This is to certify that the
dissertation entitled

DEVELOPMENTAL LEARNING WITH APPLICATIONS TO ATTENTION, TASK TRANSFER AND USER PRESENCE DETECTION

presented by

XIAO HUANG

has been accepted towards fulfillment
of the requirements for the

Ph.D degree in Computer Science

Major Professor's Signature

5/5/2005

Date

Developmental Learning with Applications to Attention, Task Transfer and User Presence Detection

By

Xiao Huang

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2005

Abstract

Developmental Learning with Applications to Attention, Task Transfer and User Presence Detection

By

Xiao Huang

How to build an intelligent machine has fascinated researchers for more than half a century. To tackle this problem, scientists have taken one of the four approaches: knowledge-based, learning-based, behavior-based and evolution-based. However, autonomous mental development (AMD) of robots did not receive sufficient attention.

Motivated by animal and human autonomous mental development from infancy to adulthood, a developmental paradigm for robots has recently been proposed. With this paradigm, a robot develops its mental skills through real-time, online interactions with the environment. We call such a robot a developmental robot.

My work focuses on motivational system and sensorimotor learning. The major contributions of this work include: (1) Proposed and implemented the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model, which integrates multimodal sensing, action-imposed learning, reinforcement learning, and communicative learning. (2) Integrated novelty and reinforcement learning into a robotic value system (motivational system) for the first time. As an important part of AMD, a value system signals the occurrence of salient

sensory inputs, modulates the mapping from sensory inputs to action outputs, and evaluates candidate actions. (3) Proposed and implemented the Locally Balanced Incremental Hierarchical Discriminant Regression (IHDR) algorithm as the engine of cognitive mapping for learning in non-stationary environments.

Based upon the above architecture and basic techniques, we have designed and implemented a prototype robot that learns the following cognitive behaviors: (1) Visual attention via novelty and rewards. We treat visual attention as a behavior guided by the value system. No salient feature is predefined but instead novelty based on experience, which is applicable to any task. (2) Covert perceptual capability development for vision-based navigation. An agent develops its covert capability via reinforcement learning through interactions with trainers. (3) Cross-task learning in developmental settings. A developmental robot learns multiple tasks incrementally and uses acquired knowledge to speed up learning new tasks. (4) Audio/Visual User presence detection. The developmental learning paradigm has been applied to a developmental agent, which detects human activities in an office using multimodal context information.

The work reported in this thesis serves as a starting point of the on-going research on developmental learning.

To my parents.

ACKNOWLEDGMENTS

I wish to thank my thesis advisor, Professor Juyang Weng, who has been the guiding force of my professional development. This dissertation would not have been completed without his insight, inspiration and encouragement. I am lucky to have such a distinguished and responsible professor as my advisor. I am also grateful to Prof. Ning Xi, Prof. Jim Zacks, and Prof. Rong Jin, for their serving on my committee, sharing ideas with me, and providing critical comments on my thesis.

During my four years at Michigan State University, I was fortunate to benefit from several respected scholars. A special acknowledgment goes to Dr. Zhengyou Zhang at Microsoft Research for his guidance in the "user presence detection" project, which is a very interesting component of my dissertation. I would like to thank Prof. Anil Jain, who served on my committee for one year and provided me with a solid pattern recognition background.

I learned a lot from many of my colleagues. The early version of HDR program provided by Wey-Shiuan Hwang was the starting point of the work reported in this thesis. Yilu Zhang, another early SAIL group member, gave me a jump-start on my research at Michigan State University. During the development of SAIL project, I

enjoyed many open discussions with my labmates: Micky Badgero, Yi Chen, Raja Ganjikunta, Jianda Han, Zhengping Ji, Ameet Joshi, Zain Kazim, Wei Tong, Shuqing Zeng, and Nan Zhang. We shared friendship and precious time together.

Thanks also go to current and former PRIPpies: Hong Chen, Rein-Lein Hsu, Xiaoguang Lu, Anoop Namboodiri, Arun Ross, and Umut Uludag, for sharing their wonderful ideas with me in the seminars, which opened a wider view over related research fields. Those wonderful group activities also made my Phd study unforgettable.

It goes without saying that I owe everything to my dear parents, Laying Kang and Hefu Huang. Their endless love, care, patience, and tremendous support accompanied me throughout my whole life. Now that I am finally graduating, I wish to turn over the diploma to them as a present.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to autonomous mental development

"Can machines think?" The fundamental question in artificial intelligence (AI) raised by Alan Turing [103] has attracted researchers for about half a century. In order to tackle this mysterious problem, four basic approaches have been proposed: knowledge-based, learning-based, behavior-based, and evolution-based.

1.1.1 Traditional approaches

In his well-known paper in 1950 [103], Turing wrote: "one might have a complete system of logical inference 'built in'." "The store would be largely occupied with definitions and propositions. The propositions would have various kinds of status, *e.g.*, well-established facts, conjectures, mathematically proven theorems, statements

given by an authority, expressions having the logical form of proposition but not belief-value." This approach is later called the knowledge-based approach in the field of AI: It is the responsibility of the human programmer to program knowledge into the machine. Knowledge-based systems emerged in the 1970's. MYCIN [15], a typical example, is an interactive program that diagnoses certain infectious diseases, prescribes antimicrobial therapy, and can explain its reasoning in detail. These systems are proved to be very successful in areas such as medical diagnosis and aviation control. In this paradigm a domain engineer finds out the rules to solve the problem and then designs data structures and algorithms. What a computer needs to do is to run the program.

By contrast, the learning-based approach allows machines to learn. Among the earliest learn-based systems, Samuel's checker player [81] was so successful that it was able to compete in some very strong human tournaments. In this framework, a human engineer proposes a learning model, the parameters of the model can be obtained by training the system. The basic difference between the knowledge-based approach and the learning-based approach is: With the former it is humans who supply the knowledge for the task at hand. With the latter humans use input data to indirectly supply a part of knowledge required for the task. How extensive the learned part is depends on the actual application task. However, the learning-based approach is still task specific — the learning algorithm is designed for a specific given task. A great deal of human engineering effort is put into analyzing and understanding the given task. Then a procedure is defined by the human designer. He specifies:

1. which features should be used;

2. which task representation model should be used so that the given task is reduced to determining a manageable number of parameters for the model;

3. what input training data are required to train the system;

4. what computational tool should be used which determines how the input training data are used to determine the parameters of the task representation model.

Rodney Brooks proposed the behavior-based approach in the 1980s [13], trying to overcome some constraints of the knowledge-based approach. A major emphasis of the behavior-based approach is to avoid explicitly modeling the world in which a robot operates. Instead, programming for the robot concentrates on modeling of robot behaviors. A behavior-based system starts with designing a set of basic behaviors instead of building a complete model of the world, which may not even exist in the first place. The system interacts with the world by manipulating and organizing these basic behaviors. In his well-known article *Intelligence Without Reasons* [11] presented during his Computer and Thoughts Award speech at the International Joint Conference on Artificial Intelligence, 1991, Brooks summarized a number of key aspects characterizing this style of work.

Situatedness: The robots are situated in the world — they do not deal with abstract descriptions, but with here and now of the world directly through their sensors and effectors.

Embodiment: The robots have bodies and experience the world directly.

Intelligence: Robots are observed to be intelligent — but the source of intelligence is not limited to just the computational engine. It also comes from the situation in the environment to which the robot is part of.

Emergence: The intelligence of the system emerges from the system's interactions with the world and sometimes from indirect interactions between its components — it is sometimes hard to point to one event or place within the system and to explain why an external action was manifested.

Unfortunately, behavior-based approaches also have fundamental limitations. It seems clear now that humans are unable to adequately describe and decompose complex behaviors, not mention their interactions. Using this paradigm, the behaviors are designed manually in advance. As a result, for complicated systems such as COG, a human-shaped robot at MIT [12], the design of behaviors becomes a serious bottleneck.

Evolutional approaches are motivated by the evolutional process in biological systems. In the 1950s, several researchers independently studied simulation systems with an idea of solving engineering problems using evolution as a search tool. The idea in these studies was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection. John Holland [40] invented "genetic algorithms" which are potentially the most "knowledge-free" way for developing intelligent machines. A strategy does not need to be as complete as an algorithm, since the detail of actions in a strategy does not need to be specified. In principle, it is possible to use an evolutionary strategy to search for an

intelligent machine "species" from scratch. However, both the cost and time required for such an evolutionary process are daunting. Therefore, researchers have been using evolutional ideas in a task-specific way.

From the above survey, we know that the traditional AI approaches generally follow such a *manual development paradigm*,

1. Given a task, the human engineer analyzes it and translates it into representations and rules that a computer program may work on.

2. The human engineer writes a program that transforms the input information into representation and follows the rules to control the machine.

3. The human engineer runs the program on the machine.

In this paradigm it is the human designer not the robot who understands the task. Even though some machine learning techniques might be used, the built-in representation and parameters defined by the human designer is task-specific. This manual development paradigm has met tremendous difficulties for tasks that require complex cognitive and behavioral capabilities, such as autonomous navigation, target finding, and human-robot interaction through gesture in unknown environments.

1.1.2 A new direction in AI – autonomous mental development (AMD)

Is it true that human brain has built-in representation for the tasks that humans generally do? Psychologist Piaget proposed assimilation-accommodation as a model of

cognitive development [74]. Assimilation involves the incorporation of new events into preexisting cognitive structures. Accommodation means existing structures change to accommodate to new information. By repeatedly accommodating to and assimilating novel environmental elements, the cognitive development takes place, which means, the cognitive system changes its internal structure and gradually evolves with maturation and experience.

A classical experiment supporting this perspective was performed by Richard Held and Alan Hein [37] with kittens raised from birth in total darkness. The kittens (shown in Fig. 1.1) were placed in pair in an apparatus called "kitten carousel" when they were old enough to walk. The first kitten in the pair was harnessed to pull the carousel so that it could learn how what it sees changes due to its own actions. The other was carried in the gondola. It saw passively and what it saw is determined by the first kitten's action. The kittens did this for three hours every day and lived in darkness in the rest of the day. After 42 days these kittens were lowered onto the surface of a visual cliff. It was observed that the passive kittens did not develop the cliff avoidance behavior but the active ones did. This experiment demonstrated that passive movements are not sufficient for normal mental development.

Recent studies of brain plasticity in neuroscience have shown that a human brain is not as task-specific as commonly believed. For example, Mriganka Sur and his coworkers rewired the visual input to an animal's (ferret) auditory cortex early in life. The target tissue in the auditory cortex, which is supposed to take auditory representation, was found to take on visual representation instead [92].

Based on evidences in psychology and neuroscience, the new mental developmental

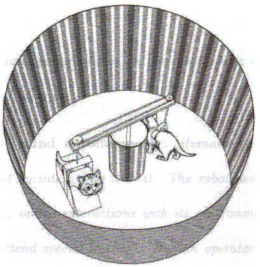


Figure 1.1: Kitten carousal.

paradigm for robots [29] is proposed.

Procedure 1 *The new* AMD paradigm *is as follows:*

1. **Designing a robot body:** *According to the general ecological conditions in which the robot will work (e.g., on-land or underwater), human designers determine the sensors, the effectors and the computational resources that the robot needs, and then the human designs a sensor-rich robot body. The computational resources include computer components such as the CPU, memory, disk, and the controllers of the sensors and effectors, etc.*

2. **Designing a developmental program:** *The human designer designs the developmental program for the robot. Since he does not know what specific tasks the machine will learn, the developmental program does not require any information about the tasks. However, the developmental program may take advantage of the sensors, effectors and the computational resources that have been previ-*

ously chosen.

3. **Birth:** *The human operator turns on the robot whose computer starts to run the developmental program.*

4. **Developing the mind autonomously:** *Humans mentally "raise" the developmental robot by interacting with it. The robot develops its mental skills through real-time, online interactions with its environment, including humans (e.g., let them attend special lessons). Human operators teach robots through verbal, gestural or written commands very much like the way parents teach their children. New skills and concepts are learned by the robots daily. The software (brain) can be downloaded from robots of different mental ages to be run by millions of other computers, e.g., desktop computers. The human operator turns on the robot whose computer starts to run the developmental program.*

This AMD paradigm does not start with any specific task and, in fact, the tasks are unknown at the time of machine construction (or programming). The hallmark of the AMD paradigm is that the human engineer does not need to understand or even anticipate the tasks to be learned by the machine. Consequently, the task-specific representation must be generated autonomously by the robot itself, instead of being designed by the human programmer. Of course, the teachers who will instruct the machine to perform the tasks must understand the tasks, but the human engineer who programs the machine would not be required to. A robot built through the autonomous mental development paradigm is called a *developmental robot.* The developmental phase of this paradigm is shown in Fig. 1.2.

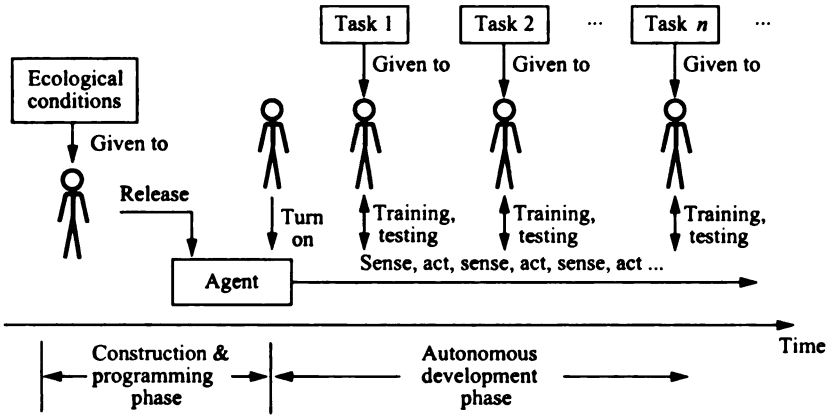


Figure 1.2: The AMD paradigm for machine agents.

1.1.3 Eight requirements of AMD

Practical robotic AMD requires that the following eight technically challenging conditions are all dealt with:

1. Environmental openness: Due to the task-nonspecificity, AMD must deal with unknown and uncontrolled environments, including various human environments.

2. High-dimensional sensors: The dimension of a sensor is the number of scaler values per unit time. AMD must deal directly with continuous raw signals from high-dimensional sensors, e.g., vision, audition and taction. For example, learning from a video camera is more difficult than learning from a laser range finder, because the former typically contains more data per unit time and the photo-electric information is affected by more factors than the range (distance).

3. Completeness in using sensory information. Due to the environmental openness and task nonspecificity, it is not desirable for a developmental program to dis-

card, at the program design stage, sensory information that may be useful for future, unknown tasks. Of course, its task-specific representation, autonomously derived after birth, does discard information that is not useful for a particular task.

4. Online processing: At each time instant, what the machine will sense next depends on what the machine does now. Off-line processing is unable to accomplish AMD.

5. Real-time speed: The sensory/memory refreshing rate must be high enough so that each physical event (e.g., motion and speech) can be temporally sampled and processed in real-time (e.g., about 15Hz for vision). Time consuming iterations must be avoided.

6. Incremental processing: This requires incremental processing. Thus, batch processing is not practical for AMD. Each new observation must be used to update the current complex representation and the raw sensory data must be discarded after it is used for updating.

7. Perform while learning: Conventional machines perform after they are built. An AMD machine must perform while it "builds" itself mentally.

8. Scale up to muddy tasks: For large perceptual and cognitive tasks, an AMD machine must be able to scale its capabilities up to handle multimodal contexts, large long-term memory and generalization, and capabilities of increasing maturity, all without catastrophic memory loss.

Table 1.1: Approaches to Artificial Intelligence

Approach	Species architecture	World knowledge	System behavior	Task-specific
Knowledge-based	Programming	Manual modeling	Manual modeling	Yes
Learning-based	Programming	Modeling with parameters	Modeling with parameters	Yes
Behavior-based	Programming	Do without representation	Manual modeling	Yes
Evolutionary	Genetic search	Modeling with parameters	Modeling with parameters	Yes
Developmental	Partly programming	Avoid modeling	Avoid modeling	No

1.1.4 Comparison of approaches

What are the major differences between the new developmental approach and the existing approaches to making an intelligent machine? Table 1.1 outlines the major differences in terms of four categories: species architecture, world knowledge, system behaviors and task- specificity.

From Table 1.1, we can see that the developmental approach stands on middle ground between two extremes: at one extreme, the agent is totally hand-programmed by human beings (the knowledge-based approach) and at the other extreme, the agent is constructed using the genetic search (the evolutionary approach)[1]. The former extreme requires a large amount of human domain knowledge and, thus, is the most domain specific and *ad hoc* in nature. The latter extreme requires the least amount of human knowledge but requires a tremendous amount of computation time for evolving a sophisticated agent. The developmental approach liberates humans from explicit

[1]It is worth noting that in principle, the evolutionary approach does not necessarily require a task-specific representation. However, due to the complexity considerations, the current genetic algorithms have been used with a human designed, task-specific chromosomes representation and only a particular part of the evolving system is searched for genetically (e.g., Animate [122], AutonoMouse [26] and the work by Steels [91]).

design of (a) any task-specific representation and knowledge and (b) system behavior representation, behavior modules and their interactions. However, the developmental program supplied at "birth" must be designed by human beings. The developmental approach is the first approach that is not task-specific.

The task-nonspecific nature of the developmental approach implies a fundamental departure from the traditional ways of developing a machine. It allows a more systematic and more tractable way of approaching artificial intelligence than all other existing approaches. This does not mean that the approach is easy. The design of developmental program is definitely very challenging, both conceptually and technically. However, since a developmental program can be written without requiring explicit knowledge about the actual tasks in the applications, all the task-related *ad hoc* messy things — things that are called muddy — are completely excluded from consideration at the time of programming. In fact, the designer of the developmental program does not need to understand or know the endless muddy tasks that the machine will end up learning. The design problem becomes systematic, easier to understand and relatively tractable. The miracle of learning various interesting tasks occurs after the programming, not before. This fundamental change in the way we approach artificial intelligence will produce a new kind of machine — developmental machine — that can demonstrate capabilities that no traditional machines have.

However, we must be aware of that all of these capabilities cannot be realized overnight. Scaling up capabilities require a significant amount of developmental experience, which in turn requires time. Furthermore, the developmental programs need repeated modifications and improvements before a high level of mental capabilities

can be developed and demonstrated. And since this thesis is a starting-point of developmental learning, we still adopt the successful components of traditional approaches, which will be shown in the following chapters.

1.2 Survey on animal learning

Since the developmental learning paradigm is derived from psychology, neuroscience and cognitive science. It is important to know the progress in these fields, especially in animal learning. Let us first examine the two different definitions of learning:

Definition 1.2.1 *Learning is an adaptive change in behavior resulting from experience.*

Compare it with the following definition [24] (page13):

> Learning is an enduring change in the mechanisms of behavior involving specific stimuli and/or responses that results from prior experience with similar stimuli and responses.

The latter definition defines learning in terms of a change in the mechanisms of behavior rather than a change in behavior itself. The main reason is that a change in behavior is determined by many factors in addition to learning. For example, whether you eat food or not depends on how hungry you are and how much you like the food. You start to eat food when you are hungry. After you have had enough, you stop eating. The second definition does not include this action change, from eating to not eating, as learning, but the first definition does. In other words, the

second definition does not include what is called *non-associative learning* explained below. In this thesis, I adopt Definition 1.2.1. Learning takes place all the time during development, whether you are eating, working, entertaining or sleeping. Every stimulus has a potential to change the behavior.

1.2.1 Nonassociative learning

Habituation and *sensitization* are examples of nonassociative learning. In habituation learning, a subject learns about the properties of a single stimulus — such as a loud noise from a toy gun or a view of a new toy. A number of autonomic changes are triggered in the human body. Heart beats faster and breathing becomes more rapid. If the same stimulus is repeated in a short interval or it sustains for a long period of time, these responses will abate. This is *habituation*, a type of learning that men experience routinely in one way or another.

With habituation an animal learns about the properties of a benign stimulus, the result of which is reduced response. With sensitization an animal learns about the properties of a harmful or threatening stimulus. A person startled by a gunshot is likely to jump at any loud noise heard minutes thereafter.

How are sensitization and habituation related? A mouse will be startled when it is exposed to a loud noise for the first time. As the same loud noise is repeated, the mouse will habituate to it and no longer respond. The startle response to the noise can be quickly restored by delivering a single electric shock to the feet of the mouse. Habitation and sensitization effects reflect how the agent ends up sorting

out what to ignore and what to respond to. Groves and Thompson's dual-process theory [36] of sensitization and habituation assumes that different types of underlying neural processes are responsible for the increases and decreases in responsiveness to stimulation. The Solomon's opponent-process theory [87] [86] assumes that an important function of mechanisms that control emotional behaviors is to minimize deviations from emotional neutrality or stability, a level called *homostasis.*

The neural basis of learning has been a very active subject of study in neural science. The weakening and strengthening of synaptic connections between sensory neurons and motor neurons (through interneurons) are found to be involved in habituation and sensitization learning in the limb withdrawal reflex of the cat and the withdrawal of the gills and siphons of the marine snail called Aplysia[2]. Enhancement of synaptic connections is sufficient for short-term habituation and sensitization where the same stimuli are repeated for a short period of time. In long term non-associative learning, the stimuli repeat for days or weeks and the change of behavior also lasts much longer. This long term change in behavior requires the synthesis of new proteins and the growth of new synaptic connections along the passway from sensory neurons to motor neurons.

In nonassociative learning, the subject learns about the properties of a single stimulus by being exposed to it repeatedly. In associative learning discussed below, a subject learns the relationship between two stimuli (classical conditioning) or the relationship of a stimulus to the subject's behavior (instrumental or operant conditioning).

[2]For a more detailed explanation of the study, the reader is referred to [90] (pages 36 - 45).

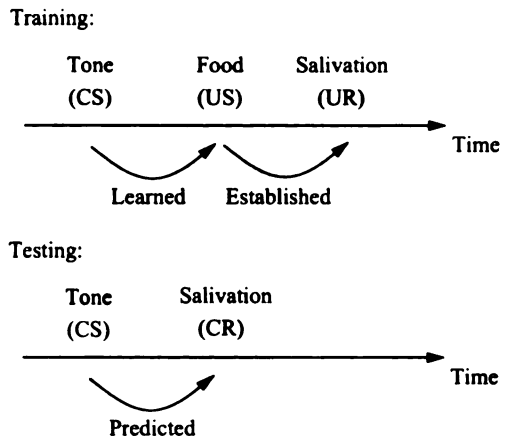


Figure 1.3: An illustration of classical conditioning.

1.2.2 Classical conditioning

Systematic studies of classical conditioning began with the great work of the Russian physiologist Ivan P. Pavlov. According to Pavlov, what animals and humans learn is not the association of ideas, as suggested by Aristotle, but the association of stimuli.

Classical conditioning involves the association or pairing of two stimuli, an *unconditioned stimulus* (US), such as food or a shock to the leg, and a *conditioned stimulus* (CS), such as a tone or a light. The unconditioned stimulus always produces a response, called *unconditioned response* (UR), such as salivation or leg withdrawal. It is called an unconditioned response because it is a hardwired (inborn) response to the unconditioned stimulus. Suppose that a tone (CS) is repeatedly followed by the presentation of food (US) which always elicits a salivation response (UR) from a horse. Then, the tone (CS) alone starts to elicit salivation (CR) even without food (UR), as shown in Fig. 1.3. Although classical conditioning is classified into appetitive conditioning (US is appetitive) and defensive conditioning (US is aversive), the

US can be neutral, as long as the US is reliably linked with the UR, either hardwired or learned. The essence of classical conditioning is a means by which animals learn to predict relationships between two events in the environment — CS predicting US.

After the training session, if the tone is repeatedly presented without food, i.e., CS is no longer followed by US, the tone is no longer able to evoke salivation. This process is called *extinction*. The available evidence in animal studies indicates that extinction is not simply forgetting. During the extinction, the animal learns something new: The conditioned stimulus no longer predicts that the unconditioned stimulus will occur; it instead predicts that the unconditioned stimulus will not occur.

In our developmental robot, the inconsistency with the past experience (surprisingness) will raise the value of the current event, which enhances the memory process. The animal is paying attention to the delivery of unconditional stimulus (US) following the conditional stimulus (CS). If the conditional stimulus does not occur, the inconsistency with the experience is detected and the value system sends an increased value to enhance the memory process for the current event.

Another important phenomenon about classical conditioning is called *blocking*, demonstrated in a three-phase experiment conducted by L. J. Kamin in 1968 [56]. In phase 1, stimulus light (CS A) is paired with an aversive unconditional stimulus, a strong electric shock, until the rats totally suppressed their lever pressing behavior (UR) whenever a light (CS A) was presented. Next, in phase 2, a new stimulus tone (CS B) was presented together with stimulus light (CS A) and paired with the electric shock (US). Finally, in phase 3, stimulus tone (CS B) was presented alone in a test trial to see if it would also suppress lever pressing (CR). Interestingly, less conditional

response (surpressing) was observed than the control rats' group (that skipped the phase 1). In this example, CS B is blocked (partially) by CS A.

The idea that the *surprisingness* of an unconditioned stimulus determines its effectiveness in producing new learning was developed into a model by Robert Rescorla and Alan Wagner [80]. An event is surprising if it is different from what is expected. If the CS perfectly predicts US, the rate of learning becomes zero. This is also consistent to our discussion of the value system that is dependent on the surprisingness.

1.2.3 Instrumental conditioning

Instrumental conditioning (also called *operant conditioning* or *trial-and-error learning*) is another major form of associative learning, systematically studied by B. F. Skinner and others. With this type of learning, the trainer is able to tune the behavior of the animal towards the direction that the trainer wants.

Instrumental conditioning can be conducted in the following way. A hungry rat is placed in a chamber which has a lever protruding from the wall. A press on the lever will release a food pellet into a cup. If pressing the lever is not something that the rat already does occasionally, it may never "discover" on its own what it has to do to obtain the food. We can use a sequence of training steps called *shaping* that has been used by animal trainers for centuries. At first, food is given if the rat does anything remotely related to the desired response. For example, at first, the rat is given a food pellet each time it gets up on its hind legs anywhere in the chamber. Once the rearing response has been established, the rat is given food only if it rears

over the lever. Once it is over the lever, the rat is given food only if the rat actually depresses the lever. With this training, whenever the rat is hungry and finds himself in the chamber, it is likely to press the lever to get food.

Classical conditioning learning forms a predicate relationship between two stimulus (CS and US). Instrumental learning can be considered as a formation of a predicate relationship between a stimulus and the animal's response. Unlike classical conditioning, where the conditional response is evoked by a specific conditional stimulus, the response in the instrumental conditioning is emitted without a recognizable stimulus (but the animal still uses some environmental cues, such as the view of the lever to evoke the behavior). In both types of learning, timing is critical. The reinforcer (food) should closely follow the operant response (pressing lever). If the time delay is too long, only weak conditioning is established.

The neural mechanisms of classical conditioning and of instrumental conditioning are very similar. If the chain of events predicts a reward, the response is emitted. If a punishment is predicted, the corresponding response is suppressed. In classical conditioning, the chain of events is CS-US-UR (e.g., tone, food, salivation). With instrumental conditioning, the chain of events is E-R-UR, where E is environmental cues (in the chamber or the view of a lever) and R is the shaped response (pressing lever) and UR is the food reward. In both cases, if UR is a punishment, the animal will evoke avoidance behaviors. In classical conditioning, CS will suppress the animal's ongoing activity or elicit the learned avoidance behavior (e.g., leg withdrawal). In instrumental conditioning, the animal would decrease the response rate or suppress the response that leads to the punishment.

To summarize what is necessary (but may not be sufficient) for nonassociative and associative learning, the following gives a list:

1. Biased sensors that define appetitive and aversive stimuli.

2. A value generation system. It decreases the output value when there is no unexpected inconsistency with the experience (lack of surprise or novelty). It increases the output value quickly when novelty is high or aversive stimuli are sensed.

3. An event prediction mechanism that is able to predict the chain of events (defined as sensorimotor signals) in time.

4. A response selection mechanism. If appetitive or avoidance of aversive stimulus is predicted, emit the corresponding response. If aversive stimulus is predicted or omission of appetitive stimulus is predicted, suppress the response.

1.2.4 Cognitive learning

The above types of animal learning does not cover a more complex learning, which is called cognitive learning in psychology. The word "cognition" comes from the Latin meaning "knowledge or thinking."

Cognitive learning covers more complex behaviors of human and animal learning. The subject is closely related to the mechanisms of organizing memory and forgetting.

Timing learning refers to the ability of learning to estimate the duration of time. Although all the above types of learning require a proper duration of time for each

event, animals are able to estimate and distinguish the length of an event and respond accordingly. For example, pigeons (or rats) can be trained to peck a red key if a light is turned on for a short period of time (e.g., 2 seconds) and to peck a green key if the light is on for a long period of time (e.g., 10 seconds). During the learning, pecks on the correct key are reinforced (e.g., with food) (see, e.g., [107] [19]).

In *serial pattern learning*, animals learn the order of events. A panel consists of five squares. Five distinct stimuli (e.g., dots, circles, and the like) were presented at the same time in the panel, each stimulus in one square. Let us call the stimuli A, B, C, D, E. The task for monkeys is to press the squares in the panel in the prescribed order: A, B, C, D, E. Training started with a presentation of stimulus A alone. After the monkeys learned to press A, B was added. After they learned the AB sequence, C was added, and so forth. After the monkeys learned the entire sequence, they were tested on subsets of two and three stimuli (AB, BC, CD, DE, ABC, BCD, DCE) and reached above-chance accuracy (see, e.g., [22]).

Perceptual concept learning is the capability of learning a concept. For example, although chairs have different colors, shapes and heights, we can all agree on what is a chair and what is not. The "chair" is an example of a perceptual concept. Pigeons have been trained to respond to the presence or absence of fish in underwater photographs [39], to the presence of the letter A as opposed to other letters of the alphabet in various fonts [66] and even to discriminate pictures of Monet from those of Picasso [108].

Language learning is probably the most complex cognitive skill learned by humans and animals. In fact, many have assumed that the linguistic skill is so complex and

specialized that it is uniquely human. Some assume that the ability of using language depends on certain innate processes that have evolved only in humans (see, e.g., Chomsky 1972 [18]). Some animals are intelligent enough to learn language when they are properly trained. Since they cannot speak a human language, the American Sign Language has been taught to Chimpanzees [33], gorillas [72], dolphins [38], sea lions [34] and African grey parrots [73]. In Allen and Beatrice Gardner's study [33], their chimpanzee, named Washoe, learned to sign well over 100 words.

As we can see above, the models of animal learning have been studied based on specific types. A model that was designed to explain a type of learning is not applicable to other types of learning. One of the major goals of this volume is to present a unified developmental model that is capable of developing an architecture that can perform all major types of animal and human learning, including nonassociative learning, classical conditioning, instrumental conditioning and the major forms of complex cognitive learning.

1.3 New machine learning types

In AMD, the environment can affect the robot brain through three channels. The first channel is its sensors. The second channel is its effectors. Affecting an effector can be done by imposing a force on it, either by stopping or by changing its motion. The third way of affecting the brain of an animal is to directly change the internal state of the brain. For example, a neural surgeon can open the human skull to perform brain surgery. For machines, we can feed data directly into the program. These ways

of accessing the brain directly change the internal state of the brain. In summary, there are three channels through which the environment can affect an agent: sensors, effectors, and internal state. The sensors are divided into two classes, biased and unbiased. If the machine has a predefined preference pattern to the signals from a sensor at the birth time, this sensor is a biased sensor. Otherwise, it is an unbiased sensor. We have a total of four entities, u, b, i, and e, representing unbiased sensors, biased sensors, internal state, and effectors, respectively.

In order to fully understand the meaning of developmental learning, we investigate learning types from three entities, b, i, and e, representing biased sensors, internal state, and effectors, respectively. Depending on whether the action e in $\{b, i, e\}$ is imposed or not, the learning can be classified into *effector-imposed* learning and *effector-autonomous* learning. *Effector-imposed* learning is such that the effector action is supplied by the trainer. For example, when an adult teaches a child to hold a pen properly, the adult can place the pen into the child's hand and manipulate the hand to hold it properly. This is effector-imposed learning. Otherwise, the learning is *effector-autonomous* learning.

Depending on whether the biased sensor b in $\{b, i, e\}$ is used or not, the learning can be classified into reinforcement learning and communicative learning. *Reinforcement* learning is such that a *biased* sensor is used to reinforce or punish certain responses from the machine agent. *Communicative* learning is such that *biased* sensors are not used and only *unbiased* sensors are. This requires the learner to correctly interpret the signal from unbiased sensors, an instruction for an action, an encouragement, or an explanation, etc. Learning by a human adult is conducted mostly in the

communicative learning mode, not in the reinforcement learning mode. For example, a student goes to a university everyday to earn his Ph.D. degree. The future of her Ph.D. degree does not give him any immediate physical pleasure. On the contrary, his studies at the university may often cause physical displeasure, such as fatigue.

1.3.1 Classification of learning types

Thus, any type of learning can be represented by a 3-tuple (i, b, e), which contains three components i, b, and e, each of which can be either represented by 0 or autonomous 1. $i = 1$ means that the internal state is imposable and $i = 0$ state-autonomous. $b = 1$ indicates reinforcement and $b = 0$ communicative. $e = 1$ denotes effector imposed and $e = 0$ effector free. Thus, depending on what attribute each component takes (0 or 1) there are a total of 8 different 3-tuples, representing a total of 8 different learning types. For example, $(i, b, e) = (1, 1, 0)$ means state-imposable, reinforcement effector-autonomous learning. This is the typical mode of reinforcement learning in the machine learning community. If we consider *ibe* as three binary bits of the type index number of learning, we have 8 types of learning defined below: We can also name each type. For example, Type 0 is state-autonomous, communicative, effector autonomous learning. Type 7 is state-imposable, reinforcement, effector imposed learning.

All traditional learning methods correspond to Types 4 to 7, which is because the representation is designed after the task is given and so are the constraints of internal representation. Autonomous development uses Types 0 to 3. However, the

Table 1.2: Eight types of learning

Type (binary)	Internal state	Biased sensor	Effector
0 (000)	Autonomous	Not used	Autonomous
1 (001)	Autonomous	Not used	Imposed
2 (010)	Autonomous	Used	Autonomous
3 (011)	Autonomous	Used	Imposed
4 (100)	Imposable	Not used	Autonomous
5 (101)	Imposable	Not used	Imposed
6 (110)	Imposable	Used	Autonomous
7 (111)	Imposable	Used	Imposed

state-autonomous nature is not sufficient to characterize the power of autonomous development, since just realizing state-autonomous is not difficult and not of much importance without the real power for dealing with muddy tasks.

1.3.2 Comparison with traditional definitions

The above definitions of learning types are different from traditional definitions for machine learning. A term *supervised learning*, often used in the machine learning and pattern recognition research communities, means that human teachers supervise the learner during the learning process. From the most strict definition of unsupervised learning, all of the above learning modes are supervised by humans to some degree. It is difficult to identify any type of learning that is completely unsupervised. For a better understanding of the eight new learning modes, the new explicit definitions in Table 1.2 are required beyond the coarse classification of supervised, unsupervised, and reinforcement learning that the scientific field is familiar with.

According to the above refined new definition, the traditional supervised learning falls into Type 5, while the traditional reinforcement learning belongs to Type 6.

Thus, the traditional learning methods belong to state-imposable learning, Types 4 to 6, because humans must define the feature space after the tasks are given (thus state-imposable). If system states are used, the meanings of the internal states are manually assigned by the programmer after the tasks are given.

Human (and animal) learning does not allow the teacher to exercise direct access to the brain (the natural counterpart of the learning program). Thus, all the modes of human and other animal learning belong to Types 0 to 3. During the early stages of human child development, Types 2 and 3 are often used and are effective. For instance, a mother rewards a good behaving toddler with candy. Types 2 and 3 are less effective for human adults. For example, offering candies or slapping on the face does not necessarily succeed in making a 30-year old do what the teacher wants. Type 1 is often used for older children whose capability of communicative learning is still limited or if the learner is a novice on the subject being taught. For example, a ballet or athletic teacher may often hold the limbs of a novice student in order to teach the student to master a technique. Type 0 learning is dominant in human learning. It occurs probably as early as the fetus stage. The majority of a human's time, either an infant, a child, an adolescent or an adult, is spent in the Type 0 learning mode. Reinforcement through biased sensors is relatively rare, compared to the time when no significant signals are received from the biased sensors.

We realize that the learning types applied to autonomous development are Types 0 to 3. How are these types being used in the actual development process? This is the major issue of this dissertation.

1.4 Thesis outline

The organization of this thesis is stated follows.

Chapter 2 proposes the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model, which integrates multi-modal sensing, action-imposed learning, reinforcement learning, and communicative learning.

Chapter 3 is mainly concerned with building the cognitive mapping engine using the Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) technique.

Chapter 4 presents our model for the value system of a developmental robot. Novelty and reinforcement learning are integrated into a value system for the first time. As an indispensable part of AMD, a value system signals the occurrence of salient sensory inputs, modulates the mapping from sensory inputs to action outputs, and evaluates candidate actions. The value system is applied to guide a robot's visual attention behavior.

Chapter 5 presents a robot system that develops convert perceptual capability for vision-based autonomous navigation behavior through online interactions with human trainers using reinforcement learning. This work is quite new comparing to the dominant supervised learning techniques used in this domain.

Chapter 6 provides the developmental agent with this cross-task learning capability. That is, an agent can learn multiple tasks and use acquired knowledge to learn new tasks.

Chapter 7 presents a user presence detection system. Context information from multi-sensory inputs is integrated to infer a user's activities in an office. We design a layered architecture to model human activities with different granularities. The prototype system verifies the effectiveness of the developmental learning paradigm.

Chapter 8 concludes with a summary of the contributions of this thesis. Some thoughts on future directions will be discussed.

Chapter 2

DOSASE MDP Model

In this chapter, we propose the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model, which integrates multi-modal sensing, action-imposed learning, reinforcement learning, and communicative learning. Two major components of the architecture: cognitive mapping engine and value system will be discussed in Chapter 3 and Chapter 4, respectively. At the end of this chapter, we will briefly present the hardware architecture of the validation platform, the SAIL robot.

2.1 Introduction

Many different architectures have been proposed in the intelligent robot community. Robot perception and perception-based behavior generation have been proved to be very difficult, especially in unknown or partially unknown environments. Some work on robot learning was motivated by human learning and development: from sim-

ple to complex. BAIRN (a Scottish word for "child") is a symbolic self-modifying information processing system used as an implementation for a theory of cognitive development[106]. Drescher[27] utilized the schema, a symbolic tripartite structure in the form of a "context-action-result," to model the ideas of child sensory-motor learning in Piaget's constructivist theory of cognitive development. Soar[59] and ACT-R[4] are two well-known symbolic systems with an aim to model cognitive processes, although cognitive development was not the goal of these efforts. The model of Albus[2] takes sensors as input and produces control signals to effectors as output, thus, allowing a finer numeric representation of sensory features when a specific task is given. The Finite State Machine (FSM) or its probability based variant, the Markov Decision Process (MDP), are two general frameworks that have been used for conducting autonomous learning with a given goal in a symbolic world (e.g., Shen[85]) or reinforcement learning (e.g., Kaelbling et al.[54]). The explanation-based neural network, called "lifelong learning," was used by Thrun[99].

Designing a program and its representation in a task-specific way using a traditional approach is typically complex, ad hoc, and labor intensive. The resulting system tends to be brittle especially in unknown and uncontrolled environments. Recently an important direction of research aims at reducing or avoiding the human imposed limitations (e.g., features, models) of the world for better environment adaptation in learning. Cresceptron[113] is a system that allows a human teacher to interactively segment natural objects from complex images through which it incrementally grows a network that performs both recognition and segmentation. SHOSLIF[45], SARCOS[105], Cog[14], and Kismet[9] are motivated by simulating infant skills via

learning through interactions with the environment.

The efforts discussed above were motivated by human learning and cognitive development to various degrees. However, these proposed architectures are fundamentally different from human learning and have not reached autonomous mental development (AMD)[119]. The SAIL robot[111] and the Darwin V robot[3] are two prototypes of developmental robot. Darwin V was designed to provide a concrete example of how the computational weights of neural circuits are determined in a controlled environment by the behavioral and environmental interactions of an autonomous device. The SAIL developmental robot was designed for developing perceptual and behavioral skills through interactions in *uncontrolled*, complex human environments. There has been a lack of systematic theory for developmental mental architecture.

This chapter introduces a theory of mental architecture appropriated to autonomous development. Section 2.2 formulates the developmental agent model. The Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model is introduced in Section 2.3. Sections 2.4 and 2.5 present the architecture and the major components of the SAIL robot, respectively. A detailed description of SAIL is given in Section 2.6. Conclusions are drawn in Section 2.7.

2.2 An agent model

To be precise in our further discussion, we need mathematical notations.

Definition 2.2.1 *Context: A context $c(t)$ of an agent is a stochastic process. It consists of two parts $c(t) = (x(t), a(t))$, where $x(t)$ denotes the sensory vector at time*

t, which collects all signals sensed by the agent at time t, $a(t)$ is the effector vector consisting of all the signals sent to the effectors of the agent at time t.

Definition 2.2.2 *Length of Context: The context related to the agent from the previous time t_1 up to a later time t_2 is a realization of a stochastic process $\{c(\tau)|t_1 \leq \tau \leq t_2\}$. The length of the context is $L = t_2 - t_1$.*

Definition 2.2.3 *Given an agent at time t_1, suppose that the agent produces different action contexts $\mathbf{a}_1$ and $\mathbf{a}_2$, from two different contexts $C_1 = \{\mathbf{c}(t) \mid t_1 \leq t \leq t_2\}$ and $C_2 = \{\mathbf{c}(t) \mid t_1 \leq t \leq t_3\}$, respectively. If $\mathbf{a}_1$ and $\mathbf{a}_2$ are considered different by a social group (human or robot), conditioned on C_1 and C_2, then we say that the agent* discriminates *two contexts C_1 and C_2 in the society. Otherwise, we say that the agent does not* discriminate *C_1 and C_2 in the society.*

Definition 2.2.4 *The last context at time t is the recent part of history, $l(t) = \{(x(t), a(t))|\tau = t - k, ..., t - 1, t\}$.*

The last context consists of a short segment of recent experience of temporal length $k + 1$. Currently in our work, k does not change with t and a different k is designed for different sensors and levels.

Definition 2.2.1 (Environment) *The* internal environment *of an agent is the brain (or "the central nervous system") of the agent. The* external environment *consists of all the remaining parts of the world, including the agent's own body (excluding the brain).*

Having defined the external and internal environments, we define the sensors and effectors associated with these concepts.

Definition 2.2.2 (Internal sensors and effectors) *An external sensor S_e is a sensor that senses the external environment. An internal sensor S_i is a sensor that senses the internal environment. An external effector E_e is an effector that acts on the external environment. An internal effector E_i is an effector that acts on the internal environment.*

Definition 2.2.5 *A state $s(t)$ is the internal representation of the context in an agent's internal information environment.*

2.3 Observation-driven Markov Decision Processes

According to the above definitions, an agent views the world as a general random process. The state gives a more compact representation than a simple concatenation of last contexts. Due to sensory uncertainty and partial observation, the state is also a random vector. The state at time instance t depends on both the context $l(t)$ and the last state $s(t-1)$. Therefore, we can recursively generate the state with:

$$s(t) = f(s(t-1), l(t)). \tag{2.1}$$

where the function f generates states using feature derivation, temporal chucking, and vector quantization. The recursive estimation process can be modeled by a Markov

decision process (MDP), where the state transition follows a conditional distribution,

$$P(s(t) = s|s(t-1) = s', l(t) = l) \tag{2.2}$$

and the representation $s(t)$ here is a high-dimensional vector. The most probable state at time t is given by,

$$s^*(t) = \arg\max_{s \in S} P(s(t) = s|s(t-1) = s', l(t) = l), \tag{2.3}$$

where S is the set of all possible states, which is infinite. In a practical implementation, we approximate the set S by a finite number of prototype states. These prototype states are incrementally merged cluster centers of many experienced state vectors. In practice, the maximization in Eq. (2.3) is too computationally expensive. We use an IHDR tree to approximate it, taking advantage of the numerical vector representation of $s(t)$ and $l(t)$.

2.3.1 Observation-driven MDP

An agent has a number of sensors and effectors. Fig. 2.2 illustrates a multi-sensor multi-effector model of an agent. The agent $A(t)$ operates at equally spaced discrete time instances $t = 0, 1, ...$ We assume that an image is produced at each time instance by the sensor, independent of the sensing modality, visual, auditory, touch, etc. Without losing the generality, we assume that the agent has two external sensors and two external effectors. Each external sensor S_{ei}, $i = 1, 2$, senses a random multi-

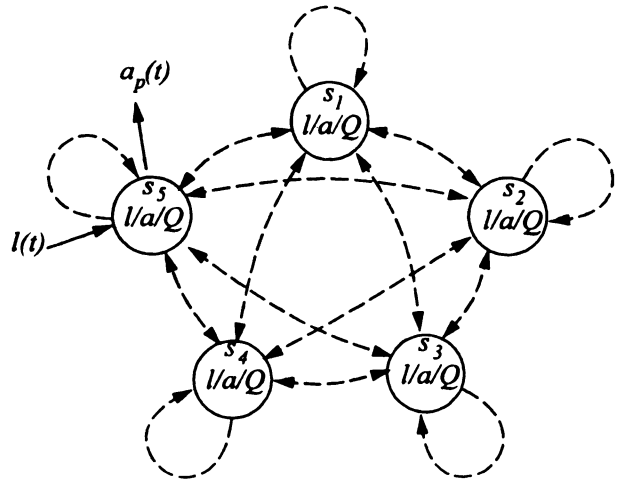


Figure 2.1: Observation-driven Markov decision process with dynamically generated states, context-driven state representation, and value-based action generation. Note: the state is not symbolic. It is a vector in a high dimensional space.

dimensional sensory frame $x_e(t) = (x_{e1}(t), x_{e2}(t))$ at each time instance t and the sensed signal is fed into the agent. Each external effector E_{ei}, $i = 1, 2$, receives from the agent an effector frame $a_e(t) = (a_{e1}(t), a_{e2}(t))$ at each time instance t.

Let $x_t \in \mathcal{X}$ and $p_t \in \mathcal{P}$ be the observations and outcome covariates (i.e., random vectors) at time t, respectively. Note that we change a variable of a vector to its subscript (e.g., change $x(t)$ to x_t) when it is more convenient to consider the variable as a discrete index number. Let H_t be the random vector of the history: $H_t = \{x_t, x_{t-1}, ..., x_0, p_{t-1}, ..., p_0\}$. At time t, the agent $A(t)$ needs to estimate the distribution of $P(p_t \mid H_t = h)$.

If t is large, H_t is too large to be practical and it contains much information that is not very related to the outcome H_t.

Definition 2.3.1 *A mental architecture is called a k-th order Observation-driven*

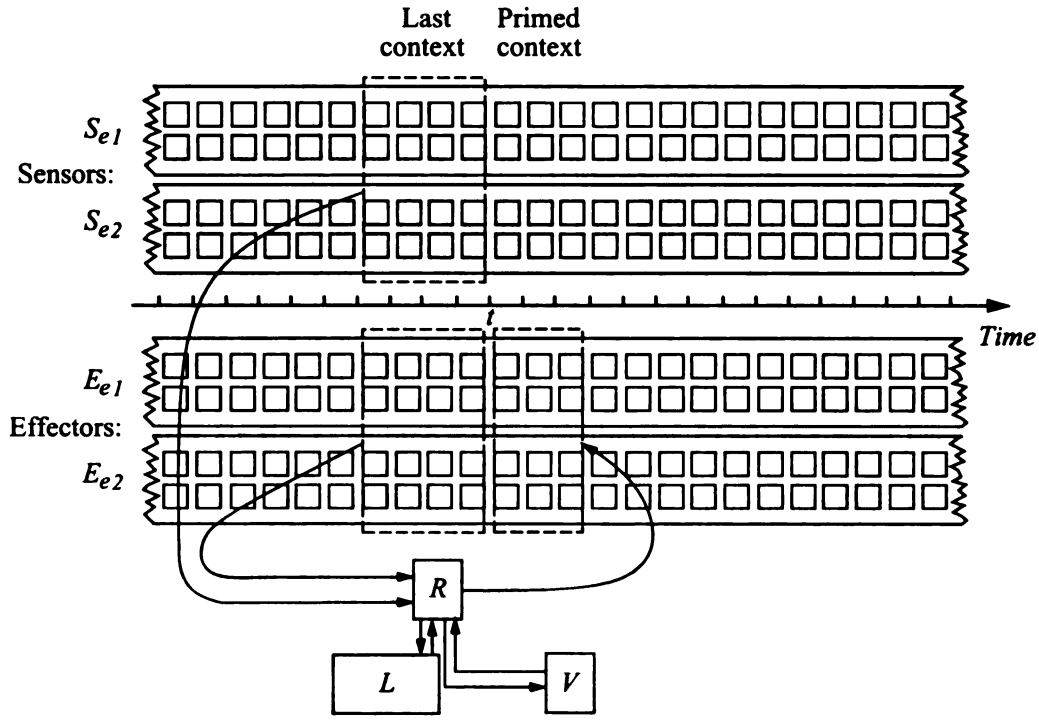


Figure 2.2: The architecture of a multi-sensor multi-effector agent: Observation-driven Markov decision process. Each square in the temporal streams denotes a smallest admissible mask. This architecture takes the entire image frame without applying any mask. The block marked with L is a set of context states (prototypes), which are clusters of all observed context vectors $l(t)$.

Markov Decision Process (MDP)[21] if the H_t to contain only the last k observations:

$$l_t = \{x_t, x_{t-1}, ..., x_{t-k}, p_{t-1}, ..., p_{t-k}\}$$

as shown in Fig. 2.2. The random observations in l_t across time $t = 0, 1, ..., t$ are the source from which the agent automatically generates states in the form of clusters $l \in \mathcal{L}$, where $\mathcal{L}$ consists of all possible observations of the last contexts $\mathcal{L} = \{H_t \mid 0 \leq t\}$. We define the state $s(t)$ in Eq. 2.1, which means the state is also kth order MDP. The predicted consequence p_t consists of predicted action a_t and the predicted value

v_t, $p_t = (a_t, v_t)$.

A major difference between a regular MDP[54, 93] (or HMM[79]) and an observation-driven MDP are that the states s_t with a regular MDP are hand-designed by a human programmer but the states with an observation-driven MDP can be automatically generated (developed). With a regular MDP, the programmer must provide initial estimates for the prior probability distribution $P(s_0)$, the state transitional probability $P(s_t \mid x_t, s_{t-1})$ and the state observation probability $P(x_t \mid s_t)$. It in turn, requires that the human programmer establishes a correspondence between the meanings of the physical events being modeled and the states. Due to the fact that physical events are not known at the time of programing for the developmental robots, the regular MDP is not suited for the developmental program. In contrast, the observation driven MDP requires no prior probability and all the probability distribution $P(p_t \mid l_t = l)$ can be estimated incrementally on-the-fly.

Another important difference between the states in the traditional MDP and the observation driven MDP is their very different nature of the representation. The states in the former correspond to some objects or events in the world by human hand-design. The entire set of states is a monolithic representation of the modeled part of the world. That results in the high brittleness of the agent in dealing with unexpected events. In contrast, the states in the latter are clusters of observational vectors. They are not monolithic in the sense that an object in the world can correspond to many state clusters. Further, each cluster may correspond to an observation of several objects in the world. Therefore, there is no strict one-to-one correspondence between a state

in the observation-driven MDP and an object of the world. It is the behavior of the agent (e.g., the same picking apple behavior from different views of the apples) that shows the discrimination and generalization power of the agent.

In practice, we implement the regressor R using the Incremental Hierarchical Discriminant Regression (IHDR)[116, 115]. Given any observed (last) context $l(t)$, the system generates the current state $s(t)$ using Eq. 2.1; the regressor R produces multiple consequences (primed context) $p_1(t), ..., p_k(t)$ with a high probability:

$$\{p_1(t), ..., p_k(t)\} = R(s(t)). \tag{2.4}$$

Thus, the regressor R is a mapping from the space of the last context $\mathcal{L}$ to the power set of $\mathcal{P}$:

$$R : \mathcal{L} \mapsto 2^{\mathcal{P}}. \tag{2.5}$$

R is developed incrementally through the real-time experience. For any $t > 0$ (after the birth), it is a total function since it is defined for all elements in $\mathcal{L}$, but it does not do well for most elements in $\mathcal{L}$ that it has not experienced. It is not an onto function since its range covers only a very small part of $2^{\mathcal{P}}$.

Therefore, we need a value system that selects desirable contexts from multiple primed ones. The value system $V(t)$ takes a set of (e.g., k) contexts from the regressor R and selects a single context:

$$V(R(s(t))) = V(\{p_1(t), p_2(t), ..., p_k(t)\}) = p_i(t) \tag{2.6}$$

where $1 \leq i \leq k$ and k varies according to experience. The value function selects the best consequence $p_i(t)$ that has the best value $v_i(t)$ in $p_i(t) = (a_i(t), v_i(t))$. For example, $V(\{p_1(t), p_2(t), ..., p_k(t)\}) = p_i(t)$ if $i = \arg\max\{v_1(t), v_2(t), ..., v_k(t)\}$.

The real-time Q-learning algorithm[110] can be used to estimate the value of each consequence $p_i(t)$, $i = 1, 2, ..., k$, and the agent selects the one (action) with the highest value.

Therefore, the value system V is a mapping from the power set of $\mathcal{P}$ to the space of $\mathcal{P}$:

$$V : 2^{\mathcal{P}} \mapsto \mathcal{P}. \tag{2.7}$$

2.3.2 Observation-driven SASE MDP

Defined in Weng[112] a Self-Aware and Self-Effecting agent not only has external sensors S_e and external effectors E_e that sense and act upon the external environment (including the robot body), but also has internal sensors S_i and internal effectors E_i that sense and act upon the internal representation of the brain (not including its body).

In neuroscience, there is no concept of internal sensors and effectors. The brain does not need a receptor to sense the signals in the brain, since the brain signals are already in the desirable form. The concept of internal sensors and effectors is introduced to facilitate computational understanding of the SASE agent.

Definition 2.3.2 (Awareness) *If an agent A senses the states (presence, absence, different forms) of object b through its sensors, we say that the agent A* senses *the*

object b. If the agent is able to recall the association between the sensed various states of b and their resulting effects in a task domain D sensed by the agent, we say that the agent is aware *of object b in the task domain D.*

By definition, an agent must use its sensors, the entry point of its sensory architecture (the input of T), to sense an object. For example, if the state of an object is fed into the architecture, e.g., through the middle of the regressor R, the motor mapping M or the value system V, the agent does not sense the object by the definition because the agent cannot use such information properly as it does with its sensors. In the above definition for awareness, we consider a task domain D because any awareness has a scope. A person who is aware of the boiling temperature of water in a domain (e.g., in a normal environment) may not necessarily be aware of the boiling temperature of water in another domain (e.g., lower in a low pressure environment).

In the definition, we require that awareness must associate with various states of b and the corresponding resulting effects. For example, if a human did not aware of the correspondence between various states of gravity (presence, absence, strong gravity) and the resulting effects (e.g., to a falling object), he would not aware of the object.

With the above definition in hand, we are able to deal with the issue of self-awareness and self-effecting.

Theorem 2.3.1 (Necessary conditions of awareness) *Suppose an agent is aware of its mental activities in a domain. Then the following points must be true: (1) It senses such activities using its sensors. (2) It feeds the sensed signal into its perceptual entry point just like that for external sensors. (3) It recalls the*

association between the different status of the activities and the resulting effects to the environment.

Proof: Point (1) is true because, according to Definition 7 for the awareness of an object, the agent must sense the object using its sensors. Point (2) is true because the status of the object must be sensed and fed into the entry point for sensors for proper perception and effect recall. Point (3) is true because the definition of awareness requires the recall of such an association. □

Based on the previous theorem, let us examine the issue of self-awareness. If an agent runs a Q-learning algorithm (or any algorithm for that matter) but it does not sense the algorithm using its sensors which are linked to its entry point for sensors, the agent is not aware of its own algorithm. For the same reason, humans do not sense the way their primary cortex works and, therefore, normally they are not aware of their own earlier visual processing. This early processing is subconscious, in the sense that it does not require a conscious decision. However, the voluntary part of the mental decision process does require a conscious, willful decision. Therefore, in the architecture design, the parts that require voluntary decisions must be sensed by the agent, and the sensed signals must enter through the entry point of the sensors.

Definition 2.3.3 *The DOSASE architecture is based on the Observation-driven MDP mental architecture, but additionally, the internal voluntary decision is sensed by the internal sensors S_i and the sensed signals are fed into the entry point of sensors, i.e., the entry point of the attention selector T. In order to recall the effects of the voluntary actions, not only is the expected reward value estimated by the value*

system, but also the primed context, which includes not only the primed action, but also the primed sensation.

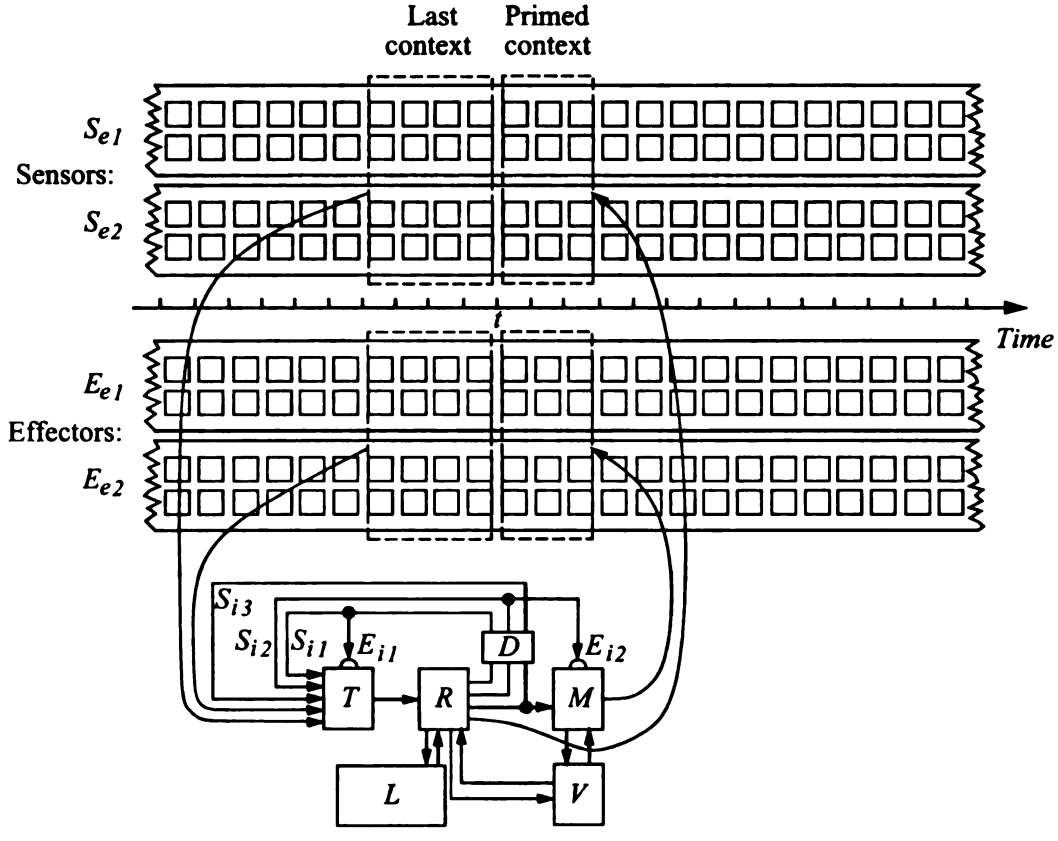


Figure 2.3: The DOSASE MDP architecture for developmental learning.

The architecture illustrated in Fig. 2.3 is the DOSASE MDP architecture. Two voluntary internal actions are modeled by E_{i1} for attention selection, and by E_{i2} for action release. Both internal actions are sensed by the internal (virtual) sensors S_{i1} and S_{i2}, respectively. The external action (not released) is sensed by the virtual internal sensor S_{i3}. Note that when the action is released, it is sensed as external action from E_{e1} and E_{e2}. The primed sensation (which predicts the sensation of S_{e1} and S_{e2}) is used by the value system in selecting the best action according to, e.g.,

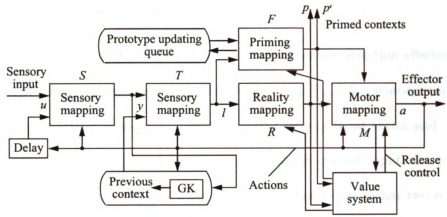


Figure 2.4: A block diagram of the architecture of a sensorimotor system. The lower cognitive mapping realizes the reality mapping and the upper one realizes the priming mapping. GK: gate keeper, an internal effector to actively control the update of the last context.

the novelty (surprise) or the nature of the reward (e.g., sweet or bitter).

The regressor R maps each state $s(t)$ to a set of multiple primed contexts, from which the value system selects a single primed context $p \in \mathcal{P}$. In other words, the composite function of R followed by V gives a mapping: $V \circ R : \mathcal{L} \mapsto \mathcal{P}$. With a SASE agent, both external context (sensed by S_e) and internal context (sensed by S_i) are available in l.

2.4 System architecture

A detailed block diagram of the architecture of a developmental learning system is shown in Fig. 2.4. Each internal and external action output feeds back, through a delay unit, into the next sensory input. This is required by the SASE agent model: the agent must sense and perceive what it does, internally and externally. The input to a sensorimotor system, indicated by the two left-most arrows in Fig. 2.4, is its

target for perception and cognition.

A sensory mapping[125] is needed wherever an attention selection effector (local analysis) or dimension reduction is needed. As shown in Fig. 2.4, a sensory mapping is used to enable attention selection from the current sensory input vector and another sensory mapping enables the attention selection of the current input, the previous context, or both. As mentioned earlier, each sensorimotor system has two cognitive mappings, the reality mapping R and the priming mapping F. A near future context from R is useful for carrying out skilled procedures. A far future context from F is needed to predict the future consequence (e.g., 10 frames or more into the future).

The source of input to the priming mapping has two alternatives: the same input as the reality mapping (as shown in the figure) or the primed context output from the reality mapping. The former allows more accurate control of the timing in the primed context while the latter may enable more efficient "abstraction" of the context in the priming mapping, since the primed context from the reality mapping has already taken the output action into account (e.g., the discriminant analysis by the IHDR tree in section 4.4).

The priming mapping, implemented by a cognitive mapping engine, IHDR needs a *prototype updating queue* whose function is to predict far future context using the Q-learning algorithm[93] in a recursive way. The reality mapping does not need such a queue because it only predicts the next near future.

The motor mapping of a sensorimotor system generates concise representation for stereotyped actions, in addition to the function of action release.

2.5 Major components

The architecture in Fig. 2.4 divides the information processing into three major mappings: the sensory, cognitive and motor mappings. The sensory mapping takes sensory inputs. It may be followed by another sensory mapping to increase the extent of the coverage of space and time. The sensory mappings are followed by a cognitive mapping.

The basic difference between the sensory mapping and the cognitive mappings is that the sensory mapping does not use the information of motor output for its feature space development, but the cognitive mapping does.

The motor mapping has two functions: (1) generating a higher motor representation in space and time for motor output space and (2) receiving signals from cognitive mappings in terms of a higher motor representation and recovering the detailed control signal for every motor that it handles.

In this section, we first discuss these three major mappings, followed by an introduction of innate value system, and finally an algorithm is given to summarize the sensorimotor architecture.

2.5.1 Sensory mapping

The sensory mapping provides representation for all possible receptive fields in space and time and allows attention selection. In the past the role of representation has been well recognized but the purpose of attention selection has not been adequately studied.

Fig. 2.5 shows the hierarchical spatiotemporal organization of sensory mapping. The major functions of the sensory mapping include:

1. Grouping different sources of sensory input (e.g., different pixels in an image or visual and auditory inputs) in space and time. The term "sensory" should be understood as including both raw sensory inputs and processed internal signals.

2. For the grouped set of input, maintaining a complete representation for a hierarchy of all possible (sampled) receptive fields, for the purpose of attention. By sampled receptive fields, we mean a large but finite number of receptive fields at different positions and sizes in space and time.

3. Automatically deriving features in the combined space as new representation. It cannot assume a predetermined representation and therefore, it does not use a symbolic representation.

4. Reducing the dimensionality of the new representation, while minimizing the loss of necessary information.

5. Execution of attention selection as an attention effector, controlled by a signal from other parts of the brain (top down control).

In [125], a simplified sensory mapping, Staggered Hierarchical Mapping (SHM), is implemented. SHM uses the Incremental Principal Component Analysis (CCIPCA) method to automatically develop orientation sensitive and other needed filters. In addition, the internal representation generated by SHM for receptive fields at different

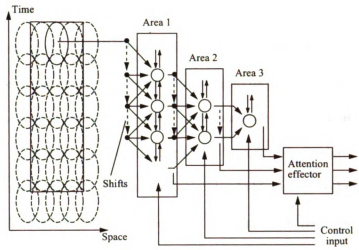


Figure 2.5: The spatiotemporal organization of areas in the sensory mapping. The ellipses represent receptive fields covering both space and time. Areas are organized in a hierarchical way, and the output of an earlier (low order) neural area is used as input to the later (higher order) neural area. Along the pathway of information processing, a neuron in a later area has a larger receptive field than one in an early area. In order to make the correct signal available at the right time for the right input line, we use a time shifting technique. Acting as a time delay unit, the shifter moves each signal to the next line at each time instant.

locations and sizes is nearly complete in the sense that it does not lose important information.

2.5.2 Complementary Candid Incremental Principal Component Analysis

Complementary Candid Incremental Principal Component Analysis (CCIPCA) is a stochastic approximation algorithm to estimate eigenvalue and eigenvector iteratively. It is the major tool to implement the sensory mapping.

PCA is a well-known technique in data compression and feature extraction. It gives a linear transform that converts a set of d-dimensional data into a lower-dimensional space by minimizing the error in the least mean square (LMS) sense.

Therefore, PCA is used to generate new representations from sensory inputs.

A well-known approach to PCA is to solve an eigensystem problem. Given A as the sample covariance matrix of the data set, one can find its eigenvectors and eigenvalues, sort the eigenvalues in descending order, and construct a $k \times d$ matrix T with the rows being the eigenvectors corresponding to the largest k eigenvalues. The matrix T is the sought transform [31]. This is the basic idea behind most techniques for PCA, such as the QR method [35]. However, since this approach requires an estimate of the covariance matrix, the data set usually needs to be completely available at the computation time. This is not appropriate for a developmental robot because of two reasons. First, a developmental robot is an online agent, which senses and responds to the environment continuously. It should not wait until all data is accumulated before doing the processing. Second, when the dimension of the data is high, both the computation and storage complexity grow dramatically. For example, in the eigenface method [57] [104], one of the promising face recognition methods that involves PCA, a moderate grey level image is of 88 rows and 64 columns, which results in a 5632-dimensional vector. Since the sample covariance matrix of a data set of d-dimensional random vectors contains $d(d+1)/2$ independent numbers, this amounts to 15,862,528 numbers!

[120] proposes an algorithm, called complementary candid incremental PCA (CCIPCA), to incrementally compute the principal components of sequentially arrived samples without estimating the covariance matrix. CCIPCA has been shown empirically to be a very efficient estimation algorithm compared with SGA and GHA for high-dimensional data [127].

2.5.3 The cognitive mapping: IHDR tree

The R and F mappings in Fig. 2.4 are implemented by two Incremental Hierarchical Discriminant Regression (IHDR) trees [47].

Learning a function $f : \mathcal{X} \mapsto \mathcal{Y}$ in real time for high-dimensional data remains a nontrivial problem, particularly when the complexity of the function to be estimated are unknown. By surveying the literature of regression in statistical learning, one can identify two classes of methods: global model fitting methods and local model fitting methods.

Local model fitting methods (e.g., IHDR) use temporal-spatially localized (thus computationally efficient) models, in the meanwhile, increase the complexity automatically (i.e., the number and organization of local models) to account for the nonlinearity and the complexity of the problem. They are more suited for incremental real-time learning, especially in the situation where there is limited knowledge about the scene and the robot, itself, needs to develop the representation of the scene in a generative and data driven fashion.

IHDR generates local models to sample the high-dimensional space $\mathcal{X} \times \mathcal{Y}$ sparsely based on the presence of data points in a vector quantization (VQ)[58] manner. IHDR enjoys two nice properties: First, IHDR derives automatically discriminating feature subspaces in a coarse-to-fine manner from input space $\mathcal{X}$. Discriminating features are automatically derived discriminating features at the internal nodes of the tree. The features are most discriminative in the sense that they maximize the trace (or the determinant) of between-class scatter. In this way input components that are

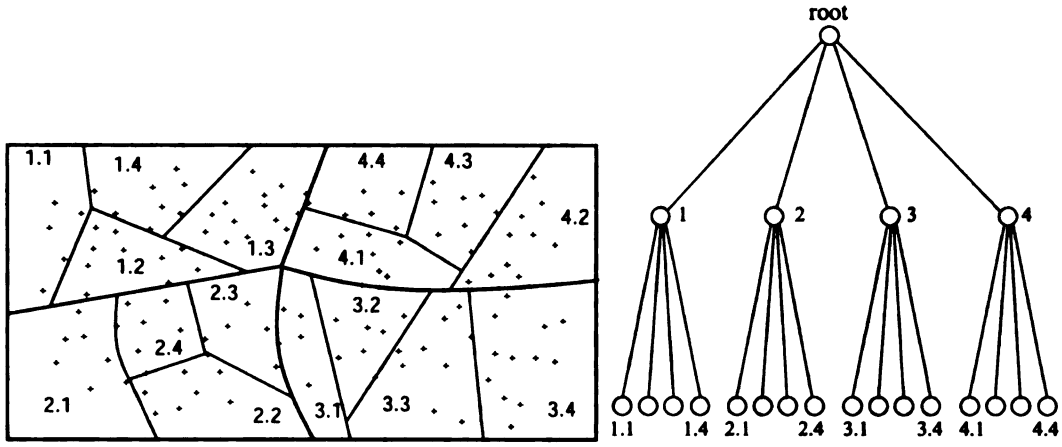


Figure 2.6: Regions in the input space marked $i.j$ where i is the index of the parent region while j is the index of child region. The leaves of the tree represent the finest partition of the space. The decision boundary of the region is of quadratic form.

irrelevant to the mapping's output are disregarded to achieve better discrimination and generalization. Second, IHDR organizes its local models in a hierarchical way, as shown in Fig. 2.6. IHDR's tree structure recursively excludes many far-away local models from consideration (e.g., an input face does not search among nonfaces), thus, the time to retrieve and update the tree for each newly arrived data point $\mathbf{x}$ is $O(\log(n))$, where n is the size of the tree or the number of local models. This extremely low time complexity is essential for real-time learning with a very large memory.

2.5.4 Motor mapping

A major goal of motor mapping is to generate a concise representation for stereotypical motor trajectories so that skilled actions that involve many motors can be represented by high-level motor primitives in a lower-dimensional space. As shown in Fig. 2.7, the architecture of the motor mapping is very similar to the spatiotemporal

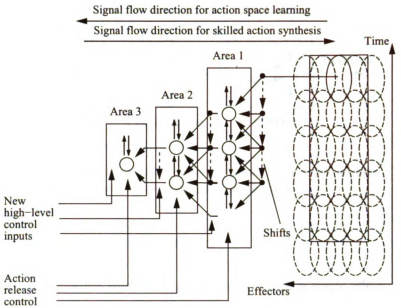


Figure 2.7: The motor mapping as a reverse application of the sensory mapping, but with signal reconstruction.

sensory mapping but it works backwards. Motor mapping receives lower-dimensional "feature" space signals (motor primitives) to synthesize higher-dimensional raw motor signals. Motor mapping has many similarities with those of sensory mapping.

Instead of having attention selection in sensory mapping, motor mapping has a gating system. The gating system plays two roles. one is the role of gating with which the motor mapping evaluates whether the intended action has sufficient action thrust to pass the gate threshold. The other is the role of subsumption [13] in subsuming the lower-level action by the integrated action, as shown in Fig. 2.8.

The primed action from the cognitive mapping includes the desired control signal for each effector as well as for action thrust. The action thrust indicates how consistently the action is issued. The higher the action thrust, the more consistent the action generator is. Therefore, enough thrust must be generated to pass the gate

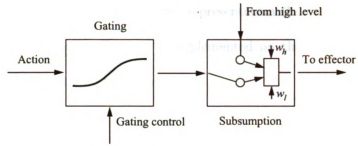


Figure 2.8: An illustration of a simple motor mapping for a single effector. The left is a gating mechanism and the right is a subsumption mechanism.

before an action can be issued to the effector.

Subsumption is needed where inconsistent actions for a single effector are issued from multiple sources, and each source has a different priority. One typical use is that the action derived from a higher level (more sensory integration) has a higher priority over the action derived from a lower level (less sensory integration). The subsumption belongs to innate internal behaviors. When the robot becomes more mature, the selection of behaviors from low levels can be overridden by learned (voluntary) internal behaviors.

If a single motor is considered, motor mapping includes only a gating system for each of the single motor just like the subsumption mechanism for integration from other sensorimotor systems as shown in Fig. 2.8. Through developmental experience, the motors that are highly correlated enable the growth of a new part of motor mapping, denoted as an attached (top right) block to the basic motor mapping in Fig. 2.4. The new part of the motor mapping plays the corresponding role of the gating system, but it is for correlated multi-motor actions. In addition, it performs not only the gating function, but also the reconstruction of higher-dimensional raw motor

signals from a lower-dimensional "feature" representation (higher motor primitives).

In [128], an action gating system is implemented on the SAIL robot to decide whether an action is actually triggered.

2.5.5 Innate value system

Given the last context $l(t)$, the IHDR tree finds the best matched context P' which is associated with a set of primed contexts $\{p_1, ..., p_k\}$. How does the controller select the best primed context? A Q value with each primed context is needed. This is represented by a list $Q = \{q_1, q_2, ..., q_k\}$, in parallel with the primed contexts. Thus, each primed context consists of three components, $(x_{pi}(t), a_{pi}(t), q_i)$, $i = 1...k$.

Definition 2.5.1 *The innate value system chooses the action that has the best value* q.

The value system receives a list of primed contexts at each time instant t and occasional environment reward signal $r(t)$, which is from the biased sensors of the robot (e.g., the aversive sensors such as a "bad button" or appetitive sensors such as a "good button"). Each element in the Q vector starts from an initial value, e.g., a zero value. The updating rule that we summarize here is adapted from Q-learning [109].To keep the temporal order of the latest retrieved primed contexts for real-time local propagation, a primed context update queue (see Fig. 2.4) is needed. The future primed context prototype (including reward) propagates back to the previous prototypes recursively for each time frame in the context update queue. Other prototypes (not in the queue) are not affected and thus are not updated.

2.5.6 Sensorimotor algorithm

We first define a sensorimotor system.

Definition 2.5.1 *A* sensorimotor system *is a system that maps a group of sensory inputs to a group of effector outputs. In addition, it may take some inputs from some sensorimotor systems and send outputs to other sensorimotor systems.*

In neuroscience, a sensorimotor system typically refers to a low-level sensorimotor system, such as a reflexive touch subsystem. However, the sensorimotor system, as defined here, is not limited by this convention. It can model both complex and simple sensorimotor subsystems. The author holds a view that all human higher level brain activities, such as language and thinking, are sensorimotor activities. My work focuses on the development of value systems and sensorimotor systems for robots.

The following is a summary of a sensorimotor system with an innate value system:

1. Initialize the mental cycle count to 0.
2. Grab the current input from the sensory mapping to form the last context $l(t) = (x(t), a(t))$.
3. Go through the sensory mapping to reduce the dimensionality of $l(t)$ while applying the internal actions to the attention selection effector of the sensory mapping.
4. Retrieve the reality (cognitive) mapping R using $l(t)$ to find the best matched prototype l' in the matched leaf node of the cognitive mapping. Its output part is a list of primed near contexts $(A_r(t), X_r(t))$.

5. The innate value system selects the near primed context $p_r(t)$ from the lists $(A_r(t), X_r(t))$.

6. Retrieve the priming (cognitive) mapping F from $l(t)$ to produce the primed far contexts $(A_f(t), X_f(t))$.

7. The innate value system selects the far primed context $p_f(t)$ from the lists $(A_f(t), X_f(t))$.

8. Feed the actions in $p_r(t)$ and $p_f(t)$ through the motor mapping. The current internal action determines whether primed near context or primed far context receives attention in the motor mapping. The attended primed context is $p(t)$.

9. Update both the reality and priming mappings using the primed context $p(t)$. If there is an imposed action from the environment (supervised learning), replace the corresponding part of $a_p(t)$ in $p(t) = (a_p(t), x_p(t))$.

10. Spatially update the primed context lists $(A_r(t), X_r(t))$ and $(A_f(t), X_f(t))$ using the Incremental Vector Quantization technique.

11. Temporally update primed contexts in the prototype update queue for the priming mapping F.

12. Push the current primed context $p(t)$ into the prototype update queue from the tail and pop out the oldest primed context.

13. The motor mapping produces internal and external actions.

14. If the mental cycle time has not been used up, sleep for the remaining time so that the exact time actually spent by this cycle is equal to the fixed pre-specified mental cycle.

15. Increase mental cycle count by 1 and go to Step 2.

The SAIL robot has successfully tested these major components of the proposed architecture. Other experiments on SAIL include: (1) vision-guided navigation[117], (2) grounded speech learning[118], (3) communicative learning[128], (4) novelty and reinforcement learning in the value system[41], and (5) sensory mapping development[125].

2.6 The SAIL robot

All the experiments presented in the following chapters have been validated on our SAIL robot (Fig. 2.9). SAIL is a human-size mobile robot house-made at Michigan State University. Its hardware components are shown in Fig. 2.10 and 2.11.[1].

The drive-base is adapted from a wheel-chair, which enables SAIL to operate both indoor and outdoor. It has two driving wheels in the middle and three supporting wheels, two front ones and a rear one. This drive-base does not have steering wheel. Its turning is achieved by the differential speed of two driving wheels.

Other two major effectors are the robot arm and the pan-tilt units. The six-joint SCORBOT-ER III robot arm was developed by Eshed Robotec (Fig. 2.12). Its controller can control eight motors totally. In addition to the six joint motors, we used one of two spare channels to control the rotary table which serves as the "neck" of the SAIL robot. The two Pan-Tilt Units (PTU) are located in the "head" of the

[1]Figures 2.10 and 2.11 are obtained from W.S. Hwang's PhD dissertation [50] with the permission of the author.

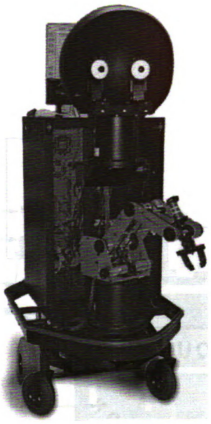


Figure 2.9: The SAIL robot at Michigan State University.

SAIL robot. They are produced by Directed Perception, Inc. Mounted on these PTUs are two CCD cameras as the "eyes" of SAIL. By controlling each of these PTUs, the "eyes" may have pan and tilt motion to cover larger vision field.

The SAIL robot has four pressure sensors on its torso. They can sense push actions and force. 28 touch sensors are distributed on its arm, neck, head, and bumper, which allow human to teach its behaviors by direct touch. These 32 inputs are multiplexed into an eight-channel A/D converter (ADR 2000) using hardwired analog multiplexers.

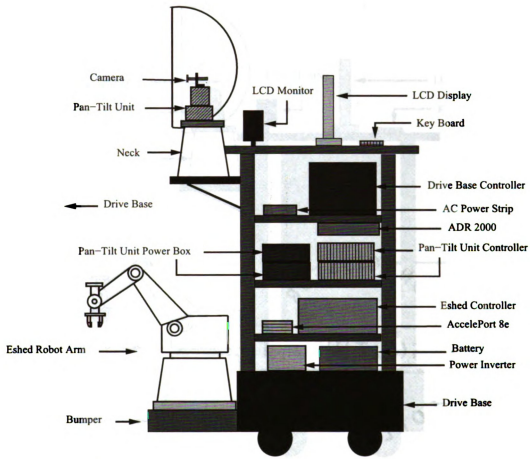


Figure 2.10: The SAIL robot system diagram: left side view.

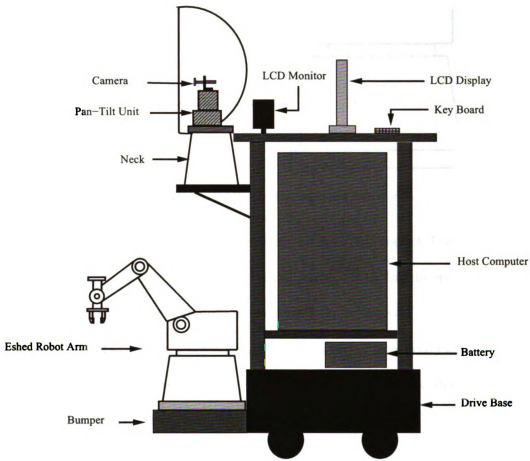


Figure 2.11: The SAIL robot system diagram: right side view.

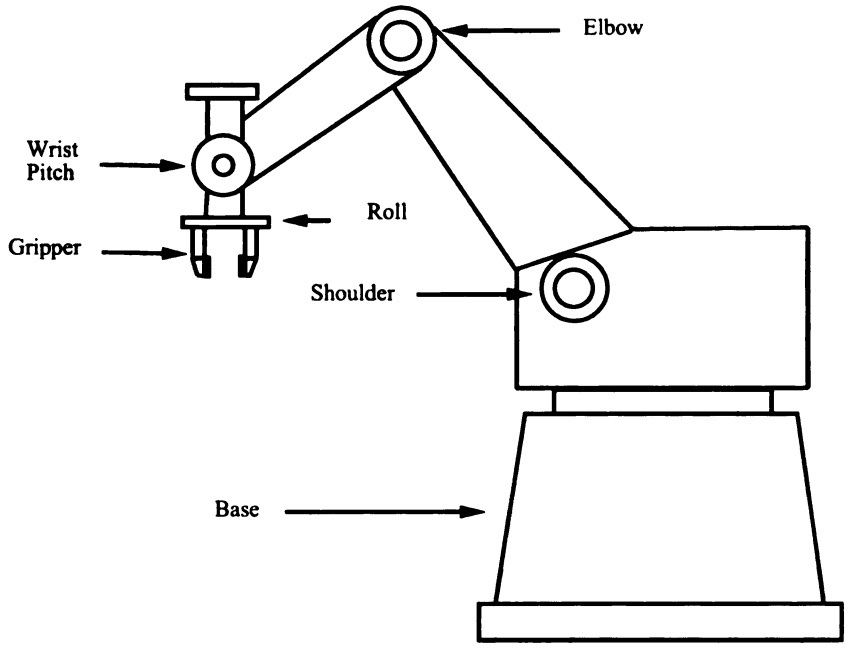


Figure 2.12: The SAIL robot: the Eshed robot arm.

The Eshed controller, the PTU controllers and the ADR 2000 A/D converter are connected to an AccelePort 8e multiport serial adapter from Digi International Inc., which enables the serial communication with the host computer.

The "eyes" of the SAIL robot are Panasonic GP-KS152 industrial Color 1/2 inch CCDs with auto gain control and auto white balance. Two Matrox Meteor II video cards are used for real-time image capturing and digitization.

The "ear" of the SAIL robot is a WMS-PRO wireless microphone system. The wireless microphone operates within a range of 250 feet. It is connected to a sound card (Creative Sound Blaster 16) on the host computer.

Currently, the host computer is a Xeon 2.2GHz dual-processor workstation with 1GB RAM. This allows a real-time memory recall and update as well as real-time effector controls. The monitor is a ViewSonic ViewPanel VPA138 14-inch LCD display,

which is flat, light and easy to carry.

The power of the SAIL robot is provided either by heavy-duty batteries or by line power. An automatic power system APS 750 from TRIPP LITE power protection is used to switch between these two sources.

2.7 Summary

This chapter introduces the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model for developmental learning. The system architecture is presented. There are three major components, e.g., sensory, cognitive, value system, and motor mappings. In the next two chapters, we will focus on cognitive mapping and value system.

Chapter 3

Locally Balanced Incremental Hierarchical Regression

In this chapter, we propose the Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) algorithm as the engine of cognitive mapping for learning in non-stationary environments.

3.1 Introduction

Decision trees (class labels as outputs) and regression trees (numerical vectors as outputs) organize the data in a hierarchy so that retrieval time can be logarithmic, which is essential for the AMD paradigm.

Traditionally, classification and regression trees use a univariate split at each internal node, such as in CART [10], C5.0 [77] and many others. This means that the partition of input space by each node uses a hyper-plane that is orthogonal to an

axis in the input space X. OC1 [67] and SHOSLIF tree [96] are two methods for constructing oblique trees. OC1 uses iterative search to find a split. SHOSLIF uses the principal component analysis (PCA) and the linear discriminant analysis (LDA) to directly compute splits. The Hierarchical Discriminating Regression (HDR) algorithm [47] casts both classification problems and regression problems into a unified regression problem. Virtual labels in the output space provide virtual labels for membership information in forming clusters in the input space.

If training samples are images and video data, the problem becomes harder. These applications give rise to high dimensional data with strong correlation among input components. CART, C5.0, OC1 and other published tree classifiers are not designed for highly correlated high dimensional data. The problem becomes even harder if the learning must be incremental.

The incremental regression problem is as follows: An unknown function $y = f(x)$ generates a stream of input-output pairs arrives as (x_1, y_1), (x_2, y_2), ..., where x_i is the input to the system and y_i is the output from the system. The goal is to approximate function f incrementally by a regressor f_i, where f_i is based on previous approximator f_{i-1} and the new samples (x_i, y_i), $i = 1, 2, ...$ Fig. 3.1 shows the procedure. In AMD, the regression technique has to satisfies all of eight requirements of AMD.

Generally, the incremental hierarchical discriminant regression (IHDR) by Hwang and Weng [49] grows the regression tree incrementally with the eight requirements as the designed goal. However, IHDR face several challenging problems. First, some data sets have high-dimensional data in input space while there are only few labels in output space. Some financial and marketing data only have two labels. For instance,

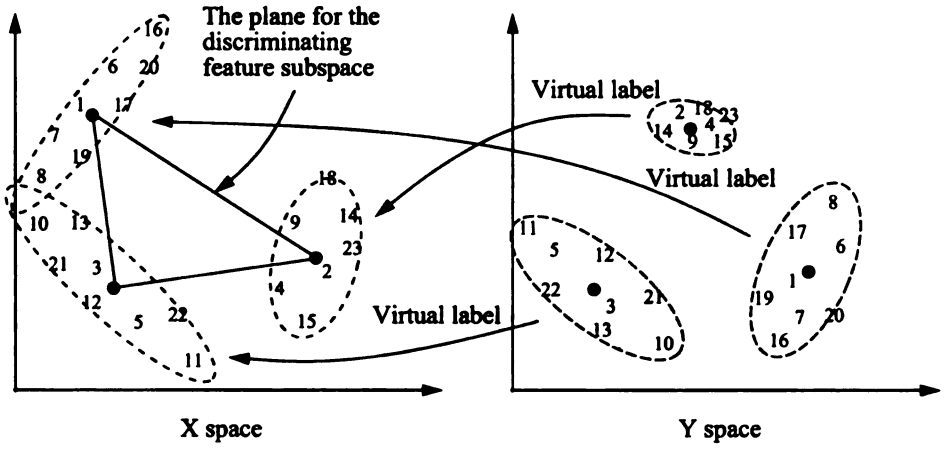


Figure 3.1: Y-clusters in space $\mathcal{Y}$ and the corresponding X-clusters in space $\mathcal{X}$. Each sample is indicated by a number which denotes the order of arrival. The first and the second order statistic are updated for each cluster. The first statistic gives the position of the cluster, while the second statistics gives the size, shape and orientation of each cluster.

usually there are two kinds of customers in the market: satisfied or not. In this case, the dimensionality of discriminant subspace of IHDR is only one, which is not enough to partition a high-dimensional input space. Second, in real-time learning the statistics, a segment of sensory stream is not stationary, but is typically very biased. Consider vision-based navigation. If the robot learns by trying first in a straight corridor, almost all incoming samples are associated with the control signal ("Go straight"). A node must spawn children if the number of stored prototypes in it hits a certain limitation. Otherwise, the retrieval time increases linearly. This situation of biased statistics is true with human mental developmental [29]. The above problems are very challenging and have not been addressed by the existing literature. We proposed a novel node self-organization and spawning strategy which generates more discriminant subspace by forming multiple clusters for one class and balancing the

number of samples in each cluster. The requirements for real-time and incremental learning are satisfied too.

3.2 Locally balanced incremental hierarchical discriminant regression

3.2.1 Incremental hierarchical discriminant regression

Many problems like content-based retrieval, vision-based navigation and intelligent data engineering can be formulated as a complicated function which maps high-dimensional input and the current state to low-dimensional output signals. We use a decision tree to approximate this function. In order to build such a tree, two types of clusters are incrementally updated at each node of the IHDR algorithm — y-clusters and x-clusters [47]. The y-clusters are clusters in the output space $\mathcal{Y}$ and x-clusters are those in the input space $\mathcal{X}$. There are a maximum of q clusters of each type at each node. The idea is that each node models a region of the input space using q Gaussians.

For each new sample (x, y), y finds the nearest y-cluster in Euclidean distance and updates the center of the y-cluster. This y-cluster indicates which corresponding x-cluster the input (x, y) belongs to. Then, the x part of (x, y) is used to update the statistics of the x-cluster (the mean vector and the covariance matrix). Each node

keeps up to q x-clusters. The centers of these q x-clusters are denoted by

$$C = \{c_1, c_2, ..., c_q \mid c_i \in \mathcal{X}, i = 1, 2, ..., q\}. \tag{3.1}$$

In online-learning, the robot updates the tree's statistics incrementally for each new training example. The average of n input examples $x_1, x_2, ..., x_n$ can be recursively computed from the current input data x_n and the previous average $\bar{x}^{(n-1)}$ by equation (3.2):

$$\bar{x}^{(n+1)} = \frac{n - \mu(n)}{n+1}\bar{x}^{(n)} + \frac{1 + \mu(n)}{n+1}x_{n+1} \tag{3.2}$$

where $\mu(n)$ is a parameter. If $\mu(n) > 0$, the new input gets more weight than old inputs. We called this implementation the amnesic average. The covariance matrix can be updated incrementally by using the amnesic average too.

$$\Gamma_x^{(n+1)} = \frac{n-1-\mu(n)}{n}\Gamma_x^{(n)} + \frac{1+\mu(n)}{n}(x_{n+1} - \overline{x}^{(n+1)})(x_{n+1} - \overline{x}^{(n+1)})^T \tag{3.3}$$

If the node is mature enough (the number of samples hits a certain limitation), it will spawn children, which means the region of space is modeled by a finer Gaussian mixture. Thus, a coarse to fine approximation of probability distribution of the training samples is realized. A more detailed description of the algorithm can be found in [49].

3.2.2 Locally balanced tree and time smoothing

However, IHDR faces several challenging problems. First, many applications in intelligent data engineering only have two labels in output space, for example, whether a customer switches from one company to another one. As shown in Fig. 3.2, supposed there are two classes specified by "+" and "-". The input dimensionality is three. m_1 and m_2 are two vectors describing the centers of the two classes. The discriminant vector D_1 is determined by Eq. 3.4.

$$D_1 = S_w^{-1}(m_1 - m_2) \tag{3.4}$$

According to the IHDR algorithm described above, the dimensionality of within-class scatter matrix S_w is one because $q = 2$ and S_w is $(q-1) \times (q-1)$. The discriminant vector is along the direction of $m_1 - m_2$. Obviously, D_1 is not enough to partition these two classes.

The second challenging problem is caused by the nonstationary statistics of a segment of sensory stream. Given vision-based navigation as an example, if the robot learns by trying first in a straight corridor, almost all incoming samples are associated with the control signal ("Go straight"). In order to make sure that the learning is conducted in real-time, a node must spawn children if the number of stored prototypes in it hits a certain limitation. Nonstationary statistics causes the following problems: (a) A segment of output contains only few values, which is similar to the first case. In some extreme cases, only one value for y. (b) The performance of a tree cannot degrade after spawning children from a node. It is challenging for new children to

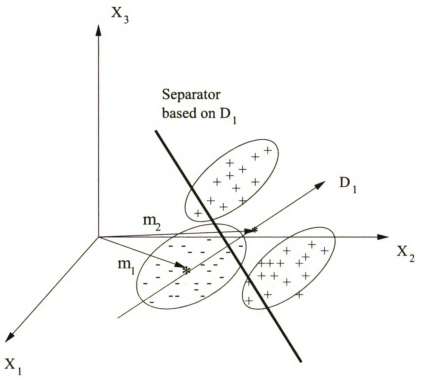


Figure 3.2: A one-dimensional discriminant subspace is formed for two classes.

maintain the performance realized earlier by the experienced parent. (c) Due to a lack of sample statistics for finer prototypes, each leaf node contains a moderate number (up to N) of micro-prototypes in the form of $(\bar{x}, \bar{y})$, where $(\bar{x}, \bar{y})$ is the average of a few similar samples. Spawning children requires distribution of micro-prototypes, which could slow down the system response without a carefully designed spawning strategy. In order to tackle these problems, we proposed a new node self-organization and spawning strategy.

Suppose there are two classes in a task, say c_1 ("+"), c_2 ("-"), as shown in Fig. 3.3. The number of micro-prototypes of each class is N_i, $i = 1, 2$. The number of clusters in a node is q. If $q > 2$ (say $q = 3$), we try to allocate some micro-prototypes of c_1 to

those clusters, which are not in use. Each class in the old tree could be mapped to multiple clusters in the balanced tree. For the ith class, the algorithm will generate q_i' clusters in the new node.

$$q_i^{'} = q\frac{N_i}{N_t} \tag{3.5}$$

where N_t = total number of micro-prototypes in this leaf node. In Eq. 3.5, the $q_i^{'}$ could be float. Eq. 3.6 converts it to an integer.

$$q_i^{'} = [\sum_{j=1}^{i} q_j^{'}] - \sum_{j=1}^{i} \overline{q}_j^{'} \tag{3.6}$$

where $i = 1, 2, ..., N_y$, N_y is the number of classes of training samples. $\overline{q}_j^{'}$ is defined by Eq. 3.7, which guarantees that at least one cluster is generated for each old class.

$$\overline{q}_i^{'} = \max\{1,\ q_i^{'}\} \tag{3.7}$$

Fig. 3.3 shows the balanced node. Originally, there were 2 clusters. Now, there are 3 clusters. Two clusters are formed for c_1. A two-dimensional subspace passes the head tips of the three center vectors: $m1$, $m2$ and $m3$. The scatter vectors are defined by the following equation:

$$s_i = m_i - \bar{m} \tag{3.8}$$

where $i = 1, 2, 3$, $\bar{m}$ is the mean of all center vectors. Let S be the set that contains

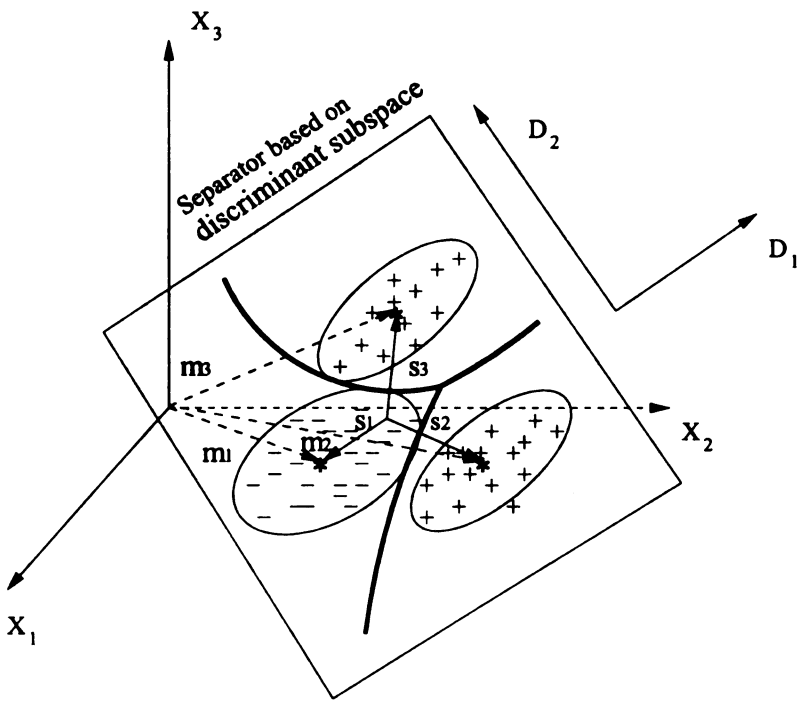


Figure 3.3: Locally balanced node. Two clusters are generated for class 1. a more discriminant subspace is formed.

these scatter vectors: $S = \{s_i | i = 1, 2, 3\}$. The discriminant subspace spanned by S, denoted by span(S), consists of all the possible linear combinations from vectors in S. By applying Gram-Schmidt Orthogonalization (GSO), we can get two orthogonalized discriminant vectors D_1 and D_2. Obviously, more discriminating features are derived from the new node. We should note that the algorithm only balances the micro-prototypes in each node, while the entire tree may not be strictly balanced. So this is a local balancing method.

After the node is balanced, the statistics of the new structure must be generated. However, in real-time navigation, the updating frequency is around 10Hz. If the robot calculates the statistical information in one time slot, it will get stuck in the road,

which is not allowed. In order to smooth the updating time for each new sample, the algorithm keeps two sets of statistics. The old statistic is used to build the tree. At the same time, the new balanced statistic is updated incrementally after the old statistic is half-mature (the number of micro-prototypes in the node hits half of the limitation). When the new statistics is mature, the old one is thrown away and micro-prototypes are redistributed and updated based on the new structure. This smoothing strategy works very well for real-time navigation. When the number of micro-prototypes hits the limitation N, a node should spawn children based on the new statistics. The performance cannot degrade after spawning. This can be resolved by redistributing micro-prototypes.

3.2.3 Algorithm

Procedure 2 `Update-tree`*: Given the root of the tree and sample* (x, y)*, update the tree using* (x, y).

1. *From the root of the tree, update the node by calling* `Update-node`.
2. *For every active cluster received, check if it points to a child node. If it does, explore the child node by calling* `Update-node`.
3. *Do the above steps until all leaf nodes are reached.*
4. *Each leaf node keeps micro-prototypes* $(\hat{x}_i, \hat{y}_i)$ *that belong to it. If* y *is not given, the output is* $\hat{y}_i$ *if* $\hat{x}_i$ *is the nearest neighbor among these micro-prototypes.*
5. *If* y *is given, do the following: If* $||x - \hat{x}_i||$ *is less than certain threshold,* (x, y)

updates $(\hat{x}_i, \hat{y}_i)$ *only. Otherwise,* (x, y) *is a new sample to keep in the leaf.*

6. *If the leaf node is half-mature, call* **Balance-node**.

7. *If the number of micro-prototypes hits the limitation required for estimating statistics in the new children, the leaf node spawns q children and is frozen. The micro-prototypes are redistributed through multiple steps.*

Procedure 3 **Update-node**: *Given a node N and* (x, y)*, update the node N using* (x, y) *incrementally.*

1. *Find the top matched x-cluster in the following way. If y is given, do (a) and (b); otherwise do (b).*

   (a) *Update the mean of the y-cluster nearest y in Euclidean distance. Incrementally update the mean and the covariance matrix of the x-cluster corresponding to the y-cluster.*

   (b) *Find the nearest x-cluster according to the probability-based distances. Update the x-cluster if it has not been updated in (a). Mark this central x-cluster as active.*

2. *Return the chosen x-cluster as active cluster.*

Procedure 4 **Balance-node**: *When the total number of micro-prototypes in this leaf is equal to half of the limitation, then balance the micro-prototypes of each cluster.* N_y *is the number of classes in the old node.*

1. *For each y, if* $q \leq N_y$*, do y-space clustering.*

2. *Otherwise, ($q > N_y$), sort X-clusters according to the number of micro-prototypes (increasing).*

3. *Calculate the number of clusters for each old class by using Eq. (3.6) and Eq. (3.7).*

4. *Conduct K-mean algorithm [28] to generate new clusters. Reallocate micro-prototypes to each new cluster.*

5. *Calculate new statistics for the balanced node.*

3.3 The experimental results

In order to show the effectiveness of the new algorithm, we applied our method to two kinds of problem: financial data engineering and vision-based navigation. First, we tried synthetic data.

3.3.1 Experiments using synthetic data

The motivation of using synthetic data for testing is to investigate the behavior of balancing a tree. This experiment used a data set that has two clusters. The number of dimension is two. Each cluster are modeled by a Gaussian distribution. The centers of the clusters are at (-2,-2) and (2,0), respectively. q is set to 3. The covariance matrix are:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 6 & 2 \\ 1 & 1 \end{bmatrix}$$

One cluster has 20 micro-prototypes; another cluster has 40. Fig. 3.4 shows the initial distribution of these two clusters. Fig. 3.5 shows the balanced distribution,

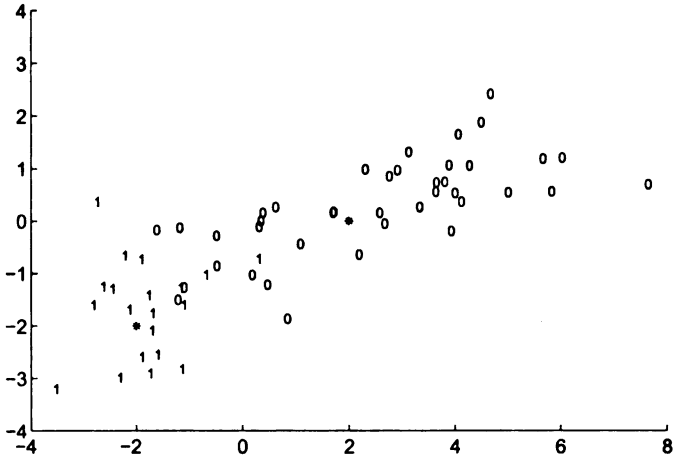


Figure 3.4: Initial distribution, two clusters.

now we have 3 clusters. Obviously, the micro-prototypes are balanced in the new

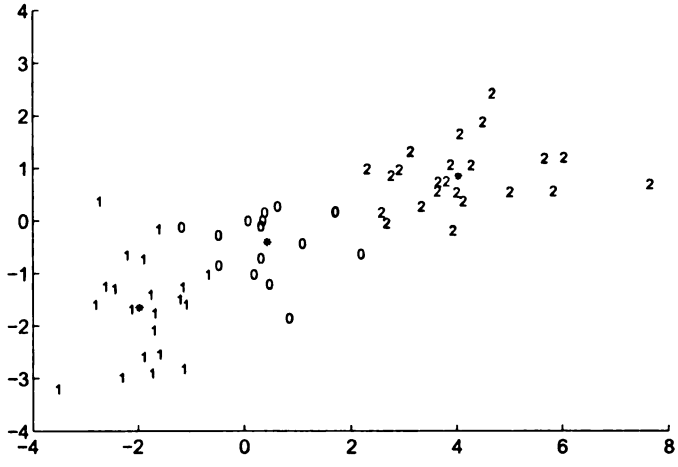


Figure 3.5: The balanced distribution, 3 clusters.

node. The center of each cluster is specified by a star.

3.3.2 Experiments using marketing data

Two kinds of marketing data were used to test the algorithm (Tab. 3.1). The first data set is about the ranking of decision making. Totally there are 12404 samples. Two thirds of all samples were used for training. The remaining samples were used to for testing. The input dimensionality is 12 and there are 16 rankings of decision making. The second data set is the ranking of customer satisfaction. Only two types of labels are provided: satisfied or not.

Table 3.1: Financial data sets

Data set	Training samples	Testing samples	Input dimensions	Classes
Data1	9306	3098	12	16
Data2	1768	590	18	2

We compared the error rate of the proposed LBIHDR algorithm with HDR, IHDR and other methods. As we can see from Tab. 3.2, if the number of classes is larger (data set 1), then the LBIHDR has the same performance as that of IHDR. However, data set 2 shows that the new algorithm improves the recognition rate. In this experiment, $q = 6$ and $N_t = 100$. The the dimensionality of discriminant space is five while that of the original IHDR is only one. More discriminant subspace is formed and the error rate reduces from 12.92% to 8.22%.

3.3.3 Experiments using navigation image data

We applied the Locally Balanced IHDR algorithm to vision-based indoor navigation on our SAIL robot, as shown in Fig. 3.6. The robot is trained online on second floor

Table 3.2: Error rates of IHDR and HDR.

	Data1	Data2
HDR	13.37%	13.57%
IHDR	12.98%	12.92%
LBIHDR	12.98%	8.22%

Figure 3.6: SAIL robot navigating through the corner of the corridor.

in the Engineering building of our University. In the training phase, at each time interval, the robot accepts a pair of stereo images and a control signal, incrementally updates its context which contains past sensory inputs and actions, and then updates the statistics of the decision tree. In the testing phase, a pair of input images is used to retrieve the best match. The associated control signal of the match is the output. This is a very challenging problem, which approximates the mapping from high dimensional input space into a low dimensional output space by using a decision tree and all the computations have to be performed in real time (about 10Hz). Some of the example input images are shown in Fig 3.7.

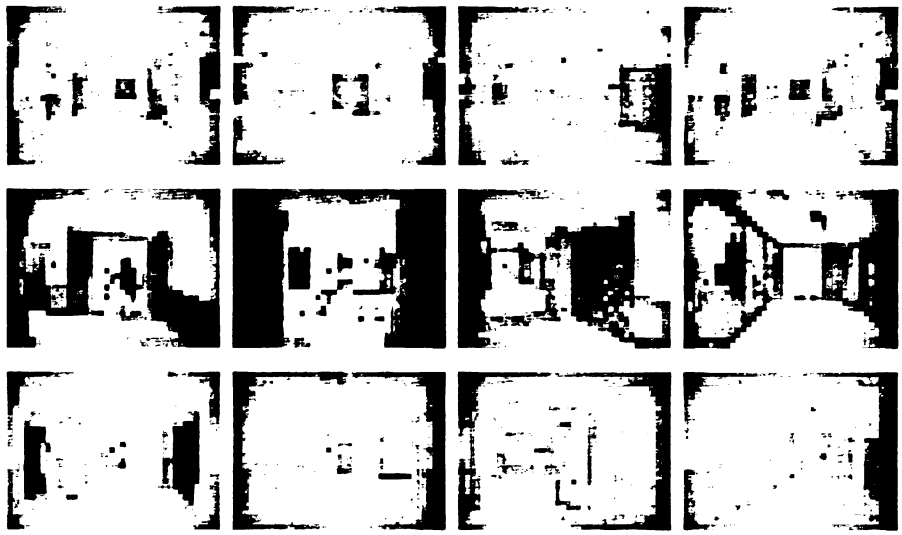


Figure 3.7: A subset of images which are inputs to guide the robot turn.

The robot is trained around the corridor. In each trip there are about 5000 thousand pairs of images and corresponding control signals. In table 3.3, a total of 5443 color stereo images with the corresponding heading directions were used for training (one trip). The resolution of each image is 30 by 40. The input dimensionality of the IHDR algorithm is $30 \times 40 \times 3 \times 2 = 7200$, where 3 is the length of context. We used the other 5764 (another trip) stereo images to test the performance of the trained system. There are three kinds of output control signals: "Go Straight", "Left", "Right". We performed both resubstitution and disjoint tests with different rejection measurements. The parameters are defined as follows: $q = 2$, the limitation of the number of samples in each node is 200. In table 3.3, R stands for resubstitution test, D stands for disjoint test. The performance is very good for such a large high dimensional data set. The rejection measure determines whether a coming sample is used to update the tree or not. If the distance between the coming sample and the

retrieved micro-prototype is less than certain measure, the sample would be rejected. For example, if the rejection measure is 0, then all samples will be saved in the tree. In this case, the tree is about 130Mb. If the rejection measure is 2000, the tree is only 45Mb and the retrieval speed is faster.

Table 3.3: Resubstitution and disjoint tests for two trips with different rejection measurements.

Rejection measure	Training set	Correct number	Correction rate
0 (R)	5443	5443	100%
2000 (R)	5443	5389	99.01%
0 (D)	5764	5434	94.28%
2000 (D)	5764	5412	93.88%

The performance is improved significantly compared with the existing IHDR [49]. As shown in Tab. 3.4, in the resubstitution test, the recognition rate of LBIHDR can reach 100%. In the disjoint test, the recognition rates of LBIHDR is 94% while IHDR can only reach 87%. The retrieval time is also less because the new algorithm balances the distribution of samples in each node. The result testifies that LBIHDR works very well for nonstationary data like visual images in navigation. The tree

Table 3.4: Comparison of IHDR and LBIHDR.

Method	Resubstitution test	Disjoint Test	Testing time (mm)
IHDR	92.72%	87.35%	28.6
LBIHDR	100%	94.28%	15.7

structure of the navigation experiment is shown in Tab. 3.5. The left part is the number of nodes in each layer; the left part is the number of samples in each layer.

Because the number of samples is balanced in each internal node, the tree will not grow too deep (only four level) that helps build up more reliable statistics in each node.

Table 3.5: The tree structure of the navigation experiment.

Layer	No. of nodes	No. of samples
1	1	0
2	8	335
3	48	2527
4	64	2702

In order to guarantee that IHDR can be implemented in real time, we applied the time smoothing algorithm. Fig. 3.8 records the updating time for each sample in different steps. Here only shows the time profile of 715 samples. The first row shows the total learning time of each sample. The second row shows the updating time except doing local batch and redistribution. Time records of calculating new statistics and redistribution are shown in third and fourth rows respectively. When the node is half mature (i.e, the node already has 100 micro-prototypes), the new statistics are computed through multiple steps; when the node is mature (i.e, the node has 200 micro-prototypes), micro-prototypes are redistributed. At these moments, the learning time reaches peaks. However, with the time smoothing algorithm, the updating rate still could reach about 3-4Hz, which is extreme important for online learning.

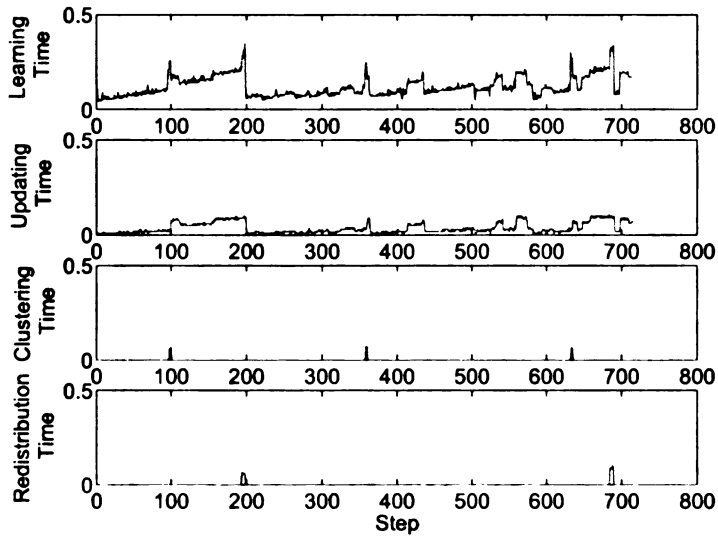


Figure 3.8: The timing recording for learning a set of 715 images.

3.4 Conclusions

In order to solve the major problems due to the nonstationary nature of input-output experience and fewer labels in output space with high-dimensional input space, a new node self-organization and spawning strategy is proposed. Partition of the input space is carried out without relying on a sufficient number of output clusters. Multiple clusters could be generated for one class. The number of samples in each cluster is balanced. Thus, more discriminant subspace can be formed. The performance of the tree is stable before and after children spawn. The distribution of computational load over time during spawning avoids a significant slow down of the system response time. The experiments with marketing data and the real-time, incremental learning of vision-based navigation showed that the new locally balanced IHDR technique performed well in the different applications.

Chapter 4

The Value System of A Developmental Robot

In this chapter, we propose a value system based on novelty and reinforcement learning will be proposed. The working of the value system is demonstrated by an experiment that producing visual attention controls.

4.1 Introduction

Developmental agents face a continuously changing stream of input data. If it sits there and does nothing, how can it develop mentally? Then we have to ask why a developmental robot should do anything in the first place: "Something has to start the process in the first place. Something has to motivate infants to look, to reach, to mouth, to seek out information about their worlds" [98]. This issue also arises in the case of a developmental robot. what drives agents to explore their environment and

learn about it? How to model the motivation? Psychologists addressed this issue by introducing the concept of intrinsic motivational forces such as "drives." Piaget [74] suggested that the need to adapt to the environment drives infants' behaviors. The essence of these proposals is that this internal driving force builds biases into the agent. These biases make the agent prefer one behavior over another. That is, they encode what is value for an agent.

4.1.1 The value system of a developmental robot

An indispensable part of a developmental program is its value system, which signals the occurrence of salient sensory inputs, modulates the mapping from sensory inputs to action outputs, and evaluates candidate actions. It provides motivation that drives mental development. Since a developmental robot is not task specific, without a value system, it simply does nothing. This is very different from a traditional robot which is designed to perform a specific task only – the task-specific representation and rules have already been programmed into the robot.

The value system of the central nervous system of a robot at its "birth" time is its innate value system. It further develops continuously throughout its "life" experience. A value system at a particular mental age after birth is called a developed value system.

4.1.2 Innate value system

The innate value system of a developmental robot is designed by the programmer. It includes the following two aspects:

1. innate spatial bias.

2. innate temporal bias.

The term "spatial" here means different sensory elements are located at different locations of the robot body. The term "temporal" means that the spatial bias changes with time. A pain signal from a pain sensor is assigned a low value and the signal from a sweet taste is assigned a high value. This is the innate spatial bias. If a pain sensor continuously sends signals to the brain for a long period of time, a newborn does not feel the pain as strong as it is sensed for the first time. This is the temporal bias.

4.1.3 The requirements of value systems

The challenges of designing and implementing the value system for a developmental robot include: (1) It must be applicable to all the possible unknown tasks. For example, it is not possible to predefine salient features, because salient features for one task (e.g. motion in intruder detection) may not be so for another task (e.g. driving). (2) It must be applicable to different maturation stages. For instance, playing with toys is interesting for youngsters but not for adults. (3) It must provide guidance for behaviors all the time, in every fraction of a second since the robot's actions are updated at this high temporal frequency. Reinforcers (e.g. sweet tastes

and pain senses) are not sufficient for a value system. A developmental robot mostly lives during a time where a search for such reinforcers is not a goal (e.g. while a child plays). (4) The value system must work with a real-time system that incrementally grows and updates representation and memory through interactions with an open, complex real physical world. Without this, the value system is unable to deal with increasingly complex tasks in unpredictable environments.

4.2 Background

4.3 Neurobiological background for value systems

Neuroscience studies have shown that a value system has the basic function of the multiple diffuse ascending systems of the vertebrate brain [83]. The detailed mechanisms of the value system and its development are mostly unknown although some characterizations of this system are available.

An adaptive organism must be able to predict future events such as the presence of mates, food, and danger. Predictions give an animal time to prepare behavioral reactions and can help improve the choices an animal makes in the future. Some theories of reward-dependent learning suggest that learning is driven by the unpredictability of the reward and sensory input. Motivational values arise either through innate mechanisms or, more often, through learning. In this way, rewards help to establish values system for behavior and serve as key references for behavioral decision. Schultz [82] described the basic reward process in the brain.

Experiments on monkeys' brains show information about rewards seem to be processed in different forms by neurons in different brain structures, ranging from the detection and perception of rewards, through the expectation of imminent rewards, to the use of information about predicted rewards for the control of goal-directed behavior. Here are a few descriptions of how the brain processes reward. "Only a limited number of brain structures seem to be implicated in the detection and perception of rewards and reward-predicting stimuli. The midbrain dopamine neurons report the occurrence of reward relative to its prediction and emit a global reinforcement signal to all neurons in the striatum and many neurons in the prefrontal cortex. The dopamine reward response might be the result of reward-detection activity in several basal ganglia structures, the amygdala and the orbitofrontal cortex. Some neurons in the dorsal and ventral striatum, orbitofrontal cortex, amygdala and lateral hypothalamus discriminate well between different rewards, which might contribute to the perception of rewards, be involved in identifying particular objects and/or provide an assessment of their motivational value."

More researchers focused their research on dopamine neurons and their relation to reward prediction and the value system. Montague and his colleagues [64] developed a theoretical framework that shows how dopamine systems could distribute to their targets a signal that represents information about future expectations. In particular, they show how activity in the cerebral cortex can make predictions about future receipt of reward and how fluctuations in the activity levels of neurons in diffuse dopamine systems above and below baseline levels would represent errors in these predictions that are delivered to cortical and subcortical targets.

4.4 Computational model of value systems

Based on abundant results in neuroscience, psychology and animal learning, scientists try to build the value system into autonomous agents.

4.4.1 Modeling operant conditioning

In the last three decades, a number of computational models of operant conditioning were produced. This method provides a way of programming agents by reward and punishment without needing to specify how the task is to be achieved. Typically, in reinforcement learning [54], statistical techniques and dynamic programming methods are used to estimate the utility of actions. A standard model is shown in Fig. 4.1. An agent is connected to its environment via perception and action. In each step

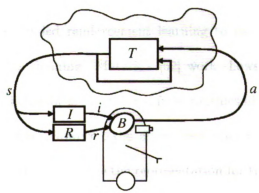


Figure 4.1: The standard reinforcement-learning model. (Adapted from [54]).

of interaction the agent receives input, i, some indication of the current state of the environment, s; the agent then chooses an action, a, to generate as output. The action changes the state, and the value of this state transition is communicated to the agent through a reinforcement signal, r. The agent's behavior, B, should choose actions

that tend to increase the long-run sum of values of the reinforcement signal. The basic idea is to learn what to do—how to map situations to actions—so as to maximize a numerical reward signal through a trial-and-error procedure. More elaborate models are TD(λ) and Q-learning [95] [110]. We should note that all of these models only consider the interaction between an agent and the external world while the internal sensors and effectors which sense and act on the internal world (the brain) are not discussed.

4.4.2 Modeling value systems

Although reinforcement learning is a breakthrough in machine learning, it is not enough for modeling all the behaviors of animals. Researchers try to apply value systems to model more complex adaptive behaviors of robots like chaining, forging, etc. Touretzky [102] extended reinforcement learning to model some functions of operant learning, such as chaining. Mataric's [62] work shows forging behaviors of mobile robots. Forging behavior is constructed from routines like wandering, homing, aggregation and dispersion. However, the above two cases use symbolic methods, which means the human trainers define the representation for the robots. That is not applicable to developmental robots.

A developmental perspective to neural model of learning

Ogmen's work [68] proposed a developmental perspective to the neural network model of learning. The choices that the network makes regarding what objects to approach are not based on minimization of a fixed, predetermined, global cost function. Instead,

the choices are based on three interacting sets of criteria: reinforcement, novelty, and habit. The model combines such design principles as adaptive resonance, opponent processing, and lateral inhibition. In this framework the interplay between reinforcement, novelty, and habit is dynamic; which of these factors is more important is heavily dependent on the environment context in which the network (or robot) finds itself. Even the stimulus criteria that maximize reinforcement can change dynamically with the context. However, only a simple simulation experiment is reported in the paper. The work is not directly applicable to complex high-dimensional sensory signals with distributed representation.

Modeling dopamine neurons

Much evidence, reviewed by Schultz [82], suggests that dopamine (DA) cells in the primate midbrain play an important role in reward and action learning. As a kind of neurotransmitters, dopamine affects brain processes that control movement, emotional response, and ability to experience pleasure and pain. Electrophysiological studies in both instrumental and classical conditioning tasks support a theory that DA cells signal a global prediction error for summed future reward in appetitive conditional tasks, which is very similar to a temporal difference (TD) learning model. In this model of primate dopamine neurons, their phasic activity reports a prediction error for future reward. Dayan [55] addresses two important sets of anomalies: generalization and novelty. Generalization responses are treated as the natural consequence of partial information; novelty responses are treated by the suggestion that dopamine cells multiplex information about reward bonuses, including exploration bonuses and

shaping bonuses. They interpret this additional role for dopamine in terms of the mechanistic attentional and psychomotor effects of dopamine, which guides exploration. However, they did not provide a source of information nor a computational model to measure novelty. Another point we need to note is that dopamine neurons are a part of the value system in the brain, more comprehensive models are necessary to model the complex functionality of value systems.

4.5 The value system of an AMD robot

Studies in animal learning testify that different reinforcers (e.g. sweet tastes and pain senses) carry different values in the animals value system. However, computational studies of reinforcement often model rewards into a single value, delivered from a separate reward channel [94] [89]. This single-value modeling facilitates understanding and simplifies computation. However, primed sensation (what is predicted by the robot) has been neglected. The value of an action under a state is determined by the rich nature of the primed sensation, not just a global value. In the value system presented here, the value is not only derived from separate reward channels but also from primed sensation.

The framework also addresses a fundamental limitation of reinforcer based value system. Reinforcers are typically sparse in time: they are delivered at infrequent spots along the time axis. Novelty from primed sensation is however dense in time, defined at every sensory refresh cycle. Thus, the framework enables continuous exploration of a developmental robot, not just exploring a task-specific setting. We should note that

the so defined just models an innate value system, which should develop through a robot's life experience. Different primed actions and primed sensations are associated with different Q-values. These Q-values determine the probability for each action to be taken. The novelty and reinforcers jointly determine the Q-value, which changes with experience. In this way, the value system is learned through a developmental robot's interactions with the world.

To demonstrate the working of the value system, we chose a challenging behavior domain: visual attention through neck pan actions. Although the degree of freedom of motor actions is only one, the difficulties lie in the highly complex, uncontrolled visual environment. Our approach is fundamentally different from traditional task-specific approaches in that we cast visual attention selection as a behavior guided by a value system, which is applicable to any task. For example, our value system does not define saliency of features, but instead novelty based on experience. A novel stimulus for one robot at one time is not novel if it is sensed repeatedly for long by the same robot. The visual attention selection in a general visual environment corresponds to a very challenging problem that cannot be systematically addressed without the general task-nonspecific approach described here.

4.6 System architecture

The basic architecture implemented for the SAIL robot is shown in Fig. 4.2. The sensory input is represented by a high dimensional vector at each time instant so that each component corresponds to a pixel. It is the cognitive mapping module that

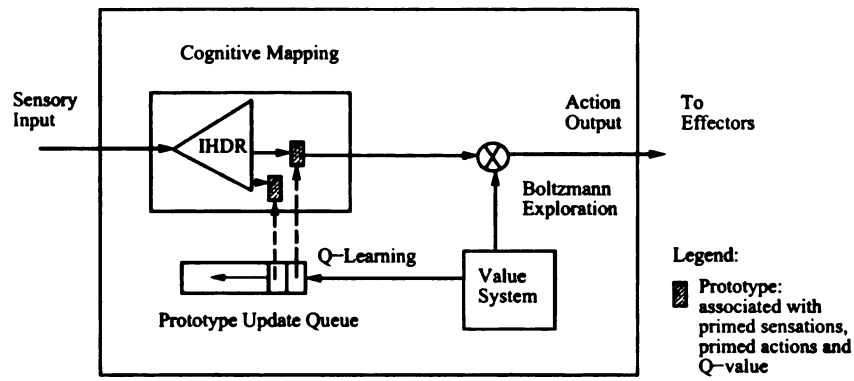


Figure 4.2: The system architecture of SAIL experiments.

derives the most discriminating features from input streams and maps each input vector to the corresponding effector control signal. The cognitive mapping is realized by Incremental Hierarchical Discriminant Regression (IHDR) [114], simulating the high dimensional mapping of a cortical area. A more detailed explanation is beyond scope. Basically, given each input vector, the IHDR tree provides an output consisting of three components: the primed sensation, the primed action and the corresponding Q-values. Primed actions are the possible actions in each state. The probability to take each primed action is based on its Q-value. The primed sensation predicts what will be the actual sensation if the corresponding primed action is taken.

Novelty is measured by the difference between primed sensation and actual sensation. In the value system, novelty is integrated with reinforcement learning so that humans can issue rewards to modulate a developmental robot's behavior. In order to let the robot explore more states, Boltzmann Softmax exploration is implemented. In the traditional Q-learning algorithms, we have to update all states. However, it is not practical for online learning. We add a prototype updating queue module to the architecture, which keeps the most recently visited states (pointed by dash lines).

Only states in that queue are updated at each time instant. Thus, state updating is local and only along the trajectory of visited states. In the following sections, we describe each component of the architecture in details.

4.6.1 Incremental hierarchical discriminant regression

Mathematically, the cognitive mapping is formulated as a mapping M: $S \times X \mapsto X' \times A \times Q$, where S is the state space, X is the current sensory space, X' is the space of primed sensation, A is the primed action space, and Q is the space of value. They are all numeric vector spaces. At every time step, M accepts the current sensory input $x(t)$ and combines it with the current state $s(t)$ and then maps to the primed sensation $x'(t)$ and the corresponding action output $a(t+1)$ and its value. The cognitive mapping is realized by the Incremental Hierarchical Discriminant Regression (IHDR) tree [51] [114]. More details of IHDR can be found in chapter 4.

It is worth noting that in this chapter IHDR is integrated with reinforcement learning. A prototype in IHDR is also a state in reinforcement learning algorithm. In order to avoid confusion, we only use state in the following sections. Thus, given an input vector x, the tree finds the best matched state s. If they are close, x is used to update s. Otherwise it is considered as a new state. Each state s (Fig. 4.3) is associated with a list of primed contexts: $\{c'_1, c'_2, ..., c'_k\}$. Each primed context, c'_i, consists of a primed sensation x'_i, a primed action a_i and a corresponding Q-value. The primed sensation is what the robot predicts to sense after taking the corresponding primed action. The Q-value is the expected value of the corresponding action. Given

the list of primed contexts, it is the value system that determines which primed action should be taken based on its Q-value. Besides primed contexts, a state consists of four kinds of information: age, learning rate, temperature, and standard deviation of the primed sensation. The age of a state is used to determine the learning rate and temperature of Boltzmann Softmax exploration when the state is visited. The standard deviation is used to calculate novelty. More details can be found in the next section.

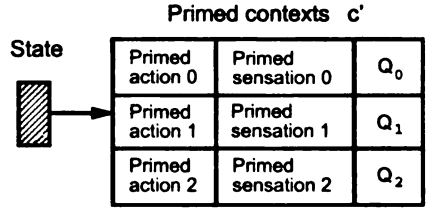


Figure 4.3: State and its primed contexts. Each state is associated with a list of primed contexts. Each primed context consists of a primed action, a primed sensation and a primed Q-value.

4.6.2 The value system

The value system reported here integrates novelty and reinforcement learning.

Novelty

As we know, rewards are as important as value but they are sparse in time. In contrast, novelty of signals is dense in time. It is defined for every time instant. In order to motivate a developmental robot at any time, it is essential to integrate novelty with rewards. As shown in Fig. 4.3, every state retrieved from the IHDR tree is associated with a list of primed contexts (c'), which include: primed sensations

$X' = (x'_{p1}, x'_{p2}, ..., x'_{pn})$, primed actions $A = (a_{p1}, a_{p2}, ..., a_{pn})$ and corresponding Q-values $Q = (q_{p1}, q_{p2}, ..., q_{pn})$. Novelty can be measured by the agreement between what is predicted by the robot and what the robot actually senses. If the robot can predict what will happen, the novelty is low. If the ith action is chosen, we can define novelty as the normalized distance between the ith primed sensation $x'_{pi} = (x'_1, x'_2 ... x'_m)$ at time t and the actual sensation $x(t+1)$ at the next time:

$$n(t) = \sqrt{\frac{1}{m} \sum_{j=1}^{m} \frac{(x'_j(t) - x_j(t+1))^2}{\sigma_j^2(t)}} \tag{4.1}$$

where m is the dimension of sensory input. Each component is divided by the expected deviation σ_j, which is the time-discounted average of the squared difference $(x'_j - x_j)^2$, as shown in Eq. (4.2):

$$\sigma_j^2(t) = \frac{t - 1 - \mu(t)}{t} \sigma_j^2(t-1) + \frac{1 + \mu(t)}{t} (x'_j - x_j)^2 \tag{4.2}$$

where $\mu(t)$ is the amnesic parameter to give more weight to the new samples. With an appropriate a, $\sigma(t)$ would represent the short-term variation of the sensation. When a state is generated for the first time, the initial value of σ_i^2 is copied from its nearest neighbor in the same leaf node. The amnesic parameter is formulated by Eq. (4.3):

$$\mu(t) = \begin{cases} 0 & \text{if } t \leq n_1 \\ c(t - n_1)/(n_2 - n_1) & \text{if } n_1 < t \leq n_2 \\ c + (t - n_2)/m & \text{otherwise} \end{cases} \tag{4.3}$$

where n_1 and n_2 are two switch points, c and m are two constant numbers which determine the shape of a. After the above calculations, $x'_j(t)$ would be set as $x_j(t+1)$, which is the new primed sensation.

Integration of Novelty and Rewards

However, novelty is only a low level measure. The system's preference to a sensory input is typically not just a simple function of $n(t)$. Besides novelty, human trainers and the environment can shape the robot's behaviors through its biased sensors. A biased sensor for a developmental robot is the one whose signal has an innate preference pattern at the birth time. For example, a biased sensor value $b = 1$ if the human teacher presses its "good" button (biased sensory) and $b = -1$ if the human teacher presses its "bad" button (another biased sensor). Furthermore, studies in animal learning show that different reinforcers have different effects. Punishment typically produces a change in behavior much more rapidly than other forms of reinforcers [25]. We integrate novelty and immediate rewards so that the robot can take different factors into account. The combined reward is defined as a weighted sum of physical reinforcers and the novelty:

$$r(t) = w_b r_b(t) + w_g r_g(t) + w_n n(t) \tag{4.4}$$

where $w_b > w_g > w_n$ are three normalized weights of punishment, reward and novelty, respectively, satisfying $w_b + w_g + w_n = 1$.

Q-Learning Algorithm and Boltzmann Softmax Exploration

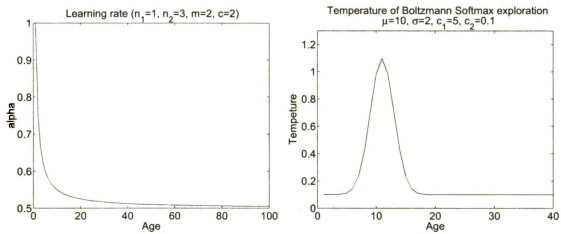
There are two major problems. First, the reward r is not always consistent. Humans may make mistakes in giving rewards, and thus, the relationship between an action and the actual reward is not always certain. The second is the delayed reward problem. The reward due to an action is typically delayed since the effect of an action is not known until some time after the action is complete. These two problems are dealt with by the following Q-learning algorithm. Q-learning is one of the most popular reinforcement learning algorithms [110]. The basic idea is as follows. At each state s, keep a Q-value for every possible primed context c', which includes primed sensation x_p, possible action a_p: $Q(s, c')$, which indicates the value of context c' at current state s. The action with the largest value will be selected as output and then a reward $r(t+1)$ will be received. A modified Q-learning updating expression is as follows:

$$Q(s(t), c'(t)) \longleftarrow (1-\alpha)Q(s(t), c'(t)) + \alpha(r(t+1) + \gamma Q(s(t+1), c'(t+1)) \quad (4.5)$$

where α and γ are two positive numbers between 0 and 1. $\alpha = (1+\mu(n))/t$ is a time varying learning rate based on amnesic average parameter ($\mu(n)$). The parameter γ is for value discount in time. With this algorithm, Q-values are updated according to the immediate reward $r(t+1)$ and the next Q-value, thus, a delayed reward can be back-propagated in time during learning.

An important difference between this design and traditional Q-learning algorithm is that we use changing learning rates based on amnesic average for each state instead a global constant learning rate for the robot. The idea is derived from human

development. In different mature stages, the learning rules of human are different. A single value is not enough to model the case. For example, the first time we meet an unknown person, we would remember him right away (high learning rate). Later, when we meet him in different dresses, we would gradually update his image in our brains with lower learning rate. The formulation of α guarantees that it has a large value at the beginning and converges to a constant smaller value through the robot's experience. Fig. 4.4 (a) shows an example of learning rate based on amnesic average.



(a) (b)

Figure 4.4: (a) Learning rate based on amnesic average (the parameters are shown in the title); (b) Temperature of Boltzmann Softmax exploration based on Gaussian density model.

We applied the Boltzmann Softmax exploration [95] to Q-learning algorithm. At each state (s), the robot has a list of action $A(s) = (a_{p1}, a_{p2}, ..., a_{pn})$ to choose from. The probability for action a to be chosen at s is:

$$p(s,a) = e^{\frac{Q(s,a)}{\tau}} / (\Sigma_{a' \in A(s)} e^{\frac{Q(s,a)}{\tau}}) \tag{4.6}$$

where τ is a positive parameter called temperature. With a high temperature, all actions in $A(s)$ have almost the same probability to be chosen. When $\tau \rightarrow 0$, the Boltzmann Softmax exploration more likely chooses action a that has a high Q-value.

The question is how to determine τ. In [42], only a constant value is set to τ. However, it meets problem when the value system is applied to the real world. As we know, when we sense a novel stimulus at the first time, we would pay attention to it for a while. In this case, a small τ is preferred because the Q-value of action "stare" would be high and the robot should choose this action. If τ is too large, the probability of each action is almost equal, which is not the case in attention. After staring at the novel stimulus for a while, the robot would feel tired and pay attention to other stimuli. Now a larger τ is preferred. After a period of exploration τ should drop again, which means the state is fully explored, the robot can take the action with the highest Q-value. If we choose a large constant τ, then the robot would explore even though it visits a state for the first time. If we choose a small τ, the robot would face the local minimal problem and cannot explore enough states. Fortunately a Gaussian density model (Eq. (4.7)) for local temperature solves this problem.

$$\tau(t) = \frac{c_1}{(2\pi)^{1/2}\sigma}\exp[-\frac{1}{2}(\frac{t-\mu}{\sigma})^2] + c_2 \tag{4.7}$$

where c_1 is a constant to control the maximal value of temperature, c_2 controls the minimal value, t is the age of the state. The plot of the model can be found in Fig. 4.4 (b). Let's talk a little about the convergence of the algorithm if the learning rate is changing. We use $\alpha(t)$ to denote the learning rate at time t. The conditions required

to assure convergence with probability one is:

$$\sum_{t=1}^{\infty} \alpha(t) = \infty \tag{4.8}$$

and

$$\sum_{t=1}^{\infty} \alpha^2(t) < \infty \tag{4.9}$$

The first condition guarantees that the learning rates are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the learning rates become small enough to assure convergence. In our case, the learning rate changes from 0.5 to 1. The second condition is not met, indicating that the estimates cannot completely converge but continue to vary in response to the most recently received rewards. But this property is actually desirable in a nonstationary environment! For a developmental robot that has to explore in unknown environments, the problems it faces are effectively nonstationary. Thus, a developmental robot will not be limited to learn a specific task, but to learn multiple tasks through its experience in the world.

Prototype Updating Queue

In the batch learning mode of a Q-learning algorithm, the back-up is applied to all states. For real-time development, this global iteration method is not applicable, due to the excessive time required. We must use a local method that only involves a small number of computations. That is why we designed the prototype updating queue in Fig. 4.2, which stores the addresses of formerly visited states. Thus, not only is

the Q-value back-propagated, so is the primed sensation. This back-up is performed iteratively from the tail of the queue back to the head of the queue. After the entire queue is updated, the current state's address is pushed into the queue and the oldest state at the head is pushed out of the queue. Because we can limit the length of the queue, real-time updating becomes possible.

Algorithm of Innate Value System

The algorithm of the innate value system works in the following way:

1. Grab the new sensory input $x(t)$.
2. Query the IHDR tree and get a state $s(t)$ and related list of primed contexts.
3. If $x(t)$ is significantly different from $s(t)$, it is considered as a new state and the IHDR tree is updated by saving $x(t)$. Otherwise, $x(t)$ updates $s(t)$ through incremental averaging.
4. Update the age of the state, calculate the temperature of the state with Eq. (4.7).
5. Using the Boltzmann Softmax Exploration in Eq. (4.6) to chose an action based on the Q-value of every primed action. Execute the action.
6. Calculate novelty with Eq. (4.1) and integrate with immediate reward $r(t+1)$ with Eq. (4.4).
7. Update the learning rate of current state based on amnesic average. Update the primed sensation using the actual sensation.

8. Update the Q-value of states in PUQ. Go to step 1.

4.7 Simulation results

In order to test the value system, a simulation environment is developed. The simulator GUI is shown in Fig. 4.5. The big window shows the viewing environment, while

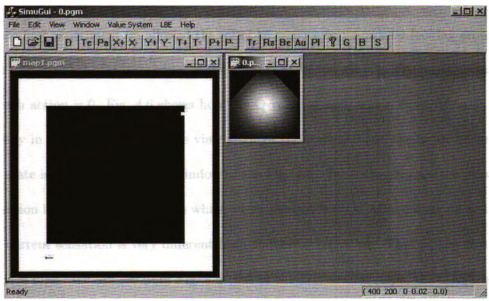


Figure 4.5: The GUI simulator. The arrow indicates the position and the viewing angle of the robot.

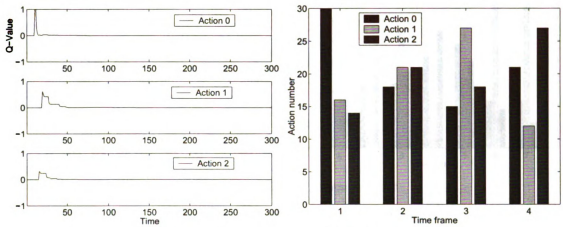
the small window shows the image the robot observes currently. There are several buttons that control the position and viewing angle of the robot. The "Good" and "Bad" buttons are used to issue rewards. In every state, the baby robot has three possible actions: stay at the current viewing angle (action 0), turn neck left 30 degree (action 1) and turn neck right 30 degree (action 2). The representation of a state consists of visual images and absolute viewing angle. The dimension of input image is 100×100. We assume that the robot cannot look backward and the number of ab-

solute viewing angle is 7. The parameters are defined as follows: $\gamma = 0.5$ in Eq. (4.5); $c = 2$, $n_1 = 1$, $n_2 = 3$, $m = 2$ in Eq. (4.3). The value of c_1 in Eq. (4.7) is 5; $c_2 = 0.1$; $\mu = 20$ and $\sigma = 2$.

4.7.1 Habituation effect

In the first experiment we allowed the robot to explore on its own by looking around. The total number of state is equal to the number of view angle (7). The initial Q-value of each action is 0. Fig. 4.6 shows how the Q-value of each action changes based on novelty in one state (the absolute view angle of the state is 0 and the input image of the state is shown in the small window of the Fig. 4.5). In the beginning, the primed sensation is set as a long vector in which every element is zero. After taking an action, the current sensation is very different from the initial sensation. That is, the novelty value is high. We can see from Fig. 4.6 that in the first several steps, the Q-values of each action increase. However, after the primed sensation is updated, it would be the same as the actual sensation if the robot takes the action again. Then the novelty becomes zero and the Q-value decreases. After a long period training (300 steps), the robot can predict the actual sensation of next step whatever action it takes. So the Q-value of each action converges to the same value (0). This means each action has the same probability to be chosen. The right part of Fig. 4.6 shows the number of each action in different time frames (60 steps in each time frame). After 300 steps, the Q-value of each action is nearly equal, so the numbers of each action are close. The experiment shows that because of the habituation effect, the robot loses the interest

of any action after exploration and chooses an action randomly. Because a state is not visited at every time instant, there are flat periods in the Q-value plot, which means that the robot is not at the state and the Q-value does not change.



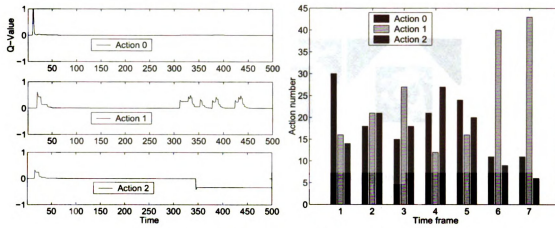
Q-value vs time Action number

Figure 4.6: Habituation effect. On the left part, the 1st, 2nd and 3rd plots correspond to the Q-value of action 0, action 1, and action 2, respectively. On the right part, the frequency of actions in different time frames.

4.7.2 Integration of novelty and positive reward

After the above experiment, we began to issue rewards. For example, when the robot turns left, a human teacher can give it a positive reward (1). For action 2, negative rewards (-1) are issued. Then the actual reward the robot receives is an integration of novelty and an immediate reward. The Q-value of action 0 converges to 0. That of action 2 turns to be negative after a punishment was issued at step 348. The Q-value of action 1 is always positive because we kept issuing positive rewards. As we can see on the left part of Fig. 4.7, after training, the Q-value of action 1 is much larger than that of other actions. As shown in Fig. 4.7 (right), gradually, action 1 was chosen the

most often.



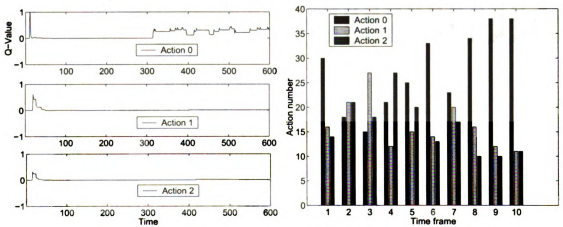
Q-value vs time Action number

Figure 4.7: Integration of novelty and immediate reward.

4.7.3 Increase novelty with a moving object

In order to show novelty preference, a moving toy was added to the simulation environment after the first experiment. The testing images are shown in Fig. 4.8. Now the total number of state is 12, which is 5 more than that of the first two experiments. Every time the robot is in the state with the absolute viewing angle 0, one of these images is generated randomly. Thus, the primed sensation of action 0 could be different from the actual sensation. As shown on the left part of Fig. 4.9, the Q-value of action 0 is positive because of high novelty. In contrast, the Q-values of action 1 and action 2 are close to zero. After training, the robot found that staying in the current state is the most interesting. So action 0 was chosen the most often.

Figure 4.8: Simulation of a moving object.

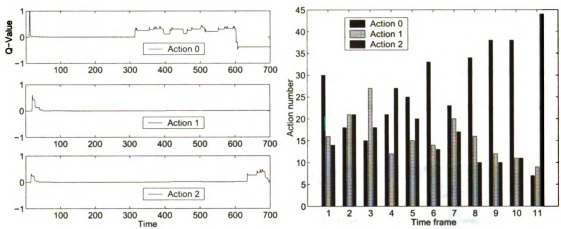


Q-value vs time Action Number

Figure 4.9: Increase novelty with a moving object.

4.7.4 Suppress novelty with punishment

After the third experiment, we issued positive rewards to action 2 (turn right), and negative rewards to action 0. Thus, even though the novelty is high when the robot stares at a moving object, the immediate rewards suppress the novelty. Gradually, the Q-value of action 2 increased. As shown in Fig. 4.10, after training, the robot chose action 2 most of time. However, action 0 and action 1 were still chosen at few times because of Boltzmann Softmax exploration.



Q-value vs time Action number

Figure 4.10: Suppress novelty with immediate rewards.

4.8 Experiments with SAIL robot

The value system is also tested on our SAIL robot (short for Self-organizing Autonomous Incremental Learner) through vision-guided neck action selection. SAIL, shown in Fig.2.9, is a human-size robot custom-made at Michigan State University. It has two "eyes", which are controlled by fast pan-tilt heads. In real-time testing, at

each step SAIL has 3 action choices: turn its neck left, turn its neck right and stay. Totally, there are 7 absolute positions of its neck. Center is position 0, and from left to right is position -3 to 3. Because there is a lot of noise in real-time testing (people come in and come out of the view), we restricted the number of states by applying a Gaussian mask to image input after subtracting the image mean. The total number of states in the real-time experiment is about 50. The dimension of the input image is $30 \times 40 \times 3 \times 2$, where 3 arises from RGB colors and 2 for 2 eyes. The resolution of the image is 30×40. Toys used as moving objects are shown in Fig. 4.11. The input representation consists of visual images and the absolute position of the robot's neck. The two components are normalized so that each has similar weight in the representation. Biased touch sensors are used to issue punishment and reward . We used the same parameters as we did in the simulation.

Figure 4.11: Toys used as moving objects (Adopted from [126]).

4.8.1 Novelty and multiple reinforcers for different actions

In order to show the effect of novelty, we allowed the robot to explore by itself for about 5 minutes (200 steps), then kept moving toys at neck position -1. At each position there could be multiple states because the input images at certain neck positions could change. Fig. 4.12 shows the information of one state at position -1.

The image part of the state is the fourth image of the first row shown in Fig. 4.11, which is the background of the experiment. The first three plots are the Q-value of each action (stay, left, right), the fourth plot is the reward of corresponding action, the fifth plot is novelty value and the last one is the learning rate of the state. After exploration (200 steps later), we moved toys in front of the robot, which increases the novelty and Q-value of action 0 (stay). After training, the robot would prefer toys and keep looking at it from step 230 to step 270. A subset of the image sequence is

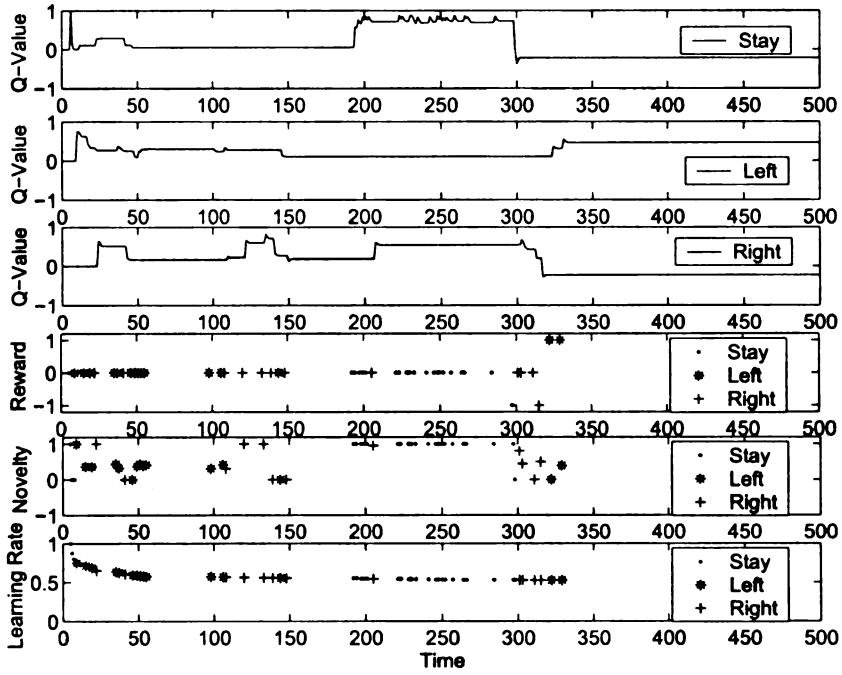


Figure 4.12: The Q-value, reward, novelty and learning rate of each action of one state at position -1 when multiple reinforcers are issued.

shown in Fig. 4.11. If the actual sensations in the second row and the corresponding primed sensation in the third row are very different, the novelty would be high. The novelty value is shown in the fifth plot. The novelty of action 0, 1, 2 is specified by '.', '*', '+', respectively.

After step 300, the trainers began to issued different reinforcers to different actions. Punishments were issued to action 0 at step 297 and step 298 (the fourth plot) and to action 2 at step 315. Rewards were issued to action 1 at step 322 and step 329. The Q-values of action 0 and action 2 became negative while that of action 1 became positive, which means that the visual attention ability of the robot is developed through the interactions with the environment. Even though the novelty of action 0 could be high, the robot preferred action 1 because of its experience. The learning rate in the fifth row shows that at the beginning the robot immediately remembered the new stimuli and then gradually updated the stimuli.

4.8.2 Boltzmann softmax exploration

As we mentioned in section 3, Boltzmann Softmax exploration is applied so that the robot can experience more states. In Fig. 4.13, only information from step 1 to step 60 of the above state is shown. The first plot is the probability of each action based on its Q-value. The total probability is 1. The probabilities of action 0, 1, 2 are plotted at the top, middle and bottom, respectively. The star denotes the random value generated by a uniform distribution. If the random value is in one range, say, the middle range, then action 1 would be taken. Because the robot is not always in the state, the plot is sparse. The second plot shows the temperature based on Gaussian density model of Eq. (4.7). At the beginning, the τ is small and the novelty of the state is high (the initial Q-values of another actions are zero), so the probability of action 'stay' is the largest one (almost 100%). The robot would stare at the stimulus

for while. Then, the temperature increases. The probabilities of each action became similar and the robot began to choose another actions and explore more states. After about 10 times, the temperature dropped to a small value (0.1) again, the action with larger Q-value would have more chance to be taken. Thus, we solved the problem in [42].

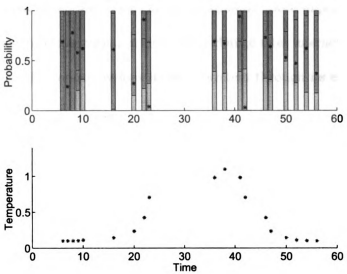


Figure 4.13: Boltzmann Softmax Exploration: The total probability is 1 in the first plot. The probabilities of action 0, 1, 2 are plotted at the top, middle and bottom, respectively. The star denotes the random value received at that time. The second plot shows the temperature.

4.9 Conclusions

In this chapter, a model for value system development of a robot is proposed. Novelty and reinforcement learning are integrated into the value system. Novelty is derived from the high dimensional primed sensation, thus value is not just a scalar number. Instead, it presents the rich information of the world. Furthermore, the primed sensa-

tion based value system can guide the developmental robot all the time since novelty from primed sensation is defined at every sensory refresh cycle while reinforcers are sparse in time. Since no salient features are predefined, the value system is applicable, in principle, to any task and provides guidance to the robot all the time. The working of the value system is shown through vision-based neck action selection. Our SAIL robot learns to pay attention to salient visual stimuli. More important is that the robot's responses to certain visual stimuli would change after interacting with human trainers. That is, the robot develops its value system through its experience in the real world.

Chapter 5

Covert Perceptual Capability Development for Vision Based Navigation

In this chapter, we propose a model to develop robots' covert perceptual capability using reinforcement learning. Covert capabilities such as attention cannot be developed using supervised learning methods. In contrast, reinforcement learning helps robots develop these capabilities via internal but non-enforceable probes. We treat covert perceptual behavior as action selected by a motivational system based on reinforcement learning. We apply this model to vision-based navigation. The goal is to enable a robot to learn the type of road boundary. Instead of deal with simple problem in controlled environments with limited states and low input dimension, We test the model on images captured in non-stationary environments for navigation. In

order to handle the problem of huge state space in the active vision domain, Incremental Hierarchical Discriminant Regression is used to generate states on the fly. Its coarse-to-fine tree structure guarantees real-time retrieval for high-dimensional input. K Nearest-Neighbor strategy is adopted to further reduce training time complexity.

5.1 Introduction

Supervised (action-imposed) learning is an effective learning mode to help a robot develop intelligent capabilities. However, in psychology and neuroscience [74] [30], there are a lot of cognitive capabilities can not be developed using this learning mode. For example, visual attention and thinking are two internal behaviors, which cannot be imposed from outside but develop through interactions with the environments. If the behavior of a robot is not visible from the outside, we call it covert behavior.

In this chapter, we apply the developmental learning paradigm to covert perceptual capability development for a robot. The goal is to teach a robot to learn the type of road boundaries for vision-based navigation. Instead of giving the the correct type explicitly, human teacher issues rewards and punishments according to the robot's guess. In this sense, the robot develops this covert capability via internal but non-enforceable probes. No action (boundary type) is imposed. After this learning stage is mature, the robot can learn the correct heading direction for navigation in the same mode. This paradigm is important and indispensable. First, action-imposed learning is not enough to model sophisticated robotic cognitive development. Second, if the robot learns wrong actions it cannot make a correction with this learning mode. On

the contrary, reinforcement learning is a good model to learn intelligent capabilities when imposed action is not acceptable. Moreover, it gives the robot the ability to recover from error.

There are quite a few studies, which apply reinforcement learning to vision and attention problems. Ballard and Whitehead's study [121] is one of the earliest attempt to use reinforcement learning as a model of active perception. Balkenius and Hulth [5] model attention as selection for action. They train a robot to pick out the correct focus of attention based on reinforcement learning. Experimental results are report on a simple simulator. Bandera [6] used reinforcement learning to solve gaze control problem. However, their application is in an artificial environment. Its performance in the real world is unknown. Minut and Mahadevan [63] proposed a reinforcement learning model of selective visual attention. The goal is to use a fixed pan-tilt-zoom camera to find an object in a cluttered lab environment. That is, the environment is stationary. In summary we can find out that most former studies, which apply reinforcement learning to model covert perceptual behaviors, test on controlled environments. The input dimension of state vector is low and the number of state is very limited. In order to reach the goal of real-time active vision development in non-stationary environments (outdoor navigation), we propose IHDR (Incremental Hierarchical Discriminant Regression) to generate continuous state space on the fly for high dimensional visual input. And its coarse-to-fine structure is able to speed up the state retrieval procedure. Furthermore, we implement K Nearest Neighbor algorithm for Q-learning so that at each time instant multiple similar states can be updated, which dramatically reduces the number of rewards human teachers have

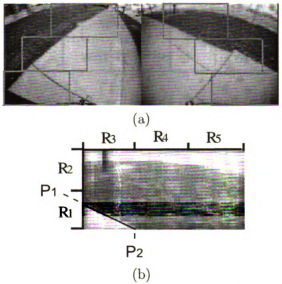
to issue. This implementation shares the same idea with [52]. Instead of using weighted-nearest neighbor, we use top-K nearest neighbor directly. We also propose a multi-layer reinforcement learning architecture [97]. The first level learning modules handle covert perceptual capability development while the second level learning module deal with navigation behaviors.

In what follows, we first describe the covert capability development problem in outdoor navigation. The detailed architecture of the developmental paradigm is presented in Section 3. The experimental results are reported in Section 4. Finally, we draw our conclusions and discuss about the future work.

5.2 Problem Description

Our goal is to enable a robot to develop its covert perceptual capability for vision-based outdoor navigation. A good example is shown in Fig. 5.1 (a). Suppose the robot has two eyes (left image and right image). Instead of using the entire image as input to each first-level learning module, we divide two input images into 6 sub-windows (specified by rectangles). Suppose the size of the original image is 160×120. The dimension of the state vector is 19200, which is almost intractable. With six windows, the dimension of state vector is reduced to $80 \times 40 = 3200$. In this way, we reduce the dimension can control the state space. And it turns out these six windows can cover most of road boundaries. Our goal is to teach the robot to learn the type of road boundary. Let's look at the top window on the right (Fig. 5.1 (b)). There is a road edge, which intersects with the window at two points (p_1, p_2).

For each intersecting point, we define five boundary types (ranges). Given p_1 as an example,there 5 possible ranges $(R_1, R_2, R_3, R_4, R_5)$ and p_1 fits in $R1$. Suppose the image of the window is $x(t)$ (a long vector and each element corresponding to a pixel). The robot needs to learning the mapping from $x(t)$ to correct road boundary type $a = \{R_1, R_2, R_3, R_4, R_5\}$. Since each window has two intersecting points, we can generate a vector $A = (a_1, a_2, ..., a_{12})$. This vector can be the input for the second level module for heading direction learning.



(a)

(b)

Figure 5.1: (a) A pair of images in vision based outdoor navigation with 6 attention windows; (b) Sub-image of one window and the type of road boundary (p_1's type is R_1).

A major part of the developmental learning paradigm is Hierarchical Discriminant Regression [51]. Using HDR tree, the robot learns the type of road boundary in each window $(A = (a_1, a_2, ..., a_{12}))$. The overall system operation architecture for robotic cognitive development is shown in Fig. 5.2. There are HDR trees at two levels. At the first level, 6 HDR trees learns the mapping from visual input to road boundary type. The second-level HDR tree maps road boundary type of all 6 windows to the correct

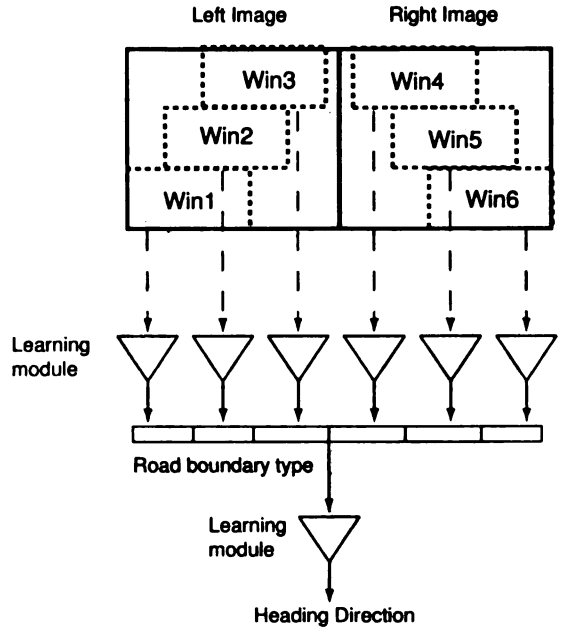


Figure 5.2: The system operation architecture for robotic cognitive development. The first-level learning modules generate the mapping from visual input to road boundary type while the second-level learning module generates the mapping from road boundary type vector to heading direction for navigation.

heading direction. It is worth noting that in this study we only conduct experiments for the first-level learning modules. How the robot learns each mapping is discussed in next section.

5.3 System Architecture of Robotic Cognitive Development

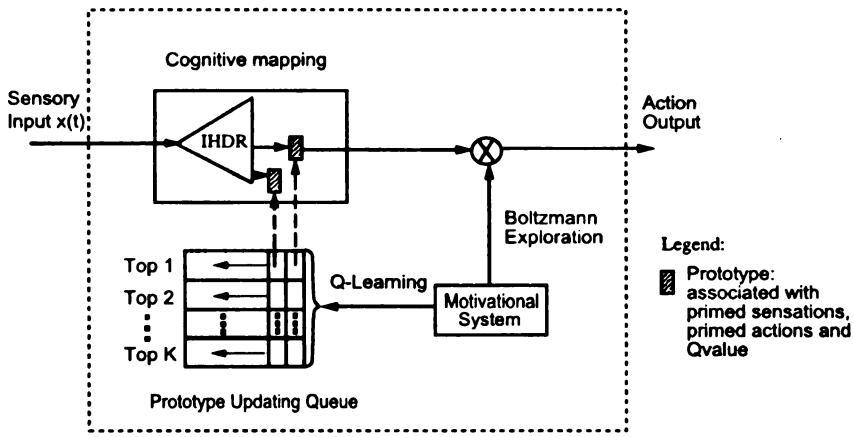


Figure 5.3: The system architecture of each learning module for robotic cognitive development.

The basic architecture for robotic cognitive development is shown in Fig. 5.3. The sensory input $x(t)$ is represented by a high dimensional vector. At the first level the input is visual image while at the second level the input is road boundary type vector. In this chapter we only discuss the first-level learning modules. $x(t)$ is fed into the cognitive mapping module. The cognitive mapping is realized by Incremental Hierarchical Discriminant Regression (IHDR) [51]. IHDR clusters $x(t)$ into current state $s(t)$ and then maps the current state to the corresponding action output. Thus, IHDR generates state for high-dimensional visual input on the fly, which is a very

difficult problem. That's why most of studies only deal with low-dimensional state space. In contrast, IHDR makes it possible to handle high-dimensional state space. In the testing phase, given a state $s(t)$, the IHDR finds the best matched s' associated with a list of primed contexts ($c' = (x', a', q)$), which include: primed sensations $X' = (x'_1, x'_2, ..., x'_n)$, primed actions $A' = (a'_1, a'_2, ..., a'_n)$ and corresponding Q-values $Q = (q_1, q_2, ..., q_n)$, where n is the number of different actions. In other words, the function of IHDR is g: $S \mapsto X' \times A' \times Q$. Primed actions are the possible actions in each state. The probability to take each primed action is based on its Q-value.

The motivational system of the developmental learning paradigm works as an action selection function $v : 2^{A'} \mapsto A$ ($2^{A'}$ denotes all the possible subsets of A'), which chooses an action from a list of primed actions. At the beginning the robot can choose random actions. Human teachers issue reward and punishment based on its actions. Through this reinforcement learning procedure the robot develops the capability to learn road type without imposed actions. In order to let the robot fully explore the action space, Boltzmann Softmax exploration is implemented. To reach the requirement of real-time updating in developmental learning, we add a prototype updating queue module to the architecture, which keeps the most recently visited states (pointed by dash lines). Only states in that queue are updated at each time instant. However, in non-stationary environments (for instance, outdoor navigation), the state space is huge. If we use the standard Q-learning algorithm, the training time is tremendous. In order to reduce the time of training, we adopt top-k nearest neighbor strategy. For each state, its top K-nearest neighbors are saved in the prototype updating queue (Fig. 5.3). If a reward is issued, the system not only

updates the current state-action pair but also updates the K-NN state-pairs, which dramatically reduces time complexity of the training procedure. Please refer to last chapter to check the description of IHDR, Q-learning and Boltzmann exploration. The major difference is that we implement a new prototype updating queue.

5.3.1 K Nearest Neighbor Prototype Updating Queue

Even though, IHDR makes learning in non-stationary environments possible, the time to train the system is still not acceptable. The major reason is: as the robot explores in new environments, the state space would continue grow. In order to reduce the training time, we adopt the K-nearest neighbor strategy [31]. Suppose the state space is S and the current state is $s(t)$. Let's define D is the distance between $s(t)$ and the kth -nearest neighbor of $s(t)$. The volume of the set of all states whose distance from $s(t)$ is less than D is defined as follows:

$$A(k, S, s(t)) = \frac{2D^n \pi^{n/2}}{n\Gamma(n/2)}. \tag{5.1}$$

Using K-NN updating rule, the updated space goes from one point $s(t)$ to A, which is a tremendous improvement.

We can also look into the training complexity issue. Suppose for each state we need to issue m rewards to get a reliable training result (Q converges.) and the number of states is N, the time complexity of training is mN. If top K updating is applied and we assume each state has the same possibility to be chosen as a match, the time complexity is $N + \frac{(m-1)N}{K}$, where the first N means that each state has to

be visited at least once while $\frac{(m-1)N}{K}$ is the number of training needed using top-K updating. The ratio of time complexity using top-1 and top-K is:

$$\frac{mN}{N + \frac{(m-1)N}{K}} = \frac{mk}{k + m - 1} \tag{5.2}$$

Even though the assumption of the equal probability of each state to be chosen as top match is too strong, we still can see the potential improvement of top-K updating.

5.3.2 Algorithm of Covert Capability Development

The algorithm of the developmental learning paradigm works in the following way:

1. Grab the new sensory input $x(t)$ and feed into IHDR tree. The IHDR tree generates a state $s(t)$ for $x(t)$.

2. Query the IHDR tree and get a matched state s' and related list of primed contexts.

3. If $s(t)$ is significantly different from s', it is considered as a new state and the IHDR tree is updated by saving $s(t)$. Otherwise, use $s(t)$ to update s' through incremental averaging.

4. Using the Boltzmann Softmax Exploration in Eq. 4.6 to chose an action based on the Q-value of every primed action. Execute the action.

5. Receive a reward

6. Update the Q-value of states in PUQ using K-NN updating rule. Go to step 1.

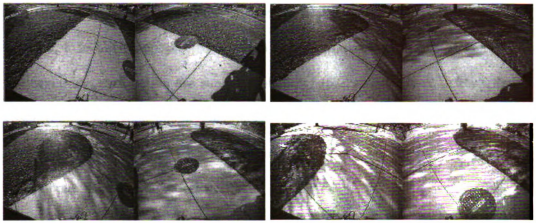


Figure 5.4: Input sample pairs for navigation test.

5.4 Experimental Results

We train the robot to learn edge boundary type for vision-based navigation. A sequence of image pairs is shown in Fig. 5.4. The dimension of each image is 160×120. The dimension of sub-image in each window is $80 \times 40 = 3200$. We compare the performance of original PUQ (top 1) and that of KNN PUQ.

5.4.1 Experiments Using Top-1 Prototype Updating Queue

Here we just show how we teach the robot to learn the road boundary type for one image (state). Totally there are 5 possible actions ($a = \{R_i | i = 0...4\}$). R_1 is the correct action. The Q-value plot of each action is shown in Fig. 5.5. Only information from step 1 to step 30 is shown. As you can see, after training the Q-value of a_1 is the largest (positive) while the Q-values of other actions converge to negative values. From Fig 5.5, we can find out that it takes about 17 visits to converge.

In Fig. 5.6, only information from step 1 to step 40 is shown. The probability

of each action based on its Q-value. The total probability is 1. The probabilities of action 0, 1, 2, 3 and 4 are plotted from top to bottom, respectively. As we can see, at the beginning, each action has similar probability (0.2) to be chosen. After training (issue reward and punishment), action 1 is chosen for most of the time. After step 18, the probability of action 1 is larger than 0.99. That is, for this attention point it takes about 18 visits to converge.

5.4.2 Experiment Using KNN Prototype Updating Queue

In this experiment, we trained 50 consecutive images repeatedly for 20 times. Top 5 nearest-neighbors updating are used.

Q value of each action

We should notice that using top-K updating mechanism, the number of updating (n_u) is different from the number of visiting (n_v). n_v means the number of times that state s is retrieved as a top 1 match while n_u means the number of times that state s is retrieved as one of the top K match. The Q-value of each action versus the number of updating is shown in Fig. 5.7. The correct action should be action 1. After training the Q-value of action 1 is the largest (positive). "+" means that the state is chosen as top 1 match. Because of the trainer's mistake , at the beginning, action 0 got several positive rewards (Q value increases) while action 1 got a couple of negative rewards (Q value decreases). However, after 30 times updating, action 1 has the largest value and the system automatically chooses it as output. The plot also verifies that KNN-based reinforcement learning can tolerate human mistakes.

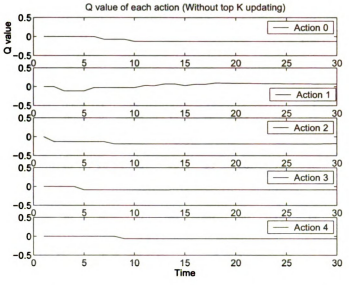


Figure 5.5: The Q-value of each action in one state using top-1 PUQ.

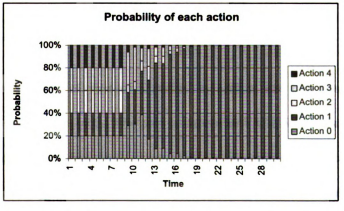


Figure 5.6: Boltzmann Softmax Exploration: The total probability is 1 in the first plot. The probabilities of action 0, 1, 2, 3, 4 are plotted from top to bottom, respectively.

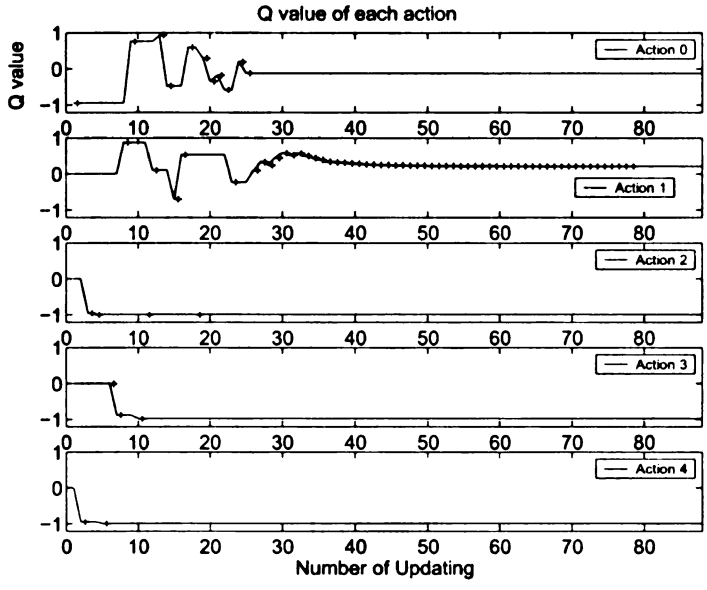


Figure 5.7: The Q-value of each action.

Boltzmann exploration

We use Boltzmann exploration to choose the action for each state. How an action is chosen is based on temperature θ. In this chapter, θ is defined as a revised Sigmoid function:

$$\theta = c_1(1 - \frac{1}{1 + e^{-(n_v - c_2)}}), \tag{5.3}$$

where n_v is the number of visit of a state. c_1 and c_2 are two constants. In our experiment, $c_1 = 10$ and $c_2 = 3$. The plot is shown in Fig. 5.8. At the beginning, θ is about 10 and each action has similar probability to be chosen. If $n_v > 8$, θ converges to 0. Greedy strategy is applied in this case. We use Boltzmann exploration to choose the action for each state. In Fig. 5.9, only information from step 1 to step 15 is shown. The first plot is the probability of each action based on its Q-value. The total probability is 1. The probabilities of action 0, 1, 2, 3 and 4 are plotted from top

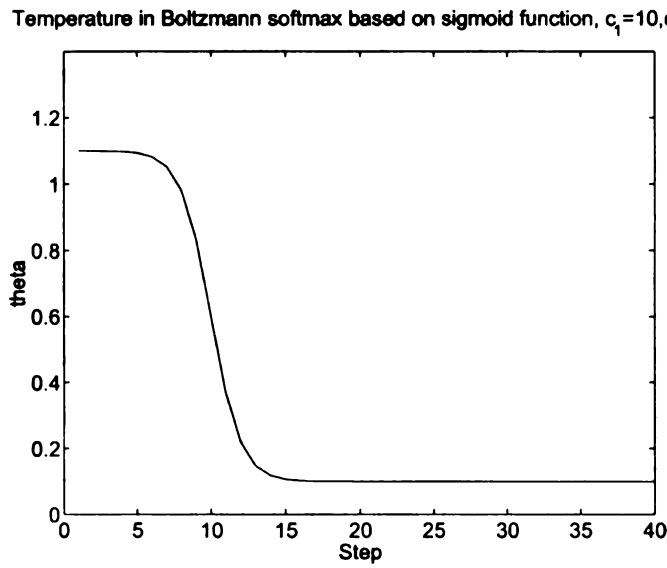


Figure 5.8: Temperature of Boltzmann exploration.

to bottom, respectively. The '+' denotes the random value generated by a uniform distribution. If the random value is in one range, say, the bottom range, then action 0 would be taken. As we can see, at the beginning, each action has similar probability (about 0.2) to be chosen. After training, action 1 is chosen for most of the time. The action sequence is show in the second plot. When $n_v > 4$, the system only chose action 1. That is, for each attention point it takes about 5 visits to converge. This again shows the advantage of KNN-based reinforcement learning.

Retrieval Time

The retrieval time of each image is shown in Fig. 5.10. The average training time of each step is 0.0186s and the highest training time is 0.12s. Even though we have 12 trees for 6 windows, the system can still work in real time. This shows how efficient IHDR is for robot mental development.

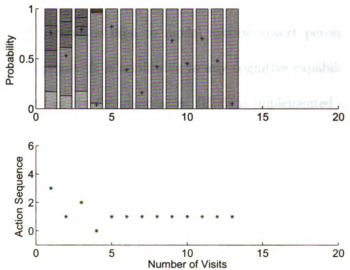


Figure 5.9: Boltzmann Softmax Exploration: The total probability is 1 in the first plot. The probabilities of action 0, 1, 2, 3, 4 are plotted from top to bottom, respectively. The "+" denotes the random value received at that time. The second plot shows the action sequence.

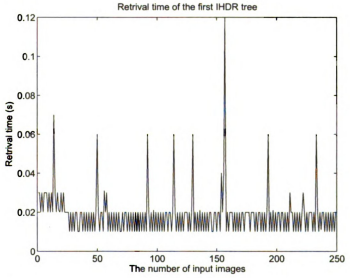


Figure 5.10: Retrieval time of each image for one tree.

5.5 Conclusions and Future Work

In this chapter, Q learning is used to model robotic covert perceptual capability development in navigation. It is applicable to any cognitive capability development when enforceable probes are not acceptable. IHDR is implemented so that learning in real-time becomes possible. And we adopt K Nearest Neighbor strategy, which dramatically reduces training time complexity in non-stationary environments. The experimental results show the effectiveness of the model, which enables a robot to learn road boundary type through interactions with teachers. We will extend the model to develop the robot's vision-based navigation capability in the future.

Chapter 6

Cross-Task Learning

Cross-task learning is essential for an artificial agent to explore in the real world. With this capability, an agent can learn multiple tasks and use acquired knowledge to learn new tasks. In this paper, the DOSASE MDP model, enables an agent to find the relatedness between different tasks and to develop cross-task learning capability by interacting with the environment. One challenge of this work is that no task is defined in advance. The trainer shapes the behavior of the agent to learn different tasks interactively and continuously through interactions. The development is conducted in real-time using high-dimensional input, which is another challenging issue. We tested the architecture on an artificial agent for vision-based navigation. The results show that DOSASE is an effective model for autonomous mental development (AMD) and dramatically reduces both time and space complexities for cross-task learning.

6.1 Introduction

Cross-task learning capability is very important for an artificial agent to explore in the real world. By cross-task, we mean that the same system must learn multiple tasks incrementally in the same mode, dealing with task specific contexts correctly. With this capability, the agent can learn different tasks and transfer learned knowledge to new tasks. Scientists in psychology proposed a lot of models to simulate human cognition in cross-task learning. Lovett [60] used an architecture called ACT-R to model how people organize knowledge and produce intelligent behaviors.They claimed that after separately fitting performance on a preliminary task, the model can make zero-parameter cross-task prediction of performance in on a second task. Computer scientists are also interested in building multiple-task learning models for artificial agents. Thrun [100] proposed the task-clustering algorithm, which learns the relationship between different tasks. When facing new a task, the algorithm first finds the most related task, then exploits information from this task. Pratt [75] and Caruana [17] investigated the similar problem: how to use information from one neural network to help a second network learn a related task. The limitations of these methods consist of: 1) human programmers know the task knowledge at the beginning. 2) in order to build up the relationship between different tasks, real-time learning is impossible. For a typical developmental agent, these limitations are not acceptable. A developmental agent has to run in real time and face different tasks, which are unknown to the agent before training.

In this paper, we applied the DOSASE MDP model to conducting cross-task learn-

ing. The advantages of the model include: 1) no task is defined at the beginning. By interacting with the trainer, the agent incrementally learns different tasks. 2) the agent can learn task-relatedness online in real-time. 3) the agent transfers learned knowledge to new tasks and dramatically reduces both time and space complexities. The model has been successfully tested on an artificial agent for vision-based navigation.

In the following section, we will first formulate what is cross-task learning. The detailed DOSASE MDP model is introduced in section 3. Experimental results are shown in section 4 and then we conclude with a summary and discussion about future works.

6.2 Problem Description

To be precise in our further discussion, we need mathematical notations.

Definition 6.2.1 *Given an agent at time t_1, suppose that the agent produces different action contexts $\mathbf{a}_1$ and $\mathbf{a}_2$, from two different contexts $C_1 = \{\mathbf{c}(t) \mid t_1 \leq t \leq t_2\}$ and $C_2 = \{\mathbf{c}(t) \mid t_1 \leq t \leq t_3\}$, respectively. If $\mathbf{a}_1$ and $\mathbf{a}_2$ are considered different by a social group (human or robot), conditioned on C_1 and C_2, then we say that the agent* discriminates *two contexts C_1 and C_2 in the society. Otherwise, we say that the agent does not* discriminate *C_1 and C_2 in the society.*

For example, given a voice command "Go to the elevator," the speech signals of the command are different from different people. However, humans consider the

commands are the same. In this case, the agent should not discriminate the above commands from different people.

Definition 6.2.2 *Cross-task Learning: 1). There are N tasks:* $\Gamma = \{T_i | i = 1, 2, ..N\}$. *2). For each task, the learning goal is to generate the mapping from context to action:* $M : C \rightarrow A$, *where A is the action space, C is the context space, which could consists of different sensory input: vision, audition, touch etc. 3). Given the first N tasks, the agent uses acquired knowledge to speed up learning the N+1th task.*

A typical setting is shown in Fig. 6.1. There are two tasks: T_1 and T_2. For example, in autonomous navigation problem, T_1 is "go around" and T_2 is "go to the elevator." The contexts of each task are C_1 and C_2, respectively.

Definition 6.2.3 *Shared context: The overlapped trajectory between these two tasks is called shared context* $C_{share} = C_1 \bigcap C_2$. *Non-overlapped trajectory is called non-shared context* $C_{non\_share} = C_1 \bigcup C_2 - C_{share}$.

Give the above figure as an example, $C_1 = \{C_{11}, C_{12}, C_{13}, C_{14}, C_{15}\}$ and $C_2 = \{C_{21}, C_{22}, C_{23}, C_{24}, C_{25}\}$. In the task space, the shared context is $C_{share} = \{C_{12}, C_{14}\}$ and the non-shared context is $C_{non\_share} = \{C_{11}, C_{13}, C_{15}, C_{21}, C_{23}, C_{25}\}$.

Definition 6.2.4 *Merge point and break point: The point, where the contexts of two or more tasks begin to overlap, is call "merge point"* (M_1, M_2). *The point, where the contexts of two or more tasks begin to diverge is "break point"* (B_1, B_2).

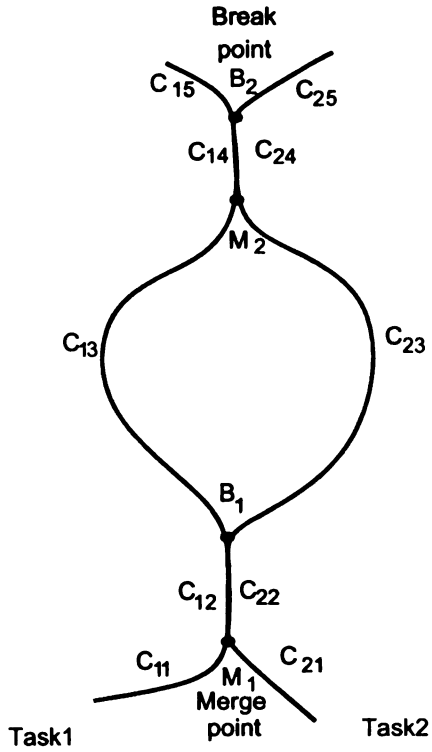


Figure 6.1: Typical setting of cross-task learning.

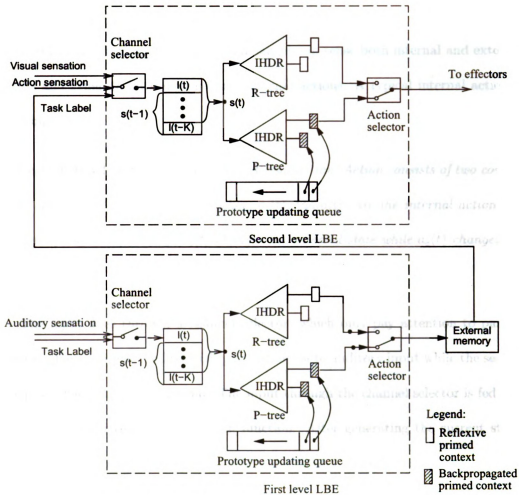


Figure 6.2: The cross-task learning system architecture.

6.3 Architecture of DOSASE MDP

We apply the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model to conduct cross-task learning. The detailed architecture of this model for cross-task learning is shown in Fig. 6.2. There are two level-building element (LBE) components. The first LBE online learns the association between task label and verbal commands. The second LBE learn the specific tasks

(for example, different vision-based navigation tasks). A developmental agent is self-aware and self-effecting, which means the agent can sense both internal and external inputs and generate both internal and external actions. A typical internal action is thinking.

Definition 6.3.1 *Internal action and external action: Action consists of two component. That is* $a(t) = (a_i(t), a_e(t))$*, where* $a_i(t)$ *and* $a_e(t)$ *are the internal action and external action, respectively.* $a_i(t)$ *changes the internal state while* $a_e(t)$ *changes the external state.*

The input goes through a channel selector, which only pay attention to part of context. For example, the first LBE pays attention to auditory input while the second one pays attention to visual input. The input through the channel selector is fed into an observation-driven state transition function. After generating the current state, the cognitive mapping engine finds the best match and outputs the associated action. It is worth noting that in the experiment only the second LBE is used for vision based navigation. The experimental result of voice command learning is not presented.

6.3.1 Observation-driven state transition function

After going through channel selector, the sensory input is fed into the observation-driven state transition component. Let's first define last context: $l(t) = f(a(t-1), x(t))$, which consists of last action $a(t-1)$ and current input $x(t)$. $l(t)$ is pushed into a context queue as shown in Fig. 6.3 and is combined with last state information to generate the current state.

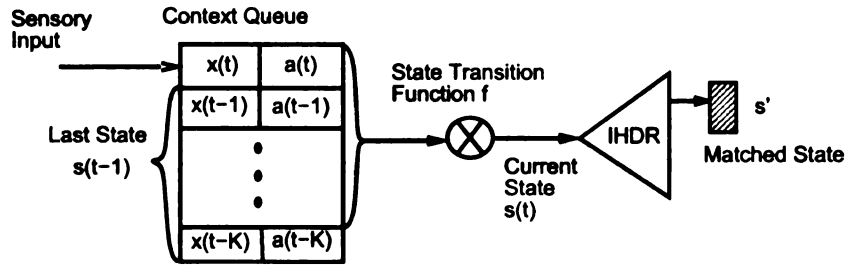


Figure 6.3: Observation-driven state transition function.

6.3.2 Online learning of one task through cognitive mapping

Many problems like content-based retrieval, vision-based navigation can be formulated as a complicated function which maps high dimensional input and the current state to low-dimensional output signals. We use a decision tree to approximate this function, which is implemented by Locally Balanced Incremental Hierarchical Discriminating Regression (LBIHDR) [43] [46].

A detailed explanation is beyond scope. Basically, given a state s, the IHDR finds the best matched s' associated with action. The mapping is done through a coarse-to-fine tree structure. In the testing phase, given s, we use K-nearest neighbor rule to find the best match in the leaf node.

$$s' = \arg \min_{1 \le i \le K} \| s'_i - s \| \tag{6.1}$$

Using IHDR, the relatedness of different task is measured by the similarity between the states generated for different tasks. If some of the states generated by two or more tasks are the same, then we say these tasks are related. That's why our architecture is better than the task-clustering method [100], which has to define different tasks in

advance to measure the relatedness. Using the DOSASE MDP model, we don't need to know the task in advance but generating states for different tasks incrementally. IHDR intrinsically has incremental online learning capability to adapt to new inputs, which is a requirement for autonomous mental development. And because of its efficiency, it can learn high dimensional input, which outperforms the typical neural network algorithm [17].

6.3.3 One-task learning algorithm by a developmental agent

The algorithm to learn one task by a developmental agent is as follows:

Procedure 5 One task learning. Learn the association between context and action.

1: Collect the current context $c(t)$.

2: Go through channel selector to get sensory input $x(t) = c_v(t)$. (For vision-based navigation task.)

3: Push $x(t)$ to the context queue and generate last context $l(t)$.

4: Use observation-driven state transition function (Eq.(1)) to generate the current state $s(t)$.

5: Find the best match s' of $s(t)$. If they are similar, use amnesic average to update s'.

6: If imposed action is given, take this action. Otherwise, choose the action (a) associated with s'.

7: Go back to step 1.

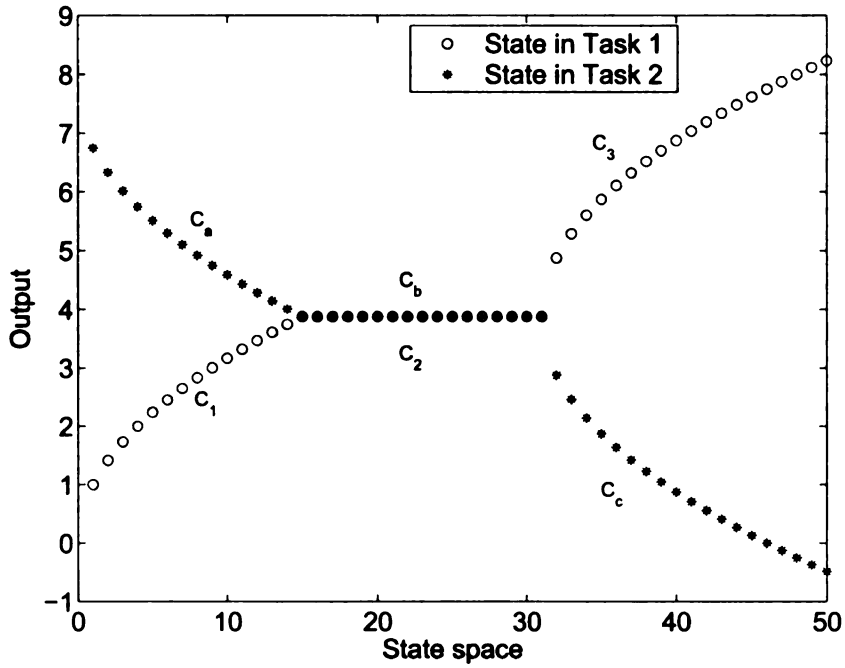


Figure 6.4: State space for 2 tasks.

6.3.4 Multiple task learning through DOSASE

How can this architecture learn multiples tasks? We know that the agent can learn one task with the model. Using the same architecture multiple tasks can be learned if the C_i and A_i are provided. Now, the problem is how to speed up learning if the agent acquires useful knowledge from former tasks? Given the tasks in Fig.6.1 as an example, if T_1("go around") has been learned, how the agent learns T_2 ("go to the elevator")? Let's take a look at the state space generated for each task.

A part of the state generated by these two tasks is shown in Fig. 6.4. The context of T_1 is $C_1 = \{c_1, c_2, c_3\}$. The state generated for T_1 is $S_1 = \{s_1, s_2, s_3\}$, where $s_j = \{s_{j1}, s_{j2}, ..., s_{jn_j}\}$. $j = \{1, 2, 3\}$ while n_j denotes the number of state generated for jth context. Now, the agent face T_2. The context is $C_a = \{c_a, c_b, c_c\}$. The state

generated for T_2 is $S_2 = \{s_a, s_a, s_c\}$, where $s_m = \{s_{m1}, s_{m2}, ..., s_{mn_m}\}$. $m = \{a, b, c\}$ while n_m denotes the number of state generated for mth context. The shared context of T_1 and T_2 is c_2 (c_b). The advantage of cross-task learning is that if the agent learns C_{share} for T_1, then there is no need to experience it again for T_2.

Lemma 6.3.1 *If the number of states generated for non-shared context of T_1 and T_2 is N_{Nshare} and the number of states generated for shared context is N_{share}: the ratio of the space complexity saved for multiple task learning is: $N_{share}/(N_{Nshare} + N_{share})$*

Now let's think about the complexity in the training time space.

Lemma 6.3.2 *If the $N+1$th task shares context with the former N tasks: $C_{share} = (\bigcup_{i=1}^{N} C_n) \bigcap C_{N+1}$, the saved time for training the $N+1$th task is L_{share}.*

6.3.5 Attention mechanism for the break point set

Suppose the agent has been trained with C_1 and C_{share}, in the testing phase, the agent would go through the same context trajectory. However, there is a problem if the agent experience s_{bn_b} (the last state of context c_b) again. Since for different tasks the external actions a_e associated with s_{bn_b} are different. For T_1, the external action should make the state move from s_{2n_2} to s_{31} while for T_2, the external action should make the state move from s_{bn_b} (equal to s_{2n_2}) to s_{c1}. What should the agent do at the "break point?"

An attention mechanism is necessary to solve the problem. Remember that the agent learn the task label through the first LBE. If the agent get into the state s_{2n_2} (e.g. s_{bn_b}), there are two external actions a_{e1}, a_{e2}. Each external action is associated

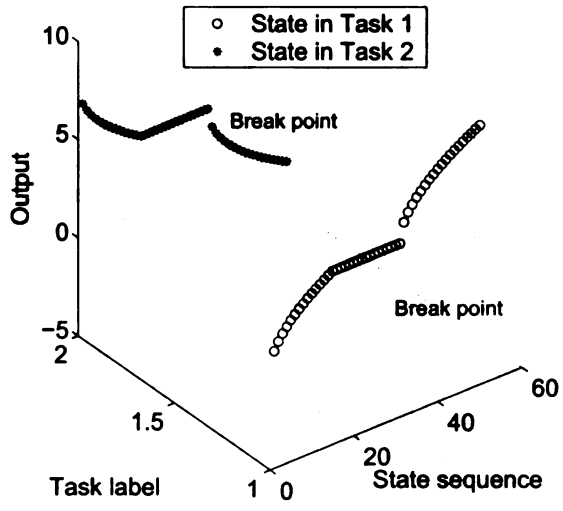


Figure 6.5: Discriminate state transition using multi-modal information.

with a task label. Using the task label information, the agent knows what action should be taken to reach next state. Fig. 6.5 shows how the task label information helps discriminate state transition. Adding task label as another dimension, the break point is split into two points. By checking current task label, the agent knows which action to. The internal action generated by the attention mechanism is defined as:

$$a_i = \begin{cases} 1 \text{ if } T_i = 1 \\ 0 \text{ otherwise} \end{cases} \tag{6.2}$$

The output external action is $a = a_e \bigotimes a_i$, where $\bigotimes$ is component-wise multiplication. Only after checking the task label information, can an action be issued.

6.3.6 Cross-tasking learning algorithm

The algorithm is as follows:

Procedure 6 Cross-task learning.

1: Collect the current context $c(t)$. $x(t) = c(t)$.

2: Push $x(t)$ to the context queue and generate last context $l(t)$.

3: Use observation-driven state transition function to generate the current state $s(t)$.

4: Find the best match s' of $s(t)$. If they are similar, use amnesic average to update s'. Check the task label. If these two states have different task labels, generate a new action for s'.

5: If imposed action is given, take this action. Otherwise, check task label, then use Eq. 6.2 to pay attention to task label and take the corresponding action.

6: Go back step 1.

6.4 Experimental Results

In order to test the cross-task learning capability of the DOSASE MDP model , a simulation environment is developed. The simulation map is shown in Fig. 6.6. The artificial agent (rectangle with an arrow) can navigate through the white area. There is an elevator on the bottom right (white rectangle). In every state, the agent has three possible actions: go straight, turn left and turn right. Some of the example input images are shown in Fig 6.7. The second and the third images on the third row are the images near the elevator. The dimension of input image is 25×25.

Figure 6.6: Simulation interface.

6.4.1 Trajectory and transferred knowledge

Only two tasks are considered: T_1 "go around" and T_2 "go the elevator." The contexts of these tasks are $C1$ and $C2$. The non-shared context is C'. The trajectories of two tasks are shown in Fig. 6.8. In the training stage, $C1$ (solid line) is trained and then C' (dot line from point 'A' to point 'B') is trained. C' includes context near the elevator. In the testing phase, the agent successfully finished these two navigation tasks. For task 2, its trajectory is different from that of task 1 around the "elevator" since this part is the non-shared. While another part of its trajectory is overlapped with that of task 1 because another part is not trained for task 2 but transferred from the knowledge of task 1.

Fig. 6.9 shows the task label of each sample when testing on task 2. The task label in the testing phase is switching between 2 and 1. Tab. 6.1 shows that there

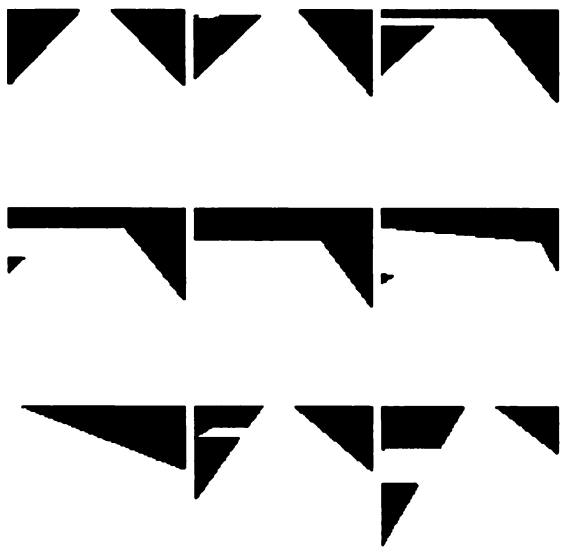


Figure 6.7: A subset of input images.

are totally 5380 samples. 2442 of them are retrieved with task label 2 while 2938 of them are retrieved with task label 1, which means that for this test about 54.61% knowledge is transferred from task 1 to task 2.

Table 6.1: Trasfered knowledge when testing on task 2.

Task	1	2	All	Transferred knowledge
No. of samples	2938	2442	5380	54.61%

6.4.2 Complexity reduced in terms of space and time

The structures of IHDR trees for different tasks are shown in Fig. 6.10. The first two plots correspond to the IHDR tree architecture if task1 and task 2 are trained separately. The plots shows the number of states in different layers. The depth of both trees is 6. In the forth layer, the tree saves most of the states. The third plot

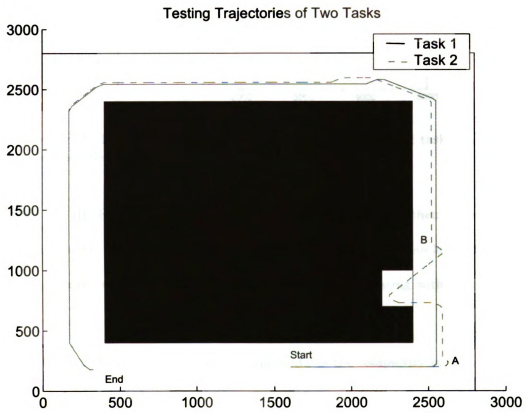


Figure 6.8: Trajectory of two tasks.

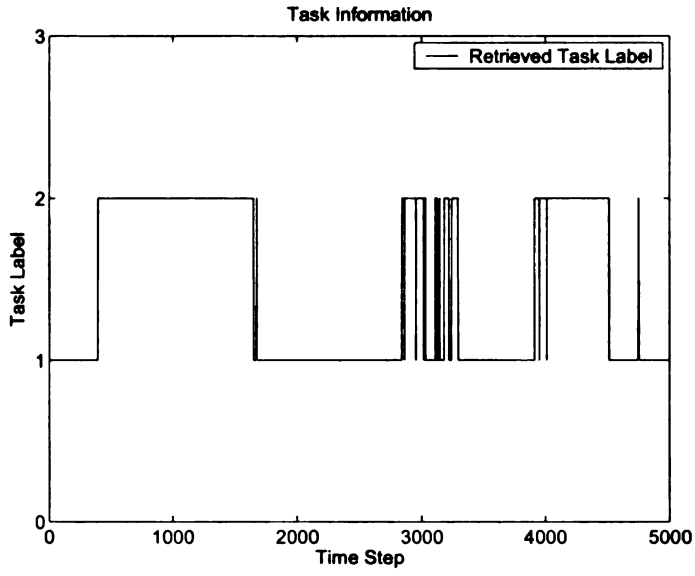


Figure 6.9: Task label information. The plot shows the retrieved task label of each sample when testing on task 2.

shows the IHDR tree structure if the agent learns $C1$ first and then learns C'. As you can see, the depth of tree is still 6, which means that the system learns two tasks but the space complexity does not increase too much comparing with learning only one task.

Table. 6.2 shows how cross-task learning reduces the complexity in terms of both space and time complexities. If we train 2 tasks separately, the size of the IHDR trees are 22.6M and 27.5M, respectively. If we train task 1 ($C1$), and then train the non-shared context (C') of $C1$ and $C2$, the size of the tree is 36.6M. The percentage of saved space is $\frac{22.6+27.5-36.6}{22.6+27.5} = 26.95\%$. The training time of task 1 and 2 is 147.42s and 143.75s, respectively. If cross-task learning is conducted, the training time is 205.83s. The percentage of saved time is about 29.31%. The experimental result shows that the model is effective in principle and cross-task learning really reduces a

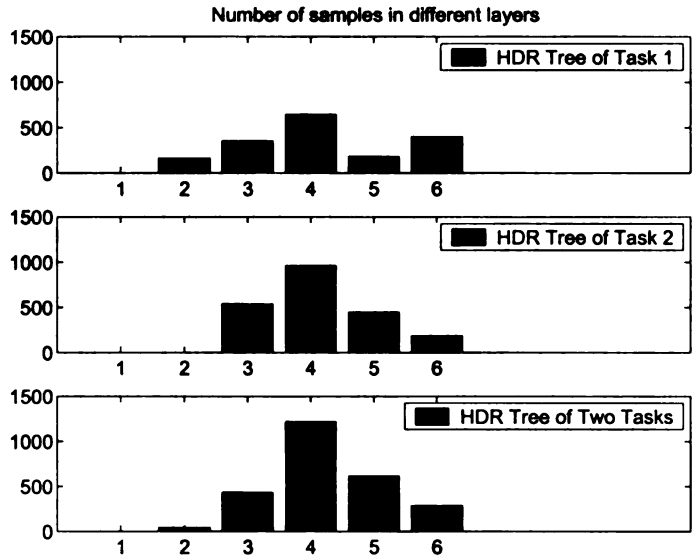


Figure 6.10: HDR tree structure. The first two plots show the number of states in different layers for task 1 and task 2, respectively. The third plot shows the tree structure if cross-task learning is conducted.

lot in terms of time and space complexities.

Table 6.2: Complexity reduced in terms of space and time.

Task	1	2	All	Complexity reduced
Tree size	22.6M	27.5M	36.6M	26.95%
Training time	147.42s	143.75s	205.83	29.31%

The training time of each step for task 1 is shown in Fig. 6.11. The average training time of each step is 0.043s and the highest training time is 0.14s. Obviously, the system can work well in real time, which is a necessary condition to conduct autonomous mental development for an artificial agent.

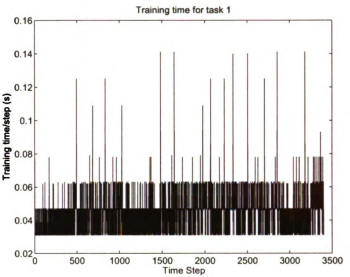


Figure 6.11: Training time of each step for task 1.

6.5 Conclusions

In this chapter, we apply the DOSASE MDP model to conducting cross-task learning for autonomous mental development. No prior knowledge is needed for each task. The model helps an agent to find the relatedness between different tasks by comparing the similarity of states generating for each task. The agent doesn't need to know all the tasks in advance. The learned knowledge in shared context can be transferred to new task, which speeds up the training procedure. The model is tested in a simulation environment for vision-based navigation. Two tasks have been incrementally learned in real time. There are big gains in terms of time and space complexities. The experimental result shows the effectiveness of the AMD paradigm. For the future work, we will test the model on a robot in the real world. This is a challenging problem since in the real world the number of state could be huge, which makes learning difficult. Also we would like to expand the model for learning using multi-

modal inputs to see how a robot develops its visual and auditory capabilities at the same time.

Chapter 7

Adaptive Multimodal Context-Aware User Presence Detection

Context-aware system has drawn increasing attention recently. The goal is to observe users' status and provide proper assistance. In this chapter, we present a user presence detection system. Both visual and acoustic contexts are integrated to infer a user's activities in an office. Since human behaviors are organized in a hierarchical manner, we implement Layered Hidden Markov Models (LHMM) to model human activities with different granularities. A more challenging issue is to make the system adapt to different users and different environments. Instead of building in all the world knowledge in advance (which is intractable), the system's adaptive capability enables it to learn user-centered knowledge (which is tractable). IHDR (Incremental

Hierarchical Discriminant Regression) algorithm is used to automatically generate models for acoustic signals related to a user. IHDR can learn new signals online and its coarse-to-fine tree structure guarantees learning in real-time. A new type of feature is developed to further improve acoustic signal classification. The prototype system has been successfully tested in open domains.

7.1 Introduction

Context-aware system [65] has drawn increasing attention from researchers and engineers. Context is defined as "the physical and social situation in which computational devices are embedded." Context-awareness is the key component of the next generation human-computer interaction technique, which tends to measure the information about "where," "what," "when," and "who." A few prototype context-aware systems have been implemented in last decade. Shafer, Brumitt & Cadiz [84] create an "EasyLiving" home environment which is similar to the intelligent office environment [69]. In [71] [101], driver's behaviors are modeled so that the intelligent assistance systems can improve the safety of driving. Wearable computing [88] is another hot area, where intelligent devices embedded in clothes, watches or glasses.

The goal of context-aware computing is to provide users with services, which are appropriate to their particular situational information. The system detects whether users are in certain environments (room, office, car, etc) and what's their status (in, out, online, offline, awake, sleeping, etc). However, current presence detection systems rely on keyboard and mouse activities, which do not reflect actual user activities.

For example, one current presence system may set the presence status to offline after 15 minutes of keyboard inactivity. If the user is left the office immediately after working on the computer, his presence status is still shown as "active." Our work aims at computing the presence reliably in real-time. In order to build such a system, we integrate visual-acoustic contextual information and other aspects of a user's information such as past states.

A significant portion of previous work related to context-aware computing focuses on recognizing human activities based on a single modality input in a specific environment. Aggarwal and Cai [1] write a review of human motion analysis. There are two popular probabilistic approaches in visual activity recognition: Hidden Markov Model (HMM) and Bayesian Belief Network (BBN). One of the earlier attempts to apply HMMs to activity recognition is found in [123]. Since then, a lot of extensions of HMMs have been tried to model different human activities. Variable-length HMM [32] is applied to exercise behavior recognition. Brand & Oliver use Coupled-HMM [8] to detect interactions between multiple people. Entropic-HMM [7] is introduced to detect and recognize activities in video. Bobick and Ivanov [53] apply a stochastic context-free parsing to recognize activities sequence generated by low-level HMMs. Bayesian Belief Network is another popular approach. Buxton & Gong [16] adopt BBN for visual surveillance. Madbhushi & Aggarwal [61] use BBN to recognize actions including sitting down, standing up, hugging, etc.

One limitation of the above studies is that there have been few studies on human activity recognition using multimodal context information. In [20], audio and visual contexts are combined to classify simple activities such as crossing road and passing

through door. Oliver [70] proposes a Layered HMM architecture integrates information from multimodal inputs to recognize six activities in an office. In order to push the limit of context-aware computing, we need further study of multi-modal context awareness systems.

A more challenging issue is the adaptation capability of context-aware systems. These systems should be able to work in different environments and adapt to different users. Most current user presence detection systems are limited to a constrained setting. A reliable system for a user in one environment is not useful for other users in different settings. For example, if we can detect the phone ring signal in an office, it would be helpful to infer whether a user is going to have a phone conversation. But each room has different telephones and each user has different cell phones. It is impossible to build a system to detect all kinds of phone ring using pre-trained classifier. A useful system must adapt to its user and learn user-centered signal on the fly in real-time.

In this paper, we propose a real-time audio/video user presence detection system to monitor the behaviors of a single human in an office. The system has the following features: 1) A multimodal context-aware system is built. We integrated both audio and visual information. A layered architecture is implemented to model human activities with different granularities. 2) IHDR (Incremental Hierarchical Discriminant Regression) is implemented to learn new acoustic signal online and adapt to new settings, which is the major difference between this work and [70]. This novel component is crucial for adaptation to unknown environments as a consumer product. Instead of building in world knowledge in advance (which is intractable), the system's

adaptive capability enables it to learn user-centered knowledge (which is tractable). And IHDR is organized in tree structure. The coarse-to-fine structure guarantees learning in real-time. 3) A new type of feature for acoustic signal is developed to further improve the performance.

In what follows, we first describe the system and challenges we have to face in Section 7.2. System architecture is discussed in Section 7.3. Then we present the experimental results and summarize our work.

7.2 System Overview and Challenges

A typical setting of a context-aware user presence detection system is shown in Fig. 7.1. A table and a chair are in an office. On the table, there are a telephone, a

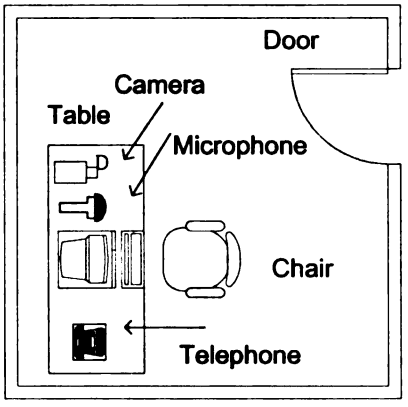


Figure 7.1: Typical setting of a user presence detection system.

personal computer, a video camera and a microphone. The sensors we use to collect context information are: 1) USB camera: a Logitech QuickCam camera sampled at 15 f.p.s. is used to detect human motions. The resolution is 640×480. 2) Microphone:

a built in microphone, associated with the camera.

The goal of this system is to detect human activities in an office. The overview of the user presence detection system is shown in Fig. 7.2. Two kinds of sensory

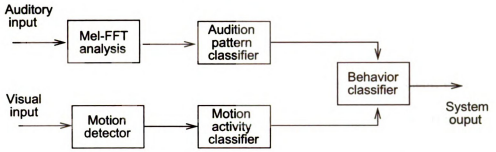


Figure 7.2: Overview of the user presence detection system.

inputs are used: auditory and visual. Audition pattern classifer discriminates four kinds of auditory patterns: "Phone ring," "Conversation," "Uncertain noise," and "Silence." A motion detector captures motion information by computing the difference between two consecutive images. A sequence of motion activities is classified into four motion patterns: "Rest," "Moving near door," "Moving in the office," and "Out." The outputs from the above two components are combined together to infer human activities in the office. There are totally four kinds of activities: "Conversation," "Other activity," "Rest," and "Nobody around."

One technical difficulty is to fuse different context inputs. In this system, the sampling rate of acoustic signal is 50Hz while the sampling rate of visual signal is 15Hz. A "Moving in the office" pattern can be detected in 0.1 second while to detect "Nobody around" the system has to wait for 2 seconds to confirm the status. Psychological studies show that human behaviors are hierarchical [124]. In order to model human activities with different granularities, we implement Layered-Hidden

Markov Model (LHMM). Low-level HMMs are used for motion pattern classification while high-level HMMs are used for human activity classification.

Adaptation to user and environment is a more challenging issue. As we mentioned in Section 1, if the system can detect the user's sound or detect his phone ring signal (telephone in office or cell phone), it would be easy to infer whether he is in the office and what is his status (face-face conversation, phone conversation, silent, etc). Since each user has unique voice and there are thousands of different phone ring tones in this world, it is impossible to has a system to recognize all these acoustic signals. If the system could adapt to each user and the related environment, we have a tractable solution. We propose the IHDR (Incremental Hierarchical Discriminant Regression) algorithm for acoustic signal classification and learning. HMM has been used for speech recognition because it has dynamic modeling capability for sequential signals. However, one limitation of HMM is that it is only a computational model in the sense that HMM is not designed to be generated automatically from observations. Engineers have to manually design the system for given settings in advance, which affects its adaptive capability in unknown environments. For example, if a new type of sensory input is added to the system, the designer has to create new HMMs and specify parameters for each HMM, which is inconvenient especially for unprofessional users. In contrast, IHDR organizes the learned patterns in tree structure. New patterns are added as new leaf nodes. One tree is enough to model all acoustic signals related to a user. Its coarse-to-fine tree structure enables efficient learning in real-time. We also develop Mel-FFT feature to further improve the performance of acoustic signal classification. Details of each component is discussed in next section.

7.3 System Architecture

The detailed architecture of the user presence detection system is shown in Fig. 7.3. Mel analysis [23] is applied to raw auditory signals to extract Mel-FFT features, which are fed into an IHDR tree to classify four kinds of auditory patterns. Sequences of motion activities are classified by low-level HMMs into four motion patterns. High-level HMMs integrate outputs from the above two components to infer human activities.

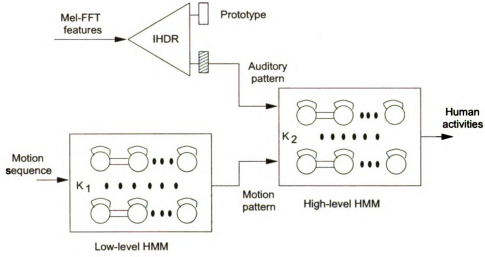


Figure 7.3: A detailed architecture of the user presence detection system.

7.3.1 Acoustic signal classification using IHDR

Feature Extraction

With the audition component, we try to discriminate the following four signals: conversation, phone ring, silence (background noise) and uncertain noise (radio, music, flapping, clapping, etc). In most speech processing systems, Mel-frequency Cepstral Coefficients (MFCCs) [23] is the dominating feature because MFCC captures the

properties of human speech signals. The procedure to calculate the MFCC feature is shown in Fig. 7.4.

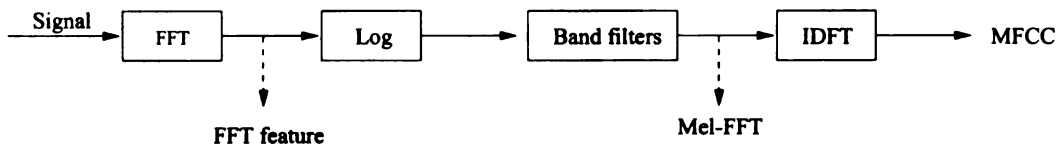


Figure 7.4: The procedure to extract FFT, Mel-FFT and MFCC features.

First, we apply FFT (Fast Fourier Transform) to get features in the frequency domain. Usually, we calculate the logarithmic energy of FFT coefficients. Critical band filters are applied to derive the appropriate frequency components for computing the MFCC. This implementation is described in Eq. 7.1,

$$Y(i) = \sum_{k=F_{is}}^{F_{ie}} log|S(k)|H_i(k\frac{2\pi}{N'}) \tag{7.1}$$

where $log|S(k)|$ is the kth log energy of the Fourier coefficient, N' is the length of the ith triangular band. H_i is a triangular filter:

$$H_i(x) = \begin{cases} 1 - |x|, & x \leq 1 \\ 0, & x > 1 \end{cases} . \tag{7.2}$$

The variable x is defined in Eq. 7.3:

$$x = \frac{k - F_{is}}{F_{is} - F_{ie}}, \tag{7.3}$$

where F_{is} and F_{ie} are the starting frequency and the ending frequency of the ith triangular band, respectively; $F_{is} \leq k \leq F_{ie}$. After we get a sequence of $Y(i)$, we can apply IDFT (Inverse Discrete Fourier Transform) to this sequence, and then we can get the MFCC feature using Eq. 7.4:

$$c(m) = \frac{1}{N'} \sum_{k=0}^{N'-1} \tilde{Y}(k) e^{jk(2\pi/N')m}. \tag{7.4}$$

where m stands for the index of the MFCC feature.

In Eq. 7.1, $Y(i)$ denotes the sum of the weighted $log|S(k)|$ within the ith critical band filter. $Y(i)$ is sometimes called the weighted *log* energy in the ith critical band. In this paper, we call $Y(i), i = 1, 2, ..N'$ the Mel-FFT feature. It is worth noting that the system doesn't need to understand the meaning of conversational signals but to classify different sounds. In section 7.4, we compare FFT feature, Mel-FFT feature and MFCC feature. It turns out that Mel-FFT is the best choice for this application.

Incremental Hierarchical Discriminant Regression

After extracting the features of acoustic signals, how do we classify them? Classification and regression problems can be formulated as a complicated function which maps high-dimensional input and the current state to low-dimensional output signals. Decision tree [10] has been a popular method for function approximation. We use the IHDR for acoutic signal classification.

Each leaf node generates quite a few primitive prototypes (block in Fig. 7.3), which represent different patterns (model generator). In testing phase, if a prototype

is reached (shadowed block), the label associated with it would be the output (computational model). A more detailed description of the algorithm can be found in [48]. The advantages of IHDR include: 1) IHDR maps the high dimension data into low dimensional subspace. LDA is conducted in the discriminating subspace, which saves a lot of computational power. 2) IHDR organizes the learning knowledge coarse to fine, which speeds up the retrieval procedure. The time complexity is $NlogN$, where N is the number of prototypes in the tree. 3) IHDR intrinsically has incremental online learning capability to adapt to new signals since the sufficient statistics are updated incrementally. This is why we use IHDR instead of traditional HMM to classify auditory patterns. For practical applications, the system has to work in different offices. Each office has different people and different telephones. In order to make HMMs adapt to new settings, a bank of new HMMs has to be created manually. In contrast, one IHDR is enough to handle all types of auditory signals related to a user. And also important is that the adaptation can be incrementally conducted in real-time.

7.3.2 HMMs for motion pattern classification

In order to recognize human motion activities, we have to find the moving body and its location. We implement a motion detector to find the bounding box of a moving body with two consecutive images. First, we calculate the difference between these two images. Then a low-pass filter is used to remove noise in the difference image. The bounding box of the moving body is determined by the histogram of the

image [76]. Now we know the size and the location of the bounding box, which is the observation of motion activities. After getting a sequential motion sub-patterns, we use discrete Hidden Markov Model (HMM) to classify human motion behaviors. An HMM is denoted by $\lambda = (A, B, \pi)$, where A is state transition probability matrix, B is the observation symbol probability matrix, π is the initial state distribution. Specification of an HMM involves the choice of the number of states N, the number of observations M. With training data, we can calculate λ by using Baum-Welch algorithm [78]. Given a model λ and a sequence of observation O=$\{O_1, O_2, ..., O_T\}$, the likelihood of the sequence is

$$P(O|\lambda) = \Sigma_i \alpha_T(i), 1 \leq i \leq N, \tag{7.5}$$

where $\alpha_t(i)$ is defined as

$$\alpha_t(i) = [\sum_i \alpha_{t-1}(i) a_{ij}] b_j(O_t), \tag{7.6}$$

where a_{ij} is the element of A and $b_j(O_t)$ is the probability for state j in the model λ of observing O_t. The meaning of $\alpha_t(i)$ is the probability of the partial observation sequence, $o_1, o_2 ... o_t$, and state i at time t, given the model λ.

Let K be the number of motion/activity patterns, we usually need to generate a bank of K HMMs (λ_k ($1 \leq k \leq K$)). The likelihood of the observation sequence in each model is $L_k = P(O|\lambda_k)$. Suppose the maximal likelihood is $L_{\max}$ and the

minimal likelihood is $L_{\min}$, the normalized likelihood is

$$L_k^{'} = \frac{L_k - L_{\min}}{L_{\max} - L_{\min}}. \tag{7.7}$$

HMMs choose the pattern $\hat{k}$ with the largest normalized likelihood as output.

$$\hat{k} = \arg\max_k \{L_k^{'}\} \tag{7.8}$$

7.3.3 Integration component

The system is able to recognize four types human activities: "Conversation," "Other activity," "Rest," and "Nobody around." Integration of the above two low-level components to infer human activities is difficult since different modalities have different updating frequency and they can be either related or unrelated. High-level HMMs are necessary for reasoning human activities based on information of low-level components. Usually the reasoning is conducted with a larger time granularity (for example, 2 seconds), while in motion pattern classification and auditory pattern classification, the granularity is less than 1 second. The outputs from low-level component are symbols, which can be fed into the integration HMMs. Since the vision component outputs 4 types of different patterns (so does the audition component), the combination of these two components gives 16 types of observations. For example, if the motion observation O_{motion} is "Moving in office" and the acoustic observation $O_{acoustic}$ is "Conversation," let's define $O_{motion} = 2$ and $O_{acoustic} = 2$. The pattern of integra-

tion observation is $O_{integration} = O_{motion} \times 2 + O_{acoustic} = 2 \times 4 + 2 = 10$. Baum-Welch algorithm [78] is used to train sequences of combination observations.

7.4 Experimental results

We conducted experiments for each of these three components.

7.4.1 Experimental results of the auditory component

We trained the system with 9 kinds of male conversation signals, 3 kinds of female conversation signals, 16 kinds of phone ring signals, background noise and uncertain noise (radio, music, flapping, clapping) using IHDR. Some signals are shown in Fig. 7.5.

Comparison of different feature vectors

First, we compared three kinds of features: FFT, Mel-FFT, and MFCC. The auditory data are digitized at 11025Hz by a normal sound blaster card. The number of Mel-FFT feature vectors is about 12000. Half is used for training, another half is used for testing. The number of training samples is about 6000 for each feature, the number of testing samples is 5174. We can find out in Tab. 7.1 that Mel-FFT is better than FFT and MFCC. The recognition rate is about 90%. The dimensions of FFT, Mel-FFT, and MFCC feature vector are 512, 21, 13, respectively. The corresponding time to train each feature vector using IHDR, is 115ms, 8.6ms and 7.3ms, respectively. In order to build a real-time online learning system, FFT is not acceptable since the

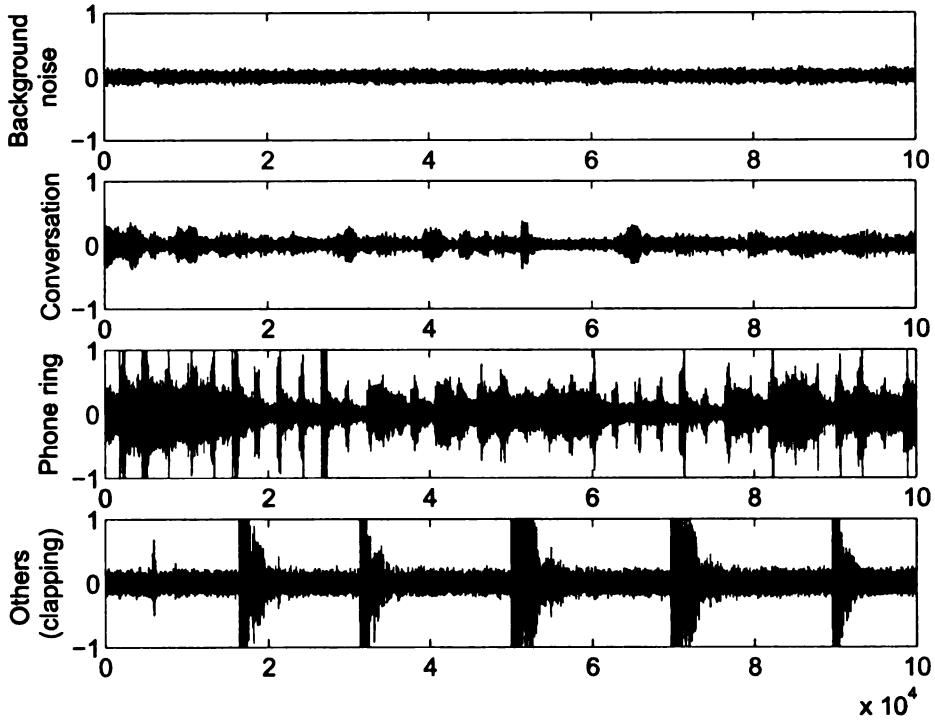


Figure 7.5: Different acoutsic signals: from top to down are background noise, conversation, phone ring and other noise (clapping). x-axis denotes the number of raw signal points, y-axis denotes the normalized energy.

system has to incrementally learn about 50 frames per second. Considering training time and recognition rate, we choose Mel-FFT as the feature of acoustic signals.

Table 7.1: Comparision of different features for acoustic signals.

Features	Dim	Training Time	Correct	Total	Rate
FFT	512	115ms	4314	5174	83.38%
MelFFT	21	8.6ms	4641	5174	89.70%
MFCC	13	7.3ms	4448	5174	85.92%

Online learning

As we mentioned earlier, we use IHDR rather than HMM for acoustic signals classification. The reason is that one IHDR tree can adapt to new signals, new users and new environments. In contrast, using HMM the designer has to create new models for new signals. In order to verify the adaptive capability of IHDR, we tested the online learning component. We divided the dataset into 3 sets: Set A consists of conversation signal and 8 types of phone ring signals; Set B consists of conversation signal and 7 types of phone rings signals; Set C consists of one type of phone ring signal, which is in A but not in B. We first trained set B offline then tested on set A. The confusion matrix is shown in Tab. 7.2, the recognition rate of phone ring signal is only 81.5%.

Table 7.2: Confusion matrix of offline training. (C=Conversation, P=Phone)

Data	C	P	Total	Rate
C	4064	254	4318	94.12%
P	275	1212	1487	81.50%

In the next experiment, using the offline generated IHDR tree, we trained the system with set C and then tested on set A again. The result of online learning is shown in Tab. 7.3. As we can see, the recognition rate of phone ring signals is improved from 81.50% to 89.64%, which proves the adaptive capability of IHDR.

Table 7.3: Confusion matrix after online training. (C=Conversation, P=Phone)

Data	C	P	Total	Rate
C	4071	247	4318	94.28%
P	154	1333	1487	89.64%

Then we tested on all four kinds of acoustic signals. The confusion matrix is shown in Tab. 7.4. Conversation and uncertain noise signals are confusing sometimes

Table 7.4: Recognition rate of each auditory set. (C=Conversation; UN=Uncertain Noise; P=Phone; S=Silence)

Data	C	UN	P	S	Total	Rate
C	2275	188	6	4	2470	92.11%
UN	216	2104	0	0	2320	90.69%
P	8	14	1768	5	1795	98.50%
S	0	0	0	1141	1141	100%

(188 conversation vectors are recognized as uncertain noise) because the uncertain noise covers a very large feature space. The overall recognition rate is 94.33%. Comparing to [44], the performance gains about 0.5%. We need to notice that the test is source-dependent. In other words, the testing set and the training set come from the same source. If we test the system with signals of a telephone we never trained, the performance would drop. That's why IHDR is important for this application since it can incrementally learn new auditory patterns.

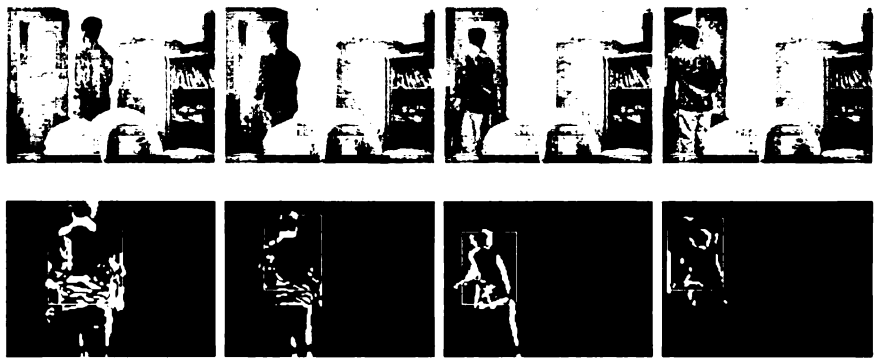


Figure 7.6: The first row shows the video sequence, which is classified as "Moving near the door"; the second row shows the bounding box of the moving object.

7.4.2 Experimental results of the motion component

Fig. 7.6 shows a motion sequence, which is classified as "Moving near the door". The first row is the video sequence while the second row shows the bounding box of the moving object.

With the size and the position of the moving body, we can feed the motion sub-pattern sequence (observations) into HMMs. In this experiment, the parameters of the low-level HMMs are: $N_s = 6$, M=4 and K=4. Different observations are: "Motion in door area," "Motion in room but not door," "Static & last motion in door area," and "Static & last motion in room but not door." The normalized likelihood of each motion pattern is shown in the first four plots of Fig. 7.7. The x-axis is time line about 400 seconds. The granularity of HMMs is 1 second. If $L'_k = 1$, then pattern k is reported. The ground truth of activity sequences is shown in the fifth plot, which goes as follows: the user firstly moved in the room (pattern 1), rested for a while (2), moved around the door (3) and then went out (4). The motion behaviors are clearly recognized. A mistake occurs around step 320, the system classified pattern (3) as

pattern (1) when the user moved near the boundary of the door and the remain parts of the room.

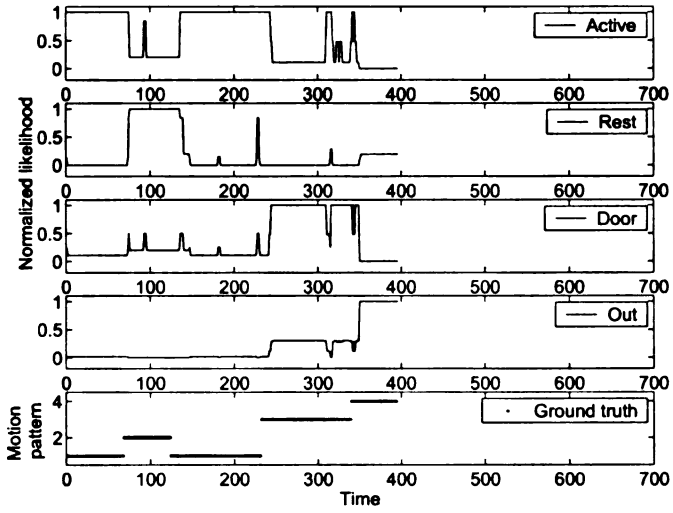


Figure 7.7: Likelihood of motion patterns over time.

7.4.3 Experimental results of the high-level reasoning

The specifications of the high level HMMs are: $N_s = 4$, M=16 and K=4. Normalized likelihood of each human activity is shown in the first four plots of Fig. 7.8. The ground truth of activity sequences is in the fifth plot, which goes like this: the user moved around in the rest (activity 1), rested (2), moved around again (1), talked for a while (3), moved to the door (1) and went out (4). The x-axis is time line about 800 time frames. "Nobody" and "Rest" are perfectly classified, while "Conversation" and "Other activity" are messed up a little because sometimes you can move and talk at the same time. 40 minutes of office activity are recorded (about 10 minutes for each activity). A half of the data is used for training; another half is used for

testing. The recognition rate of human activities is shown in Tab. 7.5. About 4% of "Conversation" is incorrectly recognized as "Other activity", which is consistent with the results in Fig. 7.8.

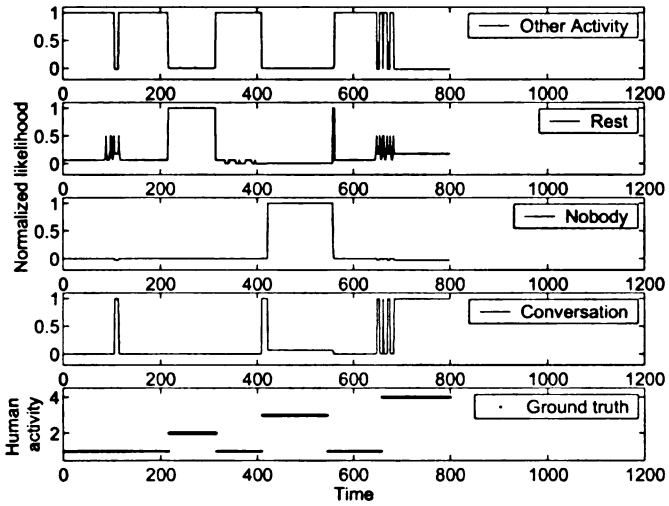


Figure 7.8: Likelihood of human activities over time.

Table 7.5: Recognition rate of human activities (N=Nobody; OA=Other activity; R=Rest; C=Conversation).

Data	N	O	R	C
N	100%	0	0	0
O	0	94.31%	0	5.69%
R	0	0	100%	0
C	0	4.52%	0	95.48%

7.5 Summary and Discussion

In this chapter, we proposed a multimodal context-aware system to detect user presence in an office. Two levels of HMMs with different granularities handle motion

pattern classification and integration of low-level outputs, respectively. A more exciting feature of the system is its adaptive capability. IHDR, as a model generator, generates representations for different auditory patterns and can easily adapt to user-centered signals. For example, IHDR can learn its owner's sound and the phone ring signal in his/her office in real-time. The initial results of the prototype system are promising. However, to build a highly adaptive system for consumers needs further study. Here are some future works: 1) Detect motion activities of multiple persons. Right now, only one user can be detected. With multi-person detection, we can find out the interactions between different people. 2) Dynamic CPU usage adaptation. In the current system, the audio and visual components use up most of the computational power. We are going to implement an attention mechanism, which can turn off a component when it is not useful. Thus, the user can run the system in the background while allocating computational power to other applications.

Chapter 8

Conclusions and Future Work

8.1 Contributions

Studies in human cognitive development have shown that interactions between a higher animal and its environment is essential for perception development and knowledge acquisition. This thesis reports some recent research on developmental robots, the robots that learn autonomously through real-time interactions with the environment. The feasibility of the developmental robots is demonstrated under eight challenging requirements for autonomous mental development (AMD) as we discussed in Section 1.1.3.

At this early stage of research in developmental robots, we are facing an array of challenging technical issues. We have developed two major techniques based on a developmental architecture, which was implemented on a real robot, SAIL, an early developmental robot prototype. A summary of the contributions is given below.

1. Propose the Developmental, Observation driven, Self-Aware, Self-Effecting, Markov Decision Process (DOSASE MDP) model, which integrates multimodal sensing, action-imposed learning, reinforcement learning, and communicative learning.

2. Build the cognitive mapping engine using the Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) technique, which has better generalization capability under non-stationary environments, especially for vision-based navigation.

3. Novelty and reinforcement learning are integrated into a robotic value system for the first time. As an important part of AMD, a value system signals the occurrence of salient sensory inputs, modulates the mapping from sensory inputs to action outputs, and evaluates candidate actions. The value system is applied to guide a robot's visual attention behavior. With this value system, developmental robots has better autonomy with novelty (dense input) and reinforcer (sparse input).

4. Present a robotic system that develops covert perceptual capability for vision-based autonomous navigation via reinforcement learning. The agent learns different road boundaries through online interactions with human trainers. All these cannot be achieved by traditional supervised learning techniques since these kind of behavior cannot be imposed.

5. Provide the developmental agent with cross-task learning capability. That is, an

agent can learn multiple tasks and use acquired knowledge to speed up learning new tasks. With capability, a developmental robot can save time and save space for multiple tasks. This is very important for developmental learning at current stage since resource is limited.

6. Applies the above techniques to a practical application funded by industry. Context information from multi-sensory inputs is integrated to infer a user's activities in an office. This is the first open-domain audio/visual learning system for user presence detection. The prototype system verifies the effectiveness of the developmental learning paradigm. Currently, we are seeking the possibility for turning the system into products.

8.2 Future directions

Autonomous mental development by a robot is an interdisciplinary research area. While more and more researchers are beginning to realize the importance and potential power, it is still at its early stage. This thesis represents initial steps in this direction. In addition to the above contributions, this work has raised many new questions and left out many interesting but unresolved problems as well.

Attention-based navigation. Vision-based navigation is a hard problem. How to use attention mechanisms to generalize learned knowledge could be a solution. Further studies are needed to fully explore the capability of DOSASE MDP model for navigation task learning. For instance, communicative learning could be a good choice to guide a robot's attention behavior.

A sophisticated value system. In this thesis, we propose a value system integrating novelty and reinforcement learning for the first time. We only test the value system for the visual attention behavior. The problem is how to scale up the value system for complex behaviors? While it is clear that the behavior of the value system should be developed through experiences, it is not clear how such a system can learn from an unstructured and highly inconsistent environment. Should this value system be a centralized universal control coordinator or a distributed system? How does a developmental robot conduct thinking under the guidance of its value system?

Applications in human machine interactions. While enjoying the improved quality of life brought by the more and more sophisticated machines, we are facing many new problems, such as the machines' flexibility, ease of usage, and efficient cooperation with human users. The related broad research areas can be grouped under human machine interactions (HMI). Machines conducting perceptual learning and behavior learning autonomously in real-time would resolve some of the above problems of HMI. It is about time to identify some HMI related applications, such as intelligent rooms and intelligent vehicles, and apply the principles and techniques of autonomous mental development. The prototype user presence detection system provides a good test-bed.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] J. S. Albus. Outline for a theory of intelligence. *IEEE Trans. Systems, Man and Cybernetics*, 21(3):473–509, May/June 1991.

[3] N. Almassy, G. M. Edelman, and O. Sporns. Behavioral constraints in the development of neural properties: A cortical model embedded in a real-world device. *Cerebral Cortex*, 8(4):346–361, 1998.

[4] J. R. Anderson. *Rules of the Mind.* Lawrence Erlbaum, Mahwah, New Jersey, 1993.

[5] C. Balkenius and N. Hulth. Attention as selection-for-action: a scheme for active perception. In *Proceedings of EUROBOT '99*, pages 113–119. 2001.

[6] C. Bander, F. Vico, J.Bravo, M. Harmon, and L. Baird. Residual q-learning applied to visual attention. In *Proc. of the 13th International Conference on Machine Learning*, pages 20–27. Bari, Italy, July 3 - 6 1996.

[7] M. Brand and V. Kettnaker. Discovery and segmentaion of activities in video. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[8] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for modeling interacting processes. In *Proc. of Int. Conf. on Computer Vision and Pattern Recognition*, pages 994 - 999, 1996.

[9] C. Breazeal. Social constraints on animate vision. Cambridge, MA, June 20-25, 2000.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, 1993.

[11] R. Brooks. Intelligence without reason. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 569–595, Sydney, Australia, August 1991.

[12] R. Brooks. Cog, a humanoid robot. Technical report, MIT Artificial Intelligence Laboratory, 1997. private communication.

[13] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[14] R. A. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, and M. M. William. The cog project: Building a humanoid robot. In C. L. Nehaniv, editor, *Computation for Metaphors, Analogy and Agents, vol. 1562 of Springer Lecture Notes in Artificial Intelligence*. Springer-Verlag, New York, NY, 1999.

[15] B.G. Buchanan and E.H. Shortliffe, editors. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.

[16] H. Buxton and S. Gong. Advanced Visual Surveillance using Bayesian Networks. In *Proc. of Int. Conf. on Computer Vision*, pages 111–123, Cambridge, Massachusetts, June 1995.

[17] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[18] N. Chomsky. *Language and Mind*. Harcourt Brace Jovanovich, New York, 1972.

[19] R. M. Church, D. J. Getty, and N. D. Lerner. Duration discrimination by rats. *Journal of Experimental Psychology: Animal Behavior Processes*, 2:303–312, 1976.

[20] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. In *Int. Joint Conf. on Acoustic, Speech and Signal Processing, ICASSP'99*, pages 3037 - 3040, 1999.

[21] D. R. Cox. Statistical analysis of time series: Some recent developments. *Scand. J. Statist.*, 8(2):93–115, 1981.

[22] M. R. D'Amato and M. Colombo. Representation of serial order in monkeys (cebus apella). *Journal of Experimental Psychology: Animal Behavior Processes*, 14:131–139, 1988.

[23] J. Deller, J. Proakis, and J. Hansen. *Discrete-time processing of speech signals*. Institute of Electrical and Electronics Engineers Press, New York, 2000.

[24] M. Domjan. *The Principles of Learning and Behavior*. Brooks/Cole, Belmont, CA, fourth edition, 1998.

[25] M. Domjan. *The Principles of learning and behavior*. Brooks/Cole Publishing Company, Belmont, CA, 1998.

[26] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[27] G. L. Drescher. *Made-Up Minds*. MIT Press, Cambridge MA, 1991.

[28] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley, New York, NY, 2001.

[29] J. Weng et al. Autonomous mental development by robots and animals. *Science*, 291:599–600, January 26, 2001.

[30] J.H. Flavell, P.H. Miller, and S.A. Miller. *Cognitive Development.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[31] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, NY, second edition, 1990.

[32] A. Galata, N. Johnson, and D. Hogg. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding*, 81(3):398–413, 2001.

[33] R. A. Gardner and B. T. Gardner. Teaching sign language to a chimpanzee. *Science*, 165:664–672, 1969.

[34] R. Gisiner and R. J. Schusterman. Sequence, syntax, and semantics: Responses of a language-trained sea lion (zalophus californianus) to novel sign combinations. *Psychology*, 106:78–91, 1992.

[35] G. H. Golub and C. F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 1989.

[36] R. M. Groves and R. F. Thompson. Habituation: A dual process theory. *Psychological Review*, 77:419–450, 1970.

[37] R. Held and A. Hein. Movement-produced stimulation and the development of visually guided behaviors. *Journal of Comparative and Physiological Psychology*, 56:872–876, 1963.

[38] L. M. Herman. Receptive competencies of language-trained animals. In J. S. Rosenblatt, C. Beer, M. C. Busnel, and P. J. B. Slater, editors, *Advances in the study of behavior (Vol. 17)*, pages 1–60. Academic Press, New York, 1987.

[39] R. J. Herrnstein and P. A. deVilliers. Fish as a natural category for people and pigeons. In G. H. Bower, editor, *The psychology of learning and motivation (Vol. 14)*, pages 60–97. Academic Press, New York, 1980.

[40] J. H. Holland. *Adaptation in Naural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[41] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Proc. Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems (EPIROB'02)*, pages 47–55, Edinburgh, Scotland, August 10 - 11, 2002.

[42] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Second International Workshop on Epigenetic Robotics*, Edinburgh, Scottland, August 10-11 2002.

[43] X. Huang and J. Weng. Locally balanced incremental hierarchical discriminant regression. In *Fourth International Conference on Intelligent Data Engineering and Automated Learning*. 2003.

[44] X. Huang, J. Weng, and Z. Zhang. Office presence detection using multimodal context information. In *Proc. of the International ConferenceAcoustics, Speech, and Signal Processing (ICASSP 2004)*, Montreal, Quebec, Canada, USA, 2004.

[45] W. Hwang and J. Weng. Vision-guilded robot manipulator control as learning and recall using SHOSLIF. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, Albuquerque, NM, April 20-25 1997.

[46] W. Hwang and J. Weng. Hierachical discriminant regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277 –1293, 2000.

[47] W. Hwang and J. Weng. Hierarchical discriminant regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, 2000.

[48] W. Hwang and J. Weng. An online training and online testing algorithm for OCR and image orientation classification using hierarchical discriminant regression,. In *Proc. Fourth IAPR International Workshop on Document Analysis Systems*, Rio De Janeiro, Brazil, December 10-13, 2000.

[49] W. Hwang and J. Weng. Incremental hierarchical discriminant regression for indoor visual navigation. In *Int'l Conf. on Image Processing*, Thessaloniki, Greece, October 2001.

[50] W.S. Hwang. *Visual learning and its application to sensorimotor actions*. PhD thesis, Department of Computer Science and Engineering, Michigan State University, 2000.

[51] W.S. Hwang and J.J. Weng. Hierarchical discriminat regression. *IEEE Trans. on Patten Analysis and Machine Intelligence*, 22(11):1277–1293, 1999.

[52] L. Baird III and A. Klopf. Reinforcement learning with high-dimensional, continuous actions. *Technical Report WL-TR-93-1147 Wright Laboratory*, 1993.

[53] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochstic parsing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[54] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[55] S. Kakade and P. Dayan. Dopamine bonuses. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

[56] L. J. Kamin. "attention-like" processes in classical conditioning. In M. R. Jones, editor, *Miami symposium on the prediction of behavior: Aversive stimulation*, pages 9–31. University of Miami Press, Miami, FL, 1968.

[57] M. Kirby and L. Sirovich. Application of the karhunen-loève procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(1):103–108, Jan. 1990.

[58] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, second edition, 1997.

[59] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[60] M.C. Lovett, L.Z. Daily, and L.M. Reder. A source activation theory of working memory: Cross-task prediction of performance in act-r. *Cognitive Systems Research*, 1:99–118, 2000.

[61] A. Madabhushi and J. Aggarwal. A bayesian approach to human activity recognition. In *Proc. of The second Int. workshop on Visual surveillance*, pages 25–30, 1999.

[62] M. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:1:51–80, 1995.

[63] S. Minut and S. Mahadevan. A reinforcement learning model of selective visual attention. In *Proceedings of the fifth international conference on Autonomous agents*, pages 457 - 464. Montreal, Quebec, Canada, 2001.

[64] P. Montague, P. Dayan, and T. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *The journal of Neuroscience*, 16(5):1936–1947, 1996.

[65] T. Moran and P. Dourish. Introduction to this special issue on context-aware computing. *Human Computer Interaction*, 16:87–95, 2001.

[66] M. J. Morgan, M. D. Fitch, J. G. Holman, and S. E. Lea. Pigeons learn the concept of an "A". *Perception*, 5:57–66, 1976.

[67] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence*, 2, August 1994.

[68] H. Ogmen. A developmental perspective to neural models of intelligence and learning. In D. Levine and R. Elsberry, editors, *Optimality in Biological and Artificial Networks?*, pages 363–395. Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1997.

[69] N. Oliver and E. Horvitz. Selective perception policies for guiding sensing and computation in multimodal systems: A comparative analysis. In *Proc. of Int. Conf. on Multimodal Interfaces*, pages 3–8, 2003.

[70] N. Oliver, E. Horvitz, and A. Garg. Layered representation for human activity recognition. In *Proc. of Int. Conf. on Multimodal Interfaces*, pages 3–8, 2002.

[71] N. Oliver and A. Pentland. Driver behavior recognition and prediction in a smartcar. In *Proc. of SPIE Aerosense2000 'Enhanced and Synthetic Vision'*, Orlando, Florida, 2000.

[72] F. G. Patternson. The gestures of a gorilla: Language acquisition in another pongid. *Brain and Language*, 5:56–71, 1978.

[73] I. M. Pepperberg. Some cognitive capabilities of an African Grey parrot (psittacus erithacus). In P. J. B. Slater, J. S. Rosenblatt, and C. Beer, editors, *Advances in the study of behavior (Vol. 19)*, pages 357–409. Academic Press, New York, 1990.

[74] J. Piaget. *The Origins of Intelligence in Children.* International Universities Press, INC, New York, 1952.

[75] Lorien Y. Pratt, Jack Mostow, and Candace A. Kamm. Direct Transfer of Learned Information among Neural Networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 584–589. AAAI, July 1991.

[76] W. Pratt. *Digital Image Processing.* John Wiley&Sons, Inc, New York, NY, 1991.

[77] J. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[78] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings IEEE*, 77(2):257–286, 1989.

[79] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[80] R. A. Rescorla and A. R. Wagner. A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black and W. F. Prokasy, editors, *Classical conditioning II: Current research and theory*, pages 64–99. Appleton-Century-Crofts, New York, 1972.

[81] A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[82] W. Schultz. Multiple reward signals in the brain. *Nature Reviews: Neuroscience*, 1:199–207, 2000.

[83] W. Schultz, P. Dayan, and R. Montague. A neural substrate of prediction and reward. *Science*, 275(14):1593–1599, 1997.

[84] S. Shafer, B. Brumitt, and J. Cadiz. Interaction issues in context-aware interactive environments. *Human Computer Interaction*, 16, 2001.

[85] Wei-Min Shen. *Autonomous Learning from the Environment.* Computer Science Press, New York, 1994.

[86] R. L. Solomon and J. D. Corbit. An opponent-process theory of motivation: Ii. cigarette addition. *Journal of Abnormal Psychology*, 81:158–171, 1973.

[87] R. L. Solomon and J. D. Corbit. An opponent-process theory of motivation: I. the temporal dynamics of affect. *Psychological Review*, 81:119–145, 1974.

[88] F. Sparacino, A. Pentland, and G. Davenport. Wearable performance. In *The First International Symposium on Wearable Computers*, Cambridge, MA, 1997.

[89] O. Sporns, N. Almassy, and G. Edelman. Plasticity in value system and its role in adaptive behavior. *Adaptive Behavior*, 2000.

[90] L. R. Squire and E. R. Kandel. *Memory: From Mind to Molecules.* Scientific American Library, New York, 1999.

[91] Luc Steels. Emergent functionality in robotic agents through on-line evolution. In R. A. Brooks and P. Maes, editors, *Articificial Life IV*, pages 8–14. MIT Press, Cambridge, Massochusetts, 1994.

[92] M. Sur, A. Angelucci, and J. Sharm. Rewiring cortex: the role of patterned activity in development and plasticity of neocortical circuits. *Journal of Neurobiology*, 41:33–43, 1999.

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

[94] R.S. Sutton and A.G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, pages 135–170, 1981.

[95] R.S. Sutton and A.G. Barto. *Reinforcement Learning - An Introduction.* The MIT Press, Cambridge, MA, 1998.

[96] D. L. Swets and J. Weng. Hierarchical discriminant analysis for image retreival. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(5):386–401, 1999.

[97] Y. Takahashi and M. Asada. Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 395–402. 2000.

[98] E. Thelen and L. Smith. *A dynamics systems approach to the development of cognition and action.* MIT Press, Cambridge, MA, 1994.

[99] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach.* Kluwer Academic, Boston, Massachusetts, 1996.

[100] S. Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. In S. Thrun and L.Y. Pratt, editors, *Learning To Learn.* Kluwer Academic Publishers, 1998.

[101] K. Torkkola, N. Massey, Bob. Leivian, C. Wood, J. Summers, and S. Kundalkar. Classification of critical driving events. In *Proc. of the International Conference on Machine Learning and Applications (ICMLA)*, pages 81–85, Los Angeles, CA, USA, 2003.

[102] D. Touretzky and L. Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5(3/4):219–247, 1997.

[103] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, October 1950.

[104] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[105] S. Vijayahumar and S. Schaal. Real time learning in humanoids: A challenge for scalability of online algorithms. Cambridge, Massachusetts, Sept. 7-8 2000.

[106] I. Wallance, D. Klahr, and K. Bluff. A self-modifying production system of cognitive development. In D. Klahr, P. Langley, and R. Neches, editors, *Production System Models of Learning and Development*, pages 359–435. MIT Press, Cambridge, Massachusetts, 1987.

[107] E. A. Wasserman, R. E. DeLong, and M. B. Larew. Temporal order and duration: Their discrimination and retention by pigeons. *Annals of the New York Academy of Sciences*, 423:103–115, 1984.

[108] S. Watanabe, J. Sakamoto, and M. Wakita. Pigeons' discrimination of paintings by Monet and Picasso. *Journal of the Experimental Analysis of Behaviors*, 63:165–174, 1995.

[109] C. Watkins. Learning from delayed rewards. Technical report, PhD thesis, King's College, Cambridge, England, 1989.

[110] C. J. Watkins. Q-learning. *Machine Learning*, 8:279–292, 1992.

[111] J. Weng. Learning in image analysis and beyond: Towards automation of learning. In C. W. Chen and Y. Q. Zhang, editors, *Visual Communication and Image Processing.* Marcel Dekker, New York, NY, 1998.

[112] J. Weng. A theory for mentally developing robots. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, pages 131–140, MIT, Cambridge, Massachusetts, June 12-15 2002.

[113] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition using the Cresceptron. *Int'l Journal of Computer Vision*, 25(2):109–143, Nov. 1997.

[114] J. Weng and W. Hwang. An incremental learning algorithm with automatically derived discriminating features. In *Proc. of Fourth Asian Conference on Computer Vision*, pages 426–431, Taipei, Taiwan, January 8-9 2000.

[115] J. Weng and W. Hwang. Online image classification using IHDR. *International Journal on Document Analysis and Recognition*, 5(2-3):118–125, 2002.

[116] J. Weng and W. S. Hwang. An incremental learning algorithm with automatically derived discriminating features. In *Proc. Asian Conference on Computer Vision*, pages 426 - 431, Taipei, Taiwan, Jan. 8-9 2000.

[117] J. Weng, W. S. Hwang, Y. Zhang, and C. Evans. Developmental robots: Theory, method and experimental results. In *Proc. 2nd International Conference on Humanoid Robots*, pages 57–64, Tokyo, Japan, Oct. 8-9 1999. IEEE Press.

[118] J. Weng, W. S. Hwang, Y. Zhang, C. Yang, and R. Smith. Developmental humanoids: Humanoids that develop skills automatically. In *Proc. First IEEE Conf. on Humanoid Robots*, MIT, Cambridge, Massachusetts, Sept. 7- 8 2000. IEEE Press.

[119] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[120] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[121] S. Whitehead and D. Ballard. Active perception and reinforcement learning. *Neural Computation*, 2:409–419, 1990.

[122] S. Wilson. Classifier systems and the Animat problem. *Machine Learning*, 2(3):199–228, 1987.

[123] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-seqential images using hidden markov model. In *Proc. of Int. Conf. on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[124] J. Zacks and B. Tersky. Event structure in perception and cognition. *Psychological Bulletin*, 127(1):3–21, 2001.

[125] N. Zhang and J. Weng. A developing sensory mapping for robots. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, MIT, Cambridge, MA, June 12-15 2002.

[126] Y. Zhang. *Online Development of Cognitive Behaviors by a Robot - A Case Study Using Auditory and Visual Sensing.* PhD thesis, Department of Computer Science and Engineering, Michigan State University, 2002.

[127] Y. Zhang and J. Weng. Complementary candid incremental principal component analysis. Technical Report MSU-CSE-01-24, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, August 2001.

[128] Y. Zhang and J. Weng. Action chaining by a developmental robot with a value system. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, MIT, Cambridge, MA, June 12-15 2002.