By Instruction:** A teacher organizes and structures the knowledge that it provides to the learner. The learner must transform the knowledge from the input language to its internal representation.

**Learning by Analogy:** Transforming existing knowledge for use in new situations with some similarity to previously encountered ones.
Learning from Examples: A form of concept learning. It can be defined as follows: given a set containing examples and counter-examples, induce a concept description that describes all of the positive examples and none of the counter-examples.

Discovery Learning: consists of having the system form its own set of concepts (categories) and then determine the description of those concepts.

Since our ultimate goal is to make robots more adaptive to the characteristics of the environment, this implies the introduction of new knowledge to the system through data gathering from interactions with the environment. Therefore, it is not surprising that the majority of robot learning research has taken an inductive approach, in particular learning from examples. In learning from examples the input is generated in one of the following ways [Carbonell et al., 1983]:

1. A teacher structures the knowledge and presents classified examples in some efficient fashion given that it has the full concept description and possible knowledge about the the algorithm used by the learner.

2. The learner controls the examples generated and an external entity such as the teacher (supervised) or environment (unsupervised) classifies them.

3. The learner has no control on the type of examples generated. The external environment generates random examples of the concept.

Carbonell et al. [Carbonell et al., 1983] also note that learning from examples can involve only positive examples or both positive and negative examples. Also, learning can be one-trial, incremental or batched. One-trial learning implies that all examples are processed as an complete ensemble. Incremental learning systems process examples as they are generated. In general incremental learners are favored since their performance more closely imitates the type of learning seen in humans, and they can learn on-line. They can be more susceptible to misleading sequential coincidences of inputs in the learning corpus that may cause the system to “garden-path” to sub-optimal solutions. Batched systems process are an intermediate form, where new groups of examples are processed intermittently.
3.3 Empirical-Model Building

A number of researchers have argued for robot learning systems that build predictive models of the environment and robot’s plant. The justification for this approach is that in some domains there may be no good predictive models available. This can be due either to the lack of an existing formalism for modelling them, or the fact that applying existing theories and tuning them to each new domain is not acceptable because high autonomy is desired [Christiansen, 1991]. In these cases, it is useful to attempt to approximate the input/output behaviors of the world. This outputs can be in terms of direct changes to state variables [Moore, 1991b; Mel, 1991; Drescher, 1989; Christiansen et al., 1991; Shen, 1989] or through a reinforcement measure that assesses the desirability of states resulting from actions [Sutton, 1988; Watkins, 1989; Whitehead, 1989; Kaelbling, 1990].

Essentially all inductive learning systems use an empirical approach that relies on observed experiential data to generate predictive mechanisms. Prediction can be interpreted as a form of concept learning, where membership in a concept may be indicated by a variety of quantities such as reward value or a resulting state transition to certain points in the state-space. If the space of possible outcomes for a given task is partitioned into concepts, then we can view concept learners as predictors. We now briefly discuss some formal definitions and issues in concept learning.

3.4 Concept Learning

Concept learning consists of forming a description of a concept given some set of instances. Haussler [Haussler, 1987] uses the following definition that is useful for describing concept learning in a succinct fashion. A concept $c$ is defined as an arbitrary subset of an instance space $X$ that is the set of all possible object instances that is termed the instance space $X$. While concepts are unrestricted subset of the instance space, in general, learning algorithm use a restricted hypothesis space $H$ that is determined by what concepts are expressible by the concept description language. This restricted hypothesis space has an inductive bias that is partially determined by the description language.

The goal of a learning algorithms is to produce a hypothesis $h \in H$ that is consistent with the examples. Given a concept $c$, we say any element contained in $c$ is a positive example, otherwise the example is a negative one. A consistent hypothesis is one that
contains all of the positive examples and none of the negative ones. Consistent hypotheses may vary in terms of their specificity, from maximally specific to most general. Usually, the most general consistent hypothesis is preferred since it permits the most generalization. This may not always be the case however, especially when the penalty for missclassification is high. The chosen inductive bias determines which hypothesis is preferable.

A learning (or recognition) algorithm for the hypothesis space \( H \) and \( X \) takes the description of the characteristics of the instance space \( X \), a randomly drawn sample set that consists of positive and negative instances of \( c \), and generates a hypothesis that is consistent with the sample set.

Concept learning algorithms may be exact learners, where no errors in classification are permitted, or they may defined as probably-approximately-correct (PAC) [Valiant, 1984] with parameters \( \delta, \epsilon \). The interpretation of these parameters is that the algorithms misclassifies with error at most \( \epsilon \) with probability greater than \( 1 - \delta \). The error \( \epsilon \) is the probability of the symmetric difference of the concept and the hypothesis. If a learning algorithm \( A \) for \( C \) with sample size \( m(\epsilon, \delta) \) exists for all sampling distributions over \( X \) then the concept class \( C \) is said to be uniformly learnable. If the learnability is contingent on \( P \) the sampling distribution, then the concept class is said to be non-uniformly learnable under that distribution.

Recognition complexity is defined as the number of computing cycles necessary to form a consistent hypothesis as function the complexity of the concept being learned. Sample complexity is defined by the rate of convergence of the algorithm to a given \( \delta, \epsilon \) as a function of concept complexity and the number of examples.

The Vapnik-Chervonenkis (V-C) dimension is a measure of the complexity of the concept class \( C \). It is defined as the cardinality of the largest set of points \( S \subseteq X \) that are shattered by \( C \). A set \( S \) is defined as shattered by a set \( C \) if the power-set \( 2^S \) of \( S \) can be obtained by iteratively intersecting \( S \) with all \( c \in C \).

Blumer et al. [Blumer et al., 1989] have shown that if the V-C dimension of the concept class is finite, then it is uniformly learnable and the learning algorithm’s sample size is linear in the V-C dimension. If the V-C dimension of \( C \) is infinite then \( C \) is unlearnable. Similar bounds hold for learning stochastic concept descriptions where the goal is to find hypotheses that agree with a sample set whose members are mislabelled with some probability [Blumer et al., 1989].
While the above results are important from a theoretical perspective, applying the results to robot domains is, in general, a difficult task. A major limitation in the above approach that it holds for boolean outputs, while robotic applications are interested in learning to characterize processes with real-valued output functions. In many tasks it is non-trivial to compute a V-C dimension a-priori. This is because in order to compute the V-C dimension, the general properties of the concept class must be available. This, in turn, requires knowing the domain where the learner will function a-priori, which is not acceptable for autonomous systems. This approach is also not amenable to multi-step learners where the classification of a given action in a situation may not be immediately available, such in delayed reinforcement tasks [Sutton, 1988]. Also, since the above results are distribution-free with respect to the collection of sample instances, they tend to be pessimistic in their bounds [Kaelbling, 1990; Buntine, 1989] since the do not take into account the possibility of prior knowledge about the sampling distribution.

3.5 State-Based Learners

A number of workers are interested in state-based approaches to learning for real-valued domains. The state are defined by partitioning the continuous state-space into discrete states. A common feature of these approaches is that they model the world as state-based automaton, where transitions between states are initiated by actions executed by the agent [Sutton, 1990; Christiansen and Goldberg, 1990; Kaelbling, 1990; Whitehead and Ballard, 1991]. In general they seek either to determine policy functions that map sensory inputs to reward-optimal output actions, or build predictive models of the world. From the robot’s perspective, a world model consists of a computational mechanism that can take the current state and an action to be issued and return the probability distribution of items in the set of resulting states.

Systems that build world models tend to be more flexible than reinforcement learners since their knowledge is not goal specific, unlike the reinforcement learner’s. The world model can be used drive a number of different planning mechanisms. Reinforcement learners are more autonomous than world modelers, since they do not require a specific planning system to be built in order to exploit the world model, but instead rely on a learned policy function to achieve the goal. What reinforcement learners lose in generality, they gain in reactivity, since typically the policy function is rapidly computable.
3.5.1 Building Models of the World

Christiansen et al. [Christiansen and Goldberg, 1990] investigate approaches to planning with stochastic actions in order to build world models. They choose a tray-tilting tasks that consists of a system with 18 possible discretized world states, consisting of nine object positions and two object orientations. Actions include twelve possible tilt azimuths at 30 degree intervals. The effects of actions on the system are modelled using a non-deterministic finite automaton, where the execution of actions leads to the transition to one of a set of successor states. The transition probabilities for the actions are approximated by observing the effect of actions over some corpus of training examples. This information is then used in either an exhaustive search that generates plans (sequences) with a greatest lower bound on failure probabilities, or a search that uses some heuristics to expedite the search and generates reasonably good solutions.

Christiansen [Christiansen, 1991] generalizes this approach in an empirical backprojection learning methodology for planning. Given a goal region, the back-projection method attempts to form a pre-image of existing states for a chosen action. The pre-image of an action $a$ consists of regions of the state-space that lead to the goal state when the action $a$ is executed there. By regressing (back-chaining) back from the goal using the learned preimages, it is hoped that the system can form plans, if they exist, from any feasible region.

G.L. Drescher’s Schema Generation System [Drescher, 1989] is an attempt to create a computational mechanism that has properties similar to those observed in Piaget’s [Piaget, 1952] theory of constructivist infant development. Drescher chooses to concentrate on the sensorimotor development stage, where the infant develops knowledge about physical objects, their properties and how to interact with them in a purposive way. The main computational component of the system is the schema. A schema makes a prediction about what the state of affairs will be if a given action is executed on the environment when its preconditions are satisfied. Each schema is composed of a context, action and a result. The system attempts to synthesize new schemas and chain them together in purposeful ways. Drescher develops a simplified sensory world simulation for an agent, with well developed visual and tactile (proprioceptive) abilities. His simulated microworld consists of a 5 by 5 spatial location matrix of possible hand and object positions, and all possible object percepts are encoded in a boolean fashion. The system is implemented on a massively-parallel
processor and able learns some interesting multi-step behaviors.

3.6 Modelling with Statistical Regression

Statistical regression is a paradigm used in a variety of disciplines and has recently begun to be used in robot learning applications [Aboaf et al., 1989a; Atkeson, 1991; Moore, 1991b]. The general regression problem is formulated as follows. Assume the process we are interested in has a behavior that can be described by some function $f(x_1, \ldots, x_p)$. Assume we do not know this function and are interested in building a good approximation of it by observing its input-output behaviors. We observe the output of this function, and we wish to approximate it from some set of noisy observations $\{(x_{11}, \ldots, x_{pn}, y_1), \ldots, (x_{1n}, \ldots, x_{pn}, y_n)\}$, where there are $n$ observations. Each observation tuple may also be weighted by some factor $w_i$. We then make the assumption that the observations come from the following process:

$$Y = f(X_1, \ldots, X_p) + \nu$$

(1)

where $\nu$ is a random variable that corrupts the observations. In regression we endeavor to reconstruct this underlying function by estimating the conditional expectation

$$\hat{Y}(x_1, \ldots, x_p) = E[Y | X_1 = x_1, \ldots, X_p = x_p]$$

(2)

This is effectively a form of learning from examples, where we are now interested in learning the function from examples. A variety of techniques exist for generating this estimate. The best known of them is linear regression although many other parametric techniques exists. However, we note that picking $f$ to come from a parameterized family of functions is a form of inductive bias, since it limits the type of underlying functions expressible and therefore learnable. The statistical community has noticed this as well, although inductive bias is in terms of model selection in their case. Since in many real-world applications the underlying functional class is unknown and constraining the classes often may lead to poor models and fits to the sampled data, numerous non-parametric techniques have been proposed. Among them are projection-pursuit regression [Friedman and Stuetzle, 1981], regression trees [Breiman et al., 1984], locally-weighted regression [Cleveland, 1979], kernel-smoothers, and spline-smoothing [Wahba, 1983]. These approaches are highly flexible and tend to have low inductive bias since they have weaker assumptions about the functional
forms for the process being characterized.

Atkeson [Atkeson, 1991] uses locally-weighted regression for modeling the forward dynamics of a manipulator. He begins with a dynamics simulation that generates random torques on the joints of simple two-link manipulator and then records the resulting output joint velocities. A locally weighted fit is used to smooth the data and generate predictions. Cross-validation techniques are used to optimize various fitting parameters such as the radius of the set of support region. Cross-validation techniques consist of splitting the learning set into m groups, performing the regression using m - 1 groups and estimating the fitting error on the “left-out” partition. This is done for each of the m subsets. Atkeson uses an estimate of the derivative of the cross-validation error to speed up the procedure for estimating the fit parameters.

Aboaf et al. [Aboaf et al., 1989a] use a compensation scheme for top-level commands to improve the performance of a juggling robot. Rather than attempt to refine the model of ballistic flight and interaction that occurs when a robot's paddle hits a ball, the system is initialized with a simple model of the dynamics. It then builds up a model of the errors that occur relative to the desired ball landing location as a function of where the ball contacts the surface of the paddle. Since during the learning process, the ball lands at discrete points on the paddle, a polynomial regression is used to smooth the compensation values so that the system can generalize in a localized fashion.

Moore [Moore, 1990] uses a nearest-neighbor memory-based approach to generate a model of torques on the cartesian velocities of a simulated manipulator. The triplets of (input, action, output) are indexed by input and output in a k-D-tree [Friedman et al., 1977] type structure for fast search. Given the current state of the manipulator and the desired state, the tree structure is searched for nearest-neighbor of the current input state and desired state and the desired action stored in the triplet node is executed. Since the algorithm permits an weighted sum of nearest neighbors, it can be considered a form of locally-weighted regression.

The Cerebellar Model Articulation Controller (CMAC) as proposed by Albus [Albus, 1972] is memory based function learner with regression-like learning characteristics. The algorithm takes points in the input space and hashes them to form a contiguous set of m memory locations. The output value for an input query point consists of the weighted sum of the contents of the m nearest memory locations for that point. The contents of the
memory cells are adjusted using a perceptron-like updating scheme. The hashing process is advantageous for high-dimensional situations because it collapses the measured points into the available finite store. Miller [Miller III, 1987] explored using CMAC memories to learn a localized inverse Jacobian function that was used to command joint velocities for a positioning task. He also investigated using CMAC's for tracking tasks as well. Subsequently, Miller [Miller III et al., 1987] has used CMAC memory for improving the performance of fixed-gain robot controllers.

3.6.1 Building Models of Reinforcement

Reinforcement Learners view the effects of actions through a reinforcement function that maps the system-state observations into a real-valued reinforcement. Therefore, the agent is interested learning to choose actions that maximize the expected reinforcement. In the most general case of reinforcement learning, the system is not told what the desired control signals are but must discover (identify) these signals through repeated experimentation. The problem with such approaches is that quite often, the dimensionality of the search space is very high, which leads to a large number of states. In these cases, convergence is often very slow since the state-space that must be searched is huge. Thus, most work using reinforcement learning has been applied to relatively simple problems.

One of the earliest successes in reinforcement learning was Barto’s [Barto et al., 1983] system that learned to balance an inverted pendulum. The system learned a control policy using a simple delayed reinforcement paradigm. The state-space of the system was partitioned and discrete control actions were associated with each of these regions. The system attempted to keep the pole balanced using the its control policy which was determined by weightings generated by the learning algorithm. At the end of each learning trial a reinforcement measure based on the duration of the trials was fed back to each of the states, actions that participated in the success were reinforced, and those that occurred before the failure were inhibited.

More recently, Sutton [Sutton, 1990] has proposed the DYNA architectures for learning. In this system, a world model is built up by observing the environment. This model is used with an incremental dynamic programming paradigm. Path planning in simple grid-like world with obstacles and goal states is learned.

Whitehead has investigated several issues relating to reinforcement learning and its re-
lation to reactive planning systems [Whitehead, 1989; Whitehead and Ballard, 1991]. In particular, he has focused on the issue of perceptual aliasing (see Section 3.14.3) and its effects on learning. Perceptual aliasing is an inconsistency between the internal representation of the world state and the actual state of the world. This is usually due to the fact the sensory system of agent is not able to disambiguate between different states in the world and therefore maps several world states to one internal state, which appears, from the agents point of view, to have non-deterministic behavior. Christiansen [Christiansen, 1991] has also noted this effect in manipulation planning.

Moore [Moore, 1991b] has investigated reinforcement learning using variable resolution dynamic programming. He uses a k-D tree and nearest neighbor lookup to generate a model of the environment. This model is then used to simulate the world and aid in a dynamic programming process that finds control rules which maximize reinforcement.

Maes et al. [Maes and Brooks, 1990] demonstrates reinforcement learning for the sequencing of behaviors in a hexapod walking robot. The paradigm assumes a fixed set of behaviors that are executed in a context-dependent fashion whenever a boolean conjunctive precondition formula is valid. The validity of a predicate represents the existence of some state of affairs in the environment. The outcome of the execution is represented in the form of either positive, negative or no feedback. The learning consists of generating candidate precondition formula that have the highest probability of generating a positive feedback for the behavior, similar to the approach of [Drescher, 1989]. The method assumes a Boolean perceptual representation and a predefined reinforcement function. It evaluates the candidate formula both for the relevance of the literals in the formula as well as reliability of the formula in predicting positive or negative reinforcement.

### 3.7 Induction of Decision Trees

Quinlan [Quinlan, 1986] has devised information-theory based metrics for generating efficient decision trees. Decision trees are a form of recognition predicate where each node selects an attribute and different branches are taken depending on the value of the currently selected attribute. The leaf nodes of the trees are labelled with different concept names. Instances are classified by testing each of their attributes according to the decision tree and then labelling according to the leaf node value. Quinlan's paradigm creates the tree adaptively by computing the expected information gain for attributes under evaluation and
picking the most informative one. By putting highly informative variables nearer to the root of the tree, more efficient trees are generated. This basic paradigm has been used in many subsequent systems.

Zrimec [Zrimec, 1990] has developed an inductive learning system that randomly explores the domain it wishes to characterize. The chosen domain was a two-dimensional world, and a set of actions for pushing objects. It then attempts to induct a qualitative model of the pushing process based on the observed examples. This process model is integrated into a planner that uses knowledge of the effects of its actions to engage in goal-directed behavior. The system first attempts to determine causality relations between actions and their affects using measures of shared information between attributes. It then attempts to determine the relevance of attributes in influencing the outcome of actions.

Tan has developed an inductive learning system that learns to grasp based on Quinlan’s ID-3 system, [Quinlan, 1986]. This system is novel in that it incorporates the cost of actions into the selection of sensing and action procedures. Cost is determined in terms of execution-time for the sensing and action procedures. Object descriptions are given in terms of appropriate grasping procedures along with an empty set of perceptual-attribute descriptions. The set of possible grasping procedures come from a collection of stereotypical grasp routines. The system begins with a training phase where it attempts to generate a low-cost sequence of sensing procedures that can discriminate between objects. The result is a low-cost decision tree for identifying objects and their correct grasping procedures. During execution, this tree is utilized to discriminate between the objects based on a cost-sensitive informativity measure for the attributes. This tends to favor sensing procedures that have good discriminability and low execution cost associated with them.

Dufay et al. [Dufay and Latombe, 1984] use an inductive method for generalizing robot plans based on the sensory traces of several successful task executions. A conventional planner is used to generate assembly plans for a given goal state. The robot then attempts to execute those plans during a training phase. If the generated plan fails for some reason during execution, the system attempts to generate “patch” plans to try and complete the task. Each attempt at the task is stored as an execution trace. Upon finishing the training phase, the execution traces form the learning set. The system then attempts to merge the various execution traces by using rewrite rules. These rewrite rules look for local matches between the execution traces. The separate traces are merged using these rules until a single
generalized robot plan for the chosen task remains.

3.8 Learning by Instruction

A number of researchers have explored supervised learning approaches for robotics tasks. The rationale for supervised learning is that they can achieve the transfer of knowledge between systems without extensive reprogramming [Simon, 1986]. This is accomplished by having one system observe another in action and then using a learning-by-observation mechanism.

Kuniyoshi et al. [Kuniyoshi et al., 1989] propose a supervised robot learning paradigm where the human acts as an instructor by performing a task while a vision-system watches. The vision system is model-based and continuously tracks the relative positions and contacts between objects while displaying its internal scene representation. The teacher watches this representation to make sure it captures the pertinent visual features and state transitions for the task. If there is disagreement, the human gives instructions to the system in terms of new features to be tracked and what the other regions of interest should be.

Ikeuchi et al. [Ikeuchi and Suehiro, 1991] have developed a similar approach, but instead of looking at dynamic information that is extracted during the motion of the parts, vision processing is attempted on the scene only when it is static. By looking at the “before” and “after” interaction part configurations in the scene, the system generates incremental actions for plans.

3.9 Rule-Based Grasp Selection

A form of supervised rote robot learning consists of directly inserting explicit heuristic rules about grasp selection into a database using expert-systems methodologies. These are not learning systems per-se since each heuristic rule must be generated by the programmer. Several expert-systems for this purpose have been developed [Cutkosky, 1989; Liu et al., 1989; Tomovic et al., 1986; Stansfield, 1990] to determine grasp choices using these heuristic rules, object geometry and task requirements. Liu [Liu et al., 1989] attempts to integrate stability issues, necessary forces to generate and necessary precision. Stansfield [Stansfield, 1990] uses a set of rules for determining which preshapes and approaches are valid given a partial or complete aspect graph of the object derived from a range image of the object. Cutkosky [Cutkosky, 1989] defines a grasp taxonomy in terms of grasp geometry, object
geometry and task requirements by interviewing and observing a number of experienced machinists with regard to which grasps they chose in given situations. He then encoded this hierarchy into a set of production rules. The system takes an input as to the task requirements and uses the rules to select grasps which satisfy those requirements.

3.10 Explanation-Based-Learning Approaches

Bennett [Bennett, 1991] proposes an explanation-based learning approach to planning for robots. The chosen problem domain is grasping of puzzle pieces. The system uses default domain rules for planning and generating initial unguaranteed plans and attempts to execute these plans. When failures occur in execution because of inadequacy in the domain model, it engages in reasoning using qualitative rules about the behavior of the uncertainties. As a result of this reasoning, it changes the set of quality functions. These functions are continuous valued and dependent on the action parameters. They are used in deciding which parameter bindings among a continuum are more favorable in a situation.

Laird et al. [Laird et al., 1991] use the SOAR learning architectures to engage in supervised learning of grasping tasks. The system begins with a default domain theory that includes a set of operators and preconditions that are a function of the sensed environment. Upon failure of a grasping task, the system must request input from an a human in terms of additional preconditions for the operator. The new operator serves to augment the domain theory by superseding the old one. They demonstrate this process with a failure that occurs when a gripper attempts to pick of an extend a triangular bar from an inappropriate orientation and fails. It then recovers by generating a new operator that checks for the salient characteristic that lead to the failure.

3.11 Model-Based Recognition

Dunn et al. [Dunn and Segen, 1988] have developed an unsupervised system for learning to pick up two dimensional puzzle pieces. The learning algorithm is purely memory based and uses unsupervised experimentation to gather exemplars. A given grasp attempt begins by reducing a thresholded image into a polygonal description. An attempt is then made to match this description to pieces it has successfully grasped. If it finds a match, then it applies the previously successful grasp to the object. Otherwise, it generates random grasps. Grasps are termed feasible if the finger trajectories intersect the object and the initial
hand configuration encloses the object. The system iteratively attempts these randomly generated grasps until it succeeds. It then stores the object description and the successful grasp parameters in its memory for future access. Grasp success or failure is established by visually monitoring the configuration of the object relative to the hand using a side mounted camera.

3.12 Classification-Based Approaches

Asada [Asada and Yang, 1989] has used a supervised learning paradigm to generate control laws for the successful deburring of plastic parts. The system attempts to build discriminant functions for a classifier. The classifier identifies vectors of sensors values that should trigger different control actions such as force directions and magnitudes. Execution traces of skilled human operators are used as the learning set.

3.13 Neuromorphic Architectures for Sensorimotor Learning

Mel [Mel, 1991] develops a system that learns path planning in a cluttered two dimensional environment with a three degree-of-freedom arm. The system consists of a real-time binary vision system, the robot arm which moves in the plane, and a workstation that simulates a large network of simple computing units. The architecture consists of various functionally distinct subnetworks of simple computing units. In each grouping, the member units represent the state variables of a corresponding sensory modality. Subnetworks represent the visual field, the direction of velocity of the hand, and the absolute angles and angular velocities of the shoulder, elbow and wrist robot joints. Mel uses the paradigm of “learning by doing” in which the system “flails” its arm around in different configurations by using joint level commands and simultaneously observes the corresponding visual field activities. The system learns to predict the effect of joint angle perturbations on the visual field and uses this as a projection mechanism for a planner that simulates possible solutions to the goal state.

Kuperstein [Kuperstein, 1988] trains a simulated network of simple elements to compute the inverse kinematics of a five degree of freedom arm from inputs derived from oculomotor and binocular disparity signals.
3.14 Discussion

3.14.1 Nearest-Neighbor Classifiers and Regression

The approaches of [Moore, 1990; Moore, 1991b; Atkeson, 1991; Aha et al., 1991] are examples of nearest-neighbor learning methods. Nearest-neighbor classification algorithms are a well established technique in pattern recognition. A nearest-neighbor algorithm classifies the membership or output value of an incoming point according to the label or value assigned to the closest point in the memory. Cover [Cover and Hart, 1967] has shown that this class of algorithms converges to a misclassification rate no greater than twice the optimal Bayesian error rate. Nearest neighbor approaches are practical either when only a small part of the entire state-space is to be populated or the dimensionality of the feature-space is low. In this case, it is possible to have a high enough local density of exemplars so that when a new observation is processed, the nearest neighbors are close enough that their values form a reliable estimate. Also, if the distribution of queries and exemplars are dense and co-local with respect to each other then the approach is feasible. The smoother the underlying function, the lower the density needed and the higher the dimensionality that can be tolerated. Often, however, this local generalization is not sufficient, such as when the task potentially requires using wide areas of the state-space and the function is not smooth. In this case, alternatives to nearest-neighbor techniques must be found.

3.14.2 Exploration

The amount of exploration that a learning method undergoes in learning is quite important and entails many tradeoffs. Since exploration is an information gathering process, naively, one might say that the more exploration permitted by a learning algorithm, the higher the probability of finding a globally optimal solution since a larger search space is explored. However, this ignores the notion of regret in learning. Regret is defined as the cumulative amount of reinforcement that is lost because the agent does not follow an optimal decision policy. The more rapid the convergence, the less regret incurred. However, if the agent has a finite lifetime, then learning algorithms that converge to close to optimal policies but do so more quickly incur less regret then a slow algorithm that eventually finds the globally best solution. On the other hand, the notion of coverage over all tasks is important. Although an action may fail to accomplish a goal for the current task, this incorrect outcome may be
the required outcome in different tasks.

Mel and Miller [Miller III, 1987; Miller III et al., 1987; Mel, 1991] train a system to approximate the inverse Jacobian for a manipulator using uniformly distributed random motions. Moore [Moore, 1991b] adopts a more directed approach by assuming a Gaussian distribution of similarity to previously observed outcomes. The width of this distribution is a function of how similar the an action is to previously observed actions. If an action is almost identical to previous actions, then with high likelihood, the outcome will be similar and the width of possible outcome distributions is narrow and centered close to the outcome of the previous action. If an action is highly dissimilar to previous actions, then little prior information is available, and the distribution width widens and begins to approximate a uniform distribution. For actions in between, the width is graded. A set of random action are generated, and the actions whose outcome distribution has the largest integrated probability overlap with the desired outcome interval is chosen as the next exploratory action. With no information, actions are uniformly distributed. If actions yield outcomes far from the desired outcome, then further experimentation near those action values is discouraged. Likewise, exploratory actions near the desired value are favored.

Christiansen et al. [Christiansen et al., 1991] has also devised the method of strategic self-training to control experimentation in a goal-directed fashion using two reliability thresholds as a guide. The acceptable threshold is defined as a value of reliability sufficient for the task. The promising threshold is less reliable than the acceptable one, but possibly good. If no plan with sufficient estimated reliability can be generated using the current world model, but a plan above a promising threshold exists, then that plan is executed to gather information. If the plan is estimated to be above the acceptable threshold or no plan can be found, then execute a random tilt. If the reliability is below the promising threshold, then the weakest link of the plan is changed by picking a uniformly distributed random action within some fixed interval around the planned action.

3.14.3 Sources of Real and Apparent Non-Determinism

Several researchers have described sources of non-determinism in models of the world [Kaelbling, 1990; Whitehead, 1989; Christiansen et al., 1991]. The interpretation of non-determinism here is in the stochastic sense, meaning that a probability distribution governs the outcome states possible with a given state and action. There seems to be agreement that the possible
sources of this non-determinism are:

1. **Perceptual Aliasing**: the agent misclassifies the actual state of affairs in the world due to insufficiencies in its perceptual system.

2. **Sensor Errors**: a sensor fails to transduce the output quantity correctly by partially or totally failing.

3. **Effector Errors**: the intended action is not executed by the system due to faulty effectors either due to a total failure, or insufficient precision in the resolution or control of the effector.

4. **Intrinsic Non-Determinism**: the process being characterized has non-deterministic state transition functions.

Kaelbling [Kaelbling, 1990] has shown that all of these failures are equivalent to an agent with perfect sensors and effectors in a non-deterministic environment. Both Kaelbling and Whitehead [Kaelbling, 1990; Whitehead and Ballard, 1991] have recently developed learning algorithms that can handle forms of perceptual aliasing.

### 3.14.4 State Discretization and Dimensionality Issues

A major dichotomy may be found between discrete-state based formalisms such as [Sutton, 1990; Christiansen *et al.*, 1991] and continuous state-spaces. In a sense, the problem of choosing a discretization for a state-space is similar to the problem of selecting primitive categories for objects in the world continuum, and for deciding when to combine regions in the segmentation of visual images. When partitions of a continuous state-space are chosen to form discrete states, assumptions are made about what regions of the state-space can be aggregated in a meaningful fashion with respect to the input/output behavior of the world. This choice is crucial in many learning algorithms. If the discretization is too fine, than there will be an overabundance of states, and this will unnecessarily increase the number of samples for statistically significant information to be gathered for each of those states. This over-resolution is equivalent to the increase in states that occurs when the dimensionality of the task increases, which makes the learning algorithm converge slowly. On the other hand, if the quantization is too crude, then perceptual aliasing will occur. Because of this inability to distinguish between areas of the state-space that may have different behaviors to actions,
apparent indeterminacy may result, which is also detrimental to the convergence-time and ultimate upper bound on performance.

In general, the problem of appropriate discrete state definition in a continuous world has been ignored except for a few attempts at automatic state discretization. Simons et al. [Simons et al., 1982] have developed a learning automaton that adaptively partitions a state-space for a peg-in-hole task. It uses the criteria that if an action executed in a region of state-space did not converge to a high or low reinforcement outcome then that region of state-space should be partitioned to a higher resolution. Moore [Moore, 1991b] has proposed a variable resolution approach to dynamic programming, where regions closer to the trajectory of the state-space have a higher resolution and those further away have lower resolution. This increases the maximum dimensionality of problems that can be solved by dynamic programming techniques since only relevant regions of the state-space are represented with high resolution.

3.14.5 Convergence Time and Environment Tracking

Obviously, in the harsh world of unstructured environments and manufacturing environments where efficiency is at a premium, learning system must quickly converge. Generally, convergence for learning automaton is given for learning in the limit [Narendra and Thathacar, 1974], which is an inappropriate measure for systems that must adapt to change within a finite lifetime. While a common justification given for learning systems is that they are adaptive to changes in the environment, it is generally difficult to quantify this in the general case. Given that the behavior of the environment changes at some discrete time or gradually, some notion of tracking ability for a learner should be developed. In order for a learner to have tracking ability at all, it must have some form of ability to forget its experiences [Atkeson, 1991], but this has not been addressed explicitly. If memory based learners have a large number of previous experiences contradicting those produced by the changed dynamics of the system, it will take a large number of new experiences for the regression to begin to capture the new characteristics and it will always experience bias. A solution to this may be in terms is an exponential decay term to be added based on proximity to new observations and output consistency. This will allow observations that are older and inconsistent to be gradually decreased in weight and extinguished when they fall below a threshold.
3.14.6 Behavior Based vs. Knowledge Learning

When background knowledge is only accessible implicitly by observing the input/output mapping function of the system, then the flexibility of the system is reduced since the run-time control of the system is subsumed by the learning process. Such approaches tend to compress the background knowledge that the system has to a low level of abstraction having advantages and drawbacks. The advantages are that generally the computation of actions can be done rapidly. This type of performance is useful in "reactive" type control where actions must be emitted with low latency time to assure competence in dynamic environment. The drawbacks of such an approach are that it does not encompass higher levels of abstraction, this limiting the flexible reuse of knowledge. Also, such procedures tend to converge very slowly compared to what a more "clever" high level reasoning system might deduce. On the other hand reactive type policies tend to avoid difficult issues of symbolic representation and the computational complexity of deduction.

3.14.7 Embedding Knowledge in Learners

Another highly relevant point is the suitability of methods for gathering assistance in the form of supervised learning. Of course, in the clever design of a learning system, the designer can often express knowledge about the structure of a task in the form of decomposition of the task into subgoals [Moore, 1990] each with their appropriately defined local reinforcement measures, correct resolution for the state-space, and in selecting parameters of the state-space that are most relevant to the success of the task.

Whitehead [Whitehead, 1991] has looked at some complexity results for reinforcement learning where the learner can observe and communicate with other agents, and also by direct supervision by an agent that suggests high reinforcement moves for the learner during training. Sutton has devised the adaptive heuristic critic [Sutton, 1988] which essentially reduces the delayed reinforcement learning into two subproblems, the first being the generation of a critic that learns to generate appropriate local reinforcement functions and the second a local reinforcement learner. Most of the time, however, the choice of the reinforcement function is left to the designer of the system, since ultimately it requires knowledge about what is required for the task.

Relevancy measures have been proposed by Zrimec [Zrimec, 1990] using an information
based metric called the normalized measure of dependence. While the measure can be quite powerful, it is quite sensitive the state partitioning decisions. If the state resolution is too fine, then the states are so sparsely populated that the metric is meaningless since the estimated probabilities are not statistically significant. A similar situation holds when resolutions are too coarse.

3.15 Conclusion

There are many possible approaches to learning in robotics, including predictive world modelling and reinforcement learning approaches, using a variety of prediction mechanisms. The actual application of learning techniques has been in modelling manipulator dynamics [Atkeson, 1991; Aboaf et al., 1989a; Moore, 1990], learning maze traversal tasks [Kaelbling, 1990], grasping [Dunn and Segen, 1988; Laird et al., 1991; Bennett, 1991; Tan, 1990] and assembly [Ikeuchi and Suehiro, 1991; Kuniyoshi et al., 1989; Bennett, 1991; Tan, 1990] and assembly [Ikeuchi and Suehiro, 1991; Kuniyoshi et al., 1989; Dufay and Latombe, 1984]. We develop a paradigm which is different from these approaches by using a non-parametric reinforcement learning algorithm for learning and an action-map approach for storing the learned grasp selection rules.

The previous discussion also points out several fundamental issues which are important in task-based learning for robots. The most important of these is that the dimensionality of a task is currently a limiting factor in learning. For example, multi-step learning with fixed actions becomes more difficult when the number of states increases, and the same occurs in single step learning when the number of parameters that control the continuum of actions increases. This is most apparent in reinforcement learning approaches for multi-step plans [Whitehead and Ballard, 1991; Sutton, 1988], which rapidly become intractable as the size of the state-space for the problem increases. Counteracting the effect of dimensionality in single step-learning has not been explored in the literature to a great extent. In single step task learning in continous domains, such as developed in sections 4 and 5, we present techniques that will mitigate the effects of increasing dimensionality on action and perception representations.

The techniques of section 4 build on the work of Atkeson [Atkeson, 1991] who suggested direct use of non-parametric regression for learning of robot dynamics. However instead of using locally-weighted regression [Cleveland, 1979] which will rapidly be useless as the dimensionality of the task increases, we use projection pursuit regression [Friedman
and Stuetzle, 1981] because it has been shown to have much better immunity to increasing dimension. The approach also differs from Atkeson’s since the regression is used to generate estimates for reinforcement values rather than direct physical quantities.

We also use a novel type of tree structure for indexing perception to action which takes its inspiration from the tree structures proposed by Omohundro [Omohundro, 1987a], and action-map building approaches of Moore [Moore, 1990]. We depart from these approaches in how we use the structures and by using adaptive methods for generating the trees more efficiently.

In section 5 we utilize some of the properties of trees and splitting rules based on uncertainty estimates as developed by Simons [Simons et al., 1982] for learning. We adopt the flexible tree structures exploited by Moore [Moore, 1991b] and Omohundro [Omohundro, 1987a], who have proposed adaptive resolution schemes that mitigate the effects of dimensionality in learning. Kaelbling’s work [Kaelbling, 1990] for probability estimation in learning provides a statistical basis for the adaptive tree generation. We extend these foundational works by proposing a novel $k$-D-tree/quadtree algorithm that controls resolution depending both on sample density and uncertainty estimates.
4 Projection Pursuit Regression Reinforcement Learning and Bootstrapping

4.1 Introduction

In this chapter we discuss learning algorithms and perception/action data structures for task-based learning. The established technique of non-parametric projection pursuit regression (PPR) is used to accomplish reinforcement learning by creating a function which approximates the expectation of task success, conditioned on perceptual and motor attribute values. It searches for \textit{generalization directions} which determine projections of high-dimensional data sets that capture task invariants. We then take the resulting estimation function and use it to create a model of the task, the perception-action map, which is used for rapid decision-making during the execution of the task.

We combine this learning algorithm with the approach of introducing additional attributes to the perceptual and action representation in an incremental way. We adopt this strategy based on evidence from the literature of developmental psychology and neurophysiology relating to the growth of sensorimotor abilities in humans and primates. This gives us a prescription for the type of developmental cycles that occur during the learning of visually-guided grasping and what representations are reasonable to employ during the development of visually-driven grasping in machine systems.

4.2 Progressive Refinement of Action and Perceptual Representations: Gibson’s Theory of Affordances

We propose the following multi-stage approach for sensorimotor learning: inductive learning must happen incrementally with respect to the number of parameters to be characterized, otherwise the learning becomes intractable due to the combinatorics of the task which requires impractically large numbers of samples. This approach motivated directly by E. J. Gibson’s theory of perceptual affordances [Gibson, 1969] and by many of the developmental results discussed in section 2. Gibson’s theory states that advances in the organism’s perceptual system are matched by those of the action system at various stages. Each synchronized advance in each of the two systems opens up new degrees of freedom (“affordances” in Gibson’s nomenclature) which must be explored and characterized relative to the interactions necessary with the world.

39
Translating this to a robotic context, we say that the perceptual system delivers increasingly differentiated information (more perceptual parameters with greater resolution) about the world, but only when the system has mastered what is currently possible with the current perceptual abilities and underlying motor control. This is paralleled by the action system, which must progressively differentiate with complex actions and controlling parameters to take advantage of the correspondingly richer perceptual representations.

The advantages of such an approach are that each learning level guides the exploration in subsequent learning levels, permitting the escape from the combinatorics of statistical learning with no prior information. Secondly, at each point, the system is competent to some level and its abilities gradually increase in a way similar to the progression in human infant development.

This approach to exploration differs from those discussed in section 3, such as the strategic self-training approaches of Christiansen [Christiansen et al., 1991] and the directed approaches of Moore [Moore, 1991b] since it exploits the assumption that the task is hierarchically learnable. Hierarchical learnability is similar to the notion of stepwise skill refinement. The idea is that complex tasks with many degrees-of-freedom (dimensions) are not learned all at once, but rather in a sequence of tasks. Each task in the sequence uses exploration strategies that exploit the acquired knowledge about the performance of earlier related tasks. Subsequent layers inherit default parameter bindings from the previous tasks. The previous bindings tend to initialize the more complex tasks close to, or inside of feasible areas in the higher-dimensional parameter spaces. The bindings limit the exploratory intervals for the subsequent tasks, yielding more information content about the task per trial than tabula rasa strategies.

We propose a series of experiments to test the validity of this approach (see Table 1). At first, the system has insensitive perceptual and action capabilities, and correspondingly, the task cannot be very demanding. In experiment one (see Table 1) we consider the following task: there is a desired object viewed in the workspace and the arm/hand must contact it to receive a tactile stimulus, although it need not grasp and lift it. If the system masters this task, it has learned that the hand and object must coincide in order for it to receive the desired sensory stimulus of a tactile contact with the object. This is the first step towards developing a target-object centered action frame for interacting with the world.

In experiment two, the system must enclose the object, although not necessarily lift it in
Table 1: The proposed sequence of tasks and representations. \( H_x, H_y \) are the \( x, y \) position of the hand in the plane, relative the robot coordinate frame. \( O_x, O_y \) are the object positions, relative to the image frame. \( H_\theta \) and \( O_\theta \) are the orientations of the major axis of the hand and object respectively.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Vision Attribute</th>
<th>Action Attribute</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((O_x, O_y))</td>
<td>((H_x, H_y))</td>
<td>Tactile Stimulus</td>
</tr>
<tr>
<td>2</td>
<td>((O_x, O_y))</td>
<td>((H_x, H_y))</td>
<td>Object Enclosure</td>
</tr>
<tr>
<td>3</td>
<td>((O_x, O_y))</td>
<td>((H_x, H_y))</td>
<td>Object Lift</td>
</tr>
<tr>
<td>4</td>
<td>(((O_x, O_y), O_\theta))</td>
<td>(((H_x, H_y), H_\theta))</td>
<td>Axial Alignment</td>
</tr>
</tbody>
</table>

In order to have succeeded. This is similar to the previous task, except that the hand/object matching constraint is much tighter since the hand must now enclose the object, not merely touch it in passing. The information from the previous task is used to guide the exploration in this level. We pick the location for a given grasp trial using the perception/action parameter bindings from the previous tactile on reinforcement task and then add a random exploratory component relative to that location to characterize the task. Since tactile stimulus is a prerequisite for enclosure, this increases the probability that a given trial will yield a positive outcome in enclosing the object. At the same time the constraint for hand/object placement is tighter for this task, so the previous binding will be insufficient and some negative trials will result. Thus, the system does not attempt grasps very far away from the location of the object, that are information poor with respect to the current task.

In experiment three, the task is similar except in order to receive reinforcement it must both enclose and successfully lift the seen object. Again, the success constraint is progressively tighter and we bootstrap our exploration using bindings from the previous tasks.

Finally, in experiment four, we add the description of the object orientation to the perceptual variables and attempt to orient the preshape in a fashion that increases the probability of succeeding in picking up the object.

### 4.3 Modeling the Plan and Execution

We model the agent, world and task using the situated automata formalism. Following the notation of Kaelbling [Kaelbling, 1990], the world is modeled as a triple \((S, A, W)\) where \(S\)
is a set of states, $A$ is the set of actions available to the agent and $W$ is the world transition function $W: S \times A \rightarrow S$ that maps a state (context) and an action into a new state. $I$ is a perceptual function $I: S \rightarrow \mathcal{I}$ that takes an environmental state and maps it into the set of perceptual states $\mathcal{I} = \{P_1 \times \ldots \times P_n\}$. Here $\{Q_1 \times \ldots \times Q_m\}$ are the action attributes that determine the instantiation values for the primitive actions selected.

A plan is specified in terms of a set of state nodes $S = \{s_1, \ldots, s_n\}$, a set of transition rules, $\delta = \{T_{ij} \mid \exists$ a planned action to transit $s_i \rightarrow s_j\}$. A transition rule $T_{ij}$ is a tuple $(s_i, s_j, L_{ij}, R_{ij}, a_{ij}, M_{ij})$ associated with two nodes $s_i, s_j, i \neq j, 1 \leq i, j \leq n$, and the transition from $s_i$ to $s_j$ occurs when the predicate $L_{ij}$ is valid by the execution of an action $a_{ij}$. The function $R_{ij}$ is the reinforcement function for that transition.

The task is executed by monitoring a set of variables to recognize the current task state and then executing the appropriate action by looking up the appropriate $M_{ij}$ action binding function that corresponds to the current state $s_i$ and parameterizing that action according to the perceptual state and some member of the returned set $Q_{feas}$ of feasible action parameterizations.

The states $s_i$ refer to meaningful points in the progression of the execution of the task. They are defined by a range of perceptual attribute values that must hold in order for that state to be currently valid. $L_{ij}$ is a boolean predicate is valid when the currently monitored perceptual attributes are all within their specified ranges for the transition template. When $L_{ij}$ becomes valid, the state tracking automaton advances by executing the next action in the plan. The generation of the plan and observer automaton is an active area of research [Sobh, 1991] but beyond the scope of this work. We assume that the system is provided with a set of action primitives and robust state recognition primitives.

Each action is parameterized by an associated binding function $M_{ij,a}$ that tries to achieve a given state transition from state $i$ to $j$ by parameterizing the chosen action $a_{ij}$. It determines the intervals for allowable values of motor attributes based on the current values of perceptual attributes, i.e. $M_{ij,a}: P_{(1)} \times P_{(2)} \times \ldots \times P_{(n)} \rightarrow Q_{feas}$ with $Q_{feas} \subseteq 2^Q$, where $Q = \{Q_1 \times Q_2 \ldots Q_m\}$, are the motor attributes. Here $Q_{feas}$ represents the set of allowable action parameter binding intervals that will achieve state $j$ with greater than some threshold probability $p$ which is based on the prediction generated by the learning mechanism.

Notice that our formalism differs significantly from the normal notion of situated-automata based learning, where normally we have a set of fixed actions to be chosen from
Figure 1: A graph specifying the sequence of states and transitions rules $T_{i,j}$ for an object retrieval task

and reinforcement that is not immediately available, such as in the various dynamic programming approaches for delayed reinforcement [Sutton, 1990; Watkins, 1989]. We have a continuum of parameter values with which we can execute an abstract action and are interested in learning the parameterization function $M_{ij}$ for the chosen actions. Our focus is on learning single step actions and determining the characteristics of $M_{ij}$ for some given state transition in a multistep sequence. Therefore, we will have the robot repeatedly practice the transition from $s_i$ to $s_j$ using action $a_{ij}$ and storing the values of $P, Q$ and $R_{ij}$ over repeated trials. The implication is that we work with tasks where reinforcement is locally (or immediately) available.

$R_{ij}$ is the reinforcement function associated with the action $a_{ij}$. It computes the effectiveness of the previously selected action with its associated parameters. Although $R_{ij}$ can be a real-valued function, we constrain it to be binary valued. This is consistent with the all-or-none measurements for task success which will be used for grasping. It returns 1 when $s_j$ (the desired state) follows $s_i$ after some action is executed and 0 otherwise.

The choice of the reinforcement measurement function is critical and task-dependent. In general we pick a reinforcement function whose value directly correlates with the desired state of affairs such as the achieving of a subgoal. The function should depend on environmental parameters that can be cheaply and reliably sensed.

4.4 Reinforcement Learning as a Form of Multivariate Regression

We may view reinforcement learning in the following way: reinforcement learning methods seek to characterize the distribution of reinforcement in the attribute space in which they operate. We view this distribution as a prediction surface. This surface forms a memory
that characterizes the distribution of the reinforcement much in the way a histogram over a parameter space does. Bins with greater ratio of successful outcomes to failures have larger values than those with lower ratio. Given a partial set of indexes into this histogram determined by what is currently perceived, we search the unconstrained indices of this histogram for values that select regions of the state-space that have a high success to failure ratio. In the same way, our reinforcement surface may have peaks and valleys of reinforcements corresponding to combinations of attributes that lead to success or failure.

The question then arises: why not just histogram the space directly and model this reinforcement surface as piecewise constant? Unfortunately, it is impractical to discretize and histogram a high dimensional state space directly, since the majority of the bins would be empty if we demanded reasonable resolution along each dimension. This is a manifestation of the curse of dimensionality, since the number of bins and samples necessary to characterize this distribution increases exponentially with the dimensionality of the state space and the resolution along each axis.

Since it is unlikely that we will encounter exactly the same percepts as previous trials, the memory must have good generalization (smoothing) properties. Generalization allows the memory to interpolate to novel instances that are similar to previous instances. Non-parametric regression techniques [Eubank, 1988] are ideal for this because of their interpolation properties given sparse measurements and their ability to tolerate noise in descriptive points that determine the surface.

We create this surface using a form of multivariate statistical regression called Pro-
jection Pursuit Regression (PPR) [Friedman, 1985] developed specifically for use in high-dimensional spaces \(d \geq 3\). It is used to approximate the distribution of reinforcement likelihood in the parameter space. Such projection-oriented techniques must be used in order to work with the small sample sizes required in learning, since there is a cost associated with completing each trial. Projection pursuit algorithms have many desirable properties that will be discussed in section 4.8, especially in comparison to locally-weighted techniques.

Having such a predictive surface yields several benefits. It is a tool for guiding task execution and subsequent learning because it provides a means of compactly characterizing the peaks in reinforcement in the space of relevant sensorimotor attributes. For example, if the surface is a function of both perceptual and action attributes \(P\) and \(Q\), then given a set of perceptual observation, we can search for values of action parameters that, when combined with perceived parameters, are inside of regions that have high reinforcement values. We then send those action valuations out to the actuators expecting to receive a desirable outcome.

Rather than searching this function for maxima each time we have partial query, we can threshold the surface and pre-partition it into regions that have a high predicted reinforcement storing them in an associative data structure such as a binary tree. This domain information may then be indexed in a rapid manner, allowing the reinforcement distribution to be efficiently accessed for decision-making during real-time execution of the actions. The iso-reinforcement surfaces of the volumes then become decision hypersurfaces whose projections onto the action parameter axes can provide feasible, as well as preferable, intervals for parameterizing a given perception/action pair.

### 4.5 Statistical Learning as a Form of Induction

It is useful to think of inductive learning as a process of searching for regularities and structure in data sets. It is a data reduction mechanism applied to stored experiential information. The discovery of such regularities corresponds to the induction of generalized rules about the data set. A system working in a continuous-valued world must be capable of data reduction from a real-valued domain to a level of granularity that permits the system to function effectively.

As an example, consider the simple task illustrated in Figure 3. We define some parameterized action primitives and simple sequencing order for the action primitives. After
taking some number of measurements of the reinforcement for different parameterizations of actions and perceived object locations in the attribute space (see Figure 4(a)) we attempt to form a least-squares response surface (see Figure 4(b)) that is used as an estimation function. This function returns the predicted reinforcement for combinations of the sensorimotor attribute valuations. The form of this function is a non-parametric least-squared fit of the data or possibly, some other non-parametric means of characterizing modes and widths of the distribution. In either case, smoothing such a distribution allows a generalization to novel instantiations over a given range using the properties of interpolation and smoothing afforded by the regression.

In order to represent the regions of high reinforcement in an efficient manner, a $2^k$-tree representation of hyper-rectangular volumes in the $k$-dimensional parameter space is used (see Figure 5 (a)). This allows arbitrarily shaped regions to be represented as unions of hyper-rectangular volumes of varying size that are accessible using time efficient $2^k$-tree structures for their storage.

Once we have an $2^k$-tree representation of the desirable regions, the question is: how do we utilize and index this information in a useful and efficient manner? Since we have the information stored in a tree representation, we can perform an associative search based on the attributes that are currently being observed. The result of such a process is illustrated schematically in Figure 5(b) where a given observation indexes through to associated vol-
Figure 4: (a) A hypothetical raw scatter plot of reinforcement values obtained by executing an action with different perceptual states of affairs. Each point in the plot has a reinforcement mass associated with it determined by the degree of success of the action. (b) A hypothetical contour plot of a non-parametric regression fit to measured data that attempts to smooth and predict the reinforcement mass density over the entire domain of input parameters.

The construction of a $2^k$ tree requires $O(2^{km})$ evaluations of the thresholded function, where there are $2^m$ intervals along an axis. This bound results from the recursive space subdivision of the parameter space for a full resolution decomposition, and is unacceptable for reasonable $k$ and resolution $2^m$ along each axis. In order to circumvent this, we make the assumption that the vast majority of the parameter space results in no reinforcement, otherwise the robot’s task would be trivial. Therefore, it makes little sense to exhaustively evaluate the function in regions that have low probability of success.

We propose a probabilistic approach to deciding whether or not to subdivide a region. The approach is similar in spirit to probably approximately correct (PAC) learning as developed by Valiant [Valiant, 1984] (see page 20). A threshold minimum probability of success $p_{above}$ is chosen by the user based on the task. This is the probability that a randomly chosen combination of percept and action values (from a uniform distribution) in
Figure 5: (a) The quadtree ($2^2$-tree) reduction of the regression surface. (b) Execution–time indexing and retrieval of domain knowledge after learning. The perceptual parameter, the observed location of the object is mapped via an associative lookup to an allowable interval that determines where the robotic hand is to be placed.

Figure 6: Binary Tree for indexing perceptions and actions. The tree is of non-uniform depth, determined by the distribution of reinforcement in the parameter space. Each node has a set of intervals associated that index the leaves beneath it.
the parameter space will succeed and have reinforcement above the threshold. Given \( p_{\text{above}} \),
the user also chooses \( p_{\text{thresh}} \). This is the probability with which at least one of a number
\( n \) of uniformly distributed queries will be above reinforcement threshold in the region if
the underlying actual probability of success is greater than \( p_{\text{above}} \). Therefore, \( p_{\text{thresh}} \) is the
probability that we detect that a success subregion exists in a region if it is present. If
a success subregion exists, then we subdivide further to attempt to characterize it with
further resolution. Thus, if the \( n \) queries return function values which are uniformly success
or failure, then we can say they are uniformly success or failure respectively, with the above
certainty limits.

The value \( n \), which is the number of exploratory queries is determined as follows. Let \( X \)
be the event which is the number of times the function is above threshold out of \( n \) queries
and \( Y \) be the event which is the number of times the function is below threshold out of \( n \)
queries. Then we have:

\[
p(X = n) = p^{n}_{\text{below}} = (1 - p_{\text{above}})^n \leq 1 - p_{\text{thresh}} = 1 - p(Y \geq 1)
\] (3)

From this follows:

\[
n \geq \frac{\log (1 - p_{\text{thresh}})}{\log (1 - p_{\text{above}})}
\]

So we must have \( n \) or more queries in the region to assure this \( p_{\text{thresh}} \) probability.

The adaptive algorithm using this approach is presented in table 2.

### 4.6 Function Learning

Using regression allows one to build up a predictive mechanism for future success as a
function of what the robot is observing and its action parameterization. This amounts to
learning the expectation of reinforcement value (the reinforcement surface) conditioned on
the valuations of the perceptual (\( P \)) and action (\( Q \)) attributes \( E(R | P, Q) \) from a series of
noisy and sparsely spaced observations. This is exactly the problem which multivariate sta-
tistical regression techniques are designed to solve. If a smooth function well-approximates
the underlying distribution, then we can extrapolate and interpolate this expectation function
to novel sensorimotor instances. In other words, we have a system that is able to
generalize with respect to the action parameters. This smoothness constraint is based on
Algorithm `adaptive_2k_tree(nodex_ptr cur_node,
leaf cur_leaf,depth,fun,P_{thresh},P_{above})`

if depth == 0 then

Label cur_node according to fun evaluation in interval and return

Compute number of evaluation $N$ for desired confidence value $P_{thresh},P_{above}$ in current interval from eqn. 4.5

Evaluate Function Outcome at $N$ points

if uniformly success then

begin

cur_node.outcome := success;
cur_node.l := cur_node.r := λ; return;
end

else if uniformly failure then

begin

cur_node.outcome := failure;
cur_node.l := cur_node.r := λ; return
end

else

begin

cur_node.l = new( node )
cur_node.r = new( node )

adaptive_2k_tree( cur_node.l, left(cur_leaf),
depth-1, fun, $P_{thresh},P_{above}$)

adaptive_2k_tree( cur_node.r, right(cur_leaf),
depth-1, fun $P_{thresh},P_{above}$)

end

Table 2: The Adaptive $2^k$-tree Creation Algorithm
Figure 7: Adaptive $2^k$ Generation. (a) Shows a closed unit circle function, whose output is 1 everywhere inside and 0 outside, along with the resulting quadtree $2^2$-tree generated. (b) Shows the points where the function was evaluated during the course of the tree creation. The process is uncertainty seeking, evaluating the function with higher density of samples along the decision boundary, and less so in highly uniform outcome areas.

The assumption that the function we are attempting to learn has smooth input/output behavior.

The idea of learning a function by a set of example input/output pairs has been used extensively in robotic learning for control. A common approach has been to use lookup tables with interpolation between measured points. Indeed, non-parametric statistical regression on a set of measurements may be considered to subsume these techniques. An early example of table lookup is Albus' CMAC [Albus, 1972]. More recently, Atkeson et al. [Atkeson, 1991; Aboaf et al., 1989a; Aboaf et al., 1989b] have explored task level robotic learning using polynomial interpolation as well as non-parametric locally weighted regression with some success. These techniques are designed to be more robust with respect to noise in the training samples. Mel [Mel, 1991] has used a connectionist approach approximate functions of several variables. These approaches are effective in some cases, but in general, suffer from high sample size requirements as the dimensionality of the input space increases. The problem of dimensionality in the learning space has been discussed to some extent by Moore et al.[Moore, 1991b] and Omohundro [Omohundro, 1987b] in the context of robot control learning.
4.7 The "Curse of Dimensionality"

All of the above approaches suffer from the "curse of dimensionality." The "curse" can be defined as the need for exponentially larger sample sizes as the dimensionality of the input space increases. A common illustration is as follows [Huber, 1986]. Consider a locally weighted regression or interpolation scheme that relies of 10% of the total samples for making an estimate of a given query point. Assume we are interested in the function over the domain of a unit 9-dimensional hypercube. If we assume uniform distribution of exemplars over this cube, then we must have 10% of the volume of this 9-d cube, i.e. \((f_i)^9 = .1\), where \(f_i\) is the fraction of the unit distance along each axis. Then \(f_i = (.1)^{1/9} \approx .77\) which is a huge portion of domain. If we attempt to narrow \(f_i\), then the fraction of volume necessary for the local fit rapidly decreases, and in order for it to contain sufficient number of points for a reasonable estimate, a huge number of samples is required. For this reason, most table lookup approaches have been applied primarily to lower-dimensional functions.

The problem of dimensionality is more than a theoretical curiosity, as was discussed in section 3.15. In evaluating the non-parametric regression approach to reinforcement learning, we first tried locally-weighted regression (LOESS [Cleveland et al., 1988]), then MARS spline regression [Friedman, 1991], and finally projection pursuit (SMART) [Friedman and Stuetzle, 1981; Friedman, 1985]. The methods were tested by attempting to fit the four-dimensional Gaussian Function

\[
f(x_1, y_1, x_2, y_2) = e^{-69.30[(x_1-x_2)^2+(y_1-y_2)^2]}
\]  

This function reaches a value of .5 at distance .1 from its center. The fitting results from 400 randomly selected points from the Gaussian function are shown in Figure 8. It can be seen that the locally weighted regression (LOESS) is unsatisfactory, while the projection pursuit method works quite well. A number of differing fit parameters were tried with the locally weighted regression with none yielding success. The MARS algorithm also showed similar unsatisfactory performance. Therefore, we selected the projection pursuit regression (SMART) approach as developed by Friedman et al. [Friedman and Stuetzle, 1981; Friedman, 1985].
Figure 8: A sample fit on a four dimensional Gaussian function in the $x_2, y_2$ space. (a) The underlying function plotted with $x_1 = .26$ and $y_1 = .1$. (b) The resulting fit using LOESS locally-weighted regression. Here $f = .5$ says that half of the observations are used in the local query estimate. (c) The same function plotted at $x_1 = .6$ and $y_1 = .6$. (d) The resulting fit using projection pursuit regress (SMART). It can be seen that the projection pursuit method performs much better than the locally weighted approach. Values of $f$ were varied widely in testing the LOESS fit with no appreciable increase in performance. The data input consisted of 400 example points in the unit cube derived from 20 uniformly distributed sampling regions with 20 random points in each region. Each sampling region has .2 dimension.
4.8 Projection Pursuit Non-Parametric Regression (PPR) Methods

We describe the Smooth Multiple Additive Regression Technique (SMART) of Friedman [Friedman, 1985]. Assume we have some underlying function \( y = f(x_1, \ldots, x_p) \) that we wish to approximate from some set of noisy observations \( \{(x_{11}, \ldots, x_{p1}, y_1), \ldots, (x_{1n}, \ldots, x_{pn}, y_n)\} \) where there are \( n \) learning trials. In our case, \( Y \) is either a binary success or a failure reinforcement value \( R \), although it could be real-valued, and \( X = (P, Q) \), the combined perceptual and action values that resulted in that reinforcement value. Each observation may also be scaled by some weighting factor \( w_i \). Assume the observations come from the following process:

\[
Y = f(X_1, \ldots, X_p) + \nu
\]

where \( \nu \) is a noise inducing random variable. In regression we endeavor to estimate the conditional expectation

\[
\hat{Y}(x_1, \ldots, x_p) = E[Y \mid X_1 = x_1, \ldots, X_p = x_p]
\]

The SMART method searches for an expansion of the form

\[
\hat{Y} = \hat{f}(X_1, \ldots, X_p) + \hat{f}(\tilde{X}) = \tilde{Y} + \sum_{i=1}^{M} \beta_i g_i(\alpha_i^T \tilde{X})
\]

where \( g_i(z) \) is a smooth "ridge" function of scalar \( z \). Here \( \alpha \) is the unit direction vector that projects the input covariates and \( \beta \) is a scalar weighting coefficient. The approach is, therefore, to simultaneously find "good" projection directions of the data and smooth functions \( g_i(z) \) that are the smoothed versions of the set of values \( \{(z_1, y_1), \ldots, (z_n, y_n)\} \), where \( z_i = \alpha^T [x_{1i}, \ldots, x_{pi}] \). By a "good" choice of direction vectors, weighting coefficients, and smooth functions, we mean those that minimize the unexplained variance of the case responses along those projections and mapped through the smooth functions.

Since the \( g_i() \)'s are the smoothed versions of all of the cases projected onto one dimension, achieving a large enough sample size is much less of a problem than methods that form estimates over the raw high dimensional neighborhoods, since we are collapsing all data points onto a unidimensional subspace.

The search for the parameter set minimizing the fitting error is done using standard minimization techniques and by grouping the parameters, holding some fixed and minimizing the others in turn, so that the residual error is decreased.
PPR can also be used to solve categorical classification problems [Friedman, 1985], that is, to come up with assignment rules conditioned on \((X_1, \ldots, X_p)\) that minimize the classification risk for a categorical response variable that takes on only one of a set of discrete and unordered values. This is useful when only binary reinforcement (a thresholded success or failure) of the task goal is available. The risk of misclassification is defined as

\[
R = E[min_{1 \leq j \leq q} \sum_{i=1}^{q} l_{ij} p(i \mid X_1, \ldots, X_p)]
\]

where \(l_{ij}\) is the loss for predicting \(Y = c_j\) when in actuality its value is \(c_i\) and \(p(i \mid X_1, \ldots, X_p)\) is the conditional probability that \(Y = c_i\) given some valuation for the predictor variables. The \(l_{ij}\) allows the incorporation of an adjustable loss in the classification. The conditional probabilities are then estimated and \(j^*\) that minimizes the \(R\) is chosen as the class for a given observation. This is the form we utilize for the experiments that follow and in table 3.

Therefore, PPR may present some advantages with respect to interpolation schemes such as CMAC etc., as well as the flexibility of non-parametric regression techniques without the problem of poor sample economy in higher dimensions.

### 4.9 Experimental Plan

We assume the following innate perceptual data reduction procedures and representations for the experiments:

1. The robot is able to discriminate the target object from the background using a simple binary thresholding procedure.

2. The object is reduced to a single location in the visual coordinates of the camera system. The location is defined to be the visual centroid of the object. Superquadric shape representation is given to the object if called for by the current experimental level.

3. Only one object at a time is presented in the visual scene.

4. Tactile assessment in terms of whether contact occurs with the target object, whether the object is enclosed and whether it is grasped.

The following action primitives support these abilities:
Algorithm **PPR LEARN**($s_i, s_j, P, Q, \delta$)

1. Initialize $M_{ij,a}$ to default mapping; assume success everywhere
2. $i = 0$
3. Do
   (a) Measure percepts $P$ and have robot execute action $a$ parameterized by $a = Q \in Q_{feas}$ region returned by $M_{ij,a}$. Set $R = 0, 1$ based on result of action.
   (b) Store $(P, Q, R)$
   (c) $i = i + 1$
   until $i > \text{sample size}$; sample size is size of batch
4. Perform Regression on current set of $(P, Q, R)$ vectors to generate $E(R | P, Q)$ function.
5. If goodness of fit measure (GOF) $> \text{fit threshold}$ then goto step 2; GOF is normalized explained variance
6. Generate Interval Tree Action Map of $E(R | P, Q)$, namely $M_{ij,a}$, based on minimum expected reinforcement threshold.
7. Goto step 2

<table>
<thead>
<tr>
<th>Table 3: The Flow of the Learning Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>approach-object</strong> has the purpose of moving the robot arm’s gripper to a desired location.</td>
</tr>
<tr>
<td>An orientation specification specifies the gripper orientation relative to the world base-frame.</td>
</tr>
<tr>
<td><strong>preshape-hand</strong> has the function of configuring the hand so that the subsequent grasp primitive can be effective. A preshape to spherical grasp and maximum aperture is chosen.</td>
</tr>
<tr>
<td><strong>lower-hand</strong> hand at the current location to some maximum depth.</td>
</tr>
<tr>
<td><strong>reflex-grasp</strong> which is initiated upon tactile contact. This contact is detected by an instrumented compliant wrist that triggers the hand closure (see Figures 10 and 11). The compliant wrist also permits the exploratory trials to be non-destructive.</td>
</tr>
<tr>
<td><strong>lift-hand</strong> lifts the object from its plane of support.</td>
</tr>
<tr>
<td>The perceptual primitives are:</td>
</tr>
<tr>
<td><strong>object-location</strong> returns the $(x, y)$ location of the target object in the image frame.</td>
</tr>
</tbody>
</table>
Figure 9: Schematic of the planar object retrieval task from the top camera view. Here, the object frame \((O_x, O_y)\), the hand frame \((H_x, H_y)\), and the no contact/contact \((R = 0, 1)\) reinforcement values are stored for each trial grasp.

**object-orientation** returns the orientation of the major axis of the object (if it has one) relative to the image coordinate frame.

**assess-hand** returns a triple of \(f_1, f_2, f_3\) which determine whether the fingers are contacting the target object or not. The finger states \(f_i\) are either 1 or 0 respectively, depending on whether they contact the object or not.

**wrist-deflection** returns the amount of deflection in the wrist upon contact with an object.

### 4.9.1 Phase 1: Tactile Reinforcement and Preliminary Results

This experiment has already been performed. It consists of learning to position the gripper in the plane given a visual observation of the centroid of the object in the plane from a top view (see Figure 9) so that the system would receive any tactile stimulus. We do not assume that the visual and action frames are related with some \textit{a-priori} transformation; the learning algorithm will determine the appropriate transformation. We use the projection pursuit method to form an estimate of \(E(R \mid O_x, O_y, H_x, H_y)\) (see Figure 9).

The action primitives are sequenced in the order **approach-object**, **preshape-hand**, **lower-hand**.
A reinforcement function is defined for this level which returns 1 whenever a tactile contact with an object is achieved and returns 0 otherwise.

A memory also exists that stores each parameterization of the move-to location \((H_x, H_y)\), the visual location of the object \((O_x, O_y)\) and the outcome as measured by the reinforcement function \(R\) for each trial.

A workspace was defined in which the object to be grasped may be placed at random. The object was a .22 Kg aluminum soda can (12 cm tall, 6 cm diameter) covered with white paper to simplify vision processing. The workspace was a rectangular 80cm by 40cm area.

A pair of numbers in the workspace are generated by a random number generator. The experimenter (human) manually positions the target object at that position. The robot arm is retracted from the workspace and the CCD camera vision system acquires a top-view digitized image of the scene. The vision software then thresholds the scene based on intensity mean, grows 8-connected regions and culls the regions by a minimum-area criterion to eliminate artifact and noise-induced smaller regions. The surviving region is then used to compute a superquadric fit and the centroid of this superquadric is stored as the position of the object. Since the task is a two-dimensional one, a monocular camera view is sufficient to determine the location of the object.

The grasping trial set consists of the following actions. The arm is retracted upwards and laterally out of the workspace to prevent visual occlusion and a visual sample is taken. The system then computes a bounding box of 40 cm around the location of the object and chooses a random location in that box from a uniform distribution. The robot hand then moves to that location and begins a downward motion. This motion is terminated by one of two conditions. Either a wrist displacement is sensed and the trial labeled a success or a positional stop at 8 cm above the table is reached. If the wrist displacement occurs, a new trial set begins by generating a new random location for the target and the object is repositioned. If the arm positional stop is reached, then the arm missed the target and the given grasping trial is labeled a failure. The arm then retracts and tries again at another random point in the bounding box. Up to 20 consecutive failures are permitted, after which a new random target location is generated and the object repositioned.
Approach  Contact  Reflex

Figure 10: The three phases of approach, contact and reflex initiation.

Figure 11: The initiation of the grasp reflex. A relative cartesian deflection or equivalent-angle axis rotation of magnitude above the given threshold value is detected by the wrist and causes the arm to stop. The hand then immediately closes.
Figure 12: The Experimental System. The PUMA560, Penn Hand and Wrist are controlled and coordinated using a MicroVAXII with shared memory. The MicroVAXII sends commands via a serial link to a high-level controller which interprets commands and servos the hand configuration to a desired force or position. The CCD camera output is digitized on the MicroVAXII and processed on the SUN4/260 via an Ethernet connection. The only real-time sensitive component of the system is the connection between the wrist and arm which occurs via the shared memory connection.

4.10 Hardware Setup

The experimental system consists of a PUMA 560, instrumented compliant wrist and Penn Hand controlled and coordinated using a common MicroVAXII with shared memory (see Figure12). The Penn Hand [Ulrich et al., 1987] is controlled using a serial link to a high-level controller which interprets commands and servos the hand configuration to desired forces or positions. The CCD camera output is digitized on the MicroVAXII and processed for a superquadric fit using on a SUN4/260 via an Ethernet connection. The only real-time sensitive component of the system is the connection between the wrist and arm which occurs via the shared memory connection within the MicroVAXII.
4.11 Results

Figure 13 represents a histogram for the three hundred and three grasping trials which were actually performed in the workspace. This figure illustrates a rough outline of the shape of the hand, since a collision causes a wrist displacement, no matter where it occurs on the hand. One can view this figure as the resulting image of the hand as yielded by the object being used as a probe to trace out the presence or lack of the hand. Notice that the width of the “fingers” of the histogram is approximately 6cm (which is the diameter of the can). Therefore the histogram also encodes information about the target object as well as the hand itself.

This data gathered from these experimental trials was used to generate an augmented data set which consisted of simulating the process of positioning the object at 20 uniformly distributed random points in a rectangular workspace of ±1m around the base of the robot. At each of the different locations, 100 points from the experimental data were rotated by a random $\theta$ in $[0,2\pi]$ and translated to the current simulated object location. This process yielded the 2000 simulated trial points shown in figure 14. Each instance is recorded as $O_x, O_y, H_x, H_y, G$, where $(O_x, O_y)$ is the perceived object location, $(H_x, H_y)$ the hand position and $R$ is either a “0” for no contact or a “1” for contact. This corresponds to randomly orienting the hand and moving it to a random point in a 40 cm by 40 cm interval around the object, and moving the hand downwards to see if it contacts the object. The larger points in Figure 14 indicate successes and the smaller points indicate failures.

The projection pursuit algorithm classification was run on this data (SMART Routines Version 10/10/84 [Friedman, 1984]) and yielded the results depicted in Figure 15. After training, the classification function was able to correctly predict, given the perceived location of the object in the plane, whether placing the hand in a given location would yield a tactile percept in a region ± 1m of the base of the robot. It was also able to generalize to regions of the workspace where empirical information was not taken as illustrated by Figure 15 which shows the correct classification given that the object is in location (−.6m, 0m) (see Figure 15 (a)), although this position was not in the learning set. Figure 16 gives a higher resolution picture of the success regions in Figure 15.
Figure 13: Histogram of centered data from 303 trial grasps in the workspace. It depicts the raw data centered at the perceived centroid of the target object and is the proportion of success to total trials summed over 50mm regions of the workspace. Notice that since the hand orientation was fixed for all trials, the histogram outlines the shape of the hand in the spherical grasp configuration.

Figure 14: The raw data used for the learning that was generated by using the empirically obtained data distribution.
Figure 15: Resulting classification based on $E(R \mid O_x, O_y, H_x, H_y)$ for objects positioned at (a) $(O_x, O_y) = (-600,0)$ mm, (b) $(O_x, O_y) = (800,600)$ mm, (c) $(O_x, O_y) = (400,-800)$ mm and (d) $(O_x, O_y) = (-1000,-800)$ mm relative to the robot base. Each density plot in the $H_x, H_y$ space represents the resulting classification where white represents an expected tactile stimulus and black represents a miss in robot-centered coordinates. From this, it can be seen that a correct decision rule for placing the hand has been induced since the spot tracks the object location. The results generalized to all positions in the plane.
Figure 16: (a) The theoretical greater than .5 contact probability region for the tactile stimulus, (b) The learned region for an object at (0mm,0mm) and (c) The learned region for an object at (300mm,200mm). Units are in millimeters. The graphs are in object-centered coordinates.
4.12 Discussion

The result shown above identify several important issues. First, the PPR method exhibits an inductive bias which searches for invariances of arbitrary distributions under affine transformations. However, since the fitting of the projected data uses a non-parametric smoother, it does not exhibit bias for preferring certain distributions over others (i.e. multimodal versus unimodal). This is advantageous in the case that the task being learned can be satisfied by several action valuations (different functional maxima) especially in conjunction with the $2^k$-tree. The $2^k$-tree is a very powerful structure for representing functional maxima. Consider training a back-propagation type neural network for a sensorimotor task. It is quite possible that radically different action parameterizations can lead to reinforcement when a given percept is seen. In that case, a network trained with $(P, Q)$ exemplar pairs would have conflicting information. Its $P \rightarrow Q$ mapping would have widely varying $Q$ values for a given $P$. This learning set would have to be clustered based on the distribution of $P$ and $Q$ and all alternatives except for one, discarded. Representing reinforcement as a function of both $P$ and $Q$ and using the $2^k$ tree to return the entire feasible set $Q_{feas}$ allows for multiple actions for a given perceptual state of affairs to be learned. This set $Q_{feas}$ can be returned to a high level planner that can incorporate other constraints in selecting which member of $Q_{feas}$ is ultimately executed.

As an example, consider the task of mating a smooth part into a cylinder with both ends uncapped. The distribution of success would have two peaks $\pm \pi$ relative to the cylinder coordinate frame. The corresponding tree representation of Figure 5 (b) would then capture the feasible bi-modal distribution of valuations for a given cylinder position. This information would then be provided to a higher level spatial planner which would use it in its plan building. This is in contrast to a connectionist type learning system which would not encode possible alternatives explicitly if they existed.

It is clear that there are several important issues in the application of this technique. The first tradeoff is between the width of the distribution of successes and the total size of the workspace. If the width is too large relative to the workspace or the sampled locations are too close together then the finding a projection direction vector which organizes the data and minimizes its variance is ill-conditioned since the ensemble variance varies little as a function of the direction chosen. This was evidenced by the fact the results for the fit on positions in the original workspace (40cm by 80cm) were poor given that the diameter
of the distribution relative to the object center is approximately 40cm, the physical hand span. Augmenting this data set using empirical data as a base and increasing the domain size to ±1m of the robot obtained the successful result shown. At the other extreme, if the width of the distribution is too small relative to the workspace, then the sample economy of the learning process will be small—many trials have to be attempted for a success to be logged causing the learning process to converge very slowly.

The results so far show that it is possible to learn the task of hand placement in the plane so as to increase the probability of a tactile stimulus occurring. This initially learned mapping to will now be used to guide exploration using the bootstrapping approach described in section 4.2.

4.12.1 Phase 2: Enclosure

In this proposed experiment\(^1\), the system must learn to position the hand so as to enclose the object. The action and representation primitives and the sequencing are the same as in the first experiment except that the grasp reflex (see Figure 11) is now enabled. This reflex consists of the instantaneous closure of the finger until either a object interaction is detected on each of the fingers or a desired position is reached with no contact.

The assess-hand function works as follows: If the positional stop occurs, then the grasp attempt is termed a failure since the finger reached its maximum position without encountering the object. The outcome \( R = 0 \) of this event is logged and a grasp trial set begins again; if 2 of the 3 fingers are still contacting the object, the finger states are logged \( R = 1 \), the arm is retracted upwards for 10 cm and the contact information saved. If the object was not enclosed at the end of the trial, another image is acquired and processed and another trial set begins.

The tasks differs in two additional ways:

1. The binding function determined by the regression in the first experiment is used to constrain the search interval for each new learning trial for this task. This results in a much higher likelihood of enclosure and so each trial is more informative with respect to the task.

\(^1\)Due to reliability problems with the three fingered gripper in phase 1, a two fingered industrial gripper (LORD) will be used in phases 2,3 and 4
2. The reinforcement function is now more stringent. The hand must now be contacting the object with at least two fingers.

4.12.2 Phase 3: Lift Task

This experiment will be basically identical to experiment 2, except the search is now limited to actions that are expected to succeed based on what will be learned in the second experiment. The reinforcement function is now altered to return 1 only if the object is still held as defined by \texttt{assess-hand} after the hand is retracted above the workspace using \texttt{lift-hand}.

4.12.3 Phase 4: Axial Task

This final experiment will incorporate an orientation component in both the coordinate frames of the object in the visual frame and the motor component of robotic move-to action. The reinforcement function is unchanged from experiment 3.

4.13 Conclusions

The previous experimental plan allows us to verify the validity of the Gibsonian approach to sensorimotor learning. In addition to carrying out and analyzing the performance of the learning methods in the sequence of experiments proposed, a series of computer simulations will be carried out. These will characterize the learning algorithm's noise immunity, sensitivity to dimensionality, and its ability to reject nuisance variables.
5 Action-Category Learning using Density-Adaptive Reinforcement Learning

5.1 Introduction

In this chapter we focus on the following problem: given a set of high level action categories and a set of objects to be grasped, how can we use human expertise and insight to train a system to use appropriate hand preshapes and approaches? We would also like the system to be adaptive so that it will rapidly learn to handle new objects as they are encountered in the environment. Given that we initially adopt a supervised learning paradigm, the question arises: how does the system handle inadequacies in the human's initial advice that turn out to conflict with the robot's experiences?

We propose a hybrid system that learns to recognize different basic-level interaction categories and generate the binding functions corresponding to each of those categories (see Figure 17). The binding functions map sensed quantities into motoric values.

This approach builds on the results of the previous section by integrating real-valued binding function learning with higher-level category learning. To accomplish the category learning we propose a Density Adaptive reinforcement learning (DARLING)\(^2\) algorithm and a forgetting algorithm to track changes in the behavior of the environment. This algorithm uses statistical tests to identify possibly "interesting" regions of the attribute space in which the dynamics of the task change. It automatically builds high resolution descriptions of the reinforcement in transition areas, and builds low resolution representations in regions that are either not populated by exemplars in the given task or have exemplars that are highly uniform in outcome. This classification learning algorithm is used together with the parameter binding learner described in section 4 to determine the real-valued parameters to be used with the selected high-level action categories.

5.2 Problem Definition: Learning Grasp from Object Shape and Pose

We define the grasp preshape problem as follows: determine the set of feasible preshape/approach direction/twist combinations for a selected target object described in terms of its superquadric extent parameters and pose relative to direction of gravity \(\bar{g}\) (see Table 4). The

\(^{2}\)Acronym due to Dr. Max Mintz
**Inputs:**
The object location $x, y, z$, orientation, extents $a_x, a_y, a_z$ and pose relative to the gravity vector $\vec{g}$.

**Outputs:**
Actions selected from among the feasible abstract action categories along with their corresponding action parameterization intervals.

Table 4: The Categorical Vision-Guided Grasping Problem

Grasp preshapes could be, for example, pinch or cylindrical. The approach direction refers to the axis along which the hand approach the object in the objects centered coordinate frame. The twist refers to the alignment of the hand with respect to the coordinate frame axes orthogonal to the approach.

We do not address the learning of control strategies for the fingers of the hand once an object is contacted, although this is an important problem. Instead we have a fixed control strategy and try to find kinematic parameters for the hand/arm actions that will be sufficient for this fixed strategy.

### 5.3 Architecture for Learning

Following the idea of partitioning into symbolic and real-valued parameter binding learning, we parcel the learning process for grasping into two processes at different levels of abstraction, the grasp action category selector and the parameter binder. The grasp action category selector outputs which grasp/approach combinations are feasible from among all of the possible user defined grasp approach combinations. The parameter binder, on the other hand, determines how to parameterize a given action category based on the geometrical description of the object returned by the vision system (see Figure 18). Both learning levels take a real-valued input vector that describes the geometry of the object.

The system learns to make two types of prediction, the first prediction is categorical, in terms of an basic-level approach/ preshape combinations. The second prediction made is a perceptual binding relation that allows the perceived geometry of the object (described in terms of real-valued continuous superquadric descriptions) to predict real-valued parameters that modulate the instantiation of basic-level action categories predicted by the categorical learner.
5.4 Why use Supervised Learning?

We propose to utilize the abilities of the human in grasp choice and transfer this knowledge through use of a learning system with an external teacher. This initial stage is followed by an unsupervised verification and adaptation phase that specializes this knowledge to the sensory and mechanical limitations of the robot and gripper based on the outcome of grasping trials.

The approach is motivated by the fact that the human is remarkably proficient at manipulating a large number of objects for a variety of purposes and we that would like to embed this knowledge in a robotic system to enhance its abilities. The cognitive abilities of humans along with their vast repertoire of everyday experience allows them to bring to bear a vast amount of knowledge to select grasp preshapes. By training using our own experience with different objects in unsupervised learning, as well as observing others people’s interactions in supervised learning, we learn effective rules for selecting grasps.

The adult human has had years of experience in learning such functionally guided grasps along with the benefit of systems specifically designed to control visuo-motor behavior (see
section 2). In practice, we seem to select grasps with almost without thought [Jeannerod, 1988]. The human cognitive system can very quickly bind the action routine for a task-object combination to the correct type of grasp to apply [Napier, 1956; Arbib, 1985]. Each time we are presented with an object to grasp, we generally do not begin with a brute force search through all possible grasps, but instead tend to put objects into categories with respect to their geometrical shape and pose when we are considering how to grasp them. It is this data reduction procedure of objects into action equivalence classes that allows us to cope with the huge variety of artifacts we see in the real world. The main advantage to this categorical representation is that it allows us a form of generalization from which we can benefit from previous experience in grasping objects that are similar in geometry to previous instances. Notice that shape alone is insufficient for grasp determination. The pose of the object with respect to the direction of gravity says much about how we should orient the approach of the hand to the object and preshape it. Additionally, the desired use of the object and the required forces and moments also alters which grasps are usable. For example, we might grasp a tall and unstable object near its top in order to minimize the chance of it tipping over.

5.5 The “Trust but Verify” approach to learning

The learning task consists of two phases. This first transfers knowledge about grasping from the human teacher. The second attempts to verify and adapt that knowledge to the manipulator hardware.

During the supervised phase, classified examples consisting of geometric object descriptions (e.g. superquadrics), object pose relative to gravity and suggested feasible grasps and approach directions are provided by a teacher along with suggested parameters. The system then attempts to generate a prediction for the categories and action bindings. Even though the examples provided by the human may not be perfectly accurate, they serve to initialize the system by providing a reasonable starting point for the system to refine its predictive rules. In the unsupervised verification and specialization phase, the categorical and binding rules are applied on new sets of objects and adapted through experience.

The ultimate aim of the system is to have a mechanism for generating feasible grasping strategies for objects described by their geometry and pose in a task independent form. Once this mechanism is available it can be adapted to specific functions in other tasks. This
proposed subsystem is fundamental since any manipulatory system requires a mechanism for generation feasible grasps. Down-stream from this generator, we may put in task-specific selection mechanisms that decide which among the feasible grasps should be ruled out due to other task considerations.

5.6 Object Representations

Superquadrics are a family of parametric shapes that are used as object shape primitives in a variety of disciplines ranging from computer vision and graphics to modelling of robotic grasping. This representation is selected both for the fact that it is flexible in describing a wide variety of objects and that robust numerical procedures exist for extracting stable shape representations from dense depth data using range scanners [Solina, 1987].

A minimal description of a superquadric shape normally consists of description of the following form \( \{x, y, z, \phi, \theta, \psi, a_x, a_y, a_z, \varepsilon_1, \varepsilon_2\} \) which is used in the following definition and is illustrated in figure 18.

**Definition** : A superquadric surface is defined as the closed surface spanned by the vector \( S \) having \( x, y \) and \( z \) components specified as functions of the angles \( \eta \) and \( \omega \) in the given intervals:
We identify components as $S_x(\eta, \omega), S_y(\eta, \omega)$, and $S_z(\eta, \omega)$. The implicit superquadric equation can be easily derived from the above definition by eliminating $\eta$ and $\omega$:

$$
S(\eta, \omega) = \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = \begin{bmatrix}
a_x \cos^{\varepsilon_1}(\eta) \cos^{\varepsilon_2}(\omega) \\
a_y \cos^{\varepsilon_1}(\eta) \sin^{\varepsilon_2}(\omega) \\
a_z \sin^{\varepsilon_1}(\eta)
\end{bmatrix} \quad \frac{-\pi}{2} \leq \eta \leq \frac{\pi}{2}, \quad -\pi \leq \omega < \pi
$$

Thus, alternatively we can define the superquadric in terms of its implicit equation, as the locus of the points $(x, y, z)$ satisfying the above equation, which is also known as the “inside-outside” function. It is named as such because the value of the left hand side is $< 1$ for points inside of the volume and $> 1$ for points outside, which is useful for a variety of intersection tests in computer graphics and for the shape recovery process in machine vision.

The parameters $a_x$, $a_y$, and $a_z$ determine the size of the superquadric in the $x$, $y$ and $z$ directions (in an object-centered coordinate system) respectively. The $\varepsilon_1$ and $\varepsilon_2$ terms describe the “squareness parameters” that control the sharpness of the shape’s edges. Based on these parameters, superquadrics can model a large set of standard building blocks, such as spheres, cylinders, parallelopipeds, as well as shapes in between.

If both $\varepsilon_1$ and $\varepsilon_2$ are equal to 1, the surface defines an ellipsoid. Cylindrical shapes are obtained for $\varepsilon_1 < 1$ and $\varepsilon_2 = 1$. Parallelopipeds are obtained for both $\varepsilon_1$ and $\varepsilon_2 \ll 1$. In our approach, the model recovery procedure allows $\varepsilon_1$ and $\varepsilon_2$ to assume values in the real interval $[0, 1]$. For values of $\varepsilon_1$ and $\varepsilon_2 > 1$ the resulting parameterized shapes define objects that will not appear in the robot’s domain during training and execution by experimental choice.

As Figure 18 illustrates, $\{x, y, z\}$ determine the location of the centroid of the object in the world space, and $a_x, a_y, a_z$ determine the magnitude of the extents of the object in these directions respectively. The canonical form dictates that $x, y, z$ directions are chosen so that $a_x < a_y < a_z$. The $\phi, \theta, \psi$ determine the $Z - Y - Z$ Euler angles for the rotational component of the transformation matrix that will bring the world frame into correspondence with the $\{x, y, z\}$ axes in the object frame. The Euler angles consists of a rotation $\phi$ around the $z$ axis followed by a rotation $\theta$ around the $y$ axis and a rotation $\psi$ around the $z$ axis.
We constrain the general orientation assumption by requiring that at least one of \( x, y, z \) to be aligned with the world \( z \) axis in the direction of the gravity vector. The direction of gravity is important in deciding whether a grasp will succeed or fail, along with the object shape. We use an equivalent-angle axis to put the axis that has the smallest angle of rotation to the \( z \) axis in register. By default, we also center the action at the centroid of the object, obviating the need to represent \( x, y, z \), since the action is implicitly object-centered. The parameters \( \varepsilon_1, \varepsilon_2 \) are held as constant and equal to 0.1 (high squareness) in order to generate an approximately parallelopiped description of the object.

The recovery of superquadric parameters for a shape is non-unique. This is not a problem in practice because the recovery procedure have side constraints for the parameters that favor certain solutions [Gupta, 1989].

In cases where the actual object has some circular cross section, such as a cylinder or sphere, then the superquadric frame axes parallel to the plane that contains the circular cross section are not uniquely defined. We therefore rotate those axes arbitrarily in a form most convenient to the representation (see Figure 19). If the plane of the circular cross section is perpendicular to the direction or gravity then rotation of those axes is arbitrary.
Table 5: Symbolic Action Attributes and Associated Legal Attribute Values.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palm</td>
<td>Palm Approach Direction</td>
<td>{x, y, z}</td>
</tr>
<tr>
<td>Twist</td>
<td>Orientation of Palm Direction</td>
<td>{0, \frac{\pi}{2}}</td>
</tr>
<tr>
<td>Preshape</td>
<td>Type of Grasp</td>
<td>{Cylindrical}</td>
</tr>
</tbody>
</table>

Table 6: Symbolic Object Description Attributes and Associated Attribute Values.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ax</td>
<td>Smallest Superquadric Extents</td>
<td>[0, \infty)</td>
</tr>
<tr>
<td>ay</td>
<td>Intermediate Superquadric Extents</td>
<td>[0, \infty)</td>
</tr>
<tr>
<td>az</td>
<td>Largest Superquadric Extents</td>
<td>[0, \infty)</td>
</tr>
<tr>
<td>Opole</td>
<td>Object Pose</td>
<td>{x_{up}, y_{up}, z_{up}}</td>
</tr>
</tbody>
</table>

If this plane is parallel to the gravity vector then we must align the coordinate axes along the direction of gravity (see Figure 19(b)).

5.7 Action Categories

Given the canonical superquadric description above, we can define a set of basic-level categories relative to this description describing the universe of possible preshape/approach/twist combinations that describe actions we allow the robot to execute.

We restrict any possible grasp to be a member of one of the following category described by the high level attributes in Table 5. Objects are described by the attributes in 6.

Proceeding as in Figure 20 from top to bottom, we imagine some abstract description of a superquadric with no world-orientation bound to it. We partition all possible object orientations into three characteristic poses relative to the world gravity vector. These orientations are \(x_{up}, y_{up}\) and \(z_{up}\). The \(z_{up}\) pose implies that the object’s \(a_z\) is aligned with the world \(\vec{g}\) axis following the convention that \(a_x < a_y < a_z\), and similarly for \(x_{up}\) and \(y_{up}\).

Assume, for the sake of exposition, that the object is in \(z_{up}\) configuration. In that case we may consider approaching the object from the \(\pm x, \pm y, \pm z\) directions. We rule out approaching from the \(-z\) direction due to the fact that objects to be handled in our domain will have a supporting surface beneath them. Once we have selected the approach direction, we must select how to orient the gripper along the approach direction. We assume
two possible orientations, one aligned with the smaller orthogonal extent and the other aligned with the larger. Exhaustively enumerating these for $x_{up}$ and $y_{up}$ leads a total of 18 possible prototypical approach preshape combinations that are the basic-level categories for the approach-grasp task (see Figure 21).

Each of the action categories have their corresponding parameters that modulate action as determined by the binding functions (see Figure 22). We choose to parameterize the grasp-approach in terms of possible termination positions relative to the centroid of the object on the approach axis using $Z_{min}$ and $Z_{max}$ (see Figure 23). $W$ is the width of the preshape. $X_{min}$ and $X_{max}$ determine the largest offsets for the lateral approach contact points for a successful grasp (see Figure 24.)

The inputs to the learner are the canonical pose $\{x_{up}, y_{up}, z_{up}\}$ and the $a_x, a_y, a_z$ values of the object. The prediction made is which of the six corresponding action categories for that pose are feasible for those shape parameters and what the action parameters are for those action categories (see Figure 25).
Figure 21: The basic-level categories
Figure 22: The binding functions for each basic-level category and their arguments. Each action binders maintains a function that binds the observed object extents to its free variables that are listed within its description.
Figure 23: The parameters for the approach depth (side view)

Figure 24: The parameters for the approach height (side view)
Figure 25: The process of selecting a grasp. The superquadric based vision system determines that the object is in the $z_{up}$ pose. The category recognition level then selects which of the six possible grasp/approach directions are feasible based on the category learning. The action parameters, e.g. the width of the gripper preshape, are then bound based on the corresponding learned binding functions and the perceived object extents.
Table 7: The scales chosen for generating objects, legal objects consist of any combination where $a_x \leq a_y \leq a_z$. The scale factors represent the length factor of each axis relative to the absolute size of each, which is chosen at 1, 2 and 4 inches for a total of 30 possible objects.

5.8 Generating the Learning Set

Since we have adopted a bounding-box representation that captures the moments of inertia of the object along with the definitions of ascending values for $a_x, a_y, a_z$, we can generate a representative learning set of objects to be grasped by the system in a simple fashion. Our objects come from the universe of possible objects that discretize the space of legal $a_x, a_y, a_z$ combinations as indicated by Table 7.

The success or failure of a given grasp will be determined by a simple test. The hand will be moved in a trajectory with accelerations that induce perturbation forces on the object. At the end of this test trajectory, the assess-hand primitive will be used to see if the object is still grasped.

5.9 Proposed Learning Method: Density Adaptive Reinforcement Learning

As discussed before, the curse of dimensionality is an important limiting factor in many learning algorithms. The projection pursuit approach discussed previously attempts to decompose higher dimensional function as a sum of functions of scalars. This type of modelling is advantageous where the distribution of points the agent will receive is expected to be sparse and uniform over the input space. On the other hand in many tasks, the distribution is non-uniform. For example, in a manufacturing task we may have a series of parts that must be assembled to form a small mechanism. If we are interested in having a robotic system learn how to grasp these parts using reinforcement learning, then it makes little sense to try and determine the reinforcement distribution over all possible part shape
descriptions since the overall size of the mechanism limits the component sizes. In that case it would make sense to concentrate the description of the reinforcement by choosing a high resolution representation of the reinforcement in the region of the parameter space that is relevant to the current task and not to try to categorize regions of the state-space that are not populated. This adaptive state-space partitioning approach has been taken before by [Simons et al., 1982] in algorithms for learning control tasks and for learning to characterize the dynamic behavior of the environment around the agent [Moore, 1991b]. Similar approaches have also been taken for recursive partition based piecewise constant regression trees [Breiman et al., 1984] that attempt to decompose functions into constant regions by recursively merging and splitting regions by optimizing an objective function that penalizes for the number of regions and rewards for goodness of fit among those regions. Omohundro [Omohundro, 1987b; Omohundro, 1987a] has also done extensive work in several geometric learning methods, including $k$-$D$-tree based methods for functional approximation and pointed out their benefits. In particular, Omohundro [Omohundro, 1987b] speculates that the uniform leaf probability densities characteristic of $k$-$D$-trees might have desirable properties for use in learning algorithms, and we exploit that property in the proposed algorithm.

Such an adaptive process requires us to have a mechanism for identifying those possibly “interesting” regions of the attribute space and to automatically concentrate on building high resolution descriptions of the reinforcement in those areas, while at the same time ignoring regions of the state-space that are not populated in the given task or are highly uniform.

We reason as follows, in regions with sparse data, it makes little sense to attempt to approximate the function with high spatial resolution since it is impossible to determine the precise location of a decision boundary with few exemplars and so a coarse description of the region will suffice. On the other hand, if the region has a high density of samples, then we might consider the extra computation worthwhile in determining the decision boundaries for concepts by building quadtrees with higher spatial resolution.

Notice, however, that a high local density of samples is not sufficient to warrant a quadtree decomposition of the region. We must somehow decide whether the region is multi or unimodal in its distribution of reinforcement. This is done by determining if the estimate of probability of a region is greater than some threshold with an associated confidence
interval. If the confidence interval is located entirely above or below some threshold success or failure value then we do not attempt to further describe it and assign it as entirely success or failure depending on whether it is above or below. If the area is not well determined because the interval lies over the threshold, then we go ahead and attempt to recursively subdivide the region using a generalized quadtree ($2^k$-tree) representation as described in section 4.5. If recursively subdividing the region does not yield better expected confidence intervals within the leaves of the quadtree, then we must assume that that region of the state-space is non-deterministic up to the maximum resolution of the perceptual and motor system. The remaining actual possibilities are that there is apparent non-determinism due to insufficient resolution or that the world is really non-deterministic in that region of the state-space. In either case the system cannot make a prediction. In fact, the system can request more exemplars in that region of state-space in order to help it disambiguate.

We accomplish this adaptivity to sample density by partitioning the real-valued attribute space into a set of hyper-rectangles with approximately uniform number of samples by generating a $k$-$D$ tree. Then in each of these hyper-rectangles, the algorithm attempts to determine its lumped probability of success by estimating its probability of success and the $(1 - \alpha)$ confidence interval for that probability. If the interval lies entirely above or below that threshold probability then we lump its probability value as appropriate. Otherwise, we build a quadtree in that region to further refine the distribution of reinforcement. As a side effect of the $k$-$D$ process, the quadtree’s domain is smaller in higher sample density regions. Therefore each leaf of the quadtree in high sample density regions has a higher effective resolution when normalized to the entire domain of system. The $k$-$D$ tree effectively “magnifies” relevant regions of the state space.

Now that we have given a flavor of the approach, we describe the of the algorithm in more detail.

### 5.9.1 Using the $k$-$D$ Tree to Adapt to Sample Density

Density adaptive partitioning of the space that contains the learning samples can be accomplished using $k$-$D$ trees as devised by [Bentley, 1975] and enhanced by [Friedman et al., 1977]. The $k$-$d$-tree was originally proposed to solve the multi-key best-match problem in an efficient manner and is capable of finding best matches to a query in a $k$ dimensional space with $n$ items in $O(\log n)$ operations, and requires $O(kn \log n)$ operations to create.
The name “k-D” stems from the fact that they operate on data described by k keys with dissimilarity measure D. They are a variant of binary trees where the root node represents the set of all data points and each child represents a partition of those data points. Each node has two children that represent additional partitionings of the parent node’s set of items. The leaf nodes of the tree contain the final partitionings of the data set. Any non-leaf node selects one of the k dimensions (the ith dimension) as a partition dimension, along with some threshold partitioning value. All items with ith key value less than that threshold value go into the left son and the remainder into the right son.

Originally, the dimension to be selected at any level L in the k-D tree was selected according to the simple rule \(i = L \mod k + 1\) [Bentley, 1975] and the discriminator threshold was determined to be the value of a randomly selected item in the current node’s partition. Optimized k-D trees [Friedman et al., 1977] improved on this approach dramatically by using the following approach. At any level in the tree, we desire to maximize the information provided by the partitioning on a dimension’s threshold. This information is maximized when the probabilities of a given item in the partition falling into the left or right child are equal. This is achieved by choosing the threshold to be the median of the distribution of the key values in the current partition along the currently selected dimension \(i\). The question still remains, how do we select the partition dimension at any given node in the tree. The selection is accomplished by picking the dimension that has the largest range of values. The partition dimension is chosen for the nearest-neighbor search based on the fact that we wish to minimize the probability that the ball containing the query’s m nearest-neighbors intersects the opposing successor partition, since that would necessitate searching the opposing node as well.

The property of k-D trees that permits them to function efficiently is that the number of items in a terminal nodes tends to be fairly uniform, which approximates a balanced binary tree. Therefore, the higher the local sample density of regions in the state-space, the more terminal nodes allocated and the smaller the volume of the nodes in the region (see Figure 26). If we consider taking each one of those nodes as the domain for a \(2^k\) tree, then we see that a \(2^k\) tree of depth l in a smaller region has higher effective resolution than one in a larger region with the same depth.
Figure 26: The $k$-D-tree tesselation of a data distribution. The distribution consists of 1000 points. Of those, 400 are Gaussian distributed, centered at $(25,25)$ $\sigma = 10$, 400 from the same distribution centered at $(75,75)$ and 200 are uniformly distributed. The leaf size of the $k$-$D$-tree was set to 10. It can be seen that the size of the leaves decreases as the local density of samples increases, yielding an adaptive resolution property.
Figure 27: The upper \((p_+)\) an lower bound \((p_-)\) probability estimates for three fixed outcome ratios and confidence bound \(\alpha = .5\). The three ratios plotted are, 0, .5, and 1. for \(n\) observation. The upper part of each component is \(p_+\) and the lower part \(p_-\). It can be seen that the estimates approach their asymptotic values rapidly as a function of sample size \(n\).

5.9.2 Determining When to Build a Binary Tree

There is no point in building a high resolution quad-tree in a region with an extremely sparse number of samples and the \(k\)-\(D\) tree prevents this from occurring, so the \(k\)-\(D\)-tree effectively controls resolution of the binary trees. Still, even though the \(k\)-\(D\) tree alone does achieve a savings, we can achieve an even greater savings it we can decide whether expanding a given \(k\)-\(D\) region with a \(2^k\) tree is worthwhile. This is done by estimating the probability of success and associated confidence intervals in the leaf-nodes of the \(k\)-\(D\) tree. This is elaborated on below.

5.9.3 Sample Size and Confidence Intervals

In the creation of the adaptive binary tree, we desire to assess whether a state can be defined as having greater than or less than some threshold probability of success with some level of confidence.

We must consider the tradeoff between state discretization (leaf size) and sample size. This can be done by attempting to estimate the true probability of a given state along with a confidence interval for that estimate. Assume the true probability of success of a leaf \(i\)
Figure 28: (a) A low \( p_i \) estimate with high confidence, (b) A high probability estimate with high confidence (c) An uncertain estimate. This may be due to insufficient resolution the leaf (more recursive subdivision necessary) or intrinsic stochasticity in the task in that region of the parameter space corresponding to the \( i \)th leaf.

In the tree is \( p_i \) and we have observed \( x \) successes out of \( n \) attempts over the course of all trials and that the underlying probability density is constant over the leaf. This then leads to a binomial process where the distribution of the number of successes \( x \) in \( n \) trials for a given random variable converges to a Gaussian distribution with \( \mu \) and standard deviation \( \sigma \) based on the central limit theorem. Here we have

\[
\mu = np_i
\]

and

\[
\sigma = \sqrt{np_i(1 - p_i)}
\]

We must then solve for an upper and lower bound \((1 - \alpha)\) limits on \( p_i \) given our observations in that leaf. This is done simply by solving

\[
x = \mu \pm g_{\frac{\alpha}{2}} \sigma
\]

Here \( g_{\frac{\alpha}{2}} \) is a confidence interval coefficient, which when multiplied times the standard deviation \( \sigma \) of the Gaussian distribution yields an area under the Gaussian curve less in the interval \((-\infty, \mu - \sigma g_{\frac{\alpha}{2}}]\) with area \( \frac{\alpha}{2} \). We desire an interval that contains the true probability value with confidence \((1 - \alpha)\).
Substituting in for the above $\mu$ and $\sigma$ and solving for $p_i$ yields

$$p_- \leq p_i \leq p_+$$

where

$$p_\pm = \frac{\sigma^2 + \bar{x} \pm \frac{\sigma^2}{\sqrt{n}} \sqrt{\frac{\sigma^2}{4n} + \frac{\bar{x}^2}{n^2}(1 - \frac{\bar{x}}{n})}}{1 + \frac{\sigma^2}{n}}$$ (13)

This equation [Kaelbling, 1990] is utilized on the finished $k$-$D$-tree partitions to determine whether a region should be expanded using a quadtree and also during the expansion of the quadtrees if they are constructed. Its behavior is plotted in Figure 27. It allows the decision as to whether evidence exists to label the leaf as a success or failure leaf, or if the leaf must be further subdivided, or if insufficient data exists to make a determination (see Figure 28). The proposed learning algorithm DARLING is summarized in Tables 8 and 9.

5.10 The Selective Forgetting Mechanism

It is well known that learners must be able to delete experiences that are in conflict with newer inputs [Moore, 1991a]. We propose a novel selective forgetting mechanism. This mechanism is implemented by associating a weighting $w$ to each observation. Each time a new exemplar is input, the weighting for the $k$th nearest observation within a neighborhood of the $m$ nearest-neighbors of the new exemplar is decreased by multiplication with $\gamma = \tau + (1 - \tau) \frac{d^2(X - X_k)}{d^2_X}$, for $d^2(X, X_k) < d^2_X$, and $\gamma = 1$ for $d^2(X - X_k) > d^2_X$ (see Figure 29). Here $X_k$ is the location of the $k$th nearest neighbor, $X$ is location of the new observation, $d$ is the Euclidean distance function, and $d^2_X$ a scale parameter for location $X$. This function is plotted in 30. The scale parameter is taken as twice the median absolute Euclidean distance of the $m$ nearest neighbors. This adapts the decay radius of influence to the local density of exemplars around the new exemplar. When a given observation's weighting falls below some threshold value it is deleted. The parameter $\tau$ determines how many nearby subsequent observations are necessary to make a given observation become obsolete. This process selectively deletes older observations only when new evidence is available which pertains to the same region of the state-space.
Figure 29: The selective forgetting mechanism. Each observation has a weight associated with it. If it is one of the m nearest-neighbors, then its weight is updated by $w_{k,n+1} = \gamma_k w_{k,n}$. When its weight decrease below a cutoff value, the observation is deleted from the learning database. The decay rate is a function of $\tau$ which is the forgetting rate. The smaller $\tau$, the fewer subsequent observations necessary in the neighborhood of an observation before it is deleted.

Figure 30: The influence function for decaying observation in the neighborhood of previous observations. The $\gamma_k$, is used as a forgetting coefficient for the $k$th nearest neighbor to the new observation which is centered at 0. It is a function of the scale parameter $d_X^2$ at which $\gamma$ is unity, $\tau$ which is the forgetting rate, and the distance from this $k$ nearest neighbor to the new point. The scale parameter is adapted to the local density of points.
Algorithm DARLING(point_set)

Generate $k$-D-tree for point_set (* perform density adaptation *)

for all leaves of $k$-D-tree

begin

    Compute $p_+$ and $p_-$ for current_leaf (* can label as success *)

    if $p_- > \text{max\_confidence}$ then cur_leaf.outcome := success

    else if $p_+ < \text{min\_confidence}$ then cur_leaf.outcome := failure (* label as failure *)

    else generate_2k_tree(new( 2k_node ), cur_leaf) (* build $2^k$-tree if indeterminate *)

end

Table 8: The Density Adaptive Reinforcement Learning (DARLING) Algorithm

Algorithm generate_2k_tree( node_ptr cur_node, leaf cur_leaf )

Compute $p_+$ and $p_-$ for cur_leaf

if $p_- > \text{max\_confidence}$ then (* terminate *)

    begin

        cur_node.outcome := success
        cur_node.l := cur_node.r := $\lambda$

    end

else if $p_+ < \text{min\_confidence}$ then (* terminate *)

    begin

        cur_node.outcome := failure
        cur_node.l := cur_node.r := $\lambda$

    end

else (* split further *)

begin

    cur_node.l = new( node )
    cur_node.r = new( node )
    generate_2k_tree( cur_node.l, left(cur_leaf) )
    generate_2k_tree( cur_node.r, right(cur_leaf) )

end

Table 9: Adaptive $2^k$-tree Construction Algorithm
5.11 Experimental Plan

Currently, the implementation of the DARLING algorithm is underway. As soon is it is complete, it will be tested in simulation to gain an understanding of its properties in terms of sample size and execution time requirements. Once the algorithm has been demonstrated to be effective in simulation, its integration with the robotics hardware and software will begin.

Software for controlling the robots and scanner has been written and tested for a MicroVaxII. However, this software must now be ported to a new controller architecture, a SUN Sparcstation IPX running RCCL. This will involve some programming effort. Fortunately, routines for controlling the scanner have already been ported to the Sparcstation by another student, Mario Campos. The interface for the wrist sensor has been upgraded by the addition of a custom six-channel instrumentation amplifier which significantly increases its sensitivity. This new amplifier has been successfully designed, constructed and interfaced by Mario Campos, Tom Lindsay and myself. The LORD gripper must also be interfaced to the Sparcstation, but this is not anticipated to be difficult, and the necessary interface hardware is already available. The superquadric software will also require some minor modifications to implement the constraints described in this section.

The hardware experimental plan consists of constructing the 30 test objects for the grasping out of a matte Plexiglas material. The experimental system is diagrammed in Figure 31. These objects will then be manually classified for the different grasp approach and preshape categories for each pose and used as inputs for the supervised learning phase. After this, the performance of the system on the objects will be monitored during the verification phase as more experiential data is gathered.

The generated object description boundaries for success and failure will be displayed to gain insight into the classification functions generated. Slices of the parameter space and the binding functions will be displayed in order to gain insight into what binding relations are generated. In particular, the learning rate will be evaluated in terms of number of presentations until good performance. The effect of noise in exemplars and process uncertainty will also be monitored.
Figure 31: The proposed experimental system. The overhead mounted camera identifies the approximate object location, the scanner then generates a depth map of the object. Vision processing is done by the SUN Sparc/IPX. The compliant wrist, hand and robot are also directly coupled into the Sparc/IPX for rapid interaction.
6 Contributions

The proposed work addresses several important issues in the fields of machine learning and robotics. As mentioned before, a major problem with many existing learning methods is that they do not scale well with the dimensionality of their chosen problems. A main thrust of this work is to develop algorithms for learning which have resistance to this dimensionality problem. Algorithms with this property will have wide application beyond that of robotic domains. The proposed learning algorithms approach the dimensionality problem in several different ways:

**Projections of High Dimensional Distributions** The projection pursuit method allows high dimensional functions to be expressed as a sum of functions of projections of that high dimensional space. For some functions, the gain in sample economy can be appreciable.

**Efficient Action-Map Building** The problem of building high dimensional action maps using the output of the regression is an important one. We address the problem of generating generalized quadtrees ($2^k$-trees) by the use of efficient probabilistic methods that adaptively control the level of recursive subdivision in high-dimensional spaces.

**Density Adaptive Reinforcement Learning** By combining $k$-$D$ trees and quadtrees to increase resolution in proportion to the local density of exemplars and utilizing estimates of non-determinism in a region, we can develop memory-based reinforcement learners which are economical in storage requirements with respect to the dimensionality of the task and also exhibit low inductive bias.

**Rules for Forgetting** In memory-based learning, forgetting is critical, otherwise the system will not be able to adapt to non-stationary environments.

**Learning in Continuous Domains** The proposed learning algorithms learn in real-valued perceptual and action domains.

**Assumptions for what is innate and what is learned** By building learning systems, we can gain insight into what innate abilities are critical in learning systems and what abilities should be learned.
Affordance Schedules for Incremental Learning By employing the theory of affordances in perceptual learning to incrementally increase the dimensionality of tasks, we can speed up learning and allow bootstrapping to higher-dimensionality task.

From a robotics standpoint, the methods provide the following advances:

Basic-Level Interactions By providing a set of basic-level interactions for grasping, we provide a representation which may better structure the learner’s world and enhance its performance.

Hybrid Architecture By providing both a symbolic level learning level in terms of a concept learner, and a numeric motor binding level in terms of regression bindings, the learning process can be structured so that it converges more rapidly.

Sample Economy By addressing the issue of dimensionality in the learning algorithms as discussed above, we hope to make robot learning techniques more practical for many applications.

Integration of Supervised and Unsupervised Learning Techniques The system’s initialization with the advice of an experienced grasper followed by the specialization and verification phase will allow the robot system to benefit from the knowledge of another agent and learn more rapidly.

Mechanical Interaction Issues in Robotic Learning By allowing the robot to experiment in manipulation tasks without damaging its hand, the instrumented compliant wrist expands the the type of tasks feasible for robotic learning.

By providing new approaches to each of these problems of learning, this work will expand the applicability of robots and make them more flexible, useful and economically feasible.
A The Capture Probability of the Penn Hand

The probability of collision between the target object and the hand in the spherical grasp configuration expressed in terms of the radial distance, \( r \) between the centers of mass of the objects can be computed as follows. Assuming the hand orientation is uniformly distributed along \([0, 2\pi]\) then the probability of collision is at a given radius is \( L_{\text{collision}} \), the length of the perimeter where the object and hand intersect, divided by \( L_{\text{total}} \), the total length of the perimeter at that given radius.

\[
p = \frac{L_{\text{collision}}}{L_{\text{total}}}
\]  

(14)

Let \( d \) be the diameter of the target object, \( w \) the width of gripper’s fingers, and \( R \) the length of the projection of the fingers in the \( z \) direction (downward).

For a spherical grasp configuration, digits are \( 2\pi/3 \) radians apart. To determine the radius below which a collision is guaranteed, we look at figure 32(a). and noting \( \sin(\pi/3) = \sqrt{3}/2 = (d + w)/2r \) when the object is closest to the center of the hand coordinate frame. For \( r \leq \frac{1}{\sqrt{3}}(d + w) \) we have \( L_{\text{collision}} = L_{\text{total}} \) and therefore \( p = 1 \).

The next characteristic region is characterized by Figure 32(b). In this case we have,

\[
p(r) = \frac{L_{\text{collision}}}{L_{\text{total}}} = \frac{3(2\theta r)}{2\pi r} = \frac{3\theta}{\pi}
\]  

(15)

where

\[
\theta = \sin^{-1}\left[ \frac{d + w}{2r} \right]
\]  

(16)

substituting in to the previous expression yields

\[
p(r) = \frac{3\theta}{\pi} = \frac{3}{\pi} \sin^{-1}\left[ \frac{(d + w)}{2r} \right]
\]  

(17)

which holds for \( \frac{d + w}{\sqrt{3}} < r \leq \sqrt{(\frac{d + w}{2})^2 + R^2} \).

The upper bound for \( r \) is determined by imagining that the target object is being slid along the digit from the intersection of the fingers until its center is aligned with the end of the fingertip, where at which point \( r = \sqrt{(\frac{d + w}{2})^2 + R^2} \).

The next region is illustrated by figure 32(c). We have

\[
p(r) = \frac{L_{\text{collision}}}{L_{\text{total}}} = \frac{3(2(\theta_1 + \theta_2)r)}{2\pi r} = \frac{3(\theta_1 + \theta_2)}{\pi}
\]  

(18)
Figure 32: The three representative regions used to compute the capture probabilities.

for \( \sqrt{(\frac{d+w}{2})^2 + R^2} < r \leq c + \frac{d}{2} \), where \( c = \sqrt{(\frac{w}{2})^2 + R^2} \). \( \theta_1 \) can be computed by a straightforward application the law of cosines, namely

\[
\theta_1 = \cos^{-1} \left[ \frac{r^2 + c^2 - (\frac{d}{2})^2}{2rc} \right]
\]

and

\[
\theta_2 = \tan^{-1} \left[ \frac{w}{2R} \right]
\]

The probability of contact, \( p \) is 0 for \( r > c + \frac{d}{2} \), where \( c \) is as above.

Summarizing the distribution,

\[
p(r) = \begin{cases} 
1 & 0 \leq r \leq \frac{(d+w)}{\sqrt{3}} \\
\frac{3}{\pi} \sin^{-1} \left[ \frac{(d+w)}{2r} \right] & \frac{d+w}{\sqrt{3}} < r \leq \sqrt{(\frac{d+w}{2})^2 + R^2} \\
\frac{3}{\pi} \cos^{-1} \left[ \frac{r^2 + c^2 - (\frac{d}{2})^2}{2rc} \right] + \tan^{-1} \left[ \frac{w}{2R} \right] & \sqrt{(\frac{d+w}{2})^2 + R^2} < r \leq c + \frac{d}{2} \\
0 & c + \frac{d}{2} < r
\end{cases}
\]

This function is plotted radially in Figure 33
Figure 33: (a) The radial probability of contact for the Penn hand with a 7cm diameter object, (b) the density plot of the same function
B Shape Extraction Hardware and Software

B.1 Mobile Laser Ranger

Shape perception is accomplished via a laser ranging system mounted on the wrist of a PUMA-560 robot so that it may flexibly explore a large work area consisting of a large portion of entire reachable workspace of the robot. Grasps are then planned for the removal of objects in that workspace using a hand/arm subsystem. The arbitrarily sized and oriented region of interest is decomposed into subregions which are merged and compensated as needed to form a complete description of the entire region. Objects may be rescanned as necessary from different directions to mitigate the effect of illumination and line of sight occlusions which are inherent in laser-stripe type scanner.

To make the system as general as possible, the input to the Image coordinator consists of \( \{x, y, z, \phi, \theta, \psi\} \) coordinates and \( n \) by \( m \) subscans scans which are merged to form a unified range image. The image coordinator is responsible for controlling the range scanning of this arbitrarily oriented rectangular patch of the workspace. Since the mobile scanner can only scan a fixed width swath of workspace, the image coordinator commands the robot to move in a trajectory which completes each subscan. The subscans consist of arm trajectories at fixed velocities.

The mobile laser ranging system presents several advantages over fixed scanners. The most important of these is that laser source and camera line of sight effects may be minimized by scanning from different directions. Maver [Maver and Bajcsy, 1990] has investigated strategies to yield maximum information using a minimum number of scans and then Merging them. Sakane [Sakane et al., 1987] has also tackled this problem in the HEAVEN system which permits efficient locating of cameras and lighting sources given the placement of objects in the scene, but this is done a-priori, not during the task. In our case, we use a simple strategy of scanning from multiple directions and Merging the different views.

Our Laser Range Imaging System consists of two components: The LOOKER and the GUS processing unit. The LOOKER is composed of a laser stripe generator and SONY XC-39 camera which generates video signal of the images obtained under the illumination of the laser stripe, and the GUS unit [Tsikos and Bajcsy, 1988] processes the continuous sequence of laser images and generates a range image of the scene in real time.

The LOOKER is called by its name because it can easily be mounted on the tip of a Puma
robot and can be made to “look” from different direction of a scene (see Figure 12). The entire system is implemented using the HEAP robot sensory driven robotics environment [Agrawal et al., 1990].

In operation, it moves linearly at a known constant velocity under robot control, thereby scanning the scene we are interested in. By geometry, it can be shown that the position of the laser stripe as observed by the camera is a measure of height of the nearest object intercepted by it. This video signal is sampled at a rate of $60Hz$ by the $GUS$ processing unit and the range image is produced in real time.

Synchronization between scanning motion and image generation is ensured by the ability to send a triggering command along a serial line connecting the host computer controlling the robot and the $GUS$ processing unit.

The imaging volume of a single scan and the resolution of the range image are summarized as follows (for a motion rate of $4cm/sec$):

<table>
<thead>
<tr>
<th>Axis</th>
<th>X (width)</th>
<th>Y (length)</th>
<th>Z (height)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imaging volume (mm)</td>
<td>135</td>
<td>164</td>
<td>172</td>
</tr>
<tr>
<td>Resolution (mm/pixel)</td>
<td>.23</td>
<td>.485</td>
<td>.672</td>
</tr>
</tbody>
</table>

Since the size of an image is limited by the imaging volume of $LOOKER$ for a given resolution, multiple number of scans are needed in order to cover whole workspace we are interested in. Having the scanner under manipulator control allows us the flexibility of variable resolution in the $Y$ direction. Noting that the resolution in the $Y$ direction (the scanning direction) is a function of velocity of the scanning motion, it is often useful to obtain a coarse large area scan (scanning at a higher velocity) in order get approximate object locations and shape.

This is useful in employing the robot for initial quick cursory scans of large amounts of the workspace. Gross forms be picked off during this phase and subsequently scanned at higher resolution finer detail is needed to characterize the object, regions with little interest may be subsequently ignored.

In surface regions where the laser stripe cannot illuminate or the camera cannot “see”, pixel values of zero are assigned. Multiple number of scans of the same scene from different direction are needed to recover the occluded part of the scene as much as possible.

Another limitation of the imaging system is that orthographic projection is assumed in the generation of the range image. Software compensation is employed to counteract errors.
of this kind, especially for tall objects [Wang and Gupta, 1989].

B.2 Vision Processing

Once all of the subscans have been performed an erosion operator is applied on each of the scans to reduce spurious measurements due to the sensing method. The subimages are then merged into a unified depth image encompassing the entire scene of interest. A height threshold of 5mm is applied to the height information. All points which pass this thresholding operator are then passed to an 8-connected region growing process. The region growing algorithm is \( O(n^2) \) where \( n \) is the dimension of the image in pixels. When this algorithm terminates, it yields a list of regions, the extremal x and y values for each subregion (to form subwindows) and an associated area in pixels. Regions with areas below a minimum size (500 pixels) are discarded since we have a minimal size which may be grasped reliably by the manipulator.

Our domain consists of objects with arbitrary height, and partially constrained orientation, in that two of the major axes of the object must be parallel to the plane of support. The objects are not currently stacked due to a the significant increase in vision computations to reliably accomplish this. Otherwise, the height and orientation in the plane is not constrained. The next phase of processing consists of generating the associated subimage for each bounding box containing the associated region’s z-values. These subimages are passed to a superquadric surface fitting procedure [Solina, 1987] which generates a set of parameters for a parametric superellipsoid which best fits the range data of the sub-image. This results in a significant data reduction from a complicated range image to a set of 11 parameters which characterize the object and its position in the scene. These eleven parameters are \( \{ x, y, z, \phi, \theta, \psi, a_1, a_2, a_3, e_1, e_2 \} \): where \( x, y, z \) describe the location of the centroid relative to the scanner frame; the \( \phi, \theta, \psi \) are Euler angles describing the rotational orientation of the principle axes of the shape; and \( e_1 \) and \( e_2 \) describe the squareness of the superquadric. The description is approximate, indicating the gross shape and pose of the object.

Finally, our system may be fooled by stacked objects which appear to the scanner as a single object and would have a reasonable goodness of fit, but are actually non rigid, being composed of multiple objects. To handle such cases would require segmentation using edge information and also exploratory procedures to characterize the mechanical degrees of freedom.
freedom between the constituent objects [Campos and Bajcsy, 1990], but this is beyond the scope of this work.

With the superquadric representation of the objects in the scene, we know the size of an object along its three major axes as well as its position and orientation, which can be characterized by a single homogeneous transform (the object frame) with the smallest $a_i$ value is defined to be in the x-axis direction and the largest in the z-axis direction. However, the 11-parameter superquadric representation of an object is not always unique [Solina, 1987]. For instance, two different roll-pitch-yaw combinations can represent the same object, but with the positive z-axis pointing in opposite directions.

**B.3 The Instrumented Compliant Wrist**

The compliant wrist serves two important functions. It controls contact forces on the hand and also serves to detect contact with the object during data gathering. Since the Penn Hand is a somewhat delicate mechanism and the PUMA is capable of large forces, we must take care to control the forces exerted on it (especially its fingers) during impacts in the data gathering phase. This is accomplished by mounting an instrumented passively compliant wrist [Lindsay and Paul, 1991] behind the hand. The wrist has intrinsic low stiffness, which dissipates impact energy on contact, thus protecting the fingers from excessive forces. The wrist also serves to detect collisions with the object in uninstrumented areas of the hand.
Since all exposed areas of the hand cannot be sensorized, a contact in a non-sensorized area will still lead to a wrist displacement.

As can be seen in Figure 11 (page 59), the wiring requirements for the hand’s actuators and sensors are significant. This leads to arm configuration dependent forces being exerted on the wrist due to the cables. These undesirable artifact forces would be sensed by the wrist and might lead to grasping reflexes being inappropriately triggered. In order to compensate for this, two low-pass filters are used on the wrist output. The first has a very low band pass which tracks the wrist positional baseline at roughly the frequency of gross arm motions. The output of the filter is subtracted from the current cartesian readings and the resultant signal is again low pass filtered, although this time with a higher pass filter. This second filter blocks arm vibrations but permits contact events to be passed. The relative displacement thresholds and filter cutoffs were empirically determined and proved quite reliable throughout the experiments. Additionally, excessive arm accelerations could lead to artifact forces and moments being generated at the wrist. Therefore, all arm accelerations were carefully limited and reflexes were only enabled during the terminal phase of the hand approach.
Figure 35: A merged range image.

Figure 36: The reduced superquadric representation of the objects.
References


