This is to certify that the
dissertation entitled

Online Development of Cognitive Behaviors by a Robot:
A Case Study Using Auditory and Visual Sensing

presented by

Yilu Zhang

has been accepted towards fulfillment
of the requirements for the

Doctoral degree in Computer Science and Engineering

Major Professor's Signature

Dec 7, 2002

Date

Online Development of Cognitive Behaviors by a Robot: A Case Study Using Auditory and Visual Sensing

By

Yilu Zhang

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

2002

Abstract

Online Development of Cognitive Behaviors by a Robot: A Case Study Using Auditory and Visual Sensing

By

Yilu Zhang

Audition and vision are two major sensory modalities for humans to sense and understand the world. Although a significant progress has been made in automatic speech recognition and visual object recognition, the fields still face a tremendous amount of difficulties.

Motivated by the autonomous development process of humans, we are interested in building a robot that automatically develops its auditory and visual cognitive and behavioral skills through real-time interactions with the environment, which typically involves humans. We call such a robot a developmental robot. To resolve the technical challenges for a developmental robot, three basic techniques have been developed and implemented: (1) A fast incremental principal component analysis algorithm, the complementary candid incremental principal component analysis (CCIPCA) algorithm; (2) A customized version of hierarchical discriminant analysis (HDR) for long temporal contexts; (3) A developmental architecture and algorithm that inte-

grate multimodal sensing, action-imposed learning, reinforcement learning, and communicative learning.

Based upon the above three basic techniques, we have designed and implemented a prototype robot that learns cognitive behaviors from simple to complex: (1) Grounded speech learning. The system develops its audition-driven behaviors through physical interactions with the environment. This is the first attempt we know that realizes developmental auditory learning from unsegmented and unlabeled speech streams without using priori knowledge such as handcrafted word models or task-specific heuristic. This emulates a mode in which human children learn. (2) Task transfer. The system applies what has been learned in one context to new contexts (tasks) through verbal instructions from trainers. This is the first system of this kind in an autonomous mental development (AMD) mode. (3) Semantics learning. The system acquires simple semantics through real-time multimodal inputs (vision and audition). This is the first robot with both developmental auditory and developmental visual capabilities to do online learning. It takes advantage of the spatiotemporal continuity of the physical world during learning. Among the above three learning types, the first one uses only the reflexive capability of the proposed architecture, the second one uses an additional priming capability, and the third one uses an additional multimodal integration capability.

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

To my parents.

ACKNOWLEDGMENTS

First and foremost, I would like 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.

During my four years at Michigan State University, I had the fortune of benefiting from several respectful scholars. Anil Jain provided me a solid pattern recognition background and he was always helpful and supportive of my research. John Deller led me into the speech recognition field and constantly cared about my career. I was especially impressed by his rigorous scholarship. Lijian Yang helped me to provide the rigorous proof of theoretical statistical problems thanks to his insight and knowledge. Sridhar Mahadevan introduced me to the broad spectrum of machine learning areas. George Stockman has the talent of turning research into a joyful experience, which made the discussions with him always a fun. I am especially grateful to professors John Deller, Anil Jain, and Lijian Yang, for serving on my committee, sharing ideas with me, and providing critical comments on my thesis.

A special acknowledgment goes to Dr. Jialin Zhong at Bell Labs Lucent Technologies, where I spent a summer as a research intern. A broad exposure to various

research problems made my internship a valuable experience.

I have benefited a lot from many of my colleagues. The early version of IHDR program provided by Wey-Shiuan Hwang was the starting point of the work reported in this thesis. He and Colin Evans, another early SAIL group member, gave me a jump-start on my research at Michigan State University. The discussions and cooperation with them were always interesting and helpful. Yong-Beom Lee wrote the early Cepstrum Analysis code and taught me my first class on speech recognition. During the development of SAIL project, I enjoyed many open discussions with my labmates, Micky Badgero, Amy Bardenhagen, Greg Bloy, Shaoyun Chen, Raja Ganjikunta, Jianda Han, Xiao Huang, Ameet Joshi, Zain Kazim, Mahesh Krishnaswamy, Jason Sperber, Georgios Theocharous, Jingliang Wang, Changjiang Yang, Shuqing Zeng, and Nan Zhang. We shared friendship as well as time together.

I wish to acknowledge some current and former PRIPpies: Paul Albee, Vera Bakic, Scott Connell, Nicolae Duta, Dan Gutchess, Rein-Lein Hsu, Xiaoguang Lu, Silviu Minut, Anoop Namboodiri, Salil Prabhakar, Arun Ross, Umut Uludag, Aditya Vailaya, and Jianguo Wang. The conversations with them and the seminars gave by them gave me a wide 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, Shouzhen Wang and Yunruo Zhang. Their endless love, care, patience, and tremendous support accompanied me through all my life, especially the last four difficult years. Now that I am finally graduating, I wish to turn over the diploma to them. They have earned it at least as much as I have.

This research was supported in part by the National Science Foundation under grant No. IIS 9815191, DARPA ETO under contract No. DAAN02-98-C-4025, and DARPA ITO under grant No. DABT63-99-1-0014.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

Speech is a convenient communication tool for human beings. Children start to acquire skills related to speech when they are very young. To most of us, this learning process is so smooth that we do not realize how complex a phenomenon speech perception actually is. However, building a spoken language interface to computers still faces a lot of difficulties after over five decades of research.

The speech-related scientific fields include automatic speech recognition (ASR), natural language understanding (NLU), computational linguistics, and speech synthesis, which together are called *speech and language processing* [41]. Research on speech and language processing has been conducted in a broad range of disciplines from linguistics, psychology, electrical engineering, to computer science, where plenty of related questions are still waiting for answers.

Consider ASR, for example. ASR is one of the few perception-related fields that have demonstrated commercial products. In the past ten years, the speech recognition community has moved from small vocabulary, isolated-word, speaker-

dependent recognition problems towards the problems of very-large-vocabulary, speaker-independent, continuous speech recognition [64] [91] [39] [41]. Commercial speech recognition products started to appear in the market in the mid 90s. Two of the well-known products are ViaVoice from IBM and Dragon NaturallySpeaking from Dragon Systems[1]. Despite of this progress and the increasing customer base, some insightful speech recognition researchers predicted an plateau in future development [70]. It has become very difficult to improve the capability of handling environmental, context, and speaker variations with continuous and large vocabulary with the state-of-the-art methodology. The technical challenges that face this community include robustness (insensitivity to environment change), portability (rapid design, development and deployment for new applications), adaptation (adaptation to changing conditions, e.g. new speakers), spontaneous speech (speech with false starts, hesitation, ungrammatical constructions), and prosody (stress, intonation, and rhythm that convey user intentions) [112].

Some major problems related to ASR can be illustrated by an example of using the dictation system of Dragon NaturallySpeaking version 5. After following the instructions step-by-step to train the system, a speaker read the following text:

> "Likewise proposals have been made to remedy some of the weak points of the symbolic approach, by introducing fuzzy versions of classic calculi, *or importing non-monotonic* techniques for dealing with incomplete information."

[1]The system was originally developed by Dragon Systems and was once acquired by Lernout & Hauspie. Now it belongs to ScanSoft Inc.

The recognized text was:

> "Likewise proposals have been made to remedy some of the weak points of the symbolic approach, by introducing fuzzy versions of classic calculi, *all it in parking noun monotonic* techniques for dealing with incomplete information."

While showing a quite amazing recognition rate, Dragon NaturallySpeaking made very interesting mistakes on "or importing non-monotonic" vs. "all in parking noun monotonic." These are the mistakes related to missing contextual information, which are not very likely to be missed by humans. So our question is: why do current speech recognition systems have harder time incorporating contextual information than human beings do?

1.1 Methodology limitations

In this section, we discuss some of the problems associated with the basic methodology of current speech and language processing techniques, which have posed fundamental limitations to its further progress.

1.1.1 Separation of natural language understanding and speech recognition

Natural language understanding (NLU) and speech recognition (SR) are two major technologies in the field of speech and language processing. There is much evidence

showing that human speech understanding involves a close cooperation between SR and NLU. For example, syntax and/or semantics of natural language may provide the valuable contextual information for identifying the spoken utterances as we have seen in the example of Dragon NaturallySpeaking above. On the other hand, prosodic information (e.g., the changes in pitch and duration of the voice) embedded in acoustic signals gives clues, such as the speaker's intention, to understand a spoken expression.

Research on integrating the NLU and ASR technologies in machine speech understanding started as early as 1970s. Nowadays, simple language models, such as trigram models, have been widely used in SR. The general stochastic techniques used in ASR are also used in in NLU to interpret meaning.

However, a deep level integration is still lack of. NLU research was originally motivated by a desire to understand cognitive processes. The theories have been mainly developed in linguistics and psychology with symbolic approaches. Practical applications have been less important than increasing intuitions about the underlying cognitive process. On the other hand, ASR research focuses on the signal-level attributes of speech. To convert the speech from the acoustic form to the string form, ASR research typically does not go into the semantics of the conveyed information. While the cooperation between NLU and ASR is desirable for both communities, the initial differences between them in motivations, interests, techniques, tools, and criteria for success made the integration very difficult [69].

1.1.2 Symbolic representation

NLU research typically uses symbolic representations. For example, in semantic analysis, to show the meaning of the sentence *I have a car*, one of the well-understood representations is the first order predicate calculus,

$$\exists x, yHaving(x) \wedge Haver(Speaker, x) \wedge HadThing(y, x) \wedge Car(y),$$

where the terms like "Speaker" are symbolic tokens. ASR research is mainly conducted in electrical engineering domain, where numerical representation is more popular. However, symbolic representations such as phonemes and words are also used, especially when compatibility with NLU research is needed in order to use some techniques from NLU, e.g., language models.

There are two major problems with symbolic representations. First, a symbol is indivisible. Once a symbol is defined, its representation limit is decided. Unless we define another symbol, no smaller unit of information may be represented. As we never know what size of an information unit is appropriate in the real world, the dilemma is that a too small unit is inefficient while a too large unit lacks discriminating power. Second, it is difficult for machines to generate new symbols. Once a system is built, its vocabulary of symbols is typically fixed. There have been discussions on dynamically creating new symbols according to some predetermined rules. Unfortunately, these efforts usually result in one of the following two situations: if the rules to create new symbols are too simple, a human designer would rather create symbols manually instead; if the rules are too complicated, a human designer could not afford

to program them. So, it is highly desirable to avoid symbolic representations for the sake of high representation power and system generalization.

1.1.3 Current training methodology

Speech is a communication tool. To successfully exchange information, the parties involved in communication use a lot of contextual information, which is either embedded in the speech environment or has been acquired from previous sensory experiences. We call the procedure of learning directly from sensory experiences, *grounded learning.* Current methodology used to train a speech processing system does not incorporate grounded learning.

Building acoustic or language models starts from collecting a large amount of speech or text data. The speech data are then manually segmented and translated into strings of symbols representing the corresponding linguistic units. This translation procedure is called *transcription*, which is very labor-intensive and requires special expertise. A transcribed speech data set is called a *speech corpus* and is used to train the models. The problem here is that after the above data collection and transcription procedure, a large amount of contextual information contained in the speech environment is discarded. A system trained using a speech corpus, therefore, fails to use any environmental cues to boost its recognition rate. This is a fundamental problem, especially for a dialog system.

Another problem related to this methodology is that the testing data is not guaranteed to "resemble" the training data. The issue of maintaining high performance

in various environments is called robustness. One way to achieve robust high performance is to use a huge training data set in an attempt to cover all possible acoustic variabilities. There is an axiom of speech research saying "*there are no data like more data*" [49]. However, a corpus generation procedure is usually very expensive, which makes it difficult to extend an application system. A more serious problem here is that statistical models trained based on data collected in a wide range of conditions will be so diffused and they will not be able to do a good job in any specific condition. No need to say, there are always new testing data unseen during training.

A lot of efforts have been made to resolve the robustness issue in ASR. Currently, the most promising approach is adaptation, especially online incremental adaptation, as discussed in a nice review by C.H. Lee [50]. Online adaptation has an issue related to data supervision. Supervised adaptation, with transcribed data, usually gives better results than those of unsupervised adaptation. However, it is more realistic to do unsupervised adaptation in practice, which does not need data transcription but requires some form of goodness of fit to verify data. While various methods have been proposed for unsupervised adaptation, they usually involve task-specific heuristic. For example, a telephone number recognition system has knowledge of non-existing area codes. So when it discovers that a recognized digit sequence spoken by a new customer does not exist, the system can modify the models with the criterion of reducing the chance to generate this sequence. The task-specific heuristic is essentially a type of contextual information, which brings us back to the grounded-learning issue we raised in the beginning of this section.

1.2 Research motivation and technical challenges

Human beings have powerful speech recognition and understanding capabilities. These capabilities are acquired gradually along with the acquisition of other complex mental capabilities, such as vision. This procedure is characterized by the following features:

- An autonomous procedure. Although human babies are usually cared for by their parents, their mental functionality development is an internal procedure. Nobody can open their brains and manipulate the information for them. In this sense, it is an autonomous procedure.

- Raw sensory experiences. Like any other biological systems, human beings access the external world through their sensory organs. The sensory experiences form the grounding to acquire knowledge.

- Active interactions with the environment. Human babies actively participate in the interaction with the environment by forming their own internal reflection of the environment and acting according to it [66].

Motivated by the developmental process of humans, we are interested in building a robot that develops its cognitive and behavioral skills through online, real-time interactions with the environment. We call such a robot a *developmental robot*[2]. This line of work is based on the following considerations: (1) Studies on human and animal cognitive development have shown that interactions between a higher animal and its

[2]A more detailed discussion on human development procedures and developmental robots is presented in Chapter 2.

environment is essential for perceptual and cognitive development. (2) To successfully communicate through speech, the parties involved should share certain common knowledge, which is usually embedded in the environment and is grounded upon the parties' sensory experiences. (3) The capability of autonomous online learning frees human engineers from labor-intensive data collection and transcription procedures in developing a traditional speech recognition system, and it may potentially enable robot systems to reach a performance level that is not practical with "spoon-fed-data" learning.

The autonomous cognitive development of a robot potentially makes the realization of machine perception easier and allows a machine to reach higher performance than the traditional methodology [24] [23]. Studies in developmental psychology implies that the new developmental approach has also the potential to address many current difficulties in computer vision and artificial intelligence in general in addition to speech recognition.

However, this new capability also raises a series of new challenges that a traditional speech learning system does not have to deal with. Due to these technical challenges in realizing autonomous developmental learning directly from continuous sensory streams, our current goal is not to demonstrate a performance as good as the one that has been reached by using manual development in constrained domains. Instead, we concentrate on architecture and self-organization schemes that realize developmental learning.

In the work presented here, we address the following challenges.

Automatic generation of internal representation. Internal representation includes, among others, features extracted from perceived signals, subspaces represented by clusters in sensory inputs, inter-connections between nodes, and states consisting of short-history information. For a traditional learning robot, the representation is typically designed by a human programmer for a particular task. The programmer takes the advantage of knowing the task and the environment by programming various constraints into the robot program. For the developmental robot we are interested in, no task-specific information is available until the system has been built, programming has been finished, and the robot has entered the user stage. During the user stage, users will be the trainers and any internal-data-level intervention by human programmers is no longer possible. Interactions between the robot and its environment, including the users, are only through the robot's sensors and effectors. Therefore, a developmental robot must collect data and generate internal representations automatically.

Unsegmented sensory stream. The environment is sensed by a robot as continuous data streams, e.g. the auditory stream from a microphone, visual streams from cameras, and tactile streams from touch sensors. To associate sensory inputs with robot actions, an important issue is to decide the appropriate contexts. For example, meaningful speech units (phonemes, words, phrases, and sentences) have varying temporal lengths. Without relying on a human designer to segment or transcribe sensory inputs, a robot must automatically associate the appropriate contexts with the desired actions. The temporal closeness between the sensory inputs and the actions is of essential importance for sensorimotor association. In addition to this,

another thing that affects the association establishment is the feedback the robot receives from the environment, such as the reward supplied by human trainers (encouraging or discouraging signals) or nonhuman environment (collisions). However, a reward is typically delayed and is often inconsistent in the delivery time.

Learning internal behaviors. Autonomous learning is not effective without developing internal behaviors. By internal behaviors, we mean the perception-invoked actions that are applied to internal effectors, such as attention effectors, action-release effectors, etc. For example, closely related to the issue of unsegmented sensory stream is the need of a selective attention behavior that allows a robot to focus on the part of the input data critical in the current situation. Another example of internal behaviors is the manipulation of internal states. Since internal behaviors are not accessible by the world outside the robot, it is very challenging to enable the robot to develop desired internal behaviors through external interactions via its sensors.

One-instance learning without local minima. Online learning implies that the learning algorithm must be very adaptive while highly reliable, which are two conflicting criteria. A system must be able to learn from as few as a single instance for association yet without getting stuck into local minima. The methods such as hidden Markov models (HMMs) typically need a preprocessing stage to estimate the initial observation probabilities and transition probabilities before the learning algorithm (e.g. Baum-welch algorithm) starts. This is because the Baum-Welch algorithm does not give a good solution starting from a random initial guess. However, an online learning algorithm must not give a bad solution even without a preprocessing stage.

There are many other technical issues, which are beyond the scope of the work reported here, e.g., the development of a high-level value system to efficiently and effectively evaluate robot behavior. Because of these new challenges, at current research stage of autonomous cognitive development, we should not expect the system performance to reach the level of a traditional speech learning system immediately, in either vocabulary size, speaker variation or recognition rate. However, as discussed in [108] and [70], the autonomous audition-related cognitive development provides new dimensions for speech learning researchers to address current difficulties in dealing with speaker variation, environmental noise, and inherent ambiguity in audition without multiple modal sensory inputs.

1.3 Thesis outline

The organization of this thesis is as follows.

Chapter 2 contains background materials on current techniques of ASR and a brief survey of human cognitive development. Developmental robots, a new direction of AI, is discussed along with some early examples of such robots.

Chapter 3 is mainly concerned with establishing the system architecture used in all the experiments of the following chapters. Two of the three major techniques, incremental hierarchical discriminant regression (IHDR) and the unified learning strategy are also discussed.

Chapter 4 presents the third major techniques, an incremental method to compute the principal components for sequentially arriving observations without esti-

mating and maintaining the covariance matrix. The proposed complementary candid incremental principal component analysis (CCIPCA) algorithm is fast in convergence rate and low in the computational complexity, compared to existing methods. A mathematical proof on the convergence of CCIPCA is given in the appendix of this thesis.

Chapter 5 presents a robot system that develops some basic audition-driven behaviors through online real-time interactions with human trainers. We show how the above architecture and three major techniques contribute to resolving some of the challenges of a developmental robot.

Chapter 6 provides an improvement of the cognitive mapping module used in Chapter 5. The resulted robot system is able to do task transfer, i.e., applying what has been learned in one context to new contexts under verbal instructions by human trainers.

Chapter 7 discusses a further enhancement of the cognitive mapping module, which helps a robot to do real-time multimodality learning. With the ability to handle perception of both audition and vision, a robot system conducts simple semantics learning and is able to answer a few verbal questions about presented objects.

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

Chapter 2

Background

In this chapter, we will briefly review some of the areas related to building a robot that develops its audition-related cognitive and behavioral skills through online, real-time interactions with the environment.

2.1 Existing speech recognition methods

Since the work reported in this thesis is highly related to the auditory system of developmental robots, it is helpful to have a brief survey on the existing ASR methods[1].

The task of ASR is to find the most probable word sequence given a sequence of acoustic observations. Shown in Fig. 2.1 is a basic architecture of an ASR system. The elements are as follows:

Raw speech. Speech is typically sampled at a high frequency, e.g., 16KHz over a microphone or 8KHz over a telephone. This yields a sequence of amplitude values

[1] A very good textbook by John Deller et al. [19] has a broad and deep coverage over speech recognition and other speech signal processing techniques.

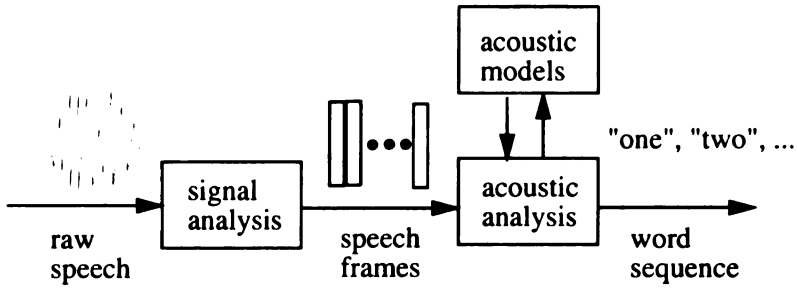


Figure 2.1: Structure of a typical automatic speech recognition system

over time.

Signal analysis. Raw speech should be initially transformed and compressed, in order to simplify subsequent processing. Many signal analysis techniques are available which can extract useful features and compress the data by a factor of ten without losing any important information. Followings are some of the most popular signal analysis conducted in speech recognition.

- Linear Predictive Coding (LPC) yields coefficients of a linear system, which produces the short segment of raw speech signals when driven by certain input excitation sequence [19].

- Perceptual Linear Prediction (PLP) takes the LPC features and modifies them in ways consistent with human hearing. For example, the spectral resolution of human hearing is worse at high frequencies, and the perceived loudness of a sound is related the cube rate of its intensity. So PLP applies various filters to the LPC spectrum and takes the cube root of the feature [41].

- Cepstral analysis represents a short segment of raw speech signals by the slowly varying part of the logarithm of its power spectrum. The slowly varying part characterizes the speech system while the "quickly varying" part is due to the

excitation [19].

Speech frames. The result of signal analysis is a sequence of speech frames, each lasts 5-20 ms and is represented by tens of coefficients. These frames may be augmented by their first and/or second derivatives, providing explicit information about speech dynamics, which typically leads to improved performance.

Acoustic models. In order to analyze the speech frames for their acoustic content, a set of acoustic models of acoustic units, such as phonemes, syllables, and words, need to be constructed during the training session. There are many kinds of acoustic models, varying in representations, granularity, context dependence, and other properties. Some of the popular models will be discussed in the following part of this section.

Acoustic analysis. Acoustic analysis applies the acoustic models over the speech frame sequence and computes the likelihood to decide the recognized words. In reality, depending on the nature of the model, the product of acoustic analysis may be other acoustic units such as phonemes.

Word sequence. The final result is a word sequence — the sentence hypothesis for the utterance. Actually, it is common to return several such sequences. Further constraints, such as language models, will be used to pick up the best one.

2.1.1 Dynamic time warping

Dynamic time warping (DTW) is one of the oldest and most important algorithms in speech recognition [95] [38] [82]. It matches acoustic models with speech frame

sequences by finding the optimal nonlinear alignment.

The simplest way to recognize an isolated word sample is to compare it against a number of stored word templates and determine which is the best match. This goal is complicated by a number of factors. First, different samples of a given word will have somewhat different duration. This problem can be eliminated by simply normalizing the templates and the unknown speech so that they all have an equal duration. However, another problem is that the rate of speech may not be constant throughout the word; in other words, the optimal alignment between a template and the speech sample may be nonlinear. DTW is an efficient method for finding such an alignment.

Consider a matrix D, where $D(x, y)$ is the Euclidean distance between the x-th input speech frame and the y-th frame of the reference template. Starting from $(0, 0)$ in D, every movement from one entry in D to an adjacent entry is called a transition. Denote $C(x, y)$ as the cumulative score along an optimal alignment path that originates from $(0, 0)$ leads to (x, y). Then we have,

$$C(x, y) = \min\{C(x-1, y), C(x-1, y-1), C(x, y-1) + D(x, y)\}$$

It is now clear that DTW is an instance of the general class of algorithms known as dynamic programming. Its time and space complexity is merely linear in the duration of the speech sample and the vocabulary size. By keeping track of backpointers, an optimal alignment path can be computed for each reference word template. The path with the lowest cumulative score is considered to be the best match for the unknown

speech frame sequence. There are many variations on DTW algorithms. For example, it is common to set local path constraints, e.g., weighing the transitions in various ways [82].

2.1.2 Artificial neural networks

Artificial neural network (ANN) was introduced in 1960s and was reintroduced in early 1980s as an powerful tool for pattern recognition problems. ANN has been used by many researchers in speech recognition. Some excellent results have been achieved in such basic tasks as voiced/unvoiced discrimination [99], phoneme recognition [97], and spoken-digit recognition [26].

There are two basic approaches to speech recognition using neural networks: static and dynamic. In static approaches, the neural network sees all of the input speech at once, and makes a single decision. By contrast, in dynamic approaches, the neural network sees only a small window of the speech, and this window slides over the input speech while the network makes a series of local decisions, which have to be integrated into a global decision at a later time.

Huang [33] demonstrated that static approach of neural networks can form complex decision surface from speech data. They applied a multi-layer perceptron with only two inputs, 50 hidden units, and 10 outputs, to Peterson and Barney's collection of vowels produced by men, women and children, using the first two formants of the vowels as the input speech representation. After 50,000 iterations of training, the network produced the decision regions nearly optimal, resembling the decision regions

that would be drawn by hand. The classification accuracy achieved was comparable to that of more conventional algorithms, such as k-nearest neighbor and Gaussian classification.

A problem with static scheme is that the input, if it includes context, is a fixed-time window without any possibility for alignment or time-stretching. "Time Delay Neural Networks" (TDNNs) solve a part of the problem by tying the weights from every frame of input to the hidden layer. The same can be done with a range of hidden-layer units. Sections of the network can be made to perform the same computation, which makes the network time-shift invariant [97]. This form of weight sharing and this imposition of constraints on the connectivity can be seen as the incorporation of prior knowledge into the ANN design.

Another dynamic approach to avoid the problem of fixed time windows is introducing cycles into the network graph. This lets the net keep information about past inputs for amount of time that is not fixed a priori, but that depends on weights and on the input data. Variations of the BP algorithm have been developed to train this kind of recurrent ANNs [76].

In addition to feed-forward or recurrent nets trained by error back propagation, there are different ANN architecture for classification tasks. Two approaches that can be mentioned briefly are learning vector quantization" (LVQ) [44], which is an algorithm to train a two-layer network for optimum discrimination between pattern classes, and radial basis function(RBF) networks, which is based on a very similar idea as the nearest neighbor method [56].

Furthermore, there are ANN architectures that are appropriate for producing new

kind of representations of complex data, for example, speech data. An example of such networks is self organizing map(SOM) [45], [46]. This kind of networks organize themselves automatically by so-called competitive learning, according to the structure of the input data (unsupervised training). Incoming speech can be mapped as the path of best-responding cells of the SOM. Such a mapping can be used as a basis of speech recognition [93].

2.1.3 Hidden Markov model

In current speech recognition techniques, hidden Markov model (HMM) is the most widely used method [34] [71] [57] [41]. General speaking, it models speech with a doubly stochastic process [73] [72].

The first stochastic process models short speech segments. Essentially, speech is a non-stationary process. However, it is shown that speech segments within a short time, typically under 20ms, can be effectively modeled as the output of a linear time-invariant all-pole filter excited by appropriate sources. So we may treat these short speech segments as stationary units and represent them by a set of filter parameters.

The second stochastic process models the piecewise stationary process over a sequence of speech frames. Speech is produced by a physical system – the human vocal system, which does not conduct dramatic changes. So the parameters of the speech frames are usually held fairly steadily over a certain period of time and evolves to another set of parameters gradually. We may identify these quasi-stationary periods and characterize how one such period evolves to the next. In HMM, these steady pe-

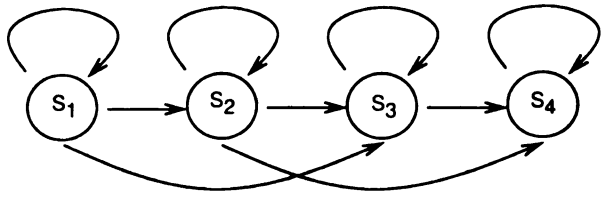


Figure 2.2: A typical HMM topology for phonemes.

riods are termed as states while the shifting procedures between periods are modeled as transition probabilities. Notice that the state in HMM is a real entry but is not accessible explicitly. That is where the term "hidden" comes from.

A typical discrete HMM can be represented by a stochastic finite automaton $A = \{S, A, B, \pi\}$ with a set of stationary states S, a transition probability matrix A, a emission probability matrix B and a set of initial state probabilities π. Usually, speech recognition systems use strictly left to right HMMs to model words, syllables, phonemes or sub-phonetic units. Often, words are modeled as a sequence of phonemes, which in turn are modeled as a sequence of HMM states. Fig. 2.2 shows the topology of a typical phoneme HMM.

There are three problems arising, when using HMMs to model speech: (1) Evaluation: What is the probability that a given HMM generated a given sequence of speech frames; (2) Decoding; Given a sequence of speech frames and a HMM, what is the most likely sequence of states through the HMM that lead to the generation of the speech frames; (3) Parameter estimation: Given a HMM and a set of speech frames to be modeled by this HMM, how can we adapt the parameters (emission and transition probability distribution) of the model to maximize the likelihood of generation.

All of the above three problems have very efficient solutions in form of special cases of dynamic programming algorithms. For instance, the evaluation problem occurs in isolated word recognition where we want to score different word HMMs according to their likelihood. It can be solved using the Forward algorithm. The decoding problems occurs in continuous speech recognition where we are seeking the most probable path through a very large HMM consisting of all possible sequences of basic sound units. Once we found this path, we can derive the most probable sequence of phonemes or words. The decoding problem can be solved using the Viterbi algorithm. The last problem, also called the training problem, can solved by the Forward/backward or Baum/Welsch algorithm, which is essentially a version of the Expectation-Maximization (EM) algorithm.

In the case of left-right HMMs with a constant small number of transitions in each state, all three algorithms have a computational complexity of only $O(NT)$, where N is the number of states in the HMM and T is the number of speech frames.

2.1.4 ANN-HMM hybrid methods

As presented above, ANN are very suitable for pattern recognition tasks where the variety and separability of the patterns can be very complicated. However, ANN has difficulty in modeling temporary structure. This is why ANN can have perfect performance in tasks such as isolated words or phonemes with a limited vocabulary but not in tasks with longer context and larger dynamic variation. On the contrary, HMM has proved to be an excellent tool for handling dynamic problems. Considering

these facts, it is natural to think of ways to combining these two techniques.

There are two different ways to combine ANN and HMM. One is to use ANN as the generator of the posterior state probabilities for continuous HMMs. This means we need to train a neural network which has as many output nodes as there are HMM states. Then we can compute the likelihood by dividing the network outputs by the prior state probabilities.

To train such a neural network, we usually need to generate target vector for each speech frame. But before we recognize the word or sentence, we do not have any idea of the frames' target vectors. An iterative procedure based on the EM algorithm is used as we did in the training problem of HMM. First, labels are generated arbitrarily. In practice, to speed up the iteration process, an existing HMM recognizer is used to find out the most probable sequence of states through the HMM by forcing Viterbi alignment, given the sequence of speech frames. Then, with the generated state labels for each frame of the utterance, a neural network is trained, using the performance on an independent cross validation set as a measurement of generalization. With the emission probabilities given by this trained neural network, the Viterbi alignment is forced again to get new labels. This procedure continues until some performance measure has been achieved. Alternatively, the Forward-Backward instead of the Viterbi alignment algorithm may be used. [7]

The second way to combine ANN and HMM is using the ANN as the vector quantizer for discrete HMMs. Similar to the case of using ANN as the posterior state probability generator, a supervised signal has to be presented during training. Usually, the vector quantizer is trained on phonetically labeled speech data. It is

applicable when the codebook size is not large. But if it is not the case, the training procedure could be intolerable. One way to handle this problem is given by [75]. The basic idea is training the ANN with the maximum mutual information(MMI) principle by relating the label stream Y resulted from a vector quantizer and the word string W representing the words of the corresponding utterance.

2.2 Brief survey on cognitive development

Cognitive development concerns the process of intellectual maturation from infancy through childhood to adolescence. Despite that there is not yet an unified theory with coherence principles to diagnose the general problems, researchers in cognitive development tends to classify the variety of developmental theories into theories from four perspectives, namely, the nativist, the environmental-learning, the constructivist, and the cultural-context ones [74] [16].

2.2.1 Nativist perspective

According to the nativist perspective, the major cause of cognitive development is maturation. The development is viewed as a largely automatic process. In other words, the basic sequence of changes that characterizes the development comes from inside the organism as a consequence of the genes the organism inherits. The environment only works as a "trigger" or "fine-tuner", but does not substantially alters the development's course and final form.

There are two major categories under the nativist perspective. Some nativists

believe in hard-wired knowledge while others emphasize the genetic constraint. The latter viewpoint is usually called "soft nativism."

One of the famous examples of the first category comes from Noam Chomsky's theory of language. He points out that most of what a child hears in everyday language experience is highly diverse, faulty, and piecemeal and thus he rejects the idea that the child learns grammar by imitating sentences heard. Chomsky concludes that all child's abilities in grammar must be inborn, in the form of a set of innate rules coded in their genes and biologically inherited [74].

According to soft nativism, knowledge is not innate, but the overall structure is. The structure determines the kinds of information that can be received, the kinds of problems that can be solved and the kind of representations that can subsequently be stored [22].

2.2.2 Environmental-learning perspective

Researchers with the environmental-learning perspective believe that the major cause of the knowledge acquisition is the sensory experience.

One of the interesting experiments supports this perspective was done on cat[2]. In the experiments reported in [31], some kittens were raised in an environment that allowed them to see only horizontal lines for several months. It was observed that they had weaker capability to detect vertical lines than normally raised cats. Further studies showed that certain cells in a cat's brain respond best to horizontal lines,

[2]Although cognitive development is an area studying human mind, experiments on animal subjects are sometimes conducted to avoid inappropriate effects on human subjects.

whereas other cells respond best to vertical lines. Both kinds of cells are present in large numbers in new born kittens that do not have any visual experience. When kittens are raised in an environment with no vertical lines, they produce far fewer of the nerve that respond to vertical lines than normal kittens do. This shows that kittens' internal representation of the world, which is one important respect of the knowledge, is largely influenced by the environmental effects.

It is observed that biological systems have some basic learning mechanisms. The environment acts through these learning mechanisms, and subsequently shapes the system development. The learning mechanisms that are believed to operate throughout the whole development process include nonassociative learning, classical conditioning and instrumental conditioning [20].

In nonassociative learning, a subject learns about the properties of a single stimulus by being exposed to it repeatedly. Habituation learning is an example of nonassociative learning. In habituation learning, a subject gradually decreases its attention paid to a repeated stimulus, for example a view of a toy. On the contrary, to pay attention again when some aspect of the stimulus has been changed (e.g., a new toy) is called dishabituation.

Classical conditioning is a process by which an organism learns which events in its environment go with each other. In the famous experiments done by Pavlov, after several experiences of hearing a tone just before food was placed in its mouth, a dog would begin to salivate in response to the tone, before it received any food. Classical conditioning is a process by which previously existing behaviors come to be elicited by new stimuli.

Instrumental conditioning is a process following such a basic idea, changes in a behavior occur as a result of the positive or negative consequences the behavior produces. In other words, organisms tend to repeat behaviors that lead to rewards and give up behaviors that lead to punishment. The behaviors of animals in circuses are good examples showing the effects of instrumental conditioning.

The major impact of the environmental-learning perspective is that it emphasizes a new cause of cognitive development, the environment. While those learning mechanisms could be innate, the concepts and the behaviors are believed to be acquired

2.2.3 Constructivist perspective

The difference between the nativist perspective and the environmental-learning perspective reflects the long-debated nature-nurture question: What controls the development process, heredity or experience?

In the constructivist perspective, nature and nurture are believed to be equally necessary. Moreover, constructivist theories attribute to an agent[3] a greater role in shaping its own development. In constructivists' opinions, the mind is an agent that actively structures the observation by forming its own internal representations and creating new functional relations between its parts.

Jean Piaget, one of the most prominent constructivists, proposes that humans are born with some initial schemata, where a schema is defined as a mental structure that

[3]By an agent here, we mean anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

provides an organism with a model for actions in similar or analogous circumstances. The initial schemata are strengthened and transformed into new schemata through two processes, "assimilation" and "accommodation." Assimilation is a process by which children incorporate new experiences into their existing schemata. Accommodation is a process by which children modify their existing schema in order to adapt to new experiences.

A classical experiment supporting the constructivist perspective was done by Richard Held and Alan Hein [29] with kittens raised from birth in total darkness. When the kittens were old enough to walk, they were placed in pair in an apparatus called "kitten carousel" as shown in Fig. 2.3. One kitten was harnessed to pull the carousel while the other was carried in the gondola. 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[4]. It was observed that the kittens harnessed in the carousal hesitated to land on the deep side of the cliff while those kittens in gondola did not try to avoid it. These two groups of kittens had roughly the same visual experiences but different interactions with the world. This experiment suggests that active interactions between the agent and the environment make a big difference in learning the interpretation of the environment. Similar phenomena were observed on human babies too[6].

[4]Visual cliff is a research apparatus with a transparent table-top. It is divided into two regions: (1) a "shallow side" in which the checkerboard pattern is flush against the glass surface, (2) a "deep side" in which the checkerboard pattern is much closer to the floor and far away from the glass surface.

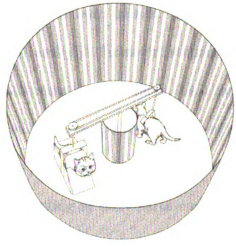


Figure 2.3: Kitten carousal.

2.2.4 Cultural-context perspective

Like the constructivist perspective, the cultural-context perspective emphasizes that the development involves the active engagement of the individual. These two perspectives are different in that the cultural-context perspective assumes that both the baby and the caretaker are active agents, and views the development as a "co-constructive" process. In addition to give more space for variability of individuality, this idea stresses the role of active environment in establishing and/or accelerating cognitive development.

As suggested by its name, the cultural-context perspective insists that cognitive development is shaped by the specific cultural-historical context a child lives in. The culture includes the designs for living, speaking, beliefs, values and customs. Adults, who have already inherited the culture from the last generation, influence the development of children through their behaviors.

There are a lot of evidence showing that the knowledge of the society determines

a lot of the behaviors of the new born babies. Japanese adults have difficulty in distinguishing phonemes /r/ and /l/. However, studies show that Japanese babies actually can perceive the difference between them. This capability to make phonemic distinctions begins to diminish at about 6 to 8 months of age [16]. Another example is that children growing up among the Oksapmin of New Guinea establish a very different counting system – counting by body parts – while they seem to have the same ability to grasp basic number concepts as those growing up in Paris or Pittsburg [58].

2.2.5 Lack of computational study

Although developmental psychology is a well-studied subject, past studies in the field have been largely qualitative in nature. This situation is probably due to the difficulties of approaching such an complex subject in exact computational terms. Not until recently, did some researchers start to address cognitive development in a computational perspective [100] [22] [3]. These recent efforts model cognitive and behavioral development through computational processes, which allows little ambiguity and the resulted model is verifiable through quantitative experiments. This direction may help to deepen our understanding of cognition, behavior and other related issues.

2.2.6 Summary

Except for the nativist perspective, the other three perspectives give a lot of credits to the effect of the environment in the course of cognitive development. While we still do not know the grand truth of where the intelligence comes from, the experiments

done on human or animal subjects seem convincing to us that the intelligence has a lot to do with interactions between an agent and its environment. This impression is the starting point of our intention of building developmental robot, as we will see in the next section.

2.3 Developmental robots

2.3.1 Motivation

With the invention of the computer, people started to raise questions such as "Can we emulate human intelligence on a machine?" A new field appeared in the middle 20th century, whose name, artificial intelligence (AI), was officially coined in a two-month workshop at Dartmouth in the summer of 1956. After 50 years, AI now covers areas ranging from game playing, logical inference, theorem proving, planning, medical diagnosis, to vision, natural language understanding, and robotics [81].

In general, AI problems can be viewed uniformly as the development of intelligent agents. While it is very obvious to imagine an autonomous robot with a real body as an agent, pure softwares, such as chess player "deep blue," are also agents. "Deep blue" perceives the coded chess configuration and acts by generating the next move. Various agents have been proposed in last 50 years, ranging from reflex agents to fully deliberative, knowledge-based agents.

Knowledge-based systems emerged in 1970s. Since Feigenbaum's MYCIN [11], these systems proved to be very successful in areas such as medical diagnosis. To

set up such a system, it typically needs a domain engineer, who is responsible for interviewing the experts in the related area and building the domain knowledge for a computer. Typically, the domain engineer finds out the rules followed by the experts to solve the problem, and designs data structures and algorithms so that a computer can run these rules. As these rules are usually not explicitly expressed by the experts, the domain engineer needs to be extremely smart to extract them out. The procedure to establish domain knowledge requires a lot of human efforts and expertise.

In contrast with the manual knowledge-feeding approach, the learning-based approach enables machines to learn. Among the earliest learn-based systems, Samuel's checker player [84] was so successful that it was able to compete on equal terms in some very strong human tournaments. Following this approach, a human engineer first needs to identify a task by specifying the input, the output and the performance evaluation procedure. Then he will decide a representation (model) and figure out a learning algorithm to train the parameters of the model. Many data are typically collected to train the system before it is released.

While learning gives a system much power to establish new knowledge, there are some problems with the current way of building a learning system. First, a learning-based system is usually designed to solve a specific task. However, intelligence is not a capability to do a single task. Moreover, there are tasks that are not separable from others, e.g., language translation is close related to culture background in addition to syntax of the languages. Second, in many current learning-based systems, the training process is off-line, i.e., training data need to be collected and, usually, edited before the training procedure starts. As human-collected training data will never cover all

the situations in the real world, the off-line learning will never be completed.

The behavior-based approach, originally proposed in 1980s [10], tries to escape from some constraints of the knowledge-based approach by avoiding modeling the world. The design of a behavior-based system starts from 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 some sense, the representation of the world is expressed through the basic behaviors, which are distributed, fragmented and even allowed to be inconsistent. One of the limitation of the behavior-based approach lies in the fact that the behaviors are designed manually in advance. It seems now clear that humans are unable to adequately describe and decompose complex behaviors. As a result, for complicate systems, such as COG, a human-shaped robot in MIT [9], the design of behaviors becomes a serious bottleneck.

From above description, we can see that existing AI systems 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 responsibility of a human engineer to tackle the task. In this sense, this paradigm may be viewed as "... solve a hard problem, you almost

have to know the answer already" [81].

By comparing the tendency in intelligent agent development to the perspectives in cognitive development, we can see surprising similarity between this major AI paradigm and the nativist viewpoint. We seem to believe that the agent should know everything by which it needs for "survival" before being "born," just as nativists believe that everything necessary has been stored in genes. The agent development, thus, reduces to a procedure of coding and feeding knowledge to the agent by the designer. Even in the cases of using learning-based method, the knowledge of how to learn and what to learn is decided by the human designer in advance according to a specific task.

However, thinking about the other perspectives in cognitive development, we seems missing some of the discoveries from our biological counterpart. R. Brooks, one of the founders of behavior-based systems, has a great insight into this problem. He once pointed out that intelligence is not limited to just computational engine but also comes from the situation in the environment to which the robot is a part of [8]. Knowledge does not necessarily need to be born possessed by the agent, just as what happened in the kitten carousal example: a kitten is not born knowing the danger of visual cliff. It figures it out by interacting with the environment.

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

A developmental robot is one that learns and practices autonomously in the real physical world by interacting with the environment through sensors and effectors, probably under human supervision. This is an important direction as discussed in a recent article in *Science [24]*.

A developmental robot follows a new autonomous mental development (AMD) paradigm [23],

1. Design a body: A human designer designs a robot body (the sensors, the effectors and the computational resources) according to the general ecological condition in which the robot will work (e.g., on-land or underwater);

2. Design a developmental program: A human programmer designs a task-nonspecific developmental program for the robot.

3. Birth: A human operator turns on the robot whose computer runs the development program.

4. Develop mind: Humans mentally "raise" the developmental robot by interacting with it. The robot develops its mental skills through real-time, online interactions with the environment which includes humans (e.g., let them attend special lessons). Human trainers teach robots through verbal, gestural or written commands in much the same way as parents teach their children. New skills and concepts are autonomously learned by the robots everyday.

While the developmental program is the central part of a developmental robot, the step of "develop mind" is the key to this new paradigm. First, in this step, the robot generates the representation autonomously according to the task it encounters instead of deciding them in advance. Second, what the robot does is determined in this step by the experiences accumulated in the real world. In contrast to the traditional paradigm, the knowledge is no longer pre-programmed and, more importantly, learning happens after the system is released for final usage instead of in the process of building the system. As we mentioned above, intelligence is not a capability to solve a single task. Only with the factor of learning after "born" may we achieve a possibility of the task-nonspecific solution.

The AMD paradigm borrows ideas from the environmental-learning, constructivist, and culture-context perspectives. First of all, the interaction with the environment is considered essential for behavior and mental development, which is the major point shared by the three perspectives. Secondly, it is emphasized that, for a developmental robot, this interactive process has to be autonomous. After it is "born," the robot is on its own to generate the representation of the world and the behaviors to conduct, instead of letting human designer to program for it. In other words, the robot must be actively engaged in this interaction, as stressed by the constructivist perspective. Thirdly, this paradigm involves a lot of participation of human trainers (not human designers). These human trainers act as the active environment of the robot and the source of knowledge, whose necessity is emphasized by the culture-context perspective. For example, the robot will learn to response to English if it is "raised" in an English-speaking environment while learning Chinese in

a Chinese-speaking environment.

2.3.3 Eight requirements of AMD

Practical AMD is technically very challenging: The following eight requirements are all necessary for practical AMD.

1. Environmental openness: Due to the task-nonspecificity, AMD must deal with unknown and uncontrolled environments, including various human environments. Although a human child cannot do everything that a human adult can do, he is exposed to nearly the full complexity of a human environment early in his life.

2. High-dimensional sensors: The dimension of a sensor is the number of scaler values per unit time. AMD must directly deal 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). Symbolic input is allowed as a clean (hand processed) but simpler sensing modality.

3. Completeness in using sensory information. Due to the environmental openness and task nonspecificity, it is not desirable for a developmental program to discard, at the program design stage, sensory information that may be useful

for some 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. Offline 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). This speed must be maintained even when a full (very large but finite) physical "machine brain size" is used. It must handle one-instance learning: learning from one instance of experience. Time consuming iterations must be avoided.

6. Incremental processing: Acquired skills must be used to assist in the acquisition of new skills, as a form of "scaffolding." 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. Muddy tasks: For large perceptual and cognitive tasks, an AMD machine must handle multimodal contexts, large long-term memory and generalization, and

capabilities for increasing maturity, all without catastrophic memory loss.

2.3.4 Early investigations

Lately, more and more researches on robots are motivated by mental development. The early examples of developmental robots include the SAIL (short for Self-organizing, Autonomous, Incremental Learner) robot at Michigan State University [106], the Darwin V robot at the Neurosciences Institute, San Diego [3], and the Illy robot at University of Illinois, Urbana-Champaign [51]. However, among these early research efforts, SAIL is the only one designed to fulfill all the eight requirements of AMD.

The SAIL robot was designed as an engineering testbed for developmental programs that are meant for scaling up to complex cognitive and behavioral capabilities [100] [105] [37]. The SAIL-2 developmental program has been tested for autonomous derivation of architecture and representation through online, real-time development of association (1) between visual stimuli of objects and eye aiming for the objects (object evoked visual attention); (2) between visual stimuli of objects and arm pre-reaching for the object (vision evoked object reaching); (3) between voice stimuli and arm actions (verbal command learning and execution) and (4) between visual stimuli and locomotion effectors (vision-guided navigation). Ongoing investigations include temporal signal processing, more sophisticated attention mechanism, and value system development [106].

The Darwin V robot by Almassy, Sprons, and Edelman was designed to pro-

vide a concrete example to show how the computational weights of neural circuits were determined by the behavioral and environmental interactions of an autonomous device [3]. Darwin V had a group of pre-designed behaviors and sensory patterns. Its visual capability was implemented by blob-pattern and stripe-pattern detectors. Darwin V has been tested for developing behaviors in response to visual stimuli at different positions and orientations (visual invariance learning). It also has been tested for the association of aversive and appetitive stimuli with visual stimuli (value learning). This work is concentrated on the development of high level plasticity of the association between stimuli and behaviors.

The Illy robot project by Levinson, Huang, and coworkers focuses on spoken language acquisition. The paper [51] reported how to teach the Illy robot to act according to short voice commands in different languages. This work paralleled our work of auditory learning by the SAIL robot. However, the speech acquisition system of the Illy robot had some preprogrammed modules such as "loud sound detection" and "end point detection," while the method reported here does not use such modules to reach a fully developmental robot. Our method is potentially applicable to both word and non-word situations, such as natural sound and music.

Another work worth mentioning is done by Roy, Schiele and Pentland in Media Lab at MIT [79] [77]. They developed a multimodal system which automatically learns shape categories and their corresponding spoken names from an input consisting of naturally spoken utterances paired with the visual representations of the objects. This work was motivated by theories of infant cognition and aimed at enabling a robot to learn semantic representations through sensorimotor experiences. However,

the learning process was not done through real-time interactions between an agent and the environment. In their experiments, the visual-auditory signal pairs were prepared *manually* from separately recorded images and speech signals. Moreover, human designers intervened the system's signal processing procedure. The auditory signals were first segmented at utterance boundaries using a recurrent neural network off-line trained by the TIMIT database. The visual representation of an object is a two-dimensional histogram of distances between edge points and relative angles of edges, which limits the representation power of the resulted system to be no more than narrowly-defined shape concepts. This kind of human involvement prevents a system from doing AMD. These limitations need to be overcome before the system can imitate the infant cognition development.

The computational mechanisms of mental development are getting attention from more and more researchers. These investigations are expected to improve our systematic understanding of the wide variety of cognitive and behavioral capabilities in humans and enable autonomous development of these highly complex capabilities by robots and other artificial systems. A Workshop on Development and Learning (WDL), funded by NSF and DARPA, was held at Michigan State University, April 5 -7, 2000. This workshop was attended by about 30 distinguished researchers in neuroscience, developmental psychology, machine learning and robotics. It resulted in a whitepaper [23] to provide the US government about this new direction. The success of WDL leads to a regularly scheduled conference, International Conference on Development and Learning (ICDL), which was first held at MIT, June 12-15, 2002 [1].

Chapter 3

A Developmental System

In this chapter, we first propose an agent model and present a basic system architecture that implements this model. Then we discuss two of the three major techniques, *incremental hierarchical discriminant regression* (IHDR) and the unified learning strategy. The third major technique, complementary candid incremental PCA, will be discussed in Chapter 4. In the end of this chapter, we will briefly present the hardware architecture of the validation platform, the SAIL robot.

3.1 An agent model

To handle numerical representations of the complex physical world, we propose an agent model for a robot, starting with the following three definitions.

Definition 3.1.1 *The history at time t is defined as $h(t) = \{(x(\tau), a(\tau))|\tau = 0, 1, 2, ..., t\}$, a realization of the random process from the "birth" time $t = 0$, where $x(\tau)$ is the numerical sensory vector acquired at time τ, and $a(\tau)$ is the numerical*

control vector of effectors at time τ.

The history includes all information available to the agent from the "birth" time, such as the auditory sensations and the status of the agent's own actions.

Definition 3.1.2 *The last context at time t is the recent part of history, $l(t) = \{(x(t), a(t))|\tau = t - b(t), ..., t-1, t\}$.*

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

Definition 3.1.3 *A state $s(t)$ is the internal representation of the context in an agent's internal information environment.*

According to the above definitions, an agent views the world as a general random process, called history. 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:

$$s(t) = f(s(t-1), l(t)). \tag{3.1}$$

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

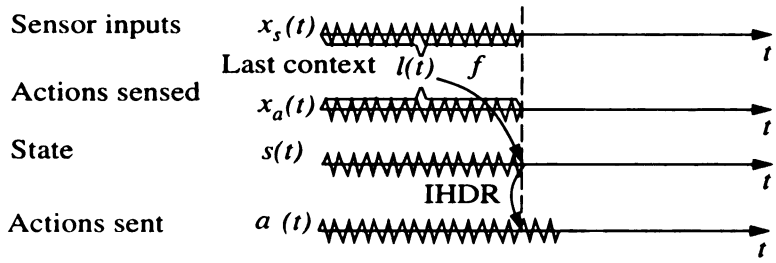


Figure 3.1: A temporal illustration of the agent model. The last context includes information from sensors and the status of the agent's own actions. Depending on both the last context and the last state, the mapping f returns a new state, associated with a list of actions. The action with the highest primed value is selected, which contains the effector control signals.

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

$$P(s(t) = s|s(t-1) = s', l(t) = l \tag{3.2}$$

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

$$s^*(t) = \arg\max_{s \in S} P(s(t) = s|s(t-1) = s', l(t) = l), \tag{3.3}$$

where S is the set of all possible states, which is infinite. In a practical implementation, we approximate the set S by an finite number of prototype states. These prototype states are incrementally merged cluster centers of many experienced state vectors. In practice, the maximization in Eq. (3.3) is too computationally expensive. We use an IHDR tree[1] to approximate it, taking advantage of the numerical vector representation of $s(t)$ and $l(t)$.

[1]See Section 3.3.

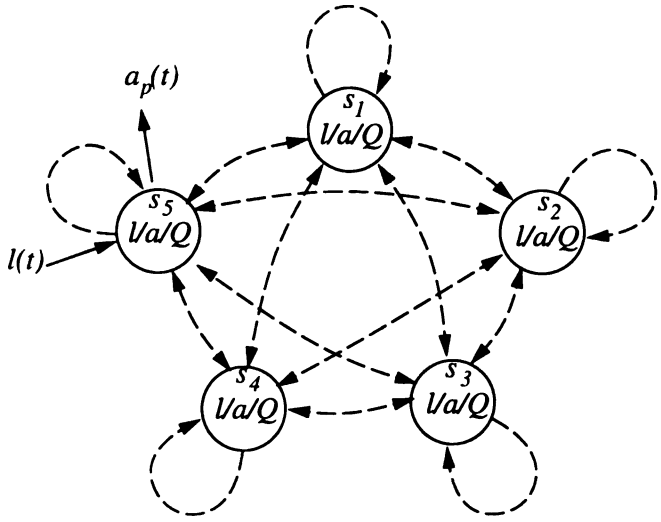


Figure 3.2: Observation-driven Markov decision process with dynamically generated states, context-driven state representation, and value-based action generation. Note: the state is not symbolic. It is a vector in a high dimensional space.

The overt behavior of the agent is determined through a value system with respect to the states. To evaluate the value of each action a at each state s, a Q-value function $q(s, a)$ realizes a mapping $q : S \times A \rightarrow Q$, where A is the action space and Q is a scalar representing the estimated value. The innate value system is such that the agent executes the action that has the highest estimated Q value. Fig. 3.1 gives a temporal illustration of the agent model.

The above way of modeling a decision process, as shown in Fig. 3.2, is called *Observation-Driven Markov model* (ODMM) [17] [110]. Different from the traditional MDPs and HMMs, all the states here are numerical instead of symbolic. They are generated "on-the-fly." Further, the state here represents the agent's internal context, instead of the state of a world event as in all existing non-developmental agents.

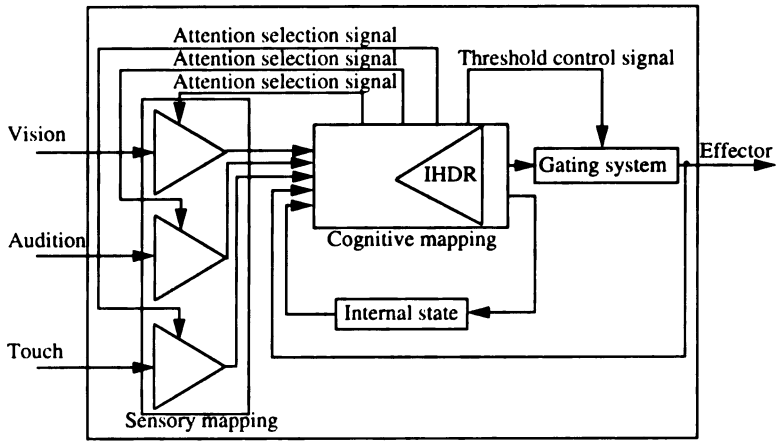


Figure 3.3: Basic architecture of the SAIL robot.

3.2 System architecture

The architecture that implements the above agent model is shown in Fig. 3.3.

Inputs from each sensor first enter a module called *sensory mapping*, which consists of a bank of developed filters. As shown in Fig. 3.4, considering the spatial and temporal expanse of the sensory inputs, we treat a sensor as a window moving in a *spatiotemporal space* with only the information inside the window getting through. We divide such a window into overlapped regions of different sizes, named *receptive fields*, each of which corresponds to an artificial neural column. These neural columns are organized in a hierarchical manner. The output from multiple lower-layer neural columns is the input to the neural columns at the next higher layer. Thus, the latter ones have larger receptive fields. If the multiple lower-layer neural columns spreads along the spatial dimension, a larger receptive field stands for a wider field of view. If the multiple lower-layer neural columns spreads along the temporal dimension, a larger receptive field stands for a longer period of context.

A neural column has two roles, developing a filter and computing the response

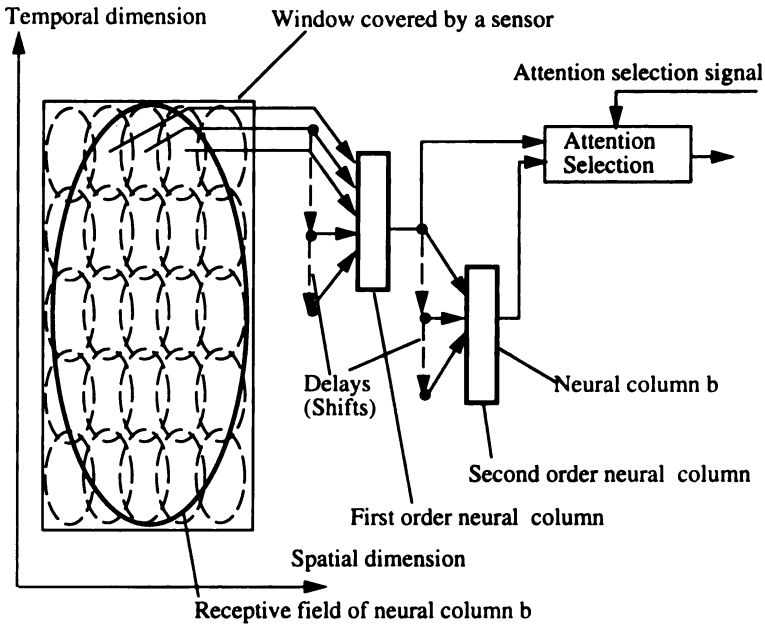


Figure 3.4: Detailed architecture of the sensory mapping module.

using the filter updated so far. The role of filter development is carried out mainly by a technique called *incremental principal component analysis* (IPCA) to compute the principal directions, along which the data has the largest variance. The response is computed by the inner product between the input vector and the principal component vectors. The outputs from the neural columns are fed into the next module, *cognitive mapping*, after being selected by the control signals from the cognitive mapping. This selection is an internal behavior that enables the selective attention mechanism to handle the unsegmented sensory stream.

The cognitive mapping module is the central part of the system. It is responsible for learning the association between the sensory information from the environment and the behaviors that the system is supposed to produce. The behaviors can be both external and internal. External behaviors correspond to generating control signals for external effectors, such as a joint motor of a robot arm, or any peripherals that

the system is equipped with to act on the environment. Internal effectors include the above-mentioned attention selection effector in the sensory mapping module, the effector that manipulates internal states, and the threshold control effector in the gating system in Fig. 3.3.

Mathematically, we have what is called observation-driven state transition function f,

$$f : S \times L \rightarrow S,$$

and the action generation function g,

$$g : S \times L \rightarrow 2^A,$$

where S is the (local) state space, L is the context space, A is the output action space, and 2^A denotes the power set of A (all the possible subsets of A). In other words, the cognitive mapping accepts the last state $s(t-1)$ and the current context $l(t)$ to generate a set of action candidates. The value system implements an action selection function v,

$$v : 2^A \rightarrow A,$$

according to the value of every action in the candidate action set. Typically, v selects the candidate with the highest expected value.

While the state transition function f is pre-defined in current implementation, the action generation function g is a learned mapping. Specifically, mapping f is done by maintaining the state as a first-in first-out queue. At every time instance, the state

is updated by replacing its oldest part with $l(t)$. In the real world, the number of all possible contexts is infinite. However, the resource of an agent is finite and so is the number of state prototypes in the memory. We have to resolve the conflict that $S \times L$ is an unbounded space while S is a bounded space. To do this, we use the method of IHDR [35] [104]. In some sense, an IHDR tree behaves as a vector quantizor (VQ). In effect, the IHDR tree does much more than a traditional VQ by enforcing a discriminant regression mechanism. It derives the features that are most relevant to the outputs and disregards the irrelevant ones. It uses a tree structure to find the best matching input prototype in a fast logarithmic time for real-time speed.

A simplest way of implementing the cognitive mapping uses one IHDR tree, which is discussed in Chapter 5. To enable the robot to handle more complicated perception and behavior, we have continuously augmented the cognitive mapping module (Fig. 3.5 and 3.6). Detailed discussions are given in Chapter 6 and Chapter 7. As one will find out, the enhanced modules are built one upon the other. A good property of such a spectrum of designs is that later modules keep all the functions of early modules. Therefore, the final system is not task specific, which is an important requirement of a developmental robot.

The gating system is a part of the value system too. It evaluates whether the intended action accumulates sufficient thrust to be issued as an actual action. In this way, actions are taken only when a sufficient number of outputs are suggested by the cognitive mapping module through the time.

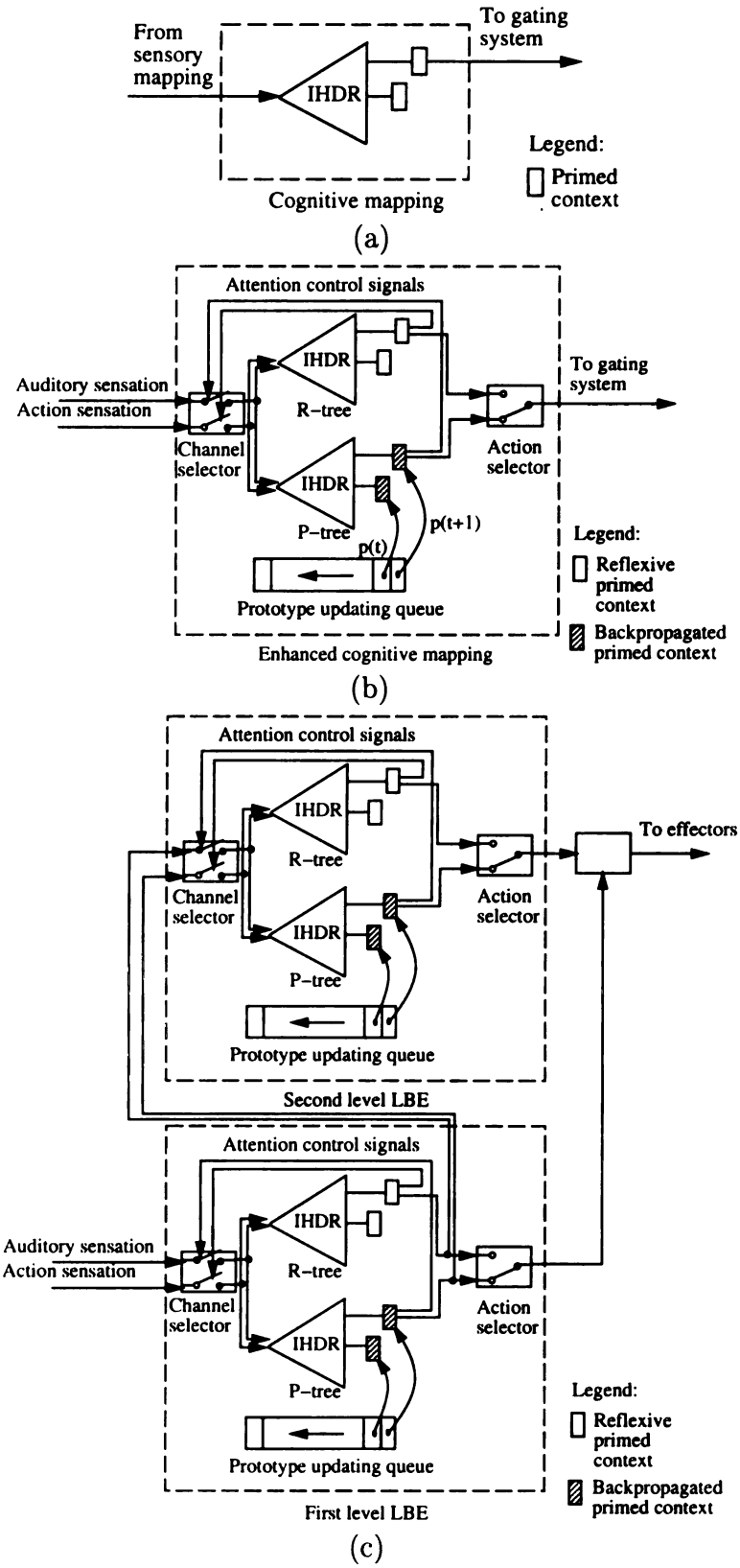


Figure 3.5: The adding-on cognitive mapping modules: (a) the basic module (b) the enhanced module (c) the two-level architecture (d) the semantics learning system architecture

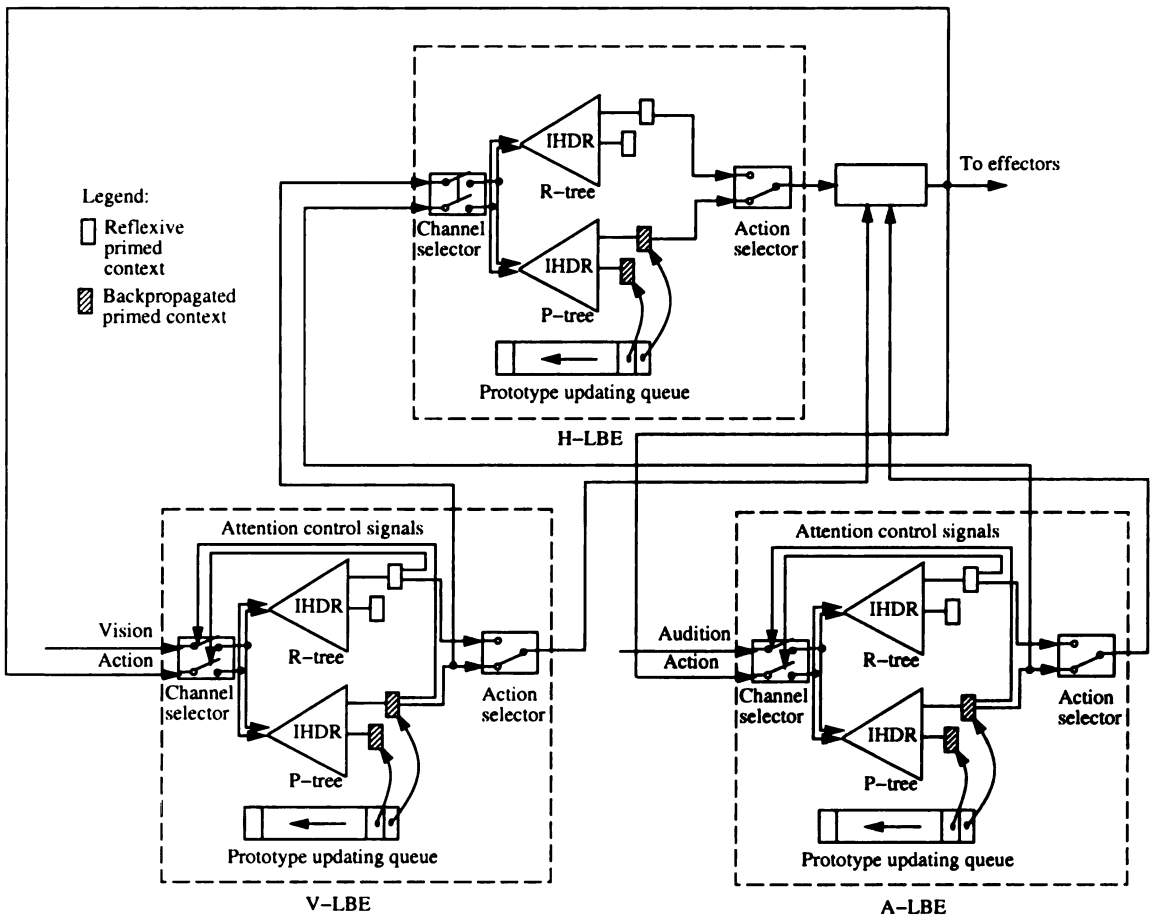


Figure 3.6: The adding-on cognitive mapping modules (Cont'd): the semantics learning system architecture

3.3 Incremental hierarchical discriminant regression (IHDR) in cognitive mapping

Hierarchical discriminant regression is a new hierarchical statistical modeling method introduced by Hwang and Weng [35]. Basically, HDR does clusterings in both input and output space with the former one done on the subspace derived according to the latter one. In this way the discrimant analysis is enforced. Both of the clusterings are done in a coarse-to-fine manner so as to reduce the computational complexity.

3.3.1 Unified analysis of classification and regression

Before getting to the details of HDR, it helps to take an unified view of classification and regression.

Typically, we have two types of problems in discrimant analysis. One is called classification problem with the desired output being a class label. The other one is called regression problem where we have numerical output. Because most of the statistical tools involves a numerical form of computation, we may cast a classification problem into a regression problem.

For a classification problem, the training samples are typically given as a set, $L = \{(x_i, l_k)|i = 1, 2, \ldots, n, k = 1, 2, \ldots, c\}$, where $x_i \in \mathcal{X}$ is an input vector and l_k is the label of x_i. The problem is to determine the class label of any unknown input $x \in \mathcal{X}$. Here are two ways to cast it into a regression problem.

1. If a cost matrix $[c_{ij}]$ is known, where c_{ij} is the cost of confusing class i and j, one can design an c-dimension output vector y_i and y_j for class i and j, respectively, so that $||y_i - y_j||$ is as close to c_{ij} as possible. Here $||\cdot||$ means any appropriate distance metric. In a simple and special case, we may have $0-1$ cost matrix, where $c_{ij} = \delta_{ij}$. The output vector y_i may be designed as a vector with its ith component being one and all others zeros. It is also called canonical mapping.

2. If the cost matrix is not known, we may design the output vector y_i according to input space. That is, set y_i as the sample mean of all x that belong to ith class. This design is equivalent to taking the distance measure in input space as the cost function. It is a good choice when the real cost matrix is unknown.

With either one of above two transformations, the original classification problem on set $L = \{(x_i, l_k)\}$ is converted to a problem on set $L = \{(x_i, y_k)\}$ where y_k is a numerical vector instead of a symbolic label. We are now ready to do double clustering.

3.3.2 Hierarchical double clustering

Consider a general regression problem: approximating a mapping $h : X \rightarrow Y$ from a set of training samples $\{(x_i, y_i)|x_i \in X, y_i \in Y, i = 1, 2, \ldots, n\}$. HDR employs the idea of Linear Discriminant Analysis (LDA) [21] to find such a mapping. LDA uses class information and seeks an optimal linear transformation from the original data space to a new space in which the samples are well separated (or discriminated). In a typical usage of LDA, one needs to estimate the between-class and within-class scatter matrices, which requires class information. However, for a regression problem, y_i is a numerical vector and there are very few samples sharing a single y_i. In other words, there are very few samples in each class. Therefore, the within-class scatter matrix will be poorly estimated, especially for high-dimensional input data, e.g., a few thousand dimensions in vision problems or a few hundred dimensions in audition problems. HDR handles this issue with a coarse-to-fine way.

First of all, q y-clusters in the output space are generated according to the output part of the training samples using a k-means-like algorithm. The training samples is thus gathered into q super groups. The x-part of the samples is accordingly clustered and we call these clusters *x-clusters* (see Fig. 3.7). Since q is typically a small number,

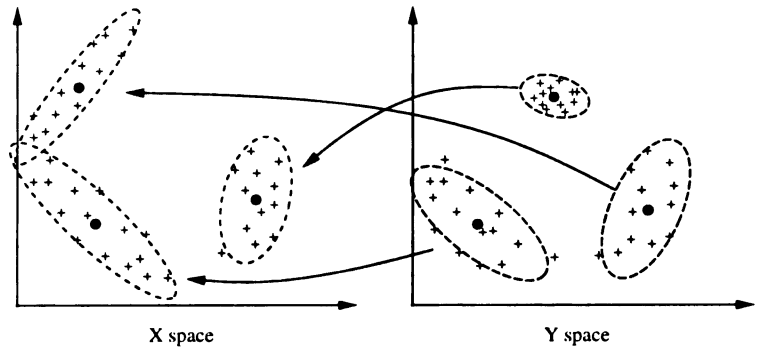


Figure 3.7: Double clustering in both input and output spaces.

e.g., 5, it is effective to estimate the between-class and within-class scatter matrices for these q x-clusters. We call this process a *double clustering* process. The q x-cluster centers span a $(q-1)$-dimensional space, which we call a *discriminant space* because it characterizes the discriminant information among x-clusters. A probability-based distance in the discriminant space is used to determine which x-cluster a test sample belongs to.

As we may immediately realize, while the double-clustering process is done on the whole training set, it can be recursively used for the data in each of the super groups (see Fig. 3.8), which ends up with a tree structure. Moreover, the deeper nodes in the tree, the smaller variance the data has. After the tree is constructed, we may generate the output of a new input x by retrieving the tree. The final output y would be the mean of the y-cluster of the leaf node on the retrieval path.

In summary, HDR has two major advantages. First, it automatically generates representations by finding the discriminant spaces according to the events (data) encountered. These subspaces are optimal for different particular tasks in terms of dicriminant capability. Second, the hierarchical structure not only reduces the

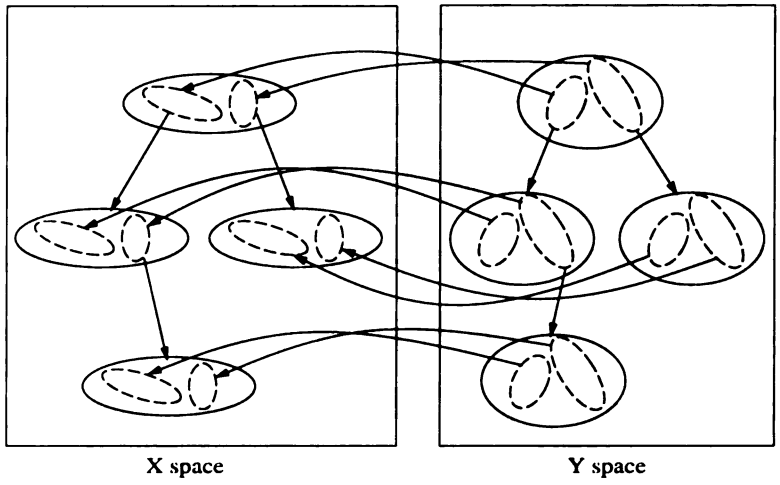


Figure 3.8: Hierarchical discriminant analysis.

searching computational complexity but also enables the system to make use of more samples at shallow levels where the variance of the data is large, which is critical to parameter estimation. At deep levels, the sample variance is tremendously reduced and correct decisions can be made with very few number of samples. Because of the first advantage, HDR has little limitation in terms of representation power and potentially can fit any data. Because of the second advantage, HDR can learn without any iterations. As a result, HDR realizes one-instance learning without local minima, i.e. zero error on training data without iterations.

3.3.3 Incremental hierarchical discriminant regression

HDR has been successfully used in face recognition[35], OCR [36]. In these applications, we typically have a training data set ready for the computation of statistics. In a system like a developmental robot, however, we are conducting online learning, which means the data are only available sequentially. In this case, above HDR method needs to be modified so as to be used incrementally [104].

A major problem of Incremental HDR (IHDR) implementation is that incremental computation of the statistics may result in bad estimation of cluster boundaries. These statistics include the sample means and scatter matrices of both x-clusters and y-clusters. Because the number of samples is small at the beginning of the incremental procedure, the clusters are usually ill-generated. So is the cluster boundary, which will influence later partitions of the tree. In IHDR, we use the amnesic average technique to gradually get rid of the effects of earlier data.

Suppose $\{a_i, i = 1, 2, \ldots, n\}$ are the data coming into the system sequentially. Their sample mean and scatter matrix are given by,

$$\bar{a}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{3.4}$$

$$\Gamma_a^{(n)} = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})(a_i - \bar{a})^T \tag{3.5}$$

Write these equations in an incremental style, we have,

$$\bar{a}^{(n+1)} = \frac{n\bar{a}^{(n)} + a_{n+1}}{n+1} = \frac{n}{n+1}\bar{a}^{(n)} + \frac{1}{n+1}a_{n+1} \tag{3.6}$$

$$\Gamma_a^{(n+1)} = \frac{n}{n+1}\Gamma_a^{(n)} + \frac{1}{n+1}(a_i - \bar{a})(a_i - \bar{a})^T \tag{3.7}$$

From equations 3.4 and 3.5, we know that these incremental estimations of the statistics are actually done by weighting each samples contribution equally $(1/n)$. As time elapses, n increases and a new sample would make less and less difference on the

estimation. Amnesic average is implement in this way,

$$\bar{a}^{(n+1)} = \frac{n-l}{n+1}\bar{a}^{(n)} + \frac{1+l}{n+1}a_{n+1} \tag{3.8}$$

$$\Gamma_a^{(n+1)} = \frac{n-l}{n+1}\Gamma_a^{(n)} + \frac{1+l}{n+1}(a_i - \bar{a})(a_i - \bar{a})^T \tag{3.9}$$

The amnesic parameter, l, changes the relative weighting of old estimation and new samples. By setting l larger than 0, new samples will have more effects and old samples are "forgotten" gradually.

3.4 Learning strategies

In general, there are three learning strategies, supervised learning, reinforcement learning and unsupervised learning. As we believe no system can really evolve without any feedback from the environment, we leave out unsupervised learning and only discuss the first two here.

3.4.1 Supervised learning

Supervised learning is the most extensively studied learning strategies in machine learning. It requires the external environment of the system to provide examples in form of input-output pairs. The system then derives the association or mapping that fits the data according to certain criteria. Under the *stationarity* assumption[2], the

[2]The stationarity assumption is introduced by Valiant [81]. Basically, it says that the training and test sets are drawn randomly from the same population of examples using the same probability

learned mapping would predict the output correctly given a new input.

Supervised learning is widely used in statistical pattern recognition problems. For example, a fingerprint recognition system is to identify a person from the fingerprint image. The provided sample would be the fingerprint image and its identity label. The learning objective is to derive the mapping from the extracted features of the image to the identity label.

Supervised learning is also used in intelligent robot area, where the provided examples are perception-action pairs. A good example for that is ALVINN (Autonomous Land Vehicle In a Neural Network) [67]. The major component of ALVINN is a multilayer perception neural network with 30-by-32 road image as input and 30-dimension vector as output to specify direction. The training data used are collected as pairs of road image and correponding correct driving direction.

While supervised learning is a powerful and successful learning strategy, the requirement of input-output pairs as training data is sometimes demanding to an agent like a real-time robot. ALVINN is lucky because it is easy to collect the training data by having a human drive the vehicle and recording the image/direction pairs. It would be extremely difficult to collect the training data when the behavior of the robot becomes complicated. Actually, even for ALVINN, many synthetic data were generated from real data by rotating the image to simulate the situation when the vehicle needs to be recovered from off course. Moreover, it is more desirable for a robot to learn from its own experience without a knowledgable supervisor around.

---

distribution.

3.4.2 Reinforcement learning

The term of reinforcement learning was first introduced in psychology domain. People observed the phenomena that biological systems, such as dog and human, increase the likelihood of certain behaviors that lead to encouraging stimuli. The central idea of reinforcement learning is that an agent receives some biased feedback of its action, such as encouragement or punishment, instead of being told the correct action itself. This is a more realistic situation that a real-time agent will meet. That is, in many cases, the feedback from the environment is the *reinforcing stimuli* or the *reward* that encourage or discourage the actions the agent takes instead of the exact guidance or demonstration of actions or action sequences. In this sense, reinforcement learning is a more powerful learning mechanism for a complete, interactive, goal-seeking agent.

A reinforcement learning algorithm has to resolve two basic issues, trial-and-error search and delayed reward [90]. Trial-and-error search means that the learner has to discover the most rewarding actions through exploration, since it is not told which action to take. The delayed reward means that the consequence of the learner's action is not necessarily adjacent to the action. They typically appears later. It is the responsibility of the learning algorithm to credit the reward to appropriate actions.

The product of a reinforcement learning system is a *policy*, which tells the agent how to behave in different situations or states. The system usually maintains a *value estimation*, which reflects the "happiness" degree under current system state. This value defines an objective function because the ultimate goal of the system is to

maximize it by following certain policy. It is updated by the *reward* received from the environment.

Q-learning is one of the most popular reinforcement learning algorithms [98] [90]. In Q-learning, each state maintains a set of action values, called Q-values. The action with the largest value will be selected as the system output. The Q-learning algorithm is as follows,

$$Q(s(t-1), a(t)) \leftarrow (1-\alpha)Q(s(t-1), a(t)) + \alpha[r(t) + \gamma \max_{a'} Q(s(t), a')], \quad (3.10)$$

where a and a' are actions associated with a state, $s(t-1)$ is the state at time $t-1$, $s(t)$ is the state the system lands on after executing action $a(t)$, $r(t)$ is the reward received from the environment, α and γ are learning rates. The algorithm shows that Q-values are updated according to the immediate reward $r(t)$ and the value of the next state, which allows delayed reward to be back-propagated in time during learning. After enough experiences of trial-and-error, the system may predict the reward correctly when similar contexts occurs the next time.

While reinforcement learning gives a more realistic learning model to a real-time agent, it is usually slow for an agent to acquire the optimal policy.

3.4.3 Unified learning strategy

In our developmental robot, above two learning strategies are implemented in an unified mode (see Fig. 3.9). A developmental robot starts to conduct behaviors since it is released to the world. At any time, its behaviors depend on what it has learned

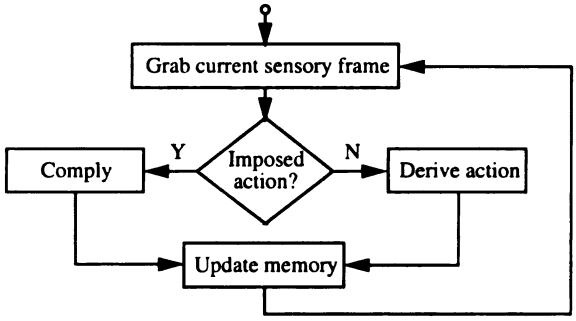


Figure 3.9: Flowchart for unified learning: the system learns while performing.

and the specific environment context. At the same time, learning goes on according to the interaction between the robot and the environment.

If the trainer imposes an action on an effector, the robot will comply it depending on whether the imposed action is consistent with what the robot, itself, plans to do. If it is, everything is fine and go ahead do it. If it is not, the robot will modify its learned behaviors so as to following the trainer's instruction (the imposed action). This is a supervised learning procedure. This learning mode is much more efficient than reinforcement learning when the input-output pairs is available.

There are situations, however, when supervised learning can not be conducted. For example, the robot has internal behaviors such as selecting receptive fields and manipulating internal states. The trainer does not have access to these internal actions. By observing the external behaviors of the robot, the trainer may give reinforcement stimuli to encourage or discourage the robot. From the viewpoint of robot, it receives certain biased sensory inputs instead of imposed actions. If the reward is positive, the robot is considered "understanding" and following the trainer's intention. If the reward is negative, the robot will modify its behavior generation strategy (policy) and derive some new internal or external behaviors. This

is a reinforcement learning procedure.

In either case, the robot memory is updated and the new association of input-output pair is internalized.

We have adapted Q-learning to integrate supervised learning and reinforcement learning strategies into a unified mode (Fig. 3.9)[3]. In our integrated learning mode, if no action is imposed by the environment, we follow the Q-learning algorithm to update the action values. If an action is imposed by the environment, we increase the value of the corresponding action with a fixed amount. Typically, we make this amount large so as to reflect the fact that the system should follow the instructions of the trainer.

Notice, for reinforcement learning case, the biased sensory input may come from both biased and unbiased sensors. By biased sensors, we mean those hardwired so that the information to represent reward directly. For example, human's preference of sweet taste and hate of bitter taste are hardwired. On the other hand, sensors, such as audition and vision, are essentially unbiased because they usually do not lead to pain or happiness at birth. However, these unbiased sensors may be turned to biased ones through experience. For example, a low-pitched sound of "bad dog!" may make a well-trained dog very upset although it does not make any difference to wild dogs at all.

[3]Related work on integrating reinforcement and supervised learning can be found in [15].

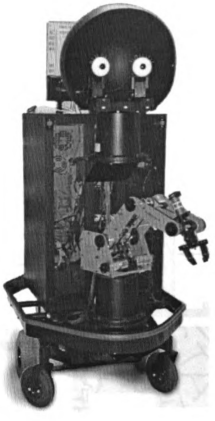


Figure 3.10: The SAIL robot at Michigan State University.

3.5 The SAIL robot

All the experiments presented in the following chapters have been validated on our SAIL robot (Fig. 3.10). SAIL is a human-size mobile robot house-made at Michigan State University. Its hardware components are shown in Fig. 3.11 and 3.12. Their connections are outlined in Fig. 3.13[4].

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.

[4]Figures 3.11, 3.12 and 3.13 are obtained from W.S. Hwang's PhD dissertation [37] with the permission of the author.

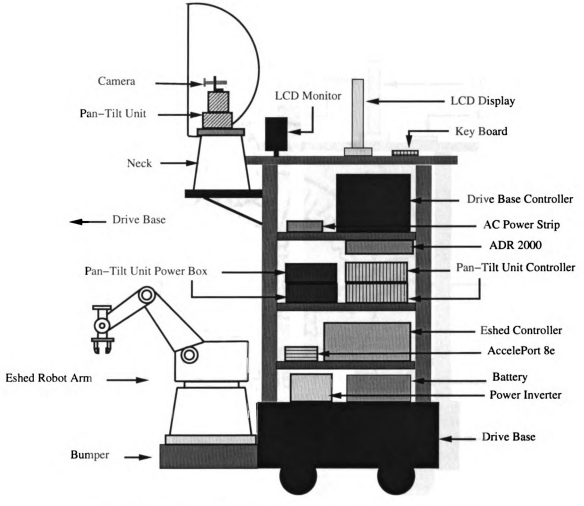


Figure 3.11: The SAIL robot system diagram: left side view.

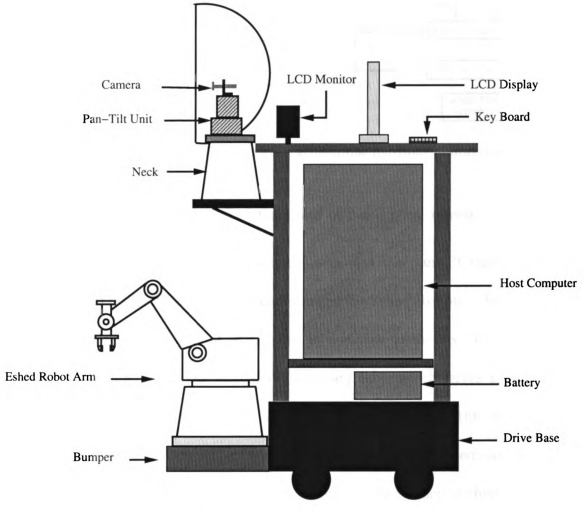


Figure 3.12: The SAIL robot system diagram: right side view.

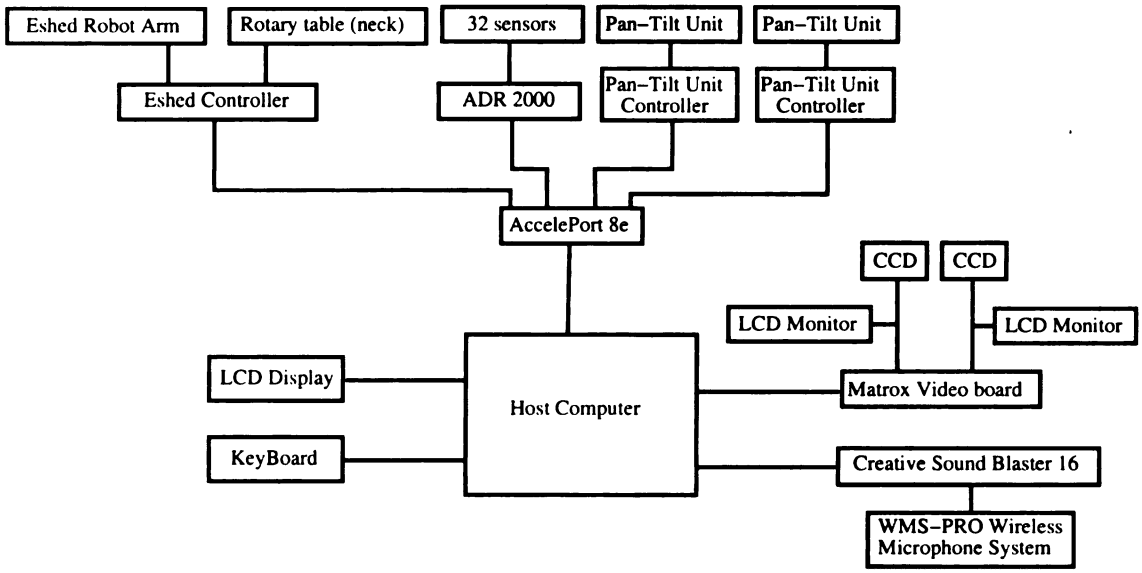


Figure 3.13: The SAIL robot system diagram: hardware connections.

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. 3.14). 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 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

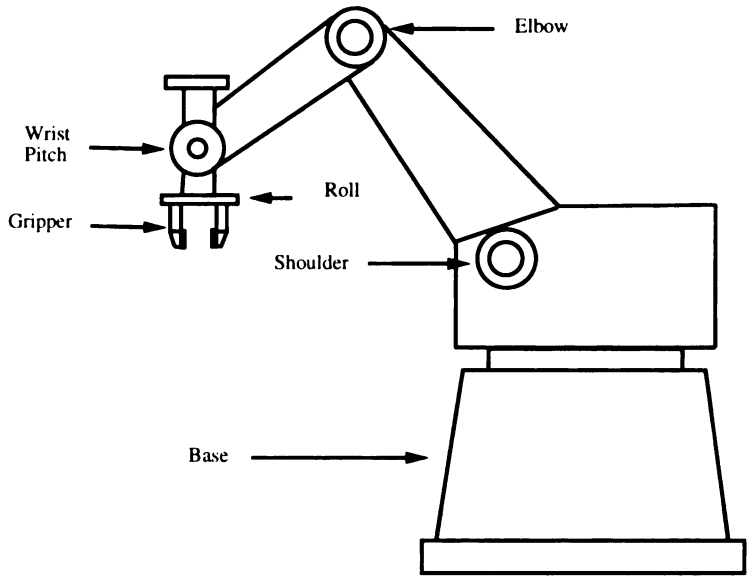


Figure 3.14: The SAIL robot: the Eshed robot arm.

analog multiplexers.

The Eshed controller, the PTU controllers and the ADR 2000 A/D converter are 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.

The host computer, the "brain" of the SAIL robot, used to be a Pentium II 333MHz dual-processor PC with 512 MB RAM and an internal 27 GB three-drive disk array. It retired after finishing the experiments of Chapter 6. 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.

Chapter 4

Incremental PCA in Sensory Mapping

4.1 Introduction

In Chapter 3, we have discussed that each neuron in the sensory mapping module is doing PCA. PCA is a well-known technique in data compression and feature extraction. It gives a linear transformation that is used to convert a set of d-dimensional data into a lower-dimensional space by minimizing the least mean square (LMS) error.

A well-known computational approach to PCA involves solving an eigensystem problem, i.e., computing the eigenvectors and eigenvalues of the sample covariance matrix, using a numerical method such as the power method and the QR method [27]. This approach requires that all the training images are available before the principal components can be estimated. This is called a batch method. This type of method no longer satisfies an up-coming new trend of computer vision research [102], in

which all visual filters are incrementally derived from very long online real-time video stream, motivated by the development of animal vision systems. Online development of visual filters requires that the system performs while new sensory signals flow in. Further, when the dimension of the image is high, both the computation and storage complexity grow dramatically. For example, in the eigenface method, a moderate grey image of 64 rows and 88 columns results in a d-dimensional vector with $d = 5632$. The symmetric covariance matrix requires $d(d+1)/2$ elements, which amounts to 15,862,528 entries! A clever saving method can be used when the number of images is smaller than the number of pixels in the image [86]. However, an online developing system must observe an open number of images and the number is larger than the dimension of the observed vectors. Thus, an incremental method is required to compute the principal components for observations arriving sequentially, where the estimate of principal components are updated by each arriving observation vector. No covariance matrix is allowed to be estimated as an intermediate result. There is evidence that biological neural networks use an incremental method to perform various learning, e.g., Hebbian learning [30].

Several IPCA techniques have been proposed to compute principal components without the covariance matrix [59][60][85]. However, they ran into convergence problems when facing high dimensional vectors. We will discuss the underlying problem of these methods and present a new method, complementary candid IPCA (CCIPCA). CCIPCA is motivated by a well-known statistical concept called efficient estimate. An amnesic average technique is also used to dynamically determine the retaining rate of the old and new data, instead of a fixed learning rate. Experimental re-

sults on high-dimensional data show the better convergence rate of CCIPCA over existing methods. A mathematical proof of the convergence of CCIPCA is given in Appendix A).

4.2 Derivation of the algorithm

4.2.1 First eigenvector

Suppose that sample vectors are acquired sequentially, $u(1), u(2), \ldots$, possibly infinite. Each $u(n)$, $n = 1, 2, \ldots$, is a d-dimensional vector and d can be as large as 5000 and beyond. Without loss of generality, we can assume that $u(n)$ has a zero mean (the mean may be incrementally estimated and subtracted out). $A = E\{u(n)u^T(n)\}$ is the $d \times d$ covariance matrix, which is neither known nor allowed to be estimated as an intermediate result.

By definition, an eigenvector x of matrix A satisfies

$$\lambda x = Ax, \tag{4.1}$$

where λ is the corresponding eigenvalue. By replacing the unknown A with the sample covariance matrix, and replacing the x of Eq. (4.1) with its estimate $x(i)$ at each time step i, we obtain an illuminating expression for $v = \lambda x$,

$$v(n) = \frac{1}{n} \sum_{i=1}^{n} u(i)u^T(i)x(i). \tag{4.2}$$

where $v(n)$ is the n-th step estimate of v. As we will see soon, this equation is motivated by statistical efficiency. Once we have the estimate of v, it is easy to get the eigenvector and the eigenvalue since $\lambda = ||v||$ and $x = v/||v||$.

Now the question is how to estimate $x(i)$ in Eq. (4.2). Considering $x = v/||v||$, we may choose $x(i)$ as $v(i-1)/||v(i-1)||$, which leads to the following incremental expression,

$$v(n) = \frac{1}{n}\sum_{i=1}^{n} u(i)u^T(i)\frac{v(i-1)}{||v(i-1)||}. \tag{4.3}$$

To begin with, we set $v(0) = u(1)$, the first direction of data spread. For incremental estimation, Eq. (4.3) is written in a recursive form,

$$v(n) = \frac{n-1}{n}v(n-1) + \frac{1}{n}u(n)u^T(n)\frac{v(n-1)}{||v(n-1)||}, \tag{4.4}$$

where $(n-1)/n$ is the weight for the last estimate and $1/n$ is the weight for the new data. We have proved that with the algorithm given by Eq. (4.4), $v_1(n) \to \pm\lambda_1 e_1$ when $n \to \infty$, where λ_1 is the largest eigenvalue of the covariance matrix of $\{u(n)\}$, and e_1 is the corresponding eigenvector (see Appendix A).

The derivation of Eqs. (4.2)-(4.4) is motivated by statistical efficiency. An unbiased estimate $\hat{Q}$ of the parameter Q is said to be the *most efficient estimate for the class D of distribution functions* if for every distribution density function $f(u,Q)$ of D the variance $D^2(\hat{Q})$ (squared error) has the minimal value given by

$$D^2(\hat{Q}) = E[(\hat{Q}-Q)^2] \geq \frac{1}{n\int_{-\infty}^{+\infty}\left[\frac{\partial \log f(u,Q)}{\partial Q}\right]^2 f(u,Q)du}. \tag{4.5}$$

The right side of inequality (4.5) is called Cramér-Rao bound. It says that the most efficient estimate is one that has the least variance from the real parameter, and its variance is bounded below by the Cramér-Rao bound. For example, the sample mean, $\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w(i)$, is the most efficient estimate of the mean of a Gaussian distribution with a known standard deviation σ [25]. For a vector version of the Cramér-Rao bound, the reader is referred to [103].

If we define $w(i) = u(i)u^T(i)x(i)$, $v(n)$ in Eq. (4.2) can be viewed as the mean of "samples" $w(i)$. That is exactly why our method is motivated by the statistical efficiency in using averaging in Eq. (4.2). In other words, statistically, the method tends to converge most quickly or the estimate has the smallest error variance given the currently observed samples. Of course, $w(i)$ is not necessarily drawn from a Gaussian distribution independently and thus the estimate using the sample mean in Eq. (4.4) is not strictly most efficient. However, the estimate $v(n)$ still has a high statistical efficiency and has a fairly low error variance as we will show experimentally.

The Cramér-Rao lower error bound in Eq. (4.5) can also be used to estimate the error variance, or equivalently the convergence rate, using a Gaussian distribution model, as proposed and experimented with by Weng et al. [103, Section 4.6]. This is a reasonable estimate because of our near optimal statistical efficiency here. Weng et al. [103] demonstrated that actual error variance is not very sensitive to the distribution (e.g., uniform or Gaussian distributions). This error estimator is especially useful to estimate roughly how many samples are needed for a given tolerable error variance.

IPCA algorithms have been studied by several researchers [61] [92] [59] [60].

An early work with a rigorous proof for convergence was given by Oja & Karhunen [59] [60], where they introduced their stochastic gradient ascent (SGA) algorithm. SGA computes,

$$\tilde{v}_i(n) = v_i(n-1) + \gamma_i(n)u(n)u^T(n)v_i(n-1) \tag{4.6}$$

$$v_i(n) = \text{orthonormalize } \tilde{v}_i(n) \text{ w.r.t. } v_j(n), j = 1, 2, \ldots, i-1 \tag{4.7}$$

where, $v_i(n)$ is the estimate of the i-th dominant eigenvectors of the sample covariance matrix $A = E\{u(n)u^T(n)\}$, and $\tilde{v}_i(n)$ is the new estimate. In practice, the orthonormalization in Eq.(4.7) can be done by a standard *Gram-Schmidt Orthonomalization* (GSO) procedure. The parameter $\gamma_i(n)$ is a stochastic approximation gain. The convergence of SGA has been proved under some assumptions of A and $\gamma_i(n)$ [60].

SGA is essentially a gradient method, associated with which is the problem of choosing $\gamma_i(n)$, the learning rate. Simply speaking, the learning rate should be appropriate so that the second term (the correction term) on the right side of Eq. (4.6) is comparable to the first term, neither too large nor too small. In practice, $\gamma_i(n)$ depends very much on the nature of the data and usually requires a trial-and-error procedure. Oja gave some suggestions on $\gamma_i(n)$ in [59], which is typically $1/n$ multiplied by some constants. However, procedure (4.6) is at the mercy of the magnitude of observation $u(n)$, where the first term has a unit norm but the second can take any magnitude. If $u(n)$ has a very small magnitude, the second term will be too small to make any changes in the new estimate. If $u(n)$ has a large magnitude, which is

the case with high dimensional images, the second term will dominate the right side before a very large number n and, hence, a small $\gamma_i(n)$, has been reached. In either case, the updating is inefficient and the convergence will be slow. It is true that $\gamma_i(n)$ can be manually selected so that it takes into account the magnitude of $u(n)$. But $\gamma_i(n)$ alone can not accomplish statistical efficiency. Further, such a manual selection is not suited for an online learning algorithm since the user does not necessarily have the required expertise and the real-time experiences may not be repeated. The algorithm must automatically compute data-sensitive parameters.

Contrasted with SGA, the first term on the right side of Eq. (4.4) is not normalized. In effect, $v(n)$ in Eq. (4.4) converges to λe instead of e as it does in Eq. (4.6), where λ is the eigenvalue and e is the eigenvector. In Eq. (4.4), the statistical efficiency is realized by keeping the scale of the estimate at the same order of the new observations (the first and second terms properly weighted on the right side of Eq. (4.4) to get sample mean), which allows fully use of every observation in terms of statistical efficiency. Note that the coefficient $(n-1)/n$ in Eq. (4.4) is as important as the "learning rate" $1/n$ in the second term to realize sample mean. Although $(n-1)/n$ is close to 1 when n is large, it is very important for fast convergence with early samples. The point is that if the estimate does not converge well at the beginning, it is harder to be pulled back later when n is large. Thus, one does not need to worry about the nature of the observations. This is also the reason that we used "candid" in naming the new algorithm.

There is a further improvement to procedure (4.4). In Eq. (4.4), all the "samples" $(w(i) = u(i)u^T(i)\frac{v(i-1)}{||v(i-1)||})$, are weighted equally. However, since $w(i)$ is generated

by $v(i)$ and $v(i)$ is far away from its real value at a early estimation stage, $w(i)$ is a "sample" with large "noise" when i is small. To speed up the convergence of the estimation, it is preferable to give smaller weight to these early "samples." A way to implement this idea is to use an amnesic average by changing Eq. (4.4) into,

$$v(n) = \frac{n-1-l}{n} v(n-1) + \frac{1+l}{n} u(n) u^T(n) \frac{v(n-1)}{||v(n-1)||}, \tag{4.8}$$

where the positive parameter l is called the amnesic parameter. Note that the two modified weights still sum to 1. With the presence of l, larger weight is given to new "samples" and the effect of old "samples" will fade out gradually. Typically, l ranges from 2 to 4.

4.2.2 Intuitive Explanation

An intuitive explanation of procedure (4.4) is as follows. Consider a set of 2-dimensional data with a Gaussian probability distribution function (For any other physically arising distribution, we can consider its first two orders of statistics since PCA does so). The data is charactrised by an ellipse as shown in Fig. 4.1. According to the geometrical meaning of eigenvectors, we know that the first eigenvector is aligned with the long axis (v_1) of the ellipse. Suppose $v_1(n-1)$ is the $(n-1)$th-step estimation of the first eigenvector. Noticing $u^T(n)\frac{v_1(n-1)}{||v_1(n-1)||}$ is a scalar, we know $\frac{1}{n}u(n)u^T(n)\frac{v_1(n-1)}{||v_1(n-1)||}$ is essentially a scaled vector of $u(n)$. According to procedure (4.4), $v_1(n)$ is a weighted combination of the last estimate, $v_1(n-1)$ and the scaled vector of $u(n)$. Therefore, geometrically speaking, $v_1(n)$ is obtained by pulling

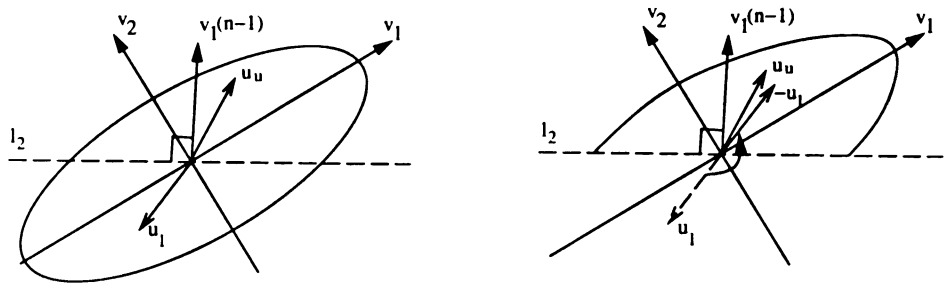


Figure 4.1: Intuitive explanation of CCIPCA.

$v_1(n-1)$ toward $u(n)$ by a small amount.

A line, l_2, orthogonal to $v_1(n-1)$, divides the whole plane into two halves, the upper and the lower ones. Because every point u_l in the lower half plane has an obtuse angle with $v_1(n-1)$, $u_l^T \frac{v_1(n-1)}{||v_1(n-1)||}$ is a negative scalar. So, for u_l, Eq. (4.4) may be written as,

$$v(n) = \frac{n-1}{n} v(n-1) + \frac{1}{n} \left| u_l^T \frac{v(n-1)}{||v(n-1)||} \right| (-u_l),$$

where $-u_l$ is an upper half plane point obtained by rotating u_l for 180 degrees w.r.t. the origin. Since the ellipse is centrally symmetric, we may rotate all the lower half plane points to the upper half plane and only consider the pulling effect of upper half plane points. For the points u_u in the upper half plane, the pure force will pull $v_1(n-1)$ towards the direction of v_1 since there are more data points to the right side of $v_1(n-1)$ than those to the left side. As long as the first two eigenvalues are different, this pulling force always exists and the pulling direction is towards the eigenvector corresponding to a larger eigenvalue. $v_1(n-1)$ will not stop moving until it is aligned with v_1 when the pulling forces from both sides are balanced. In other

words, $v_1(n)$ in Eq. (4.4) will converge to the first eigenvector. As we can imagine, the larger the ratio of the first eigenvalue over the second eigenvalue, the more unbalanced the force is, and the faster the pulling or the convergence will be. However, when $\lambda_1 = \lambda_2$, the ellipse degenerates to a circle. The movement will not stop, which seems that the algorithm does not converge. Actually, since any vector in that circle can represent the eigenvector, it does not hurt not converging. We will get back to the cases of equal eigenvalues in Section 4.2.4.

4.2.3 Higher-order eigenvectors

Procedure (4.4) only estimates the first dominant eigenvector. One way to compute the other higher order eigenvectors is following what SGA does: Start with a set of orthonormalized vectors, update them using the suggested iteration step, and recover the orthogonality using GSO. For real-time online computation, we need to avoid the time-consuming GSO. Further, breaking-then-recovering orthogonality slows down the convergence compared with keeping orthogonality all along. We know eigenvectors are orthogonal to each other. So, it helps to generate "observations" only in a complementary space for the computation of the higher order eigenvectors. For example, to compute the second order eigenvector, we first subtract from the data its projection on the estimated first order eigenvector $v_1(n)$, as shown in Eq. (4.9),

$$u_2(n) = u_1(n) - u_1^T(n)\frac{v_1(n)}{||v_1(n)||}\frac{v_1(n)}{||v_1(n)||}, \tag{4.9}$$

where $u_1(n) = u(n)$. The obtained residual, $u_2(n)$, which is in the complementary space of $v_1(n)$, serves as the input data to the iteration step. In this way, the orthogonality is always enforced when the convergence is reached, although not exactly so at early stages. This in effect well uses the sample available and thus speeds up the convergence. That is the reason we use "complementary" in referring to the proposed algorithm.

A similar idea has been used by some other researchers. Kreyszig proposed an algorithm, which finds the first eigenvector using a method equivalent to SGA and subtracts the first component from the samples before computing the next component [47]. Sanger suggested an algorithm, called generalized hebbian algorithm (GHA), based on the same idea except that all the components are computed at the same time [85]. However, in both cases, the statistical efficiency was not considered.

4.2.4 Equal Eigenvalues

Let us consider the case where there are equal eigenvalues. Suppose ordered eigenvalues between λ_l and λ_m are equal:

$$\lambda_{l-1} > \lambda_l = \lambda_{l+1} = \ldots = \lambda_m > \lambda_{m+1}.$$

According to the explanation in Section 4.2.2, the vector estimate will converge to the one with a larger eigenvalue first. Therefore, the estimate of eigenvectors e_i, where $i < l$, will not be affected anyway.

The vector estimates of e_l to e_m will move into the subspace spanned by themselves

but do not converge inside the subspace since the shape in Fig. 4.1 is now a hypersphere. This does not hurt because any vector in that subspace is a good estimate.

According to Section 4.2.3, those eigenvectors with smaller eigenvalues are estimated in the subspace orthogonal to the space spanned by all the eigenvectors corresponding to larger eigenvalues. Although those estimates with equal eigenvalues keep moving within the subspace, the subspace itself does not move and, therefore, nor does its complementary subspace. So, the estimates of eigenvectors with small eigenvalues will not be affected, either.

When two eigenvalues are nearly equal, the shape in Fig. 4.1 is nearly a hypersphere. The convergence will be slower than with an elongated shape. But the negative effect is minimal since the corresponding vectors are almost equivalent as we discussed above for the equal eigenvalue case.

4.2.5 Algorithm summary

Combining the mechanisms discussed above, we have the complementary candid IPCA algorithm shown in Fig. 4.2.

4.3 Empirical results on convergence

We performed experiments to study the statistical efficiency of the new algorithm as well as the existing IPCA algorithms. The experiments were conducted on two data sets, one with medium dimensionality (130-D) and the other with very high dimensionality (5632-D).

For $n = 1, 2, \ldots$, do,

1. $u_1(n) = u(n)$.
2. For $i = 1, 2, \ldots, k$, do,
   - 2.1. If $i < n$,
     $$v_i(n) = \frac{n-1-l}{n} v_i(n-1) + \frac{1+l}{n} u_i(n) u_i^T(n) \frac{v_i(n-1)}{||v_i(n-1)||},$$
     $$u_{i+1}(n) = u_i(n) - u_i^T(n) \frac{v_i(n)}{||v_i(n)||} \frac{v_i(n)}{||v_i(n)||}.$$
   - 2.2. If $i = n$, initialize the ith eigenvector,
     $$v_i(n) = u_i(n).$$
   - 2.3. If $i > n$, initialize the ith eigenvector,
     $$v_i(n) = 0.$$

Figure 4.2: CCIPCA algorithm: compute first k dominant eigenvectors, $v_1(n), v_2(n), \ldots, v_k(n)$, directly from $u(n)$, where $n = 1, 2, \ldots$.

4.3.1 Experiments on speech data set

The first data set consisted of the utterances of ten numbers ("one" to "ten") by 15 persons collected in pattern recognition and image processing (PRIP) lab at MSU. Each number had five utterances. The sampling rate was 11.025 kHz. Totally we had 374-second speech data. The 13-order Mel-frequency Cepstral Coefficients (MFCCs) [19] were computed for speech frames generated using a 256-point Hamming window. The MFCCs of ten consecutive frames were concatenated as a single vector, which served as a sample. Therefore, the sample dimensionality is 130 and there are totally 20612 samples for the 374-second speech data.

We first computed the eigenvectors using a batch PCA with QR method and used them as our ground truth. The code to do batch PCA was adopted and modified from the C recipes of [68]. Then, SGA, GHA, and CCIPCA were implemented to estimate the eigenvectors incrementally. We divided the whole data set into 20 subsets. When the data went through the IPCA algorithms, the estimations of the eigenvectors

were saved after each subset was passed. In SGA, the learning rate was chosen according to the suggestions in [59, page 54], i.e., $\gamma_1(n) = 0.7/n$, $\gamma_2(n) = 2.5/n$, $\gamma_3(n) = 10/n$, $\gamma_4(n) = 20/n$, and, $\gamma_5(n) = 32/n$. Since there were only suggestions on the first 5 diagonal components of Γ_n, we extrapolated them and chose $\gamma_6(n) = 46/n$, $\gamma_7(n) = 62/n$, $\gamma_8(n) = 80/n$, $\gamma_9(n) = 100/n$, and, $\gamma_{10}(n) = 130/n$. In GHA, we set $\gamma(n)$ as $1/n$. The amnesic parameter l was set to be 2 in CCIPCA.

The coherence between the estimated unit eigenvector v and the one computed by the batch method v', also normalized, is represented by their inner product $v \cdot v'$. Because $||v - v'|| = 2(1 - v \cdot v')$, we have $v = v'$ iff $v \cdot v' = 1$. In other words, the closer to 1 the inner product is, the better the coherence of the two vectors is. As we can see from Fig. 4.3, SGA does not converge at all while GHA shows some trends to converge during the last portion of the experiment. On the contrary, the eigenvectors obtained by CCIPCA converge very well. $||v_i||$ converges to λ_i reliably as shown in Fig. 4.4, where the vertical axis is the ratio, $\frac{||v_i||}{\lambda_i}$, the magnitude of the first 10 eigenvectors obtained by CCIPCA over the corresponding eigenvalues computed by the batch PCA.

To show more clearly why CCIPCA works better than SGA, let us divide both sides of Eq. 4.4 by $||v_i(n-1)||$ and we have,

$$\frac{v_i(n)}{||v_i(n-1)||} = \frac{n-1}{n}\frac{v_i(n-1)}{||v_i(n-1)||} + \frac{1}{n||v_i(n-1)||}u_i(n)u_i^T(n)\frac{v_i(n-1)}{||v_i(n-1)||}. \quad (4.10)$$

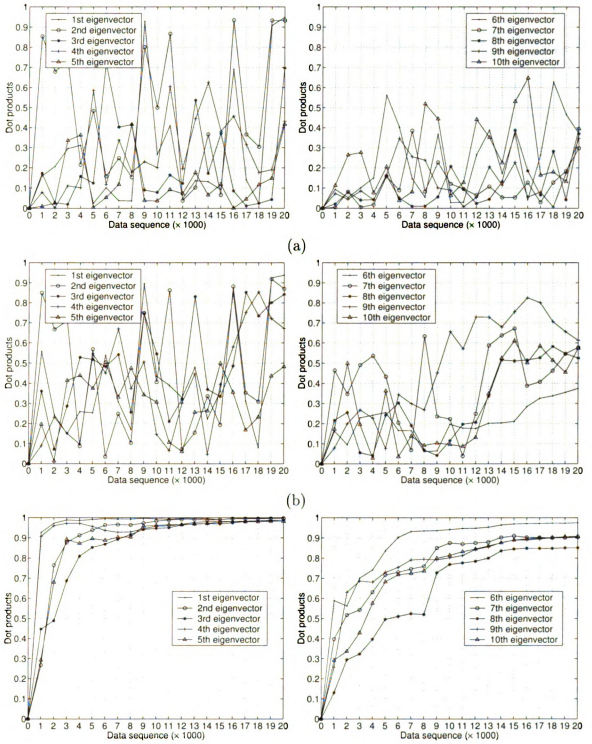


Figure 4.3: Number utterance data set: the correctness, or the coherence, represented by dot products, of the first 10 eigenvectors computed (a) SGA (b) GHA (c) CCIPCA, respectively.

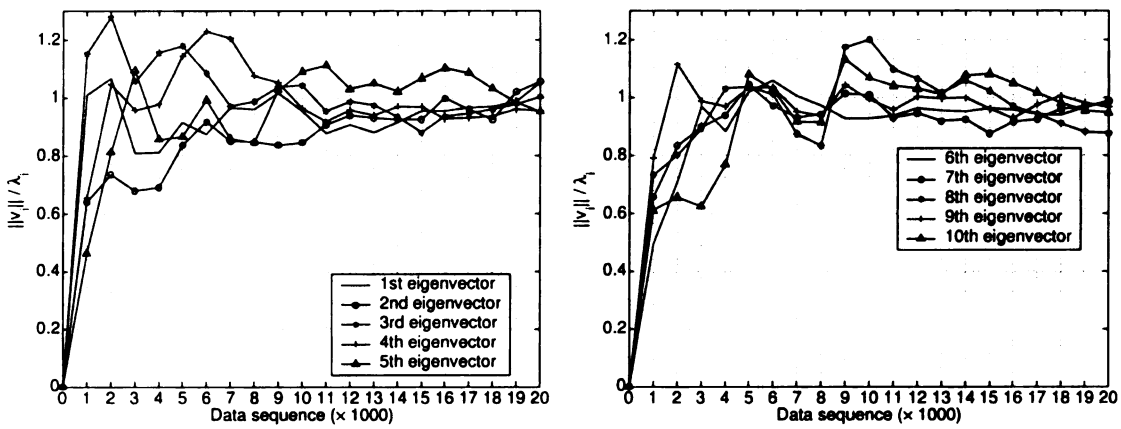


Figure 4.4: Number utterance data set: the correctness of the eigenvalue, $\frac{||v_i||}{\lambda_i}$ by CCIPCA.

Denoting $\frac{v_i(n)}{||v_i(n-1)||}$ as $\tilde{v}_i(n)$, we may rewrite Eq. 4.10 as

$$\tilde{v}_i(n) = \frac{n-1}{n}\frac{v_i(n-1)}{||v_i(n-1)||} + \frac{1}{n||v_i(n-1)||}u_i(n)u_i^T(n)\frac{v_i(n-1)}{||v_i(n-1)||} \tag{4.11}$$

Eq. 4.11 is very similar to Eq. 4.6 if we have $\gamma(n) = \frac{1}{n||v_i(n-1)||}$. In other words, the CCIPCA algorithm can be viewed as an SGA algorithm with a learning rate adaptable according to the magnitude of $v_i(n-1)$. Considering $||v_i(n-1)|| \to \lambda_i$, where λ_i is the i-th largest eigenvalue obtained by the batch PCA, we set $\gamma_i(n) = 1/(\lambda_i n)$ in Eq. 4.6 and did the experiment again. Interestingly, SGA converged much better than before as shown in Fig. 4.5, which means $1/(\lambda_i n)$ was a very good learning rate for SGA. Comparing Fig. 4.5 with Fig. 4.3 (c), we observe that the results of SGA with the finely-tuned learning rate assemble the results of CCIPCA, especially in the cases of the first few principal components. However, for a real-time application we can not use trial-and-error method to find the eigenvalues in advance. Therefore, CCIPCA has a great advantage over SGA.

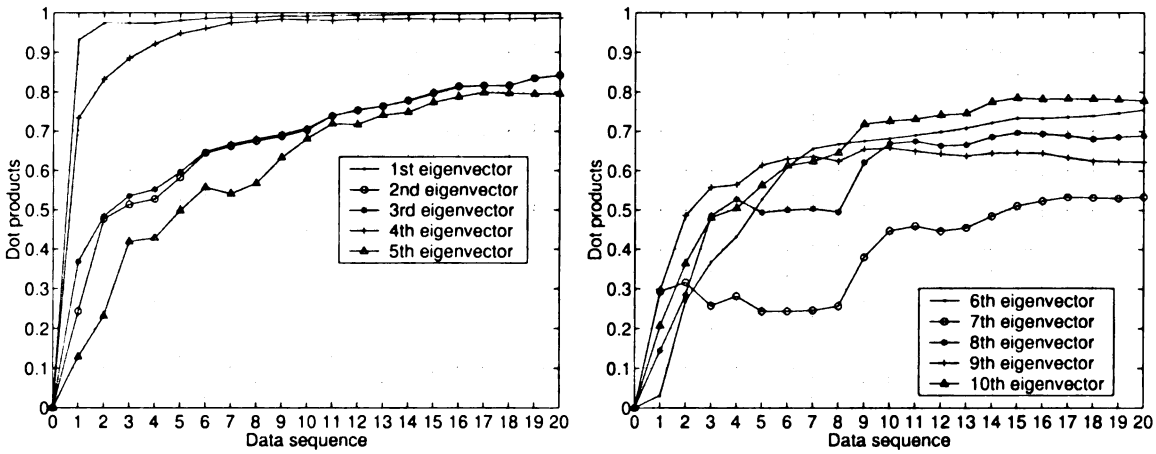


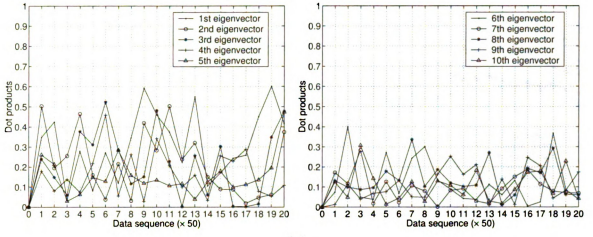
Figure 4.5: Number utterance data set: the correctness, or the coherence, represented by dot products, of the first 10 eigenvectors computed by SGA with finely tuned learning rate.

4.3.2 Experiments on FERET face data set

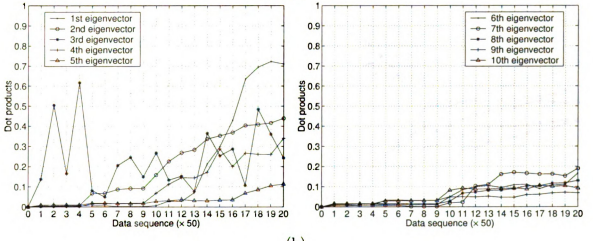
As a second example, we conducted some experiments on the FERET face data set [65] with even higher dimensionality. We used the frontal views of 457 subjects from the data set. Most of the subjects have two views while 34 of them have four views and two of them have one view, which results in a frontal face data set of 982 images. The size of each image is 88-by-64 pixels, or 5632 dimensions.

Again, we first computed the eigenvectors using the batch PCA algorithm. Then we divided the entire data set into 20 subsets and saved the estimates of the eigenvectors after each subset went through the IPCA algorithms. The learning rates for SGA and GHA were set the same as in speech data set experiment. Fig. 4.6 and Fig. 4.7 give a summary of the result, which shows similar convergence patterns as in number data set experiment.

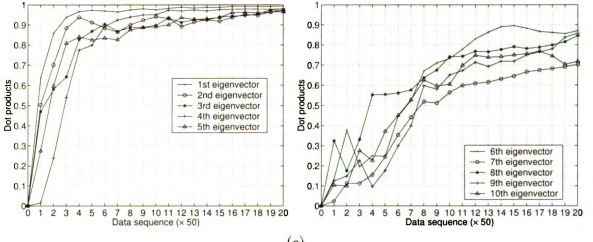
To demonstrate the effect of amnesic parameter l in Eq. (4.8), we show the result of eigenvector estimate with $l = 0$. Comparing Fig. 4.8 with Fig. 4.6 (c), we can see



(a)



(b)



(c)

Figure 4.6: FERET data set: the correctness, or the coherence, represented by dot products, of the first 10 eigenvectors computed by (a) SGA (b) GHA (c) CCIPCA with the amnesic parameter $l = 2$.

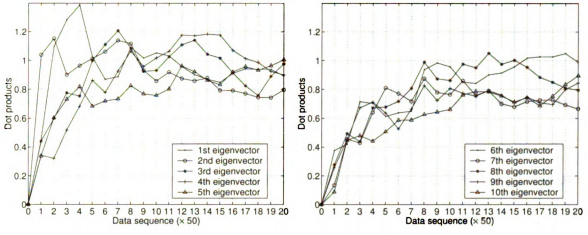


Figure 4.7: FERET data set: the correctness of the eigenvalue, $\frac{||v_i||}{\lambda_i}$ by CCIPCA.

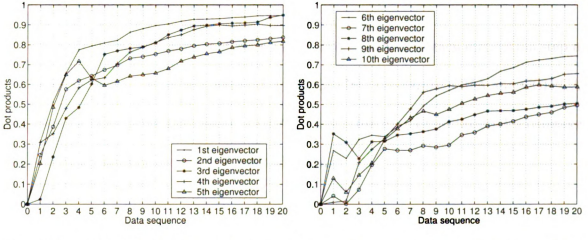


Figure 4.8: FERET data set: the effect of the amnesic parameter. The correctness of the first 10 eigenvectors computed by CCIPCA, with the amnesic parameter $l = 0$. A comparison with Fig. 4.6 (c).

that the amnesic parameter did help to achieve faster convergence.

A further experiment was done to show the performance of the algorithm with a long data stream. Since the statistics of a real-world image stream may not necessarily be stationary (for example, the mean and variance may change with time), the changing mean and variance make correctness evaluation difficult. To avoid this effect, we fed the same set of FERET image data repeatedly into the algorithms to simulate a statistically stable long data stream. Fig. 4.9 shows the result after

Table 4.1: The average execution time.

SGA	GHA	CCIPCA
0.42s	0.083s	0.072s

20 epochs. As expected, all IPCA algorithms converged better with the long data stream while CCIPCA was the quickest one.

In order to provide a visual view about the eigenvectors computed, Fig. 4.10 shows the first 10 eigenfaces obtained by batch PCA and CCIPCA (with amnesic parameter $l = 2$), respectively as images. The corresponding eigenfaces computed by the two very different methods are very similar.

The average execution time of SGA, GHA, and CCIPCA in each estimation step is shown in Table 4.1. As shown, without doing the GSO procedure, GHA and CCIPCA run significantly faster than SGA. CCIPCA has a further computational advantage over GHA because of a saving in normalization.

4.4 Conclusions

This chapter concentrates on a challenging issue of computing dominating eigenvectors and eigenvalues from incrementally arriving high dimensional data stream without computing the corresponding covariance matrix and without knowing data in advance. The proposed CCIPCA algorithm is fast in convergence rate and low in the computational complexity. Our results showed that whether the concept of the most efficient estimate is used or not plays a dominating role in convergence speed for high dimensional data. An amnesic average technique is implemented to further improve the convergence rate.

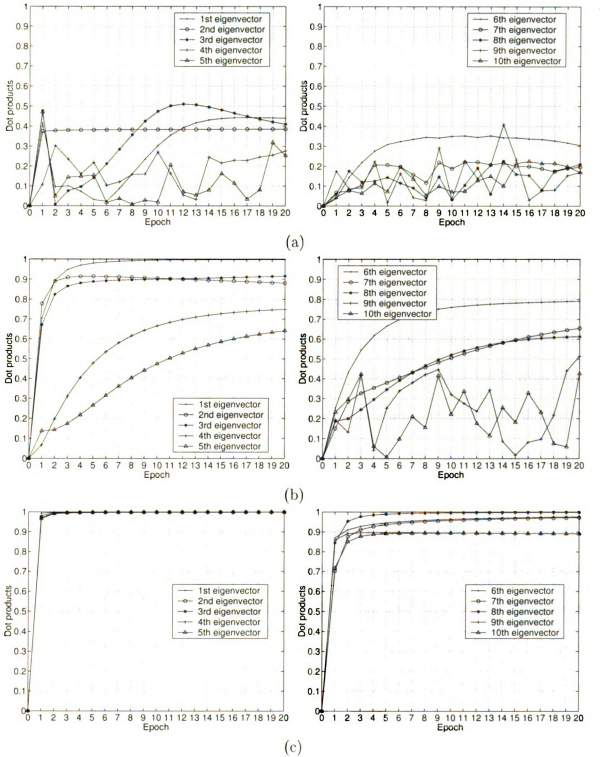


Figure 4.9: FERET data set: the correctness of the first 10 eigenvectors computed by (a) SGA (b) GHA (c) CCIPCA (with the amnesic parameter $l = 2$), respectively over 20 epochs (a long data stream).

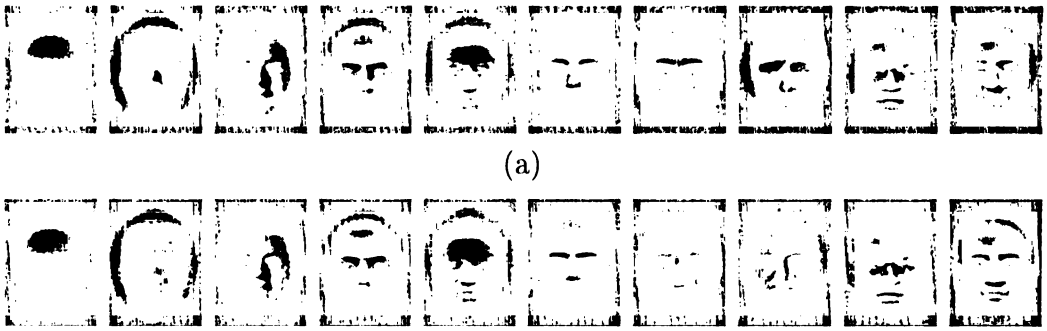


Figure 4.10: FERET data set: the first 10 eigenfaces obtained by (a) batch PCA, and (b) CCIPCA (with amnesic parameter $l = 2$), shown as images.

The importance of the result presented here is potentially beyond the apparent technical scope interesting to the computer vision community. As discussed in [102], what a human brain does is not just computing – processing data – but more importantly and more fundamentally, developing the computing engine itself, from real-world, online sensory data streams. Although a lot of studies remain to be done and many open questions are waiting to be answered, the incremental development of a "processor" plays a central role in brain development. The "processor" here is closely related to a procedure widely used now in appearance-based vision: inner product of input scatter vector u with an eigenvector, something that a neuron does before sigmoidal nonlinearity. What is the relationship between IPCA and our brain? A clear answer is not available yet, but Rubner & Schulten [80] proved that the well-known mechanisms of biological Hebbian learning and lateral inhibition between nearby neurons [42] (pages 1020, 376) result in an incremental way of computing PCA. Although we do not claim that the computational steps of the proposed CCIPCA can be found physiologically in the brain, the link between incremental PCA and the developmental mechanisms of our brain is probably more intimate than we can fully appreciate

now.

Chapter 5

Grounded Speech Learning

5.1 Introduction

We view a speech recognition system as an intelligent agent with audition-driven behaviors. The development of such a system is an agent development problem. We have discussed some of the challenges of such a system in Chapter 1. In this chapter, we show how the architecture and the techniques we developed in the last two chapters contribute to resolving the problems. We would like to show that an artificial system can develop its audition-dependent behaviors through online, real-time interaction with the environment. This is one step toward fully autonomous learning, which may lead ASR systems to be autonomous.

5.2 Implementation details

5.2.1 IHDR

The first requirement for a system to learn directly from its real-time sensory experiences is to eliminate the manual design of internal representation. Internal representation includes, among others, features extracted from perceived signals, subspaces represented by clusters in sensory inputs, inter-connections between nodes, and states consisting of short-history information.

As discussed in Chapter 3, we choose the method of incremental hierarchical discriminant regression (IHDR) to achieve this goal. IHDR does clustering in both input and output spaces. When constructing (or learning) the tree, we first derive the clusters in the output space, according to which the clusters in the input space are formed. When using the constructed or partially constructed tree to do the mapping, we retrieve the tree in the input space to find the primitive prototype and then locate the corresponding prototype in the tree of the output space to generate the output. To guarantee the performance, IHDR needs the sample distributions in the input and output spaces to have the same topology, i.e., the order of distance between samples should be consistent in both spaces. Otherwise, the input space clusters will not be appropriately constructed. For example, in Fig. 5.1, because y_2 is close to y_3 and x_2 is close to x_3, the corresponding input and output clusters are well organized. But it is not very desirable to form the input space cluster of x_1 and x_4, which corresponds to the output space cluster of y_1 and y_4.

However, for audition-driven behavior development, the distribution of the input

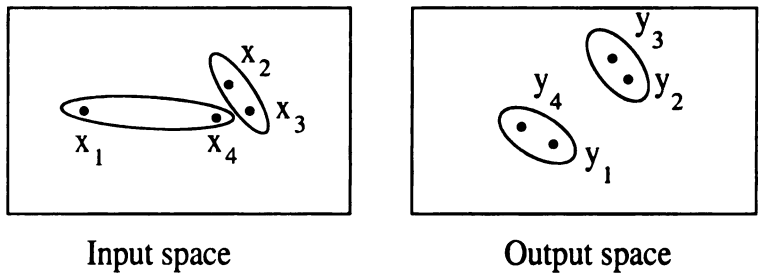


Figure 5.1: The sample distributions in the input and output space should have the same topology.

to the system, the auditory signals, does not necessarily maintain the same topology as that of the output, the motor control signals. For example, there are cases that the motors do not move while there are non-silent auditory signals heard by the robot. We surely do not want to group all these non-silent auditory signals into a big cluster corresponding to a zero output.

To compile with the requirement of the IHDR method, we did not directly use the auditory signals and the motor control signals as the input-output pairs entering the IHDR tree. Instead, we set the output part as the concatenation of the auditory signal and the motor control signal,

$$l(t) =< x_s(t), x_a(t) >,$$

where $l(t)$ is the *output part* in the input-output pair, $x_s(t)$ and $x_a(t)$ are the auditory sensation vector and the control signal vector, respectively. In this way, when the robot is not doing any actions, $x_a(t)$ is zero and the IHDR tree is constructed according to the auditory signals only. When the robot is conducting certain actions, both $x_s(t)$ and $x_a(t)$ are used to guide the clustering in output space and eventually guide the

clustering in input space.

Since the auditory signal part had different dimension, mean, and variance from the control signal part, to avoid the domination of either part, we normalized them before the concatenation,

$$\tilde{v}_i(t) = \frac{v_i(t) - \bar{v}_i(t)}{\sigma_i(t)}, \quad i = 1, 2,$$

where $v_1(t) = l(t)$, $v_2(t) = x_a(t)$, $\tilde{v}_i(t)$ is the normalized signal, $\bar{v}_i(t)$ is the mean of $v_i(t)$, and $\sigma_i(t)$ is the variance of the norm of $v_i(t)$.

5.2.2 Auditory sensation

For a robot, the auditory signal enters the microphone as a continuous stream. Without the help of a human designer to edit or transcribe the sensory input, the meaningful speech units, such as phonemes, words, phrases, and sentences, do not exist to a robot any more, which has a twofold effect. On one hand, the robot only needs to make sure actions are conducted under appropriate auditory contexts and does not need to decide whether the context corresponds to a phoneme, a word, or a sentence, which makes the system design simpler. On the other hand, the context has to be determined automatically and appropriately by the robot itself through its interactive real-time experiences, which is very challenging. The sensory mapping we proposed (see Section 3.2) is an attempt to resolve the challenge, although it may not be the best or the final solution. Nevertheless, our experiments did show some interesting results by enabling a selective attention mechanism, modeled as an internal behavior,

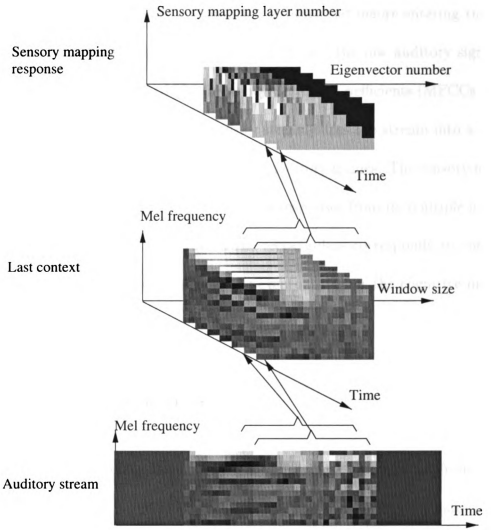


Figure 5.2: The process over the auditory sensation before it enters the cognitive mapping module.

to choose the context.

Speech is a linear signal in the sense that the information is distributed over time. Therefore, a single and short auditory frame usually does not contain enough information for behavior decision making. In our implementation hereafter, if not specified explicitly, a newly captured auditory sensation includes 20 auditory frames in the history at each time instance. This short history information proved to be enough for short phrases which are typical in early audition development stage.

Fig.5.2 illustrates how auditory sensation is processed before entering the cognitive mapping module. Cepstrum Analysis is done over the raw auditory signals and generates an auditory stream of Mel-frequency Cepstral Coefficients (MFCCs [19]). A sliding window over the continuous auditory stream turns the stream into a series of last auditory contexts with each containing 20 auditory frames. The sensory mapping modules take these last contexts and generates responses from its multiple layers. In the upper panel of Fig.5.2, each row of the rectangulars corresponds to one layer's response. These sensory mapping responses eventually enter the cognitive mapping.

5.2.3 Behavior generation

The nature of autonomous development of our system prevents us from defining the behaviors in advance. While this offers the flexibility of the ultimate behavior capability of the system, it immediately introduces a problem — how to acquire the desired actions among such a very large number of possible ones?

As we can imagine, the action space is determined at the system programming time. For example, after the robot arm is built, the action space is defined, although not all points in this space are reachable. The system can generate all the possible external actions from a small set of internal action increments. The actual action outputs of the system will be the accumulation of these action increments. For example, to touch every (x, y) position in a bounded 2-D space, a minimum of four increment vectors are sufficient: $(1, 0)$, $(-1, 0)$, $(0, 1)$, $(0, -1)$.

5.2.4 Learning procedure

At each time instance, the robot conduct the following learning procedure.

1. Grab new auditory sensation $x_s(t)$.

2. Update each neural column of the sensory mapping module by doing PCA incrementally.

3. Concatenate the outputs from the neural columns in the sensory mapping module and the reading from the internal action sensor, $x_a(t)$, into a single vector, the last context $l(t)$. Suppress the outputs from some of the neural columns to zero according to the attention selection signals (part of $x_a(t)$).

4. Replace the oldest part of the internal state $s(t-1)$ with $l(t)$ to get a vector $s'(t)$[1].

5. Query the partially constructed IHDR tree and get a primitive prototype, $s(t)$, that is closest to $s'(t)$ using the fast tree search.

6. If $s'(t)$ is significantly different from $s(t)$, it is considered as a new sample and we update the IHDR tree using the input-output pair $< s'(t), l(t) >$. Otherwise, $s'(t)$ updates $s(t)$ through incremental averaging.

7. After updating the tree, we query it again using $s'(t)$ to get the primitive prototype $s(t)$, which would be the new state. Thus we finish the state transition as shown in Fig. 3.2.

[1]See page 48.

8. If an action is given (imposed by a trainer) through the touch sensors as $a_i(t)$, increase the value of the action associated with $s(t)$ that is most similar to $a_i(t)$, and return $a_i(t)$ as the action to be executed at this time instance. Otherwise, update the Q-values of $s(t-1)$ using Eq. (3.10), and return

$$a(t) = \arg\max_{a'} Q(s(t), a'),$$

as the action to be executed at this time instance.

9. Return to step 1

5.2.5 Communicative learning

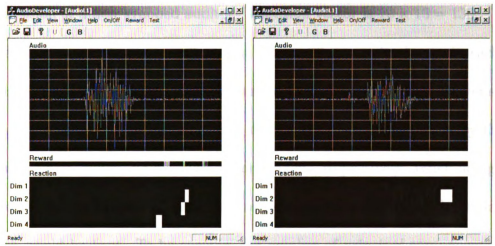
While both supervised learning and reinforcement learning have some advantages and they compensate each other in our unified learning strategy, they are relatively low-level learning mechanisms and demand a lot of training efforts in complex behavior learning. For example, to teach a robot to "go left at the corner," the human trainer has to either push the robot to turn (supervised learning) or give some rewards when the robot happens to turn left at the corner (reinforcement learning). Actually, in animal learning and human learning, there is a higher level of learning mode, which we call *communicative learning.* Communicative learning takes advantage of a higher animal's capability of language acquisition to transfer sophisticated skills from the trainer to the learner through language instructions. In the case of the previous example, suppose "recognizing a corner" and "following a 'turn left' command" are two learned behaviors. Teaching "go left at the corner" would be as easy as saying

"turn left" when the robot reaches the corner. As a preliminary investigation of communicative learning, we taught the robot to learn vision-guided navigation after it acquired some simple language capability and the results are reported in Section 5.6.

5.3 Experiments on audition-driven behavior development with reinforcement learning

The first experiment we conducted was to train a system to develop audition-driven behaviors through reinforcement learning.

We built a real-time software agent called AudioDeveloper. AudioDeveloper had two kinds of sensors, a microphone as the auditory sensor and two touch sensors as reward sensors. A simulated 4-joint robot arm was the effector. As shown in Fig. 5.3, the GUI of AudioDeveloper displays the audio wave (the top part), the rewards (the middle part), and the actions (the bottom part) along the horizontal time axis. Human trainers first spoke to AudioDeveloper and then gave the rewards by pushing the "G" and "B" buttons on the GUI toolbar for "good" and "bad", respectively, according to AudioDeveloper's behavior. To facilitate the training procedure, another software agent, AudioTeacher, was programmed as the real-time virtual teacher. AudioTeacher played the pre-recorded sound through a set of computer speakers, and generated and sent the reward to AudioDeveloper according to AudioDeveloper's response. The training procedure was run in real-time for 6 hours, during which the resulting developed system was saved every half an hour so that we could examine



(a) (b)

Figure 5.3: The GUI of AudioDeveloper: (a)During online learning; (b)After online learning.

the performance later.

The speech was contributed by 140 persons with a variety of nationalities (American, Chinese, French, Indian, Malaysian, and Spanish) and ages (from 18 to 50). Each person made five utterances for each of the four vowels, /A/(hard), /E/(head), /i/(heed) and /c/(haul)[2]. There was silence lasting about 0.5s between two consecutive utterances. The sound was digitized at 11.025 kHz. In this way, we got an speech data set with 2800 isolated utterances. For the five sets of vowel utterances, we did a 5-fold leave-one-out cross-validation in the experiment. That is the AudioTeacher played four out of five sets of utterances during training and played the independent set of utterances during testing.

Before the speech data reached the sensory mapping, the 13-order MFCCs were computed over every 256 sound sample points (a single auditory frame of about 20ms) before the data reached the sensory mapping module. Covering 10 auditory frames,

[2]We use single symbol version of ARPAbet to represent the phonetic alphabets.

the dimension of a new grabbed auditory sensation vector is $10 \times 13 = 130$.

The sensory mapping module had four layers. The neural columns in each layer took as the input the outputs from ten neural columns of the next lower layer spreading along the temporal dimension. Thus, the length of context covered by the four layers was about 200ms, 400ms, 800ms, and 1600ms, respectively. When doing PCA, we determined the number of principal components so that 95% of the variance in the data would be kept. To be specific, the number of principal components in the four sensory mapping layers was 31, 27, 22, and 18, respectively.

The control signal to the effector of the system was a 4-D action vector. Four desired behaviors were defined, each for one of the four vowels. Behaviors were identified by the component of the action vector with the maximum value. For example, if the first component of the action vector had the maximum value, it was identified as action 1. Internal actions were defined as the increment on each component of the action vector. There was an extra internal action that reset the whole action vector to zero.

The rewards were decided as follows. If the system made a correct action within a range of (-200ms,200ms) at the end of an utterance, the system would receive a reward 1. If the action was wrong or there was no action made within that time window, the system would get a reward -1. In all other cases including the silence period, the system got reward -0.001.

The performance was evaluated as follows. Within a short period before or after the end of an utterance, if the system reacted correctly once or more than once within that time window, we counted it as a success. Otherwise it was an error. Fig.5.4 shows

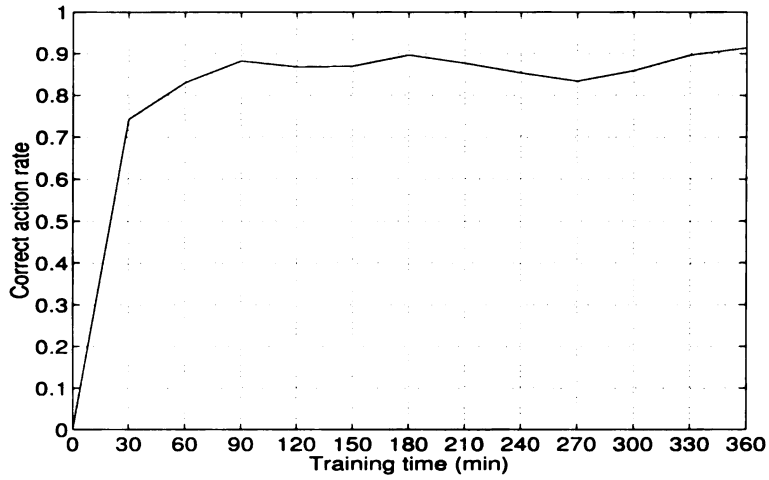


Figure 5.4: A real-time system: correct action rate vs. training time.

Table 5.1: Simulation: confusion table for test.

Expected Behavior vs. Actual Behavior	/A/	/E/	/i/	/c/	Rejection
/A/	125	3	0	5	7
/E/	1	127	9	0	3
/i/	0	5	134	0	1
/c/	5	0	0	133	2

that the system gradually and steadily improved its behavior through practice as the time passed by.

To examine the behavior of the system in more details. We did a simulation experiment in the same way as the real-time system experiment above. The only difference was that the MFCCs were computed in advance and fed into the system manually. We call a round of feeding all the training data an *epoch*. The performance evaluated after each epoch is shown in Fig. 5.5, which has the similar trend as that of the real-time system. The confusion table of the expected behaviors (E.B.) vs. the actual behaviors (A.B.) is shown in Table 5.1.

To take a closer look at the behavior of the system, we traced one of the primitive prototypes in the IHDR tree that corresponded to vowel "a." All the state's internal

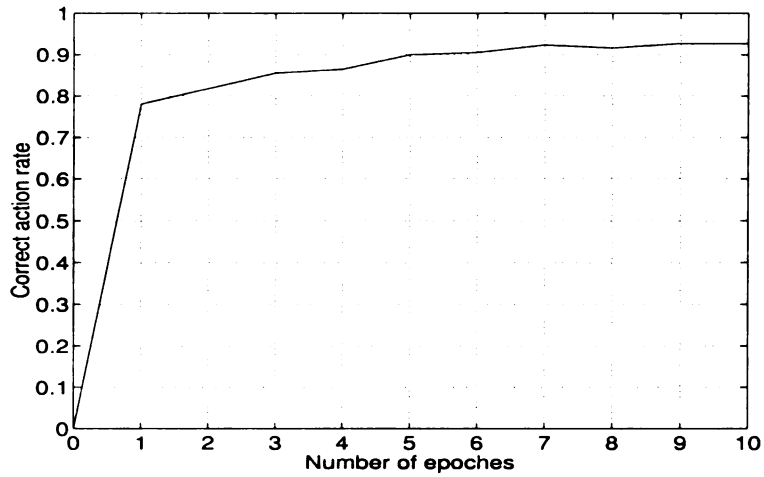


Figure 5.5: Simulation: correct action rate vs. epoches.

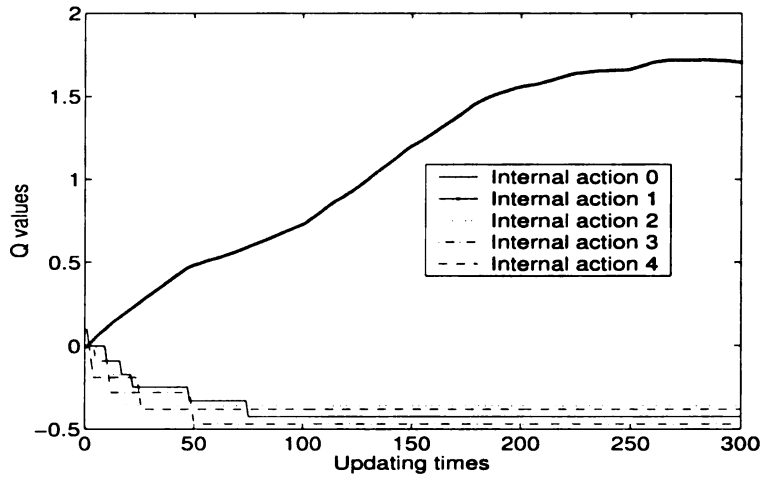


Figure 5.6: Simulation: smoothed Q values of the internal actions of one of the states vs. the number of times it was updated.

action values were set to zero initially except for that of the extra "reset" action, whose initial value was 0.1. This meant that the system preferred to do nothing at the very beginning. As the development process went on, the Q-value of internal action 1 increased while those of other internal actions decreased (Fig.5.6). This suggested that the system gradually and reliably figured out that it should increase the first component of the action vector (as defined by the meaning of the internal actions) under this state, which was exactly what we wanted.

Table 5.2: Selective attention: correct rate (C.R.) for using different sensory mapping layers

Layer	1	2	3	4	All
C.R.	88.6%	89.7%	83.3%	78.25%	92.7%

5.4 Experiments on selective attention learning

In the experiments presented in Section 5.3, the attention selection signals were set in advance so that the outputs from all sensory mapping layers were selected.

We did this because we observed that selecting all layers had better performance than that of any single layer for this task, as shown in Table 5.2. We owed this phenomenon to two reasons. On one hand, higher layer lost some information when we ignored the higher order principal components. So, the performance of layer 4 degraded comparing to lower layers. On the other hand, by covering different length of context, different layers extracted uncorrelated information, which did good to the final decision making in different cases. Hence, their combination gave us better performance than each single layer.

To specifically test the system's attention selection capability, we collected speech data of ten numbers (1-10) from five people, which had more context information than vowel data. The simulation experiments were conducted in a way similar to that of last section. The difference included that the sensory mapping module here had only two layers and the system had one more internal action to select the output from one of the layers. After being trained for ten epoches, the correction rate of the external actions were constantly improved (Fig. 5.7). We examined and recorded the system's choices of attention at the time the system made the external behaviors. In Table 5.3,

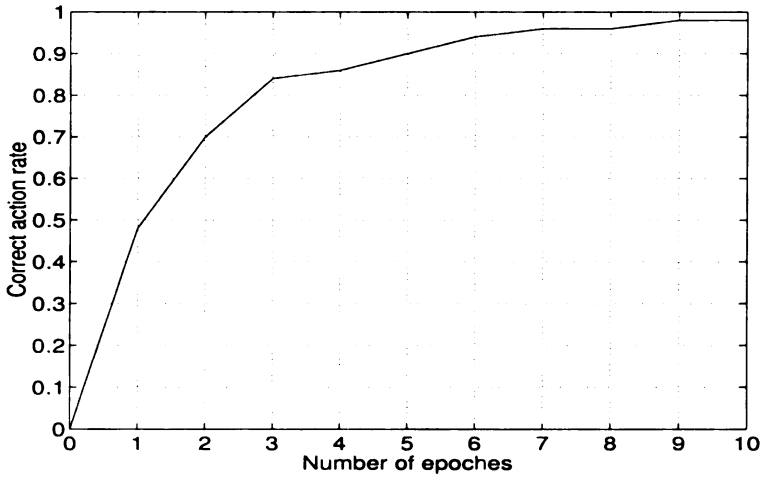


Figure 5.7: Selective attention: overall performance vs. training epoches.

Table 5.3: Frequency of the attention selection behavior per external behavior

External Behavior	1	2	3	4	5	6	7	8	9	10
1st layer	0	1.00	0.73	0.83	0.66	0.74	0	0.63	0.25	0.40
2nd layer	1.00	0	0.27	0.17	0.33	0.26	1.00	0.38	0.75	0.60

the first line shows the external behaviors represented by their corresponding number utterances. The second and third lines show the frequency that the system selected the output from the first or the second layer of the sensory mapping when firing external behaviors.

An interesting thing happened to external behaviors 1 and 7 as we expected. The system exclusively chose the second layer when making decisions. After a second thought, one will realize that the tail parts (about 200ms) of the utterances of "one" and "seven" were similar. Therefore, the first layer did not cover enough context for distinguishing these two words and the system chose the second layer to make decision. For "nine" and "ten", which had similar tail parts, the system showed its preference by selecting the second layer too. In other words, the attention selection behavior was successfully acquired.

5.5 Experiments on the SAIL Robot

In the experiments presented above, we showed that a real-time artificial system could learn to react to auditory inputs through interaction with the environment. However, the vocabulary size was very small and the behaviors were simple. In this section, we will show the work done on a real robot that learned to follow more complicated verbal commands.

5.5.1 Preliminary experiment on number data

Before working on the SAIL robot, we did a preliminary experiment on a larger-size vocabulary. The whole settings were the same as those in Section 5.3, except following changes,

- The effector of the system was represented by a 10-D action vector. Ten desired behaviors were defined, each for one of the ten numbers ("one" to "ten"). Behaviors were still identified by the component of the action vector with the maximum value.

- To simulate the situation a robot would face in a real world, the eigenvectors in the sensory mapping were derived from more auditory data, 5-hour radio programs that included both news and music.

- The training procedure consists of two phases, supervised learning followed by reinforcement learning. In supervised learning, the imposed actions were given to the system by the end of each utterance. The settings for reinforcement

Table 5.4: Results on number recognition

External Behavior	Correct Rate (%)	Incorrect Rate(%)	Rejection Rate(%)
1	98.4	1.6	0
2	95.2	3.2	1.6
3	93.7	3.2	3.2
4	96.8	3.2	0
5	95.2	4.8	0
6	93.7	3.2	3.2
7	96.8	3.2	0
8	92.1	3.2	4.8
9	93.7	3.2	3.2
10	93.7	3.2	3.2
Average	94.9	3.2	1.9

learning were the same as before.

- The auditory data were collected at the same time when those vowel data were collected. Each of the 63 persons made five utterances for each of the ten numbers, "one" to "ten". There was a silence of a length of about 0.5s between two consecutive utterances. The data set had totally 3150 isolated utterances.

The test was done using the 5-fold leave-one-out cross-validation. The results summarized in Table 5.4 show that our system reliably responded to the complicate auditory inputs after online interactive learning. Fig. 5.8 shows part of the resulted IHDR tree.

5.5.2 Real robot experiments

To teach the SAIL robot, in the supervised learning phase, a trainer spoke to the robot with a spoken command C and then imposed a desired action A by pressing a pressure sensor or a touch sensor that was linked to the corresponding effector. In

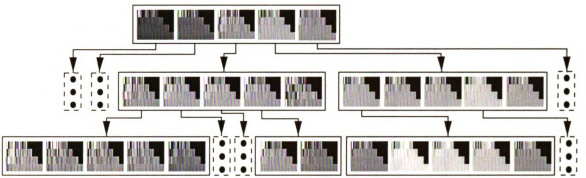


Figure 5.8: Part of the IHDR tree after training using the number data set. Shown in each image is the x-cluster mean.

reinforcement learning, a "good" button and a "bad" button were used to simulate appetizing and aversive sensors, respectively. The reward was decided in the same way as in the experiments presented in Section 5.3. We still set the SAIL robot a default attention selection behavior to select all sensory mapping layers.

The training process was conducted online in real-time through physical interactions between a trainer and the SAIL robot. After being trained for 15 minutes, the SAIL robot could follow commands with about a 90% correct rate. Table 5.5 summarizes the performance of the SAIL robot when it was guided by the verbal commands to navigate through the corridors of the Engineering Building at MSU (see Fig. 5.9). The arm and eye commands were issued ten times each at different locations.

To further test the system's capability of dealing with speaker variations, we conducted a multi-trainer experiment. Eleven persons participated in training. They spoke each of the 15 commands for five times which resulted in 825 utterances. The speech data (four out of five utterances of each commands) was fed into the SAIL robot off-line appended with appropriated actions at the end of each utterance. The SAIL robot with partially trained "brain" started to run in real-time. Then, a trainer,

Table 5.5: Performance of the SAIL robot in one-trainer case.

Commands	Total times	Correct rate(%)
Go left	35	97.1
Go right	23	91.3
Forward	65	93.8
Backward	7	100.0
Freeze	5	80.0
Arm left	10	100.0
Arm right	10	90.0
Arm up	10	100.0
Arm down	10	100.0
Hand open	10	90.0
Hand close	10	90.0
See left	10	100.0
See right	10	100.0
See up	10	100.0
See down	10	100.0

being the 12th trainer, taught the SAIL robot through physical interactions four times for each command. In this way, we simulated the situation that a partially developed SAIL robot continuously developed its audition-driven behaviors.

After training, the 12th trainer tested the SAIL robot by guiding it through the second floor of the Engineering Building, just as was done in the one-trainer case. The performance is summarized in Table 5.6. More trainers introduced more variance in speech data. The results show that the performance of the SAIL robot in the multi-trainer case degraded a little compared with the one-trainer case but it was still reasonable. The performance for other trainers was evaluated off-line using the left-out utterances. The performance is summarized in Table 5.7.

To our knowledge, this is the first work on online speech learning without a pre-designed model and with the number of words and the number of speakers totally

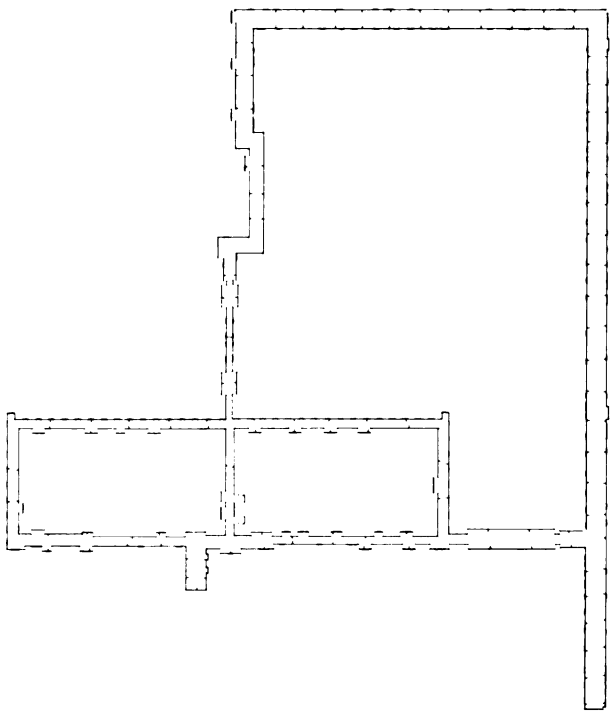


Figure 5.9: Engineering Building 2nd floor at MSU.

open[3]. Although the performance described above has not matched those of the traditional SR systems in term of vocabulary size, this work has made a solid progress for the very difficult new learning mode.

5.5.3 Speed issues

Execution time is a very important issue for a real-time system. In each computation loop, the SAIL robot needs to collect the sensory data from sound card, touch and

[3]The work in [107] was for recognizing five vowels in a simulated online mode.

Table 5.6: Performance of the SAIL robot when following the 12th trainer's command in multi-trainer case

Commands	Total times	Correct rate(%)
Go left	36	88.9
Go right	28	89.3
Forward	70	92.8
Backward	8	87.5
Freeze	9	88.9
Arm left	10	90.0
Arm right	10	90.0
Arm up	10	100.0
Arm down	10	100.0
Hand open	10	90.0
Hand close	10	80.0
See left	10	100.0
See right	10	100.0
See up	10	100.0
See down	10	100.0

pressure sensors, compute the MFCCs for each auditory frame, construct and retrieve the IHDR tree, and submit the control signals to the controllers of the effectors. As each speech frame covers 256 points with 56 point overlap and the auditory digitization frequency is 11.025 kHz, above computations should be finished in 18.1ms, which is very tight in time. While the system worked smoothly in this experiment, we would like to keep track of its speed performance. We recorded the execution time of MFCCs calculation and IHDR tree construction/retrieval in each loop, which were the most time-consuming portions (Fig. 5.10). The average execution time is shown in Table 5.8. As we can see, the MFCCs calculation took the major execution time and the total average time was well under 18.1ms. The shape of resulted IHDR tree is shown in Fig. 5.11, where the horizontal axis is the depth of the tree and the vertical axis is the number of nodes.

Table 5.7: Performance of the SAIL robot on off-line test data in multi-trainer case

Commands	Correct rate(%)
Go left	94.5
Go right	89.9
Forward	92.7
Backward	100.0
Freeze	100.0
Arm left	100.0
Arm right	90.9
Arm up	96.3
Arm down	92.7
Hand open	89.9
Hand close	89.9
See left	90.0
See right	92.7
See up	100.0
See down	100.0

Table 5.8: The SAIL robot: the average execution time in each loop (one-trainer case)

	Computation of MFCCs	IHDR tree retrieval	Total
Ave. exec. time/loop	8.3ms	0.59ms	8.9ms

5.6 Experiments on communicative learning

The above experiments demonstrated a process of simple language acquisition with rich semantics and simple syntax[4]. The semantics was naturally acquired because the speech was learned within the physical context. Upon learning to follow verbal commands, the SAIL robot was taught to perform vision-guided navigation through communicative learning[5].

Communicative learning includes two phases. The first phase is language acquisition, which has been achieved above. The second phase is teaching using language (Fig. 5.12). Using the trained commands, the trainer guided and instructed the robot

[4]It is typically the case for a child.

[5]This work is jointly done by Wey Hwang and me.

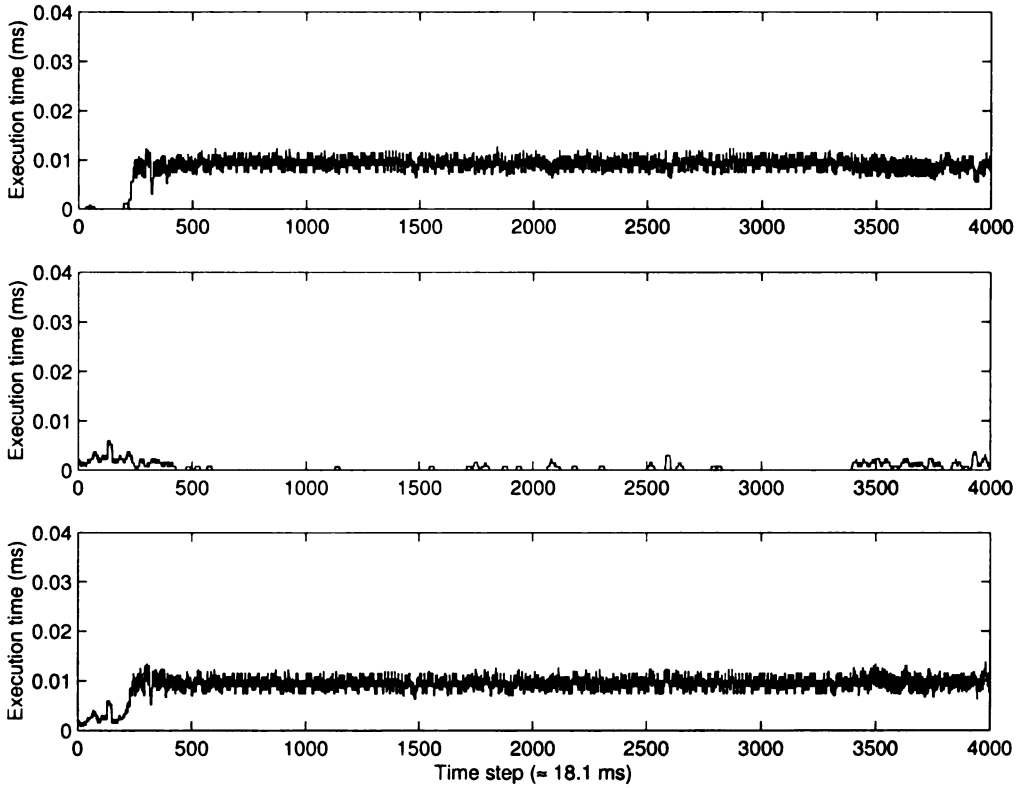


Figure 5.10: The SAIL robot: the execution time in each loop (one-trainer case). (a) MFCCs calculation; (b) IHDR tree construction/retrieval; (c) Total execution time.

to go through the entire second floor of the Engineering Building at Michigan State University. The actions executed under the corresponding voice commands were associated with the visual context by another IHDR tree in real time. Fig. 5.13 shows some of the images seen by the SAIL robot. Later, when the robot saw a similar visual context, it would retrieve the IHDR mapping and find the practiced action. In the experiment presented here, the switch of the attention between vision and audition was an innate (programmed-in) volume-based behavior of the robot. The voice command had a higher priority to control the robot than the learned vision-guided controls. After training, the SAIL robot was tested for ten turns in the Engineering Building until the robot batteries ran low. It needed further voice instructions at

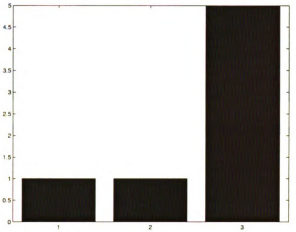


Figure 5.11: The SAIL robot: the shape of IHDR tree in one-trainer case. The horizontal axis specifies the depth of the tree and the vertical axis specifies the number of nodes.

Figure 5.12: The SAIL robot is running autonomously in the hallway of the Engineering Building of Michigan State University.

only five locations which were difficult for vision-guided navigation because of the changing lighting (windows) or very narrow corners. The experiment of communicative learning reported here saved the trainer a lot of effort in training by using verbal commands.

In the next chapter, we will discuss a more sophisticated architecture for communicative learning – learning complex behaviors through verbal instructions after acquiring simple ones.

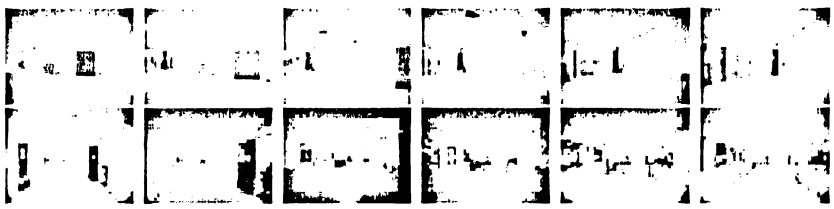


Figure 5.13: Examples of the image sequence seen by the SAIL robot.

5.7 Conclusions

We have demonstrated that it is feasible for an artificial system to develop its audition-driven behaviors through online, real-time interaction with the environment. This work is among a few works that enable a machine to learn directly from unsegmented and unlabeled speech streams, a mode in which human children learn.

IHDR played an important role in the success of the experiments. It automatically derives discriminant features and thus automatically generates internal representations. The behaviors of the robot were developed through a unified learning strategy. Both external behaviors, such as body movement, and internal behaviors, such as selective attention, were acquired.

Our experiments show that after a short process of grounded simple language acquisition, the robot can produce desired behaviors upon hearing verbal commands. We have also showed that this simple language capability can be used later for other behavior learning, such as vision-guided navigation. In the next chapter, we will report more results on teaching a robot to conduct complex behaviors through verbal instructions.

Chapter 6

Task Transfer

6.1 Introduction

Task transfer is a capability of autonomously applying what has been learned in one context to new contexts (tasks) [13]. It is more than memorizing and learning. Here is an example of task transfer. Suppose that a tiger in a circus has learned two skills: (1) Jump onto the table top immediately after hearing "Table!" command from the trainer. (2) Jump through the fire ring immediately after hearing "Ring!" command. The new task is to perform (1) and (2) in sequence after hearing a new command "Start!" without stepwise commands "Table!" and "Ring!"

Task transfer is an important capability to test the success of what is called *cognitive learning by humans and animals.* Cognitive learning is considered as a higher-level learning than classical conditioning [63] and instrumental conditioning [53]. Unlike classical conditioning and instrumental conditioning, there is no reinforcement involved in cognitive learning. As far as we know, this is the first work to formulate

and realize task transfer in the AMD paradigm.

Symbolically, classical conditioning has been described as the following so-called *acquisition procedure* (A.P.):

$$CS \rightarrow US \rightarrow UR \Rightarrow CS \rightarrow CR \tag{6.1}$$

where, CS stands for conditioned stimuli (e.g., tone), US for unconditioned stimuli (e.g., food), UR for unconditioned response (e.g., salivation), CR for conditioned response (e.g., salivation), $\rightarrow$ means "followed by," and $\Rightarrow$ means "develops." Other conditioning protocols that have been widely studied include secondary conditioning, where one CS is preceded by another CS,

$$CS_2 \rightarrow CS_1 \rightarrow CR \Rightarrow CS_2 \rightarrow CR \tag{6.2}$$

and instrumental conditioning, where the association between a stimulus (S) and a response (R) is affected by the following positive or negative reinforcement (reward).

The major problems of the traditional computational models of these animal autonomous learning phenomena include:

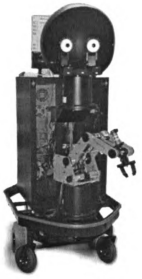
1. They are symbolic models in that the stimuli or responses are considered as an atomic entity [88] [89] [43] [4], instead of spreading over real-time sensory streams. See [5] for an excellent survey. Some of these models have been implemented on robots. For example, in [94] [83], a robot functioned upon a set of perception and behaviors which were defined in advance. In [87], a mobile

robot learned to associate the visual sensations of predefined classes (blob and strip patterns on cubes) with predefined action sequences. The computational models are not directly applicable to multimodal real sensory streams, just like the case where a text language model does not directly apply to raw auditory signals.

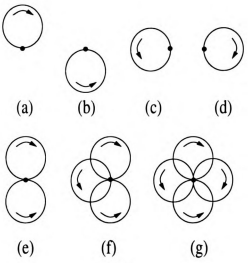
2. They are ad hoc in nature. A model for instrumental conditioning is not applicable to classical conditioning and verse visa. There is a dilemma of model applicability when they are applied to a general autonomous learning setting: Which model is applicable at any time?

3. They overlook the role of autonomous attention on the learning agent part. Traditionally, task-related information is manually extracted for symbolic description as in A.P. (6.1) and (6.2) without modeling why only this information is relevant among many other environmental stimuli and bodily internal signals, e.g., motor signals.

In this chapter, we augment the architecture presented in Chapter 5 and make it applicable to all kinds of associative learning, including classical conditioning, secondary conditioning, instrumental conditioning, and the more general class, cognitive learning. In order to fully explain how an AMD agent works instead of partially modeling one kind of learning, this architecture captures also the important cross-modality attention mechanism in the process of autonomous learning.



(I)



(II)

Figure 6.1: (I) The SAIL robot house-built at Michigan State University. (II) Behaviors represented as gripper tip trajectories of the SAIL robot. (a)-(d): Individual behaviors as petal drawing, each of which starts from the black dot. (e)-(g): Drawings consists of more than one petal, as behaviors developed through multiple task transfers.

6.2 Problem description

Suppose a robot has learned to draw a correct petal following the commands "left," "right," "upper," and "lower," respectively, as shown in Fig. 6.1 (II). Now the teacher wants the robot to transfer the learned skills of drawing individual petals to a new task: drawing a flower consisiting of multiple petals triggered by a new command. Of course, these basic skills can be transferred to any number of new composite tasks, each corresponding to a new command.

6.2.1 Principles of autonomous learning

Although the literature in psychology has clearly classified several animal learning types, such as classical conditioning, instrumental conditioning, and cognitive learn-

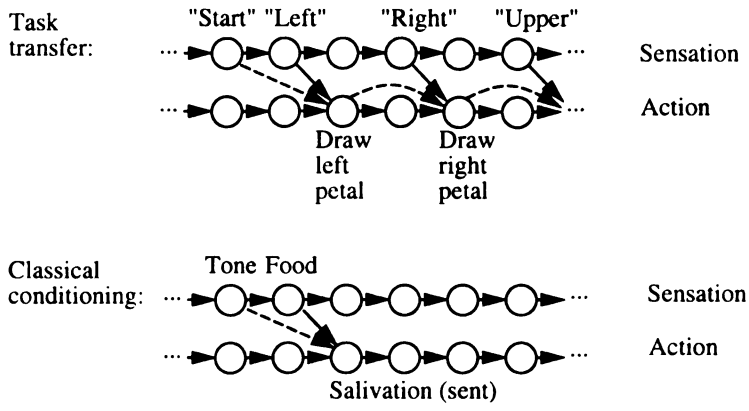


Figure 6.2: A new view of task transfer and classical conditioning. Two vertically aligned circles constitute a context at a particular time. Solid arrows indicate previously established association, and dash arrows denote newly established association through priming.

ing, they are enabled by a single brain architecture. The challenging issue here is to investigate the architecture and implement it to test whether the architecture works on a developmental robot. We hold a view that different learning types described in psychology are enabled by a set of principles, which involves context, association, priming, attention, and value.

To see these important principles, we illustrate task transfer and classical conditioning in an unconventional way in Fig. 6.2. In the figure, a circle in the upper row corresponds to a sensation context and a circle in the lower row corresponds to an action context. In the same row from left to right, different circles denote contexts at different time instances. A context at a time instance is denoted by two vertically aligned circles, a sensation context at the upper row and an action context at the lower row.

The attention control at each time instance determines which part in the current context is attended to form the current attended context. If an attended context C_1 is repeatedly followed by another attended context C_2 in time, an association from

C_1 to C_2 is established in memory, where C_1 and C_2 can be a sensation context, an action context, a part of each, or a combination thereof. Once such an association is established, C_2 will be primed whenever there is a context similar to C_1. However, human brain does not just prime the context for the *next* time instance, it also primes farther, as indicated by the dash arrows in Fig. 6.2.

Some associations are innate (e.g., food-to-salivation association in classical conditioning) while others are learned. If C_1 corresponds to a sensation and C_2 corresponds to an action, the value of C_2 estimated by the value system (also called motivational system) is crucial to determine whether the action should be sent to motors for execution. The value system of a developmental agent starts with an innate value system at birth time, which prefers appetitive stimuli (e.g., sweat taste) and avoids aversive stimuli (e.g., pain sense). The added learned value system can greatly enrich and alter such an inborn value system.

Our SAIL developmental robot is based on our above theory of human and animal learning, which is different from the traditional type-specific models of animal learning. This theory serves as a guide in the development of the SAIL general architecture. However, designing a practical architecture from the above theory faces a series of challenges. We discuss two of them here. One is the challenge of numerical representation. The other is the challenge of the missing contexts.

6.2.2 Numerical representations

It is important to note that each circle in Fig. 6.2 corresponds to a realization of a random process, or in other words, the (discrete) experience of a developmental robot after "birth" can be represented by a series of random vectors,

$$E = \{C(t) = (X(t), A(t)) | t = 0, 1, 2, ...\},$$

where $C(t)$ denotes the context random vector at time t, $X(t)$ and $A(t)$ denotes the sensory and action context, respectively. The context is the part of information that the agent attends to at time t. It is stored in working memory. Since hand-segmentation and hand-labeling of sensory streams are not allowed for the AMD mode, we cannot use a symbolic label to represent a context. Instead, each context is a numerical vector representing information in (sensory and motor) space and time (the last few context vectors in time, probably processed). At each time instance t, the sensory input $s(t)$ is a long vector collected by the real sensors, such as cameras, microphones, and touch sensors, as well as internal sensors such as those that sense selective attention effectors, covering last several time frames, $t-k, t-k+1, ..., t-1, t$. The dimension of $s(t)$ is as high as a few hundreds (for audition) to a few thousands (for vision) and beyond. With the same sampling rate, the vectors of control signals are sent to the motor controllers. Depending on the number of motors, the vector of control signals may vary from 1 to a few dozens in the case of the SAIL robot.

The numerical representation is motivated by the distributed neural coding in biological brains. An atomic symbol loses its similarity information with respect to other

symbols but a numerical vector representation does not. The high-dimensional vector representation potentially has a higher generalization capability than the symbolic one, as the number of classes is dynamic and virtually unlimited. Therefore, a numerical representation enables the same program to learn new stimuli and behaviors. However, it also poses greater challenges than dealing with symbolic representations. First, many different sensory contexts $X(t)$ require the same action vector. For example, different people have different voices but they can speak the same word. Second, different action contexts $A(t)$ are perceptually equivalent, just like the fact that a word can be hand-written differently at each time. Third, it is not known which part of the sensory input is related to the action output. For instance, the appropriate temporal length of a context vector is hard to determine at the "birth" time.

To keep the discussion clear, we will still use symbols to denote random processes although the internal representation does not use any symbol as a concept. Therefore, the task transfer process can be written as the following A.P.,

$$C_c \rightarrow C_{s1} \rightarrow A_{s1} \rightarrow C_{s2} \rightarrow A_{s2} \Rightarrow C_c \rightarrow A_{s1} \rightarrow A_{s2} \tag{6.3}$$

where, C_c is a composite command (e.g., "Start"), C_{s1} and C_{s2} are simple commands (e.g., "Left"), A_{s1} and A_{s2} are the actions (e.g., drawing individual petals).

6.2.3 Missing context

Another challenge in task transfer comes from the missing contexts, the commands C_{s1} and C_{s2}, in A.P. (6.3). Logically, C_c should not elicit $A_{s1} \rightarrow A_{s2}$ without the pres-

ence of C_{s1} and C_{s2}. The teacher wants to make use of the learned skills, $C_{s1} \rightarrow A_{s1}$ and $C_{s2} \rightarrow A_{s2}$, to save the training process. So, on the learner side, there should be some mechanism to handle the missing contexts. The study of classical conditioning shows that animals take advantage of the causal property of the physical world (the orderly event sequence) to achieve task transfer, which is a general mechanism fulfilling our *task-nonspecific* requirement. However, because of the numerical representation, C_{s1} and C_{s2} are random processes with varying length of duration. A powerful architecture design is needed.

6.3 Single-level architecture

In the previous chapters, we have proposed an architecture that handles numerical representations, which enabled the SAIL robot to conduct grounded speech learning. Without any task-specific information available, such as pre-defined acoustic models, the SAIL developmental program generated internal representations and architecture autonomously according to the events encountered. All the learning processes of SAIL were conducted online in real-time through *physical interactions* between trainers and the SAIL robot. It is based on such acquired sensorimotor skills that the task transfer reported here can take place.

6.3.1 Handling the missing context

The cognitive mapping module of the architecture presented in Chapter 3 (Fig. 6.3) is essentially a reflexive one, which produces the corresponding behavior output given a

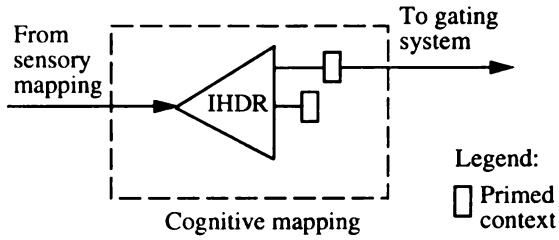


Figure 6.3: The cognitive mapping module

context input. It cannot look ahead and predict sufficiently far. If there is any context missing, the IHDR tree will give a badly matched state and the agent's behavior is not predictable. To handle the missing context, we augmented the reflexive cognitive module by adding another IHDR tree (Fig. 6.4). The new tree is identical to the old one in the architecture except that it is associated with a *prototype updating queue* (PUQ). We call the original tree the reality tree, or *R-tree*, and the added one the priming tree, or *P-tree*. The goal of PUQ for the P-tree is to enable a looking-ahead (farther priming) mechanism. The PUQ maintains a list of pointers to the primed contexts retrieved by the P-tree. A primed context is a sensation-action-value tuple, $p = (x, a, Q)$, where the primed sensation (mental image) x is useful for developing high-level value system and for planning, the primed action a is a possible action at the current state, and Q is its value, estimated by the value system.

At every time instance, a pointer to a newly retrieved primed context enters the PUQ while the oldest one moves out. When the pointers are kept in PUQ, the primed contexts they point to are updated with a recursive model adapted from Q-learning [98],

$$p^{(n)}(t) = p^{(n-1)}(t) + \frac{1+l}{n}(\gamma p^{(n-1)}(t+1) - p^{(n-1)}(t)) \tag{6.4}$$

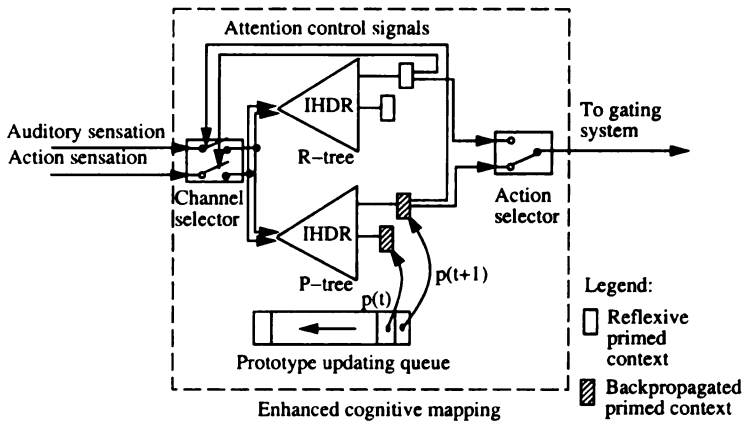


Figure 6.4: The enhanced cognitive mapping.

where, $p^{(n)}(t)$ is the primed context at the time instance t, n represents the number of times $p^{(n)}(t)$ has been updated, and γ is a time-discount rate. l is an amnesic parameter introduced by us, (e.g., $l = 2$), which is typically positive and is used to give more weight on the newer data points.

Reorganizing Eq. (6.4), we have,

$$p^{(n)}(t) = \frac{n-1-l}{n} p^{(n-1)}(t) + \frac{1+l}{n} \gamma p^{(n-1)}(t+1) \tag{6.5}$$

which shows more clearly that a primed context $p^{(n)}(t)$ is updated by averaging its last version $p^{(n-1)}(t)$ and the time-discounted version of the current primed context $p^{(n-1)}(t+1)$. In this way, the information embedded in the future context, $p^{(n-1)}(t+1)$ in model (6.4), is recursively backpropagated into earlier primed contexts. Therefore, Eq. (6.4) is effectively a prediction model. When an earlier context is recalled, it contains the expected future information. Note that this kind of manipulation on context can only be done on a numerical representation, which makes the power of numerical representation clear.

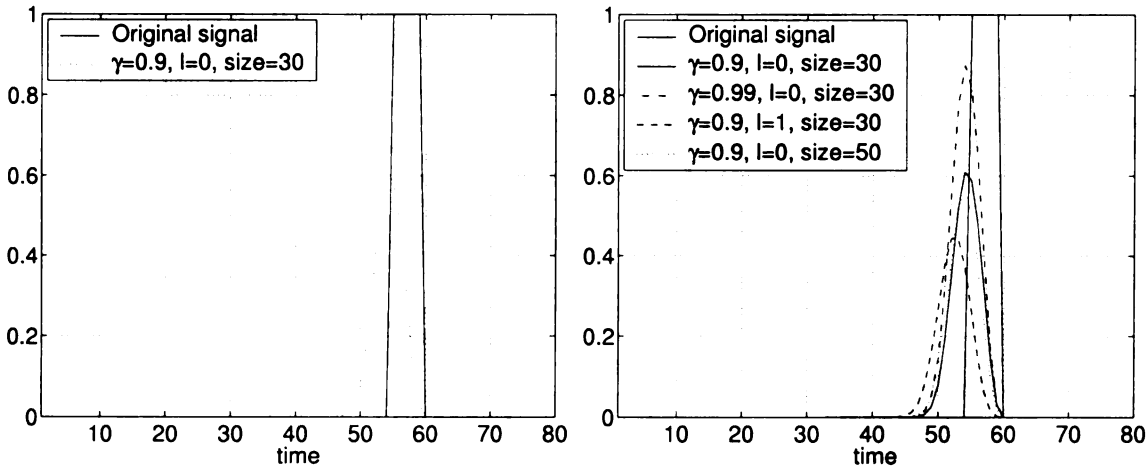


Figure 6.5: The behavior of prediction model (6.4) with different γ, l, and PUQ size.

To view this effect more intuitively, we show the behavior of the prediction model on a simple example. Suppose the primed contexts appearing over time are represented by a series of scalers. A scaler with value 1 means the primed context contains certain information while 0 means no information is embedded. An example of a series of the primed contexts is shown with a solid line in Fig. 6.5 (a), where there is certain information over the five consecutive time instances ($t = 55, 56, \ldots, 59$) and nothing elsewhere. Applying the model (6.4) with $\gamma = 0.9$ and $l = 0$ using a PUQ of size 30, we get the dotted line in Fig. 6.5 (a), where the information has been backpropagated with the peak appearing at $t = 54$. In other words, at an earlier time instance, the model predicts about 60% of the information.

Three parameters have influence on the behavior of model (6.4): the time-discount rate γ, the amnesic parameter l, and the size of the PUQ. As shown in Fig. 6.5 (b), a larger γ helps the model to predict more information while a larger l enables the model to predict longer in the future (by propagating the information back more in

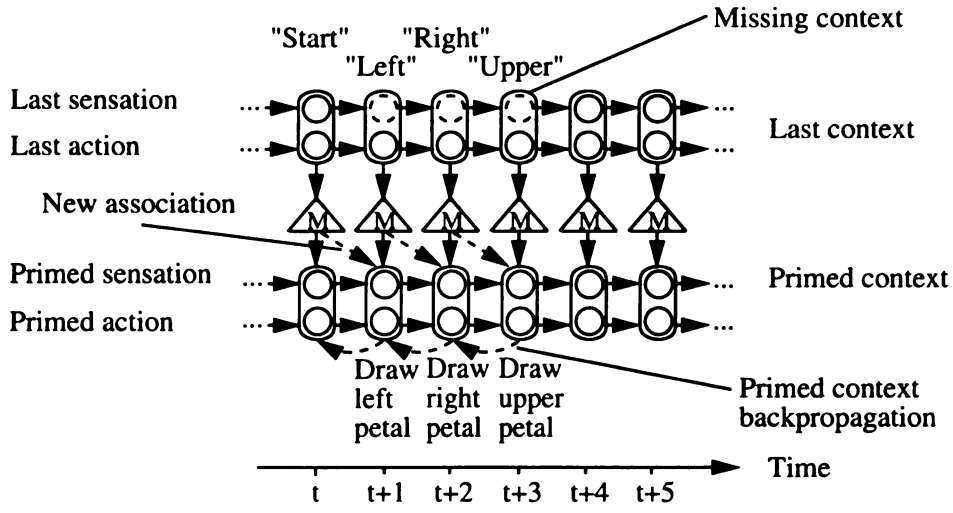


Figure 6.6: Handling the missing context.

time). The performance is not very sensitive to the PUQ size but the PUQ should be long enough so that the newly entered primed context can still affect the one that is about be moved out of PUQ.

With the prediction model (6.4), the process of handling missing context can be viewed in Fig. 6.6. At each time instance, the robot decides the primed context, especially the primed action, based upon the observed last context. At the same time, the operations in PUQ makes the primed contexts to be backpropagated over time. When there is a missing context, e.g., missing "Left," "Right," and "Upper," the drawings will still be executed because of the newly established association between last contexts and primed contexts in the P-tree. Therefore, one way to look at the roles played by the R-tree and the P-tree is as follows: Both trees do the mapping from the last context to the primed context. While the R-tree does immediate mapping, the P-tree does mapping with the prediction to a near future.

6.3.2 The value system

One problem of having a double-tree architecture in the cognitive mapping module is decision making. In other words, when both trees provide action vectors, which one should the agent choose and send to the effectors? This is the responsibility of a value system. Subsumption [10] is a simple example of a value system, while the state-action value in Q-learning is another one. A sophisticated value system should be developed from experience.

In the value system of the presented work, each primed context includes a value, $Q(p)$. It is assigned with an initial value when a prototype is stored in the IHDR tree. If there is an action imposed at the time, the initial value is set to be 1. Otherwise, it is 0. Since the prediction model (6.4) is also applied to this value when the primed context enters PUQ, the value will be distributed among consecutive primed contexts. In addition to this value, another factor affecting decision making is the goodness of match $d(s)$, the Euclidean distance between the current state s and the prototype state retrieved by the tree. A small $d(s)$ means that the agent recalls the new context with a high confidence and, therefore, the decision is reliable.

Put the above two items together, we build a quantity called *confidence index*,

$$i(t) = \frac{Q(p(t))}{d(s(t)) + \varepsilon},$$

where ε is a small number to avoid a zero denominator. The action selector in Fig.6.4 simply compares the confidence index associated with the action vector given by both the R-tree and the P-tree. The one with higher confidence index gets through and

goes to the corresponding effector.

6.3.3 Mechanism for cross-modality attention

We have two trees doing different mappings and a value system selecting the outputs from them. The robot still can not do task transfer without a cross-modality attention mechanism.

In the work here, we only consider one complex sensory modality, audition. There is one more internal sensing modality, the sensation of the agent action. Each voluntary effector needs a corresponding sensor. This sensation is very important because it informs the agent its current status which may affect its later behavior. One way of treating these two modalities is to put the sensation vectors together as a single last context entering the IHDR tree. However, we will see that such a naive way does not work.

First, we need to understand that a last context covers a short duration instead of a single time instance. That means even when an utterance is finished, there is still some non-silent auditory sensation presented to the robot. The same thing happens to the action sensation when each action lasts more than one time instance. This is shown in the legend of Fig. 6.7. We represent each last context with a color bar. Since there are two sensing modalities, the last context at each time instance can be described by a color bar pair. For example, the first last context shown in Fig. 6.7 is $< indigo, white >$, where *indigo* represents the last auditory sensation and *white* represents the last action sensation. The external action is also presented as a color

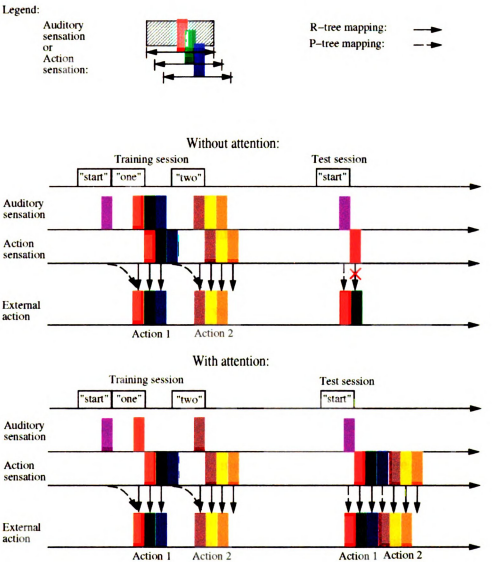


Figure 6.7: The comparison of a system with and without cross-modality attention. This figure is presented in color.

bar in Fig. 6.7. Notice that the sensed action is the external action at the last time instance. Therefore, the action sensation is a shifted bar sequence of the external action. In general, a white bar means no sound or no action is sensed or executed.

During training, the teacher gives all the three commands, "start" (C_c), "one" (C_{s1}), and "two" (C_{s2}). The R-tree learns the immediate mapping from a last context to a primed context which includs a primed action. This mapping is shown by the solid arrows in Fig. 6.7. The P-tree learns a mapping with prediction to a near future as shown with the dash arrows in Fig. 6.7. The choice to execute the primed action of the R-tree or the P-tree is made by the value system.

Now we take a look at a system running without a cross-modality attention. During testing, after the utterance of "start," the P-tree successfully gives the mapping $< indigo, white > \rightarrow < red >$ and the value system chooses the output of the P-tree as the external action. Everything is perfect so far. Unfortunately, the last context configuration in the next time instance, $< white, red >$, has never showed up during training, leading to unpredictable behavior. The new task skill is not acquired.

In fact, a cross-modality attention mechanism will resolve the problem. Notice that although $< white, red >$ has never appeared during training, the red action bar is always followed by the green action bar. Suppose the robot pays attention to its action sensation only (the red action bar) after it starts to move during training. Then the green and blue bars of auditory sensation will not appear in the auditory sensation. Now we have the last context configuration of $< white, red >$, which is mapped to the external action $< green >$ by the R-tree. When action 1 is finished, the auditory sensation should be restored to sense the utterance of "two" and the

similar switch to action sensation is needed after the action 2 starts to be executed. In general, it is unlikely to have a matched global context. Selective attention brings related information to the recognition process.

In summary, we need a mechanism that can switch between the two sensation channels. The principle is that the channel with higher short-term variation is more salient and would receive the exclusive attention. To do this, we designed a simple cross-modality attention module, the channel selector shown in Fig. 6.4. The standard deviation $\sigma(t)$ of the magnitude $m(t)$ of the sensation from each modality is computed recursively as,

$$\sigma(t) = \frac{t-l-1}{t}\sigma(t-1) + \frac{1+l}{t}m(t),$$

where l is the amnesic parameter to give more weight to the new samples. With an appropriate l, $\sigma(t)$ would represent the short-term variation of the sensation. The channel selector implements a "winner-takes-all" criterion by blocking the channel with lower $\sigma(t)$.

6.3.4 Algorithm

As a summary of above, at each time instance, the agent executes the following learning process. Since there is no explicitly separated training and testing sessions, the algorithm is executed all the time.

1. Collect the sensation from both the auditory sensor, $x_s(t)$, and the action sensor, $x_a(t)$.

2. The channel selector forms a single sensation vector (the last context), $l(t) =$

$(x_s(t), x_a(t))$, by replacing $x_s(t)$ or $x_a(t)$ with a zero vector.

3. Both the R-tree and the P-tree conduct learning from $l(t)$, using IHDR tree.

4. The best-matched states, or the prototypes, $s_R(t)$ and $s_P(t)$, are retrieved from two trees together with their associated primed contexts, $p_R(t)$ and $p_P(t)$, respectively.

5. If there is an imposed action, $a_i(t)$, the primed action part of $p_R(t)$ is set to be $a_i(t)$ and its value is set to be 1. Otherwise, the primed action part of $p_R(t)$ is set to 0 (the default non-action) and its value is set to be 0, too.

6. Compute the confidence index for both trees and decide the external action, $a_e(t)$. Its associated value is termed as $Q_e(t)$.

7. The primed context $p_P(t)$ of $s_P(t)$ enters the PUQ and each entry in the PUQ is updated according to the model (6.4). Specifically, the sensation part is updated using $s_P(t)$, the action part using $a_e(t)$, and the value part using $Q_e(t)$.

8. Send $a_e(t)$ to the corresponding effectors.

6.4 Multilevel Architecture

The above described system learns the basic actions through supervised learning. While supervised learning has the advantage of efficiency, the system is tedious to teach and does not allow "extinction" or teacher errors. For example, suppose the robot has been taught to lift its arm when hearing the command "one." Later on, the

teacher does not want the robot to lift its arm any more, which is known in psychology as "extinction." Therefore, we need to add a reinforcement learning mechanism into the system.

One might think that this can be done by simply feeding reinforcement signals to the system and modified the prediction model for updating Q into the following one:

$$Q^{(n)}(t) = Q^{(n-1)}(t) + \frac{1+l}{n}[r(t) + \gamma Q^{(n-1)}(t+1) - Q^{(n-1)}(t)]$$

In this way, one can tune the Q value in the primed context through the reinforcement signals and, consequently, tune the behavior of the system.

However, this straightforward strategy did not work well. Because of the prediction model (6.4) and the PUQ, the primed context of a particular primitive prototype would be propagated to other prototypes so that multiple primitive prototypes may have a similar primed context. However, in a real application, the exact context is not guaranteed to be repeated, which means not all the propagated primitive prototypes may show up in the following training session. As a result, while we may adjust the primed context of certain primitive prototypes, we may not be able to adjust all the propagated ones. This is the "abstraction-from-signals" issue in the new challenging AMD mode.

To solve this problem, we designed the architecture shown in Fig. 6.8. We call the whole module of Fig. 6.4 as a level building element, LBE. The new architecture has two levels of LBEs. All the LBEs run the algorithm of Section 6.3.4 except that the second-level one takes the primed sensations of the lower-level one as the

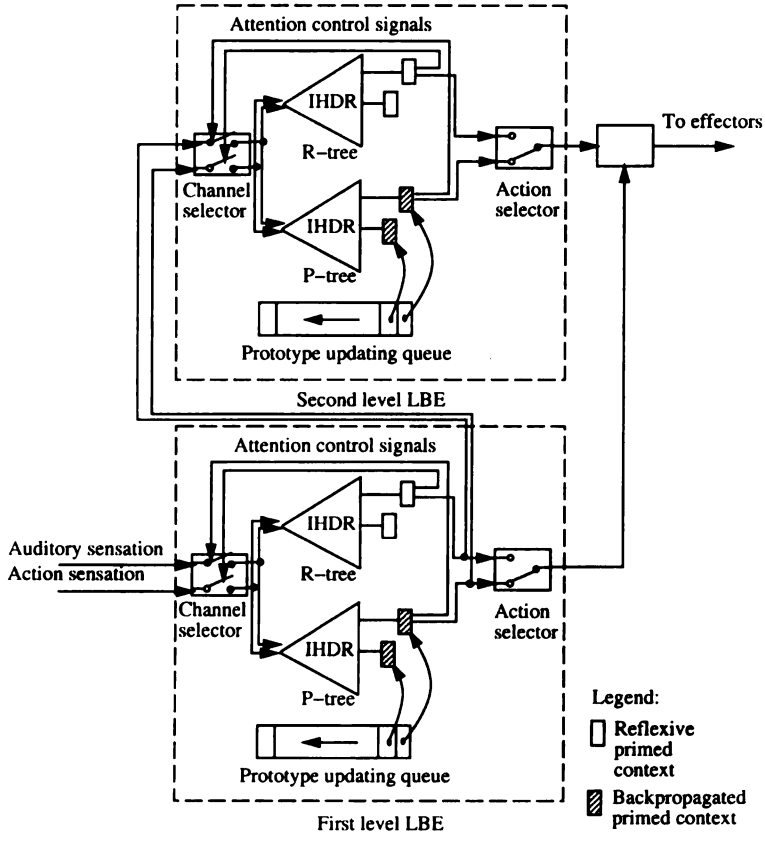


Figure 6.8: A two-level architecture of SAIL.

input. The underlying idea of the two-level system is as follows. Because of the prediction model (6.4), the primed sensation vectors are the averaged version of its future context vectors. This means that, the vectors which used to fall into different primitive prototypes in the first level LBE L_1 may be grouped into one primitive prototype of the second level LBE L_2, as shown in Fig. 6.9. As a result, even though it may be rare for a particular primitive prototype to be revisited in L_1, it is very likely for L_2 to cover the context represented by this primitive prototype when a similar context appears.

To make decisions on actions from multiple levels, L_2 checks the Q value of the action that is closest to the one selected by L_1. If it is larger than zero, the corre-

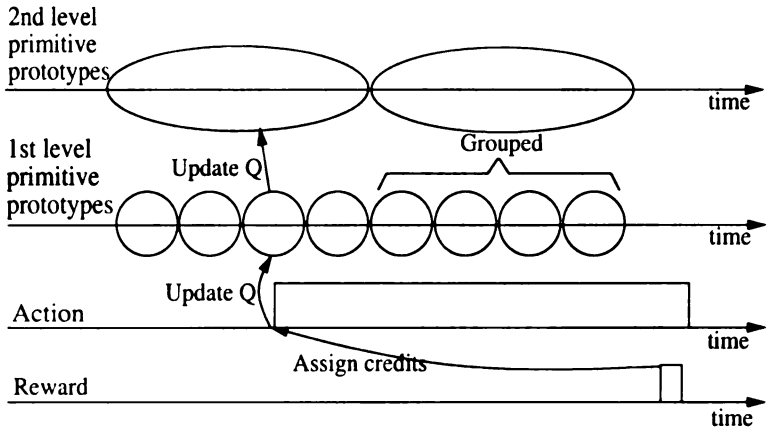


Figure 6.9: Internal mechanism of the two-level architecture.

sponding control signals would be sent to the effectors. Otherwise, the control signals would be blocked. In other words, no action would be taken. The Q value given by L_2 will enter L_1 as an internal reward so the behavior of L_1 will be tuned even if there is no constant environment feedback.

In the current implementation, the R-tree of L_2 was not used. It will be used when a higher level is built. To quickly assign the reward to the right context, a credit-assignment criterion is preprogrammed by targeting the reward to the starting point of a recently conducted action, as shown in Fig. 6.9.

6.4.1 Algorithm

In summary, the following is the developmental learning algorithm of the two-level system at each time instance.

1. Collect the sensation from both the auditory sensor, $x_s(t)$, and the action sensor, $x_a(t)$.

2. For L_1, do steps 2 through 5 of the algorithm of Section 6.3.4. Denote the

primitive prototypes retrieved by R-tree and P-tree as $s_{R1}(t)$ and $s_{P1}(t)$, respectively. Find the primed context with highest confident index among the primed contexts associated with $s_{R1}(t)$ and $s_{P1}(t)$ and denote it as $p_1(t)$.

3. Take the primed sensation part of $p_1(t)$ as the input to L_2 and do step 2 through 5 of the algorithm of Section 6.3.4 for L_2. Denote the primitive prototype retrieved by P-tree as $s_{P2}(t)$. Find the primed context of $s_{P2}(t)$ with the primed action part most similar to that of $p_1(t)$ and denote it as $p_2(t)$. Let $p_2(t)$ enter the PUQ of L_2 and update according to model (6.4). For updating Q, use model (6.6).

4. If the Q value of $p_2(t)$ is larger than zero, send the primed action part of $p_1(t)$ to the corresponding effectors. Otherwise, send the zero vector to the corresponding effectors.

5. Let $p_1(t)$ enter the PUQ of L_1 and update according to model (6.4). For updating Q, use

$$Q^{(n)}(t) = Q^{(n-1)}(t) + \frac{1+l}{n}[r(t) + Q_2^{(n-1)}(t) + \gamma Q^{(n-1)}(t+1) - Q^{(n-1)}(t)]$$

where $Q_2^{(n)}(t)$ is the Q value of $p_2(t)$.

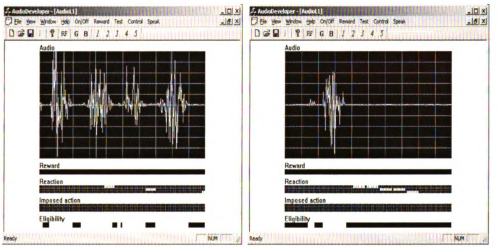
From the above discussions, we can see that the role of L_2 is effectively a module evaluating the behaviors of L_1. This means that our system have some characteristics of the well-known actor-critic methods [90, page 151]. The major differences here are (1) our method starts from raw sensors instead of symbolic input, (2) both the "critic"

L_2 and the "actor" L_1 learn from the environment, (3) the critic is not task-specific. It gives the system a quicker response to the change of the environment by not letting the actor wait until the critic realizes the change. Another important characteristic of our system is that the "critic" and the "actor" have similar architecture and learn in the same manner, which makes the architecture move systematic. L_2 was motivated by higher-order cortex in the biological brain [42] but further discussion is beyond the scope of this paper.

6.5 Experiments on AudioDeveloper

The task transfer performance of the agent model has been validated on a real-time software agent, called AudioDeveloper. This software testing and evaluation environment provides detailed performance record that is not readily available from a real robot. AudioDeveloper has two kinds of sensors, a microphone and the touch sensors (the buttons on the toolbar of the GUI of AudioDeveloper). The touch sensors are used by the trainer to impose actions. The agent has another internal sensation to sense its own actions. A simulated 3-joint robot arm is the effector. In the GUI (Fig. 6.10), the top panel shows the audio waves. The reactions and the imposed actions of the agent are shown in the third and the forth panel of the GUI, along the horizontal time axis. The time interval between two consecutive time instances is about 18.1ms. There are three simple behaviors (A_{s1}, A_{s2}, and A_{s3}) corresponding to three voice commands, (C_{s1} ="one," C_{s2} ="two," and C_{s3} ="three"), respectively. The behaviors are identified by the component of the action vector with the maximum

value. For example, if the first component of the action vector has the maximum value, it is identified as A_{s1}. The new task A_c is defined, in the teacher's mind, as $A_{s1} \rightarrow A_{s2} \rightarrow A_{s3}$ and the corresponding voice command (C_c) is "start." The developmental program was written without any task-specific information.



(a) (b)

Figure 6.10: The GUI of AudioDeveloper. (a)During online learning; (b)After online learning.

Before learning the new tasks, the agent should be able to perform the simple actions. So, the whole experiment are divided into two steps: (1) The agent learns simple skills following voice commands; (2) The agent learns complex skills following new voice commands through task transfer. All the learning of the above two steps is done in the same mode as required by AMD, i.e., real-time physical interactions between the agent and the trainer. As we discussed above, whether an interaction is for training or testing is only in the mind of the teacher. The same algorithm works in a single mode all the time. In the following discussion, when we mention the test session, we simply mean "freezing" any changes to the cognitive mapping module so that we may evaluate the system.

The training process of step (1) is as follows,

1. A voice command is spoken to AudioDeveloper.

2. At the end of the utterance, a corresponding action is imposed by pressing a button in the tool bar of AudioDeveloper.

3. Wait for a couple of seconds and go back to 1. As long as two consecutive commands and actions are separated by a sufficient amount of time, it does not matter how long the separation time is.

The above training process is a grounded speech learning step, which has been discussed in more details in [111]. While a supervised learning procedure was conducted in this experiment, it can also be done with reinforcement learning [109]. After training, the voice commands were spoken to AudioDeveloper again with no actions imposed. As we will see in the experimental results below, AudioDeveloper correctly conducted the appropriate actions after the commands, "one," "two," and "three."

The training process of step (2) is as follows,

1. A command, "start," followed by "one," "two," and then "three," is spoken to AudioDeveloper.

2. Wait for a couple of seconds and go back to 1.

This is essentially a process that the trainer teaches the system to do the new task through verbal instructions. Fig. 6.11 shows a fraction of this training session. In Fig. 6.11, the upper panel presents the energy of the voice commands along the time axis. The next three panels display the 3-D action vectors of the R-tree, the P-tree,

and the enhanced cognitive mapping. The behaviors of the robot were identified by the component of the action vector with the maximum value. For example, if the first component of the action vector had the maximum value, it was identified as action 1. The fifth panel of GUI shows the confidence of the R-tree and the P-tree on their outputs. The bottom panel shows the attention of the system over the auditory sensation and action sensation channels. The sequence of the external action vectors shows that the system responded to commands C_{s1}, C_{s2} and C_{s3} accordingly.

Let us take a closer look at how the system behaved around time instance 100 when the command "one" was being spoken (Fig. 6.12). Note that the confidence subplot is shown in a semi-logarithmic way in this figure. At time instance 98, the P-tree started to show significant trend to conduct action A_{s1} and it had a higher confidence than the R-tree. So the enhanced cognitive mapping fired the external action A_{s1}. Once the action was started, the attention was switched from the auditory sensation channel to the action sensation channel. With the sensation for the starting of action A_{s1}, the R-tree began to have higher confidence and kept conducting action A_{s1} until time instance 104 when the R-tree decided to stop the action. For a better view of the transition of the confidence of the R-tree and the P-tree between time instances 100 and 104, the reader is referred back to Fig. 6.11.

After training, when only the composite command "start" was given, the system successfully repeated the action sequence, $A_c(= A_{s1} \rightarrow A_{s2} \rightarrow A_{s3})$, as shown in Fig. 6.13. Again, let us give the system behavior a closer look in Fig. 6.14, which expands the time axis round time instance 460[1]. During training, the utterance of

[1]The confidence subplot is shown as semi-logarithmic.

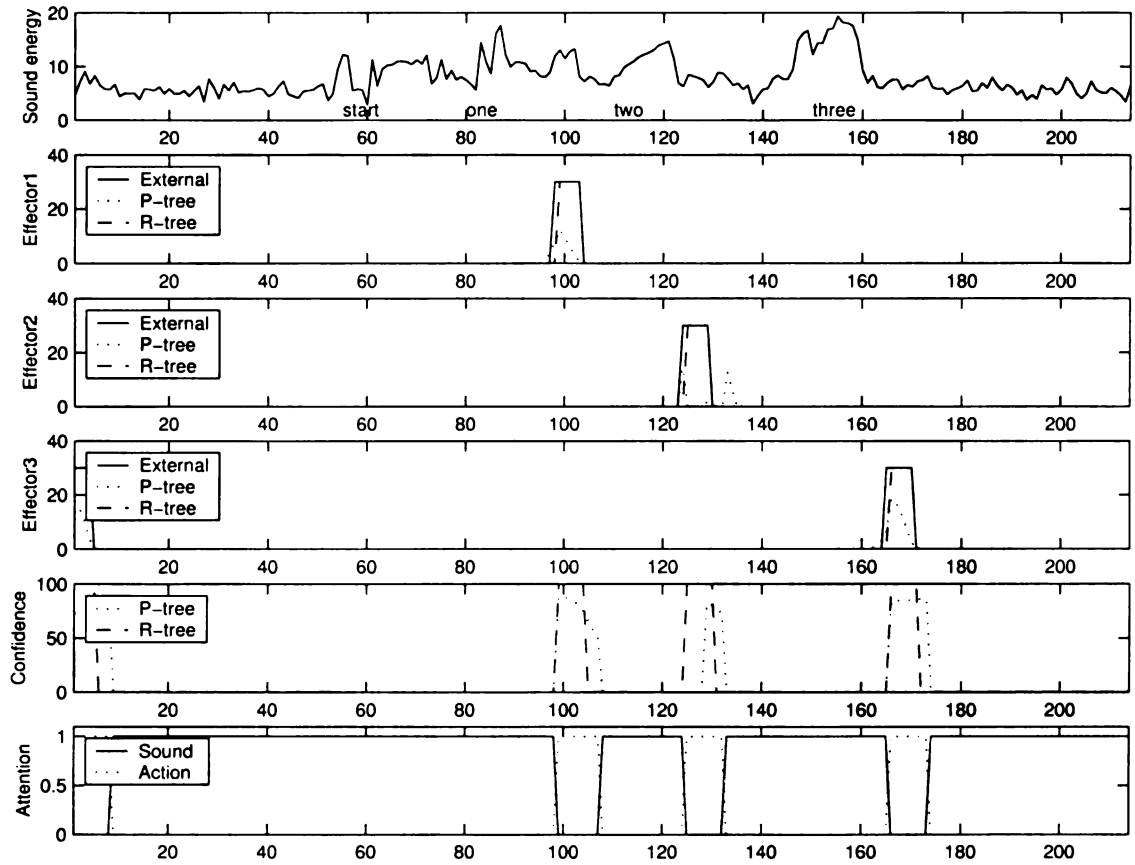


Figure 6.11: A fraction of the training session of step (2).

"start" was always followed by the utterance of "one." Because of the backpropagation of the primed context, upon hearing "start," the P-tree learned to produce the primed context associated with "one." Therefore, without hearing "one" during testing, the P-tree started to have significant trend of executing action A_{s1} after hearing "start" at time instance 464. Since the P-tree also had higher confidence than the R-tree at the time, action A_{s1} was fired as the external action of the enhanced cognitive mapping. The attention was then switched to the action sensation channel at time instance 465 and the R-tree took over until action A_{s1} was stopped at time instance 471 since it had a higher confidence (see Fig. 6.13 for an expanded view of

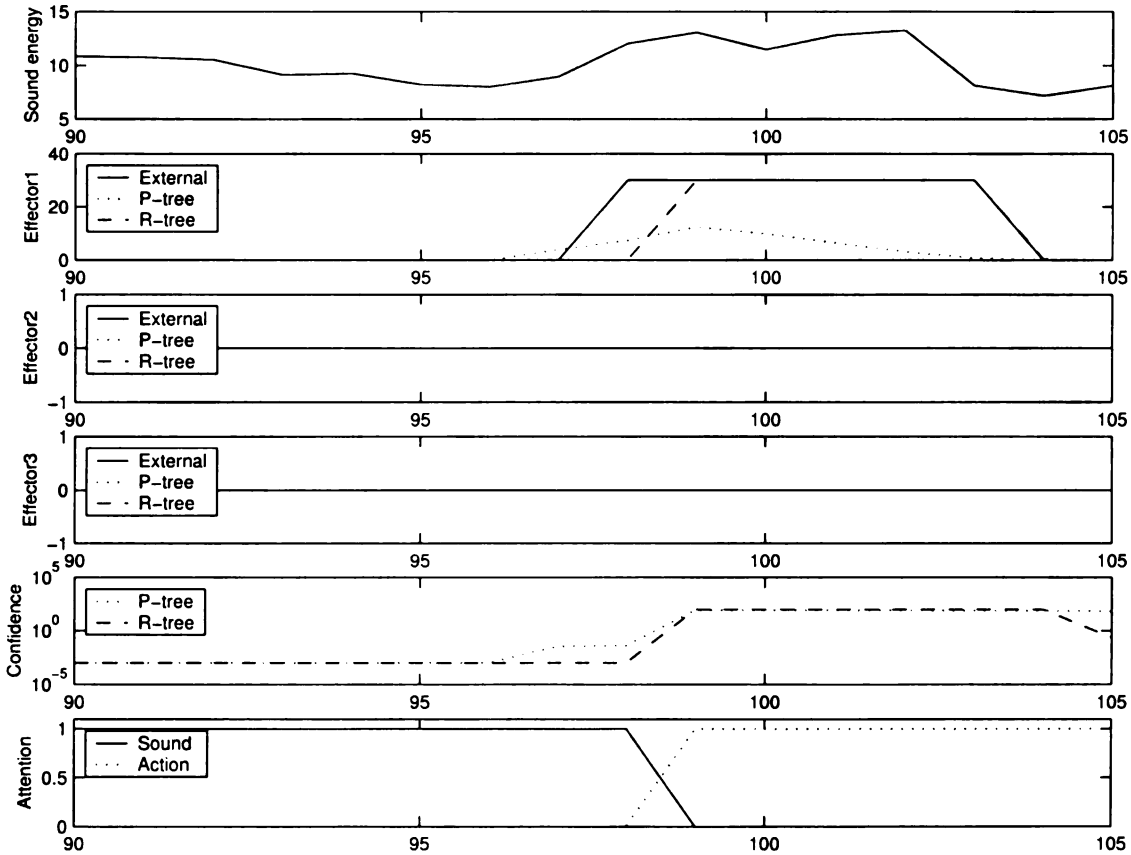


Figure 6.12: A closer look at the training session of step (2).

confidence.).

With the numerical representation, an utterance of a voice command typically corresponds to multiple primitive prototypes saved in the IHDR tree, depending on the duration and the auditory changes of the utterance. During training, the primed context of all these multiple prototypes may be changed because of the primed context backpropagation. Therefore, there were more than one cases that the P-tree could start action A_{s1}. If the simple action lasted too short, this simple action would be fired for a second time. This is why we see the action was repeated after the system finished it once at time instance 472. In the real world, the duration of an action is typically longer than an utterance. So such action repetition would be avoided as we

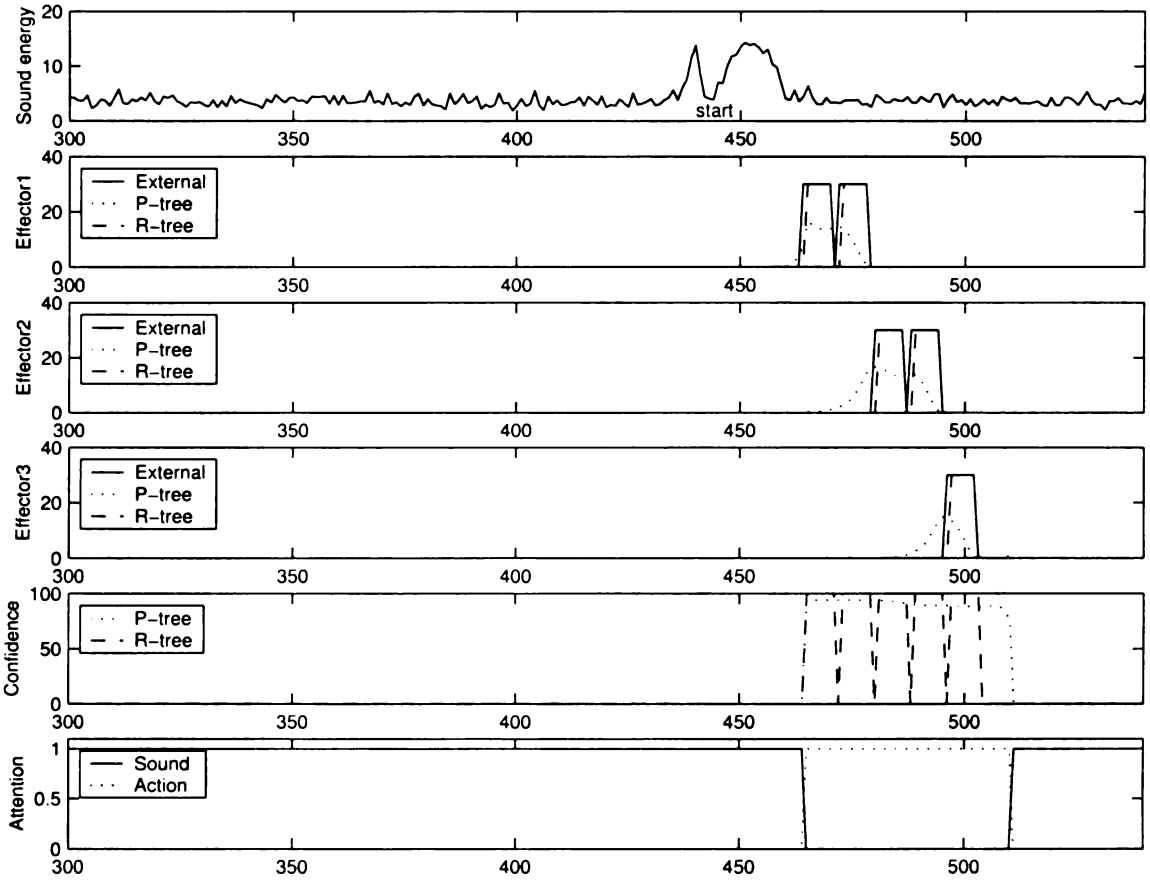


Figure 6.13: A fraction of the test session of step (2).

would see in the experiment on a real robot below.

Similar to the case that "start" elicited action A_{s1}, during training, the P-tree "saw" that action A_{s1} was followed by the utterance of "two" for many times. Therefore, the primed context of the utterance of "two" was backpropagated to the last context of seeing action A_{s1}. As a result, the P-tree started to show a high confidence of conducting action A_{s2} at time instance 480 although "two" was not presented. The above process was repeated until the end of action A_{s3}, which means the agent successfully learned to conduct the new task.

Multiple-user results. Above experimental results were obtained with the inter-

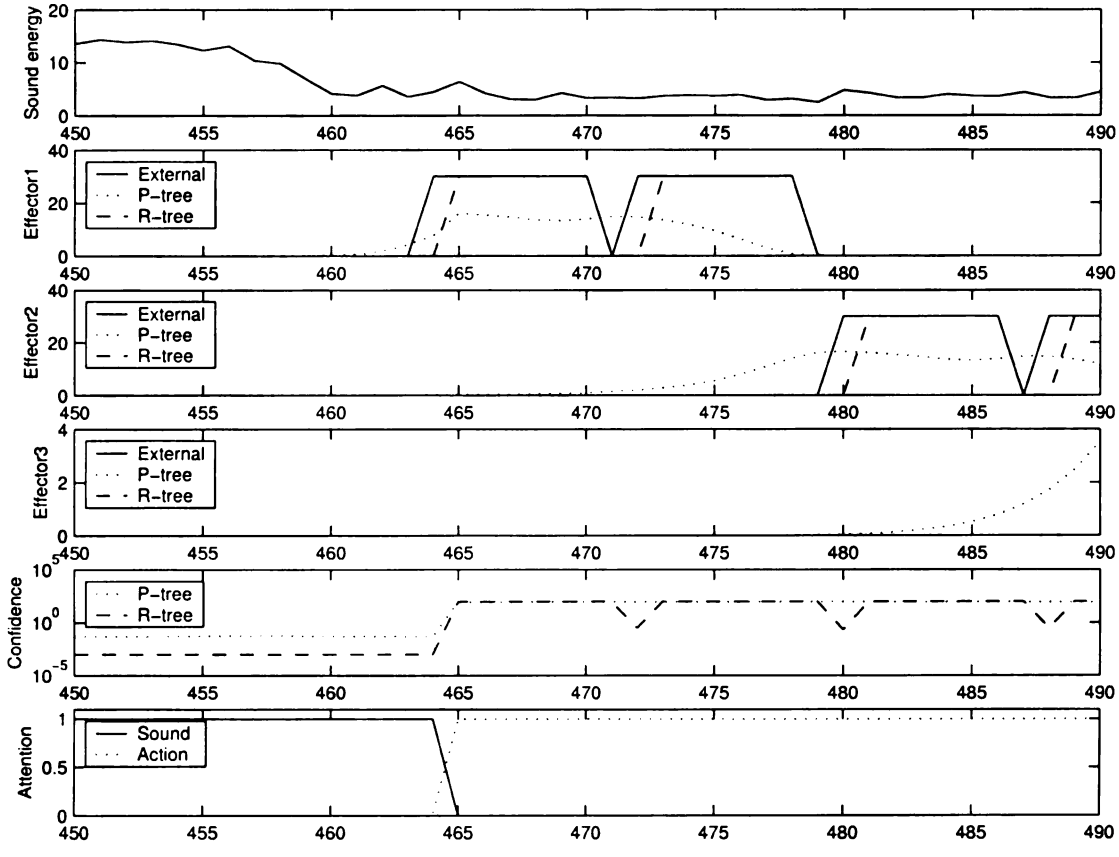


Figure 6.14: A closer look at the test session of step (2).

action between AudioDeveloper and one trainer. To test AudioDeveloper with wider variance, we conducted more experiments. To save the human efforts in training AudioDeveloper, first we built another software agent, AudioTeacher, as a "teacher." It can play pre-recorded sound and give imposed actions to AudioDeveloper.

The auditory data was taken from the number data set contributed by 63 persons with a variety of nationalities (American, Chinese, French, Indian, Malaysian, and Spanish) and ages (from 18 to 50). Each person made five utterances for each of the ten numbers, one to ten. In the experiment here, we only used the utterances of "one," "two," and "ten," where the first two were the simple commands and the last one was the composite command. Among the five utterances of each number, three

Table 6.1: Performance of AudioDeveloper trained by multiple users. C.R.1 and C.R.2 represent the correct rate for action A_{s1} and the new task, respectively.

Epoch No.	6	7	8	9	10	11	12	13
C.R.1 (%)	42.86	57.14	60.32	71.43	74.60	74.60	74.60	77.78
C.R.2 (%)	1.59	9.52	26.19	38.89	42.06	57.94	67.46	73.81
Epoch No.	14	15	16	17	18	19	20	
C.R.1 (%)	77.78	77.78	77.78	77.78	77.78	77.78	77.78	
C.R.2 (%)	76.98	81.75	84.92	84.92	89.68	91.27	92.86	

were used in training and two were used in testing. In effect, in the step (2) testing, only the utterances of the composite command were needed.

AudioTeacher was programmed to lead the two-step training process as we discussed above. Since we did not have enough speech data from each subject, the data was replayed after each round. We call one cycle through the data set "an epoch." In the step (1) training, one epoch was sufficient for AudioDeveloper to acquire the simple actions. In the step (2) training, multiple epochs (reviews and practices) were needed for AudioDeveloper on the new task. We stopped the training after 20 epochs. Starting from epoch six, the cognitive mapping were saved after each epoch for later performance evaluation.

The entire training process took about 6 hours for the 63 subjects' data. The performance of AudioDeveloper was evaluated as follows. After a composite command was spoken, if AudioDeveloper made action A_{s1} for one or several times followed by action A_{s2} for one or several times, we counted it as a success. Otherwise, it was a failure. Table 6.1 shows the percentage of successfully acquiring the new task after each epochs. After 20 epochs, AudioDeveloper learned the new task reliably with a correct response rate of about 93%. Also shown in this table are the success rates for action A_{s1}. As expected, A_{s1} was acquired earlier than the acquisition of the new

task. The reason it did not reach very high percentage was that there were cases where the acquisition of both action A_{s1} and the new task occurred within one epoch. In these cases, the success of acquiring action A_{s1} was not counted.

The failed cases were related to three subjects. We examined them one by one and found that the reasons for failure were the same. These three subjects' utterances were very short. Since AudioTeacher played the utterances one after the other in the step (2) training, the utterances of these three subjects tended to be very close, which prevented them from being recognized correctly. The problem was if the simple commands were not recognized correctly, the step (2) training was not really effective. After adjusting the separation time between the utterances for these three subjects, the new task was successfully acquired after about ten epochs. A lesson we learned here is that it is important to provide sufficient pause time between tasks to avoid unwanted association, a phenomenon well recognized in animal learning.

6.6 Experiments on the SAIL robot

SAIL shown in Fig. 6.1 (I) is a human-size mobile robot house-made at Michigan State University. It has a drive-base, a six-joint robot arm, a neck, and two pan-tilt units, on which two CCD cameras (as eyes) are mounted. A wireless microphone functions as an ear. SAIL has four pressure sensors on its torso and 28 touch sensors on its eyes, arm, neck, and bumper. Its main computer is a dual-processor PII PC workstation with 512 MB RAM and an internal 27 GB three-drive disk array. All the sensory information processing, memory recall and update as well as effector controls

are done in real-time.

We have conducted some experiments on the SAIL robot by interacting with it through its auditory sensor, a microphone, and the micro-switch sensors on its arm. One of the switch sensors were defined as a biased sensor to accept reinforcement signals. Specifically, if the reading of that biased sensor was positive, the reward was 1, and if the reading was negative, the reward was -1. Other two switch sensors were used to impose the four basic arm actions, drawing four petals as shown in Fig. 6.1 (II).

We first taught the robot the basic actions in the same way as we did in Section 6.5. Specifically, the training process went as follows: (1) The trainer spoke one of the voice commands ("one," "two," and "three," in this experiment). (2) At the end of the utterance, the trainer pressed the switch sensor of SAIL to impose one of the four actions. (3) Wait for a couple of seconds and go back to (1). The trainer might need to repeat some of the commands for SAIL to practice. But according to our observation, within three repeats, a single behavior can be established with the correct rate of 95-100%. This type of fast learning is not possible with iterative mapping learning algorithms such as artificial neural networks. Thanks to IHDR which enables one-instance learning.

6.6.1 Behavior extinction

We first report an experiment showing how to extinct the established behaviors of the robot. Suppose the robot has learned to draw the upper-petal after hearing "one"

and, later, the teacher does not want it to move the arm after the same command. The training process goes as follows: (1) The trainer speaks "one." (2) If the robot moves its arm, the trainer presses a reward switch sensor to give a punishment. (3) Wait for a couple of seconds and go back to (1).

To trace the internal changes of the robot, we saved the information related to decision making at every time step at an interval of about 20 millisecond. The upper panel of Fig. 6.15 shows the sound volume along the time axis. The vertical dash lines show the time instances when the robot started the upper-petal drawing. Among them, the first one is when the trainer imposed the action. The lower panel of Fig. 6.15 shows the changes of Q values of two actions associated with a particular primitive prototype. A non-action means the robot stays still and action 1 means upper-petal drawing. In the real-time experiment, there were typically more than one thousand primitive prototypes saved in each IHDR tree. We got hold of the best matched primitive prototype when it was retrieved. For the time instances when a particular primitive prototype was not retrieved, we simply show its Q value when it was retrieved previously. Therefore, the Q value of a particular prototype-action pair changed only when the prototype was visited. This was also the reason that the changes of the Q values shown up much later than the reinforcement signals. However, we do see the Q value of action 1 starting to grow after the action was imposed. At the same time, that of the non-action dropped. This means the robot gradually preferred action 1 to the non-action. After receiving a punishment reinforcement signal at the time instance shown by the vertical dash line in the lower panel of Fig. 6.15, the robot's preference reversed by the changing Q values. From the upper panel, we can

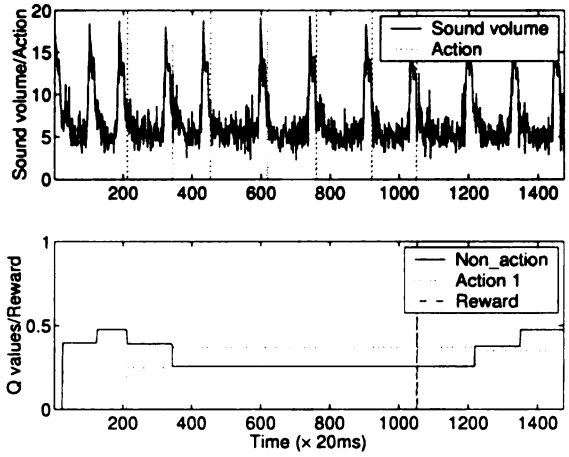


Figure 6.15: The behavior changes of SAIL with the two-level architecture.

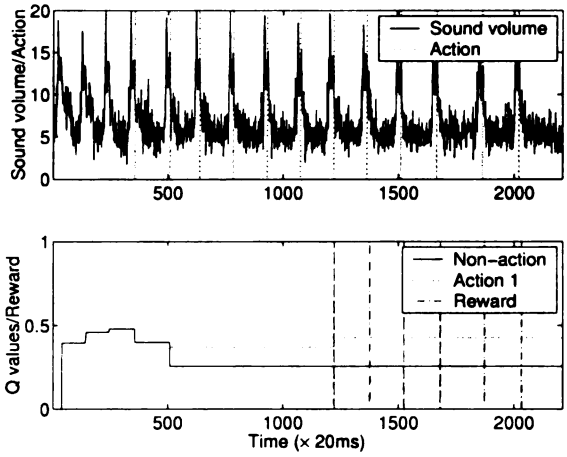


Figure 6.16: The behavior changes of SAIL with the one-level architecture.

see that the previous established behavior was extinct thereafter.

For comparison, we did the same experiment on a single level system. As shown in Fig. 6.16, although the system can establish the actions, it took a much longer time to extinct the established behavior, which shows the power of multilevel architecture.

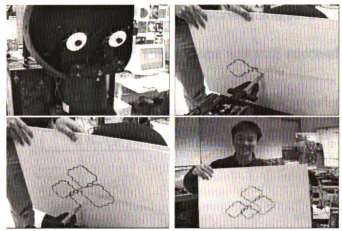


Figure 6.17: The SAIL robot learned the new task after verbally instructed by human trainers.

6.6.2 Task transfer

In this experiment, we teach the robot to do the composite action. The training process goes as follows: (1) A trainer speaks "start," followed by "one," "two," "three," "four." (2) Wait for a couple of seconds and go back to (1). This is essentially a process that the trainer teaches the system to do the composite action through verbal instructions. After repeating the procedure several times, the trainer speaks only "start." If the robot continuously do the four actions successfully, we count it as a success. Fig. 6.17 shows that the SAIL robot learned to execute an extended action sequence upon a new stimulus (the new command) by transferring previously learned skills.

We repeated 20 times the experiment of teaching SAIL drawing individual petals and then drawing the 4-petal flower upon learning the new start command. Table 6.2 shows that there was only one case among 20 when the robot failed to learn the new task. In that particular case, the robot confused the command "start" with

the command "three" because their beginning portion sounded similar. Even in this failed case, the invokation of the first two actions was still successful.

Table 6.2: Performance of SAIL doing task transfer

No. of Actions	1	2	3	4
C.R. (%)	100	100	95	95

6.6.3 Execution time and memory size

The IHDR trees keep only clusters, instead of samples. Typically, the more complex the sound distribution is, the larger the trees are. And larger trees size means (slightly) longer retrieval time because the IHDR has a logarithmic time complexity. To push the system to its extreme in terms of tree size, we played loud music to it and recorded the execution time of above computations within each time step.

Fig. 6.18 shows that the execution time tended to grow at the beginning and it flattened out after about 100 seconds. Overall, the execution time of each time cycle is well under the interval of 18.1ms, though it varies all the time because a primitive prototype may be stored at and retrieved from different depths of the trees. The total size of the three trees reached about 300MB after 550 seconds of "music listening." We do not expect that the time is a major issue for very large IHDR trees because of its logarithmic time complexity. To provide an idea about the structure of the trees, some structure data about the resulted IHDR trees is shown in Fig. 6.19, where the horizontal axes denote the depth of the tree and the vertical axes are the number of nodes or the number of primitive prototypes saved. The P-tree of L_2 is significantly smaller than the R-tree and the P-tree of L_1 because of the "grouping" we mentioned

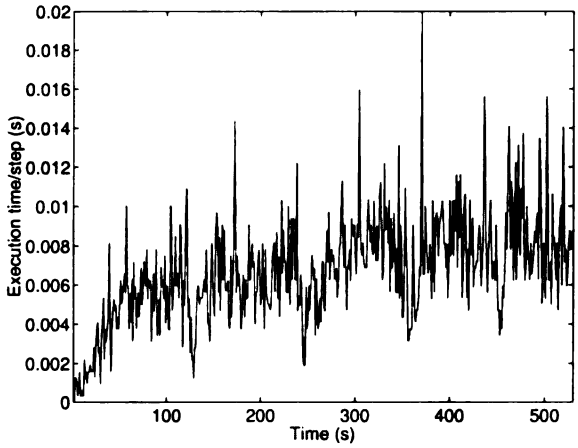


Figure 6.18: The average execution time of the two-level system in each execution step is lower than 18.1ms, the required interval of each speech frame.

in Section 6.4.

6.7 Conclusions

The capability of transferring acquired skills from one task to another setting is crucial for a robot to autonomously learn complex cognitive behaviors. In this report, we presented principles, architecture, and computational framework of a developmental agent capable of task transfer. Our experiments showed that, it is theoretically sound and experimentally practical to construct an artificial device that can autonomously learn new complex cognitive skills by applying its previously learned skills. The implication of this conclusion lies in the fact that everything is done online in real time, without a human programmer to supply a task-specific representation.

A major contribution of this work is that the learning process of the agent can be conducted through online real-time interactions between the agent and trainers, which has not been achieved by any previous research. The high-dimensional numerical rep-

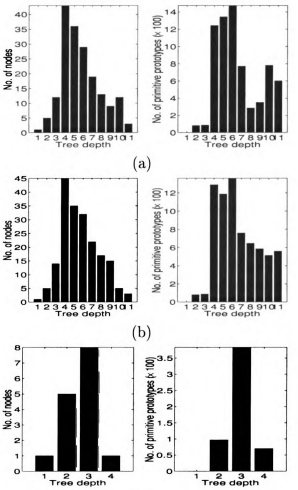


Figure 6.19: The node distribution and primitive prototype distribution in the trees: (a) R-tree of the first level LBE; (b) P-tree of the first level LBE; (c) P-tree of the second level LBE.

resentation made this possible although it posed very challenging technical issues of high-dimensional incremental regression. Thanks to the grounded speech learning capability, all the perception and the actions, including even the actual number of their classes, need not be available before the programming was finished. The advantage of this architecture is that the artificial agent has a great flexibility for perception and behavior learning.

The missing context problem requires the robot to look ahead or use the knowledge

about what will happen in the future. A system using future knowledge is not causal and can not be implemented directly. We followed the idea of Q-learning and proposed a model to do backpropagation over the primed contexts. This backward model effectively realized a prediction mechanism and enabled the robot to tolerate missing some context while doing task transfer.

The multilevel architecture resolved the abstraction issue on perception and facilitated the behavior adjustment process.

Probably having implication to the understanding about how humans form abstract symbolic concepts from numerical sensory inputs, we have demonstrated that symbolic behaviors, including complex phenomena such as task transfer, can emerge from non-symbolic distributed vector-type internal representation. There is no clear boundary between audition, speech, and language in such a developmental robot. This idea has been successfully used in a later research project reported in the next chapter, where a robot integrates vision and audition in a dynamic world via online dialogue.

Chapter 7

Semantics Learning through Multiple Modalities

7.1 Introduction

While communication seems so natural and easy for human beings, the situation in the counterpart, machines, is so much worse. Many efforts have been made to duplicate the fascinating language capability of human on an artificial agent. However, even though we see the performance of automatic speech recognition (ASR) technologies getting constantly improved in the past decade, in terms of communication, none of the existing systems is even close to the capability of a three-year old kid.

Language is composed of two inter-related components: syntax and semantics. Syntax studies the structure of well-formed phrases (spelled out as sound sequences). Semantics deals with the way syntactic structures are interpreted. In some sense, syntax plays the role of a protocol in communication while semantics is the content.

If two people share the same syntax and semantics, they can communicate with each other. Between syntax and semantics, the latter seems more essential for communication. We all have the experience that two people from different countries can have some simple communication even neither of them know the other's language. The acquisition of semantics is a very challenging part of machine language learning.

Semantics is the meaning of a string in a language [2], shared by the users of the language. So, the acquisition of semantics is the development of this shared understanding. There is not a common definition of "meaning" or "shared understanding" among cognitive scientists yet [96], which partially contributes to the difficulty of machine language learning. Here we would like to propose a working definition. All the activities of an agent can be regarded as the motor control driven by the sensory input. This is quite obvious for external behaviors such as locomotion and speech while it is also true for internal behaviors such as visual attention [101]. Our definition of understanding meaning or semantics is to conduct appropriate behaviors under appropriate context. By context, we mean a configuration of available information that an agent uses for making sense of language in particular situations.

Under this definition, a semantics acquisition process is a process of internalizing and organizing the context while a communication process is a process of retrieving appropriate context and producing corresponding verbal behaviors. As we have been discussing in the previous chapters, during both of these two processes, an important requirement is grounding [40] [28] [14]. Grounding means that representations inside machine should be connected to their referents in the external world. For example, the representation of "dog" should be related to the presence of actual dogs in the

environment. Grounding is accomplished through real time sensory experiences. The fact that people speaking different languages can communicate with each other is a side support to the importance of grounding: since people share the same physical world, we develop similar semantics and we can communicate even we do not have the same syntax.

In this chapter, we present an embodied system that acquires simple semantics in real time. We explore some of the difficulties in the process of grounded semantics learning and propose an computational model to resolve the challenges.

7.2 Problem description

Semantics is grounded upon rich sensory inputs. In the process children developing language capability, they take in information through all the senses - sight, hearing, smell, touch, and taste, and they integrate the information and act upon it. There are evidence showing that if visual, auditory, and tactile inputs never have the chance to occur together, there is no opportunity to develop an integrated linkage between what is seen, heard and felt [31]. Therefore, our study on robot semantics learning is based upon multimodal information.

In communication, we make sense of what someone says (semantics) based on whatever information resources we think are relevant. These contextual resources are likely to be found in such things as [55],

- the physical surroundings;
- the past shared experience and relationship of the speakers;

- the speakers' shared tasks or goals;

- the speakers' experience of similar kinds of conversation.

Retrieving contextual information from different resources involves complex mental activities, which a young baby is not competent of. Neither is a early-stage learning system. So, we limit the contextual resources of a robot to immediate physical surroundings or situations at this research stage.

In summary, we would like a robot to conduct appropriate behaviors given a similar visuoauditory context. In particular, we present a system that can answer verbal questions appropriately, given visual stimuli of some objects. This is a simple semantics acquisition process, which requires the robot to develop visual and auditory perception and associate visuoauditory context with appropriate behaviors, all in real time. We have presented real-time audition learning in previous chapters. However, real-time multimodality learning has its own challenges.

Visual representation of objects. To interact successfully with objects in the environment, the robot must be able to recognize them across changes in their orientations. The rich studies of visual representations of objects can be divided into two major types: object-based models and image-based models. The former ones suggest that objects are represented as structural descriptions of their 3-D parts and the relations between those parts in a manner that is independent of the objects' orientation relative to the observer [54]. The latter ones propose that objects are represented as a set of 2-D views or snapshots taken as specific orientations relative to the observer. While the former models sound intuitively more plausible to reach

view-point invariance, the latter ones receive more supports by recent psychophysical and neurophysiological studies [12] [52] [62].

One advantage of image-based models over object-based models is that image based models do not require manually designed object-specific 3-D models. Since a developmental robot needs to handle various objects without manual interference, we adopted the image-based models for the visual representation of our system. Our previous studies have shown the feasibility of real-time visual learning [106]. However, given the combination of vision and audition, the speed of the system is yet to be tested, especially considering the time-critical nature of the audition information.

Imperfect alignment. Many existing works on multimodality learning rely on the strict coupling between vision and audition information, such as the movement of mouth and the utterance produced [18] [32]. In our simple semantics learning problem, the alignment between the visual stimuli and the auditory stimuli is imperfect. In the real world, the presence of an object is usually coupled with the related auditory signals, such as the noise made by the object or the verbal label given by a teacher. However, since image-based models are viewpoint-dependent, the observations from different views of an object are different. The auditory signals are not likely to appear exactly when the same visual observation is acquired. In other words, if we call a visual-auditory stimulus pair at a particular time instance *a visuoauditory context*, it is unlikely that a particular visuoauditory context will be exactly repeated.

A recent study proposed a mutual-information-based framework to resolve the problem [78]. But, in the reported experiment, the visual stimuli were unrealistic still images of the objects, although the auditory data was very challenging real mother-

baby verbal interaction. We view this problem as an abstraction problem and use the spatiotemporal correlations among contiguous visual and auditory information to resolve it.

7.3 Architecture and algorithm

In Chapter 6, we introduced the module of LBE with two IHDR trees to handle immediate reflexive behaviors and missing or delayed behaviors, respectively. The primed contexts of the P-tree possess future information, i.e., the sensations about to be captured, and the action about to be made together its value. Since the prediction model (Eq. (6.4)) is actually a low-pass filter, the primed sensation changes slowly compared to the corresponding last sensation. Taking advantage of this, we constructed a multi-level system using LBE as the basic element to handle different abstraction at different levels. The abstraction idea can also be used in the simple semantics learning problem.

As discussed above, one of the major challenges here is that an object appears to the robot as a sequence of images captured from different viewpoints. We need a representation upon which the robot can group these different images into a single cluster, i.e. an object, before the object-specific action can be learned. If manual transcription is allowed, we may design a representation, say a label, and assign it to the corresponding images one by one. However, in the developmental paradigm, such a task-specific design and data structure level manipulation are not allowed. They are not practical to do for a robot running in an unstructured and complex environment,

either. Fortunately, there is a very important property of the physical world we may take advantage of, i.e., the continuity.

In the real world, an object does not emerge from nothing and it would not disappear like a magic. We may make use of the shared image features of the spatiotemporally contiguous views of an object. Moving in and out the agent's field of view, two consecutive views of an objects are similar by a large when the capturing speed is high enough. If we filter out the high-frequency components, the images change even more slowly and may be considered as identical in some cases, which is exactly what we need.

Fig. 7.1 shows the architecture we used to do simple semantics learning. It has three LBE modules, a vision LBE (V-LBE), an audition LBE (A-LBE), and a high-level LBE (H-LBE). The visual sensation is the original image captured by a CCD camera. We do not do any pre-processing on the images to derive any low-level features such as edge histogram. The important discriminant features are derived automatically by the IHDR trees. The auditory sensation is captured by a sound blaster card through a microphone. We do Cepstrual analysis on the original sound signals before the data entering the A-LBE. Since sound is a linear signal in the sense the information is distributed over a period of time, each auditory sensation vector actually covers 20 speech frames as you would see in the experimental results. The primed sensations from the P-trees of both V-LBE and A-LBE are inputs to H-LBE. After the low-pass filtering in PUQ, the primed sensation only keeps the low-frequency components of the last context. We would discuss how the architecture works in details while presenting the experimental results.

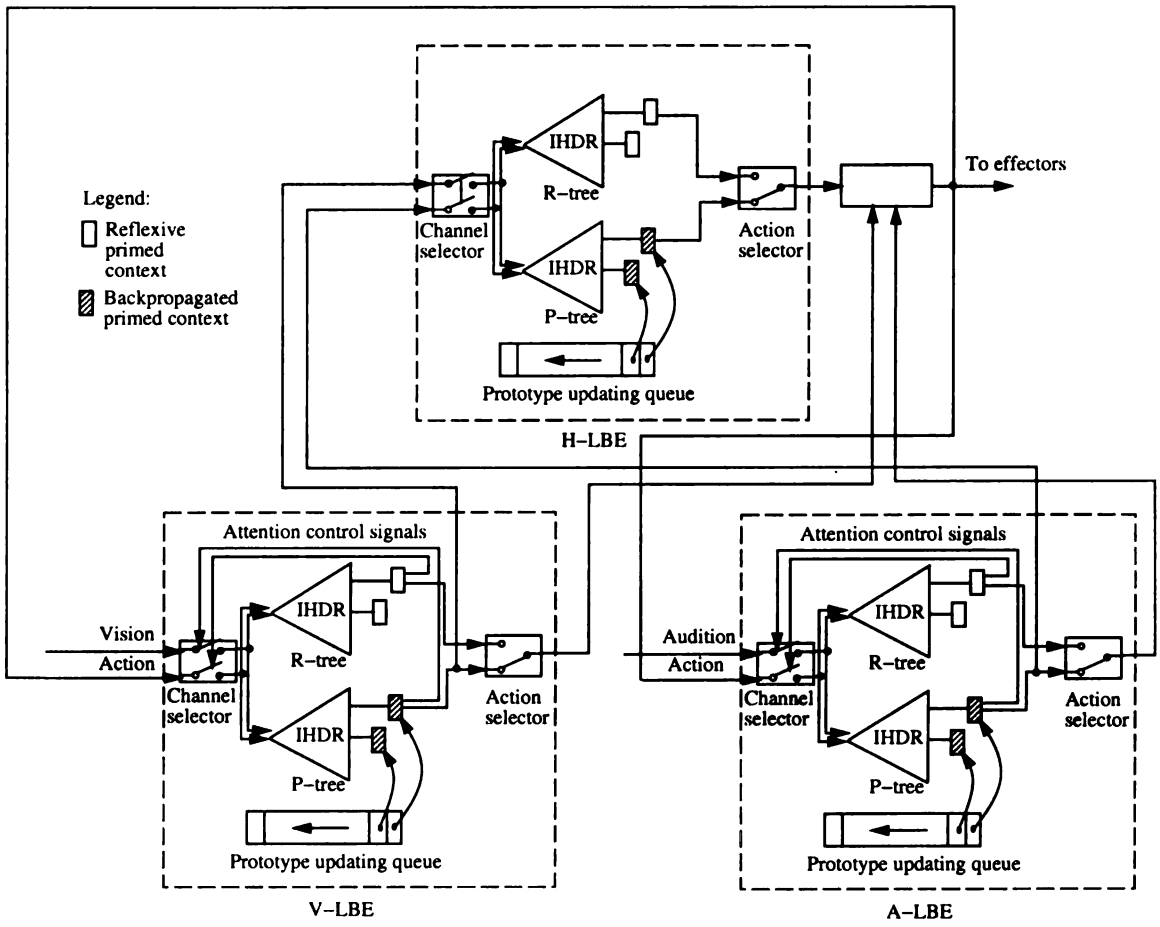


Figure 7.1: The semantics learning system architecture.

7.4 Experimental results

The multilevel semantics learning architecture has been implemented on the SAIL robot.

The experiment was done in the following way. After SAIL started running, the trainer mounted objects one after another on the gripper of SAIL and let SAIL rotate the gripper in front of its eyes at the speed of about 3.6s per round. Totally 13 objects were presented (Fig. 7.2) to SAIL. All these real-world objects were of very complex shape, for example, the hair of "Harry Porter," and non-rigid form. It was extremely difficult, if not impossible, to model them using 3-D representations. The

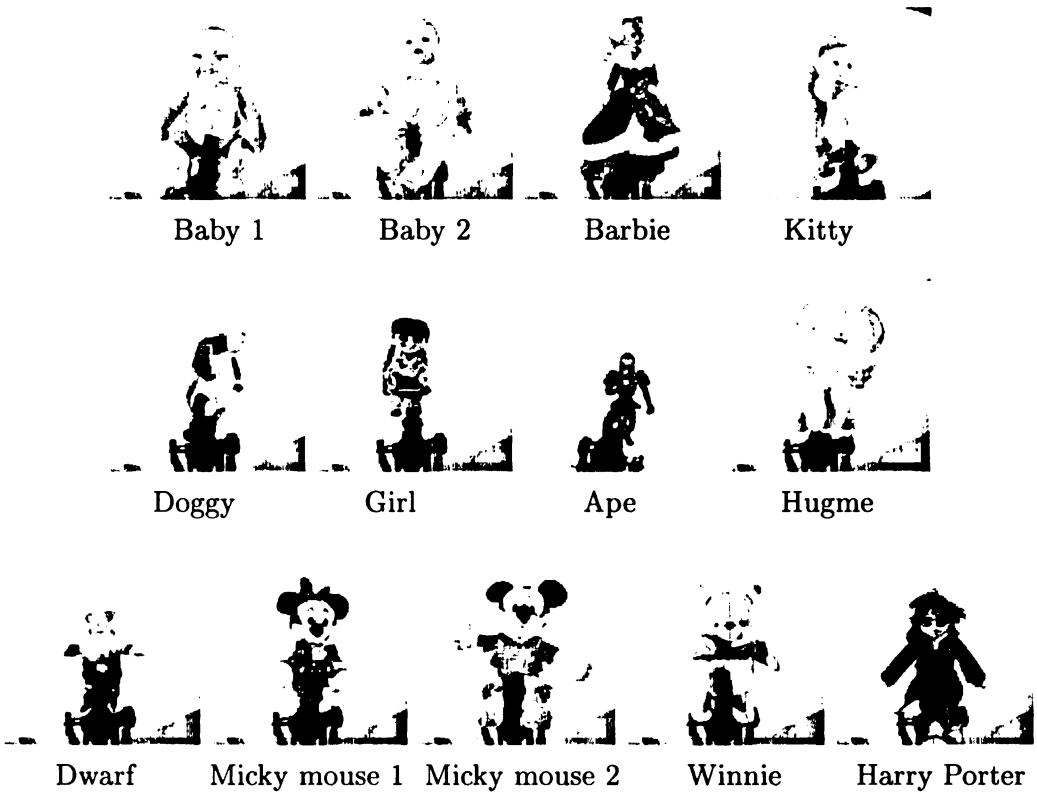


Figure 7.2: The objects used in the experiment.

properties of each object was taught to SAIL in a question-and-answer session. First, the trainer verbally asked the questions, such as "name?" and "size?", when an object was presented. And then the trainer gave the appropriate answers by pushing the switch sensors of SAIL, where different switch sensor status represented different answers. Since the objects were rotated, and moved in and out of SAIL's field of view continuously, the orientation and the positions of the objects kept changing. There were hardly chances that SAIL could see the same images of the objects, when the same question was asked again. A sample video sequence seen by SAIL is shown in Fig. 7.3. We expected SAIL to correctly answer the taught questions when the objects were presented and the question were asked the next time.

The images were captured by a Matrox Meteor II board as gray-scale images at

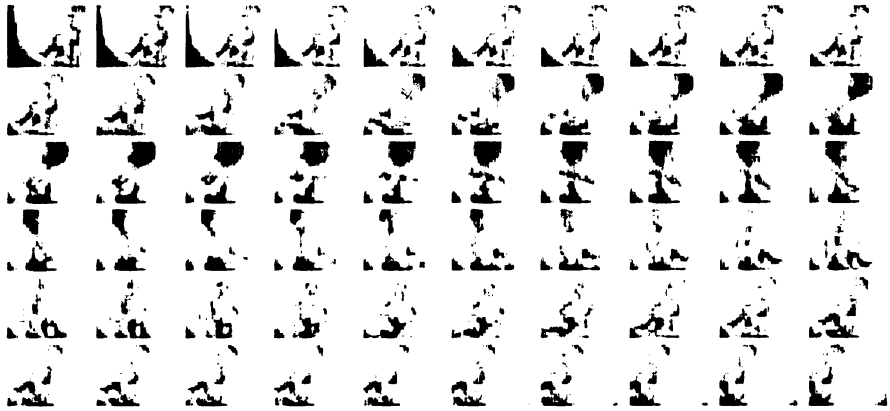


Figure 7.3: Part of a sample image sequence.

30 frames per second. The dimension of the images was 25-by-20. The speech data were digitized at 11.025kHz by a normal sound blaster card. We did Cepstral analysis on the speech data and 13-order Mel-frequency Cepstral Coefficients (MFCCs) were computed over 256-point wide frame windows. There was an overlap of 56 points between two consecutive frames. Therefore, the MFCCs entered the auditory channel of SAIL at the rate of about 50Hz. We concatenated 20 consecutive MFCC vectors together as a single auditory sensation vector because a 20ms speech frame is too short to convey any meaningful information. To compensate the slower capture rate of image data, the cognitive mapping module of SAIL would use the last image captured accompanying the new vector of MFCC when a new image was not available.

To examine the behavior of SAIL in detail and evaluate the performance, we pursued an experiment on pre-recorded data first. The image data of each object were five video sequences of the object moving in SAIL's field of view, rotating for roughly one round, and then moving out of SAIL's field of view. Each image sequence contained 350 frames,

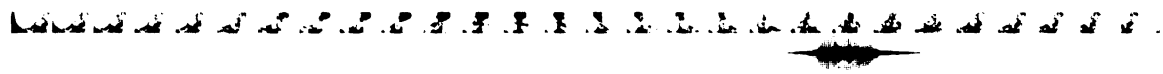
- Frame 1-50: background images;

- Frame 51-100: an object moving to the center of SAIL's; field of view;
- Frame 101-300: the object rotating along its center axis;
- Frame 301-350: the object moving out of the SAIL's field of view.

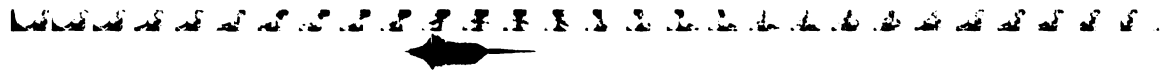
The speech data was part of the number data set used in Section 5.5.1. Ten subjects were randomly selected from the total 63 ones. Each number were spoken five times by each subject. We used the utterances of "one" to represent "name" and "two" to represent "size." During training, the switch sensor inputs (a numerical vector) were given after the utterances were finished, which was the time SAIL was taught the answers. Of all the five sets of image and speech data, we used four of them in training and the left-out one for testing. So, with 13 objects, ten persons, and two questions, SAIL was taught 1040 times in training and evaluated for 260 times in testing.

To emulate the situation that the trainer would not be able to ask questions when the objects were presented with exactly the same orientations and positions in testing as in training, we randomly choose the point to align the image sequences and speech sequences (Fig. 7.4). Specifically, the end point of questions was aligned with image No. 300 during training. When testing, we aligned the end point of questions with image No. 100, 150, 200, 250, and 300, respectively.

The behavior of SAIL were evaluated in two different ways. First, we counted the total number of times when SAIL responded with certain answers after the question utterances. This number was usually larger than the number of image sequences because there were chances when SAIL responded more than once after a single



(a)



(b)

Figure 7.4: The alignment of image and speech data (a) during training (b) during testing.

question. For each of the responses, if it was correct with respect to the object presented at the time, we counted it as correct. So, we got the first correct answer rate (C.A.R.1),

$$C.A.R.1 = \frac{\text{no. of correct responses}}{\text{total no. of responses}}.$$

In the second evaluation way, we counted all the responses during one object image sequences as one trial. During each object image sequence, if the majority of the responses were correct, we counted this trial as correct. Otherwise, we counted it as wrong. Here came the second correct answer rate (C.A.R.2),

$$C.A.R.2 = \frac{\text{no. of image sequences with correct majority responses}}{\text{no. of image sequences}}.$$

We plot the correct answer rates with respect to the question positions during testing in Fig. 7.5. The visuoauditory scenes were never exactly same during testing as during training when the questions were asked. When the question-position difference between training and testing was not large, SAIL maintained high correct answer rate. With the increase of the question-position difference, the correct answer rate dropped gradually. Also, during the time the objects moving in or out of SAIL's field of view, SAIL's performance was also low because SAIL did not have an attention mechanism

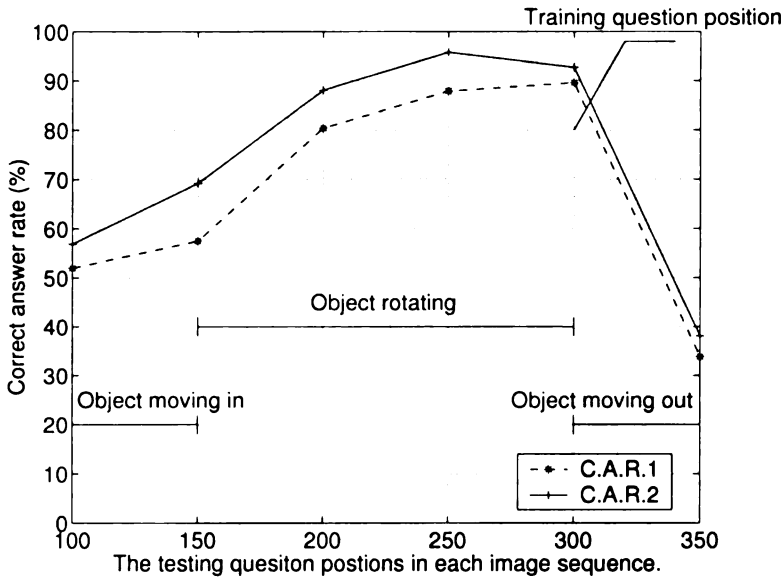


Figure 7.5: The two correct answer rates of SAIL v.s. the question positions in each image sequence.

to locate the object in the center of its field of view.

Particularly, a detailed confusion table (Table 7.1 and Table 7.2) shows SAIL's C.A.R.1 on different objects and different questions when the questions were aligned with image frame No. 250. The average C.A.R.1 is 87.89%. Table 7.3 shows SAIL's C.A.R.2 with an average rate of 95.77%.

The size of the whole "brain" after training was 806MB. The shape the three major trees of the three LBEs are shown in Fig. 7.6. Because of the tree structure, the average execution time at each time step is 3.4ms, much lower than 18.1ms, the required interval of a single speech frame.

To see why SAIL was able to respond when the questions were not asked at the exactly same time in testing as in training, we show the primed sensation of V-LBE in Fig. 7.7. Since the operation done in the PUQ of V-LBE was a low-pass filtering, the primed visual sensation was a blurred version of the real visual sensation. The

Table 7.1: SAIL's responses on question 1 ("Name?") when the questions were aligned with image frame No. 250.

Objects/ Answers (%)	None	Baby 1	Baby 2	Barbie	Kitty	Dwarf	Doggy	Girl	Ape	Hugme	Micky mouse 1	Micky mouse 2	Winnie	Harry Porter
None	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Baby 1	0.00	93.33	2.50	6.00	6.00	5.00	5.00	7.50	5.00	5.00	5.00	5.00	5.00	5.00
Baby 2	0.00	5.00	86.83	0.00	0.00	0.00	0.00	0.00	0.00	24.83	0.00	0.00	0.00	0.00
Barbie	0.00	0.00	0.00	91.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Kitty	0.00	0.00	0.00	0.00	94.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Dwarf	0.00	0.00	0.00	0.00	0.00	92.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doggy	0.00	0.00	0.00	0.00	0.00	0.00	92.50	0.00	0.00	0.00	0.00	0.00	1.67	0.00
Girl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	69.83	0.00	0.00	0.00	0.00	0.00	0.00
Ape	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	95.00	0.00	0.00	0.00	0.00	0.00
Hugme	0.00	1.67	10.67	0.00	0.00	2.50	2.50	0.00	0.00	70.17	0.00	0.00	5.83	0.00
Micky 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.67	0.00	0.00	92.50	3.33	0.00	0.00
Micky 2	0.00	0.00	0.00	2.50	0.00	0.00	0.00	0.00	0.00	0.00	2.50	89.17	0.00	0.00
Winnie	0.00	0.00	0.00	0.00	0.00	0.00	0.00	16.00	0.00	0.00	0.00	2.50	87.50	0.00
HarryPorter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	95.00
Big	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Small	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7.2: SAIL's responses on question 2 ("Size?") when the questions were aligned with image frame No. 250. The italic numbers represent the correct rates.

Objects/ Answers (%)	None	Baby 1	Baby 2	Barbie	Kitty	Dwarf	Doggy	Girl	Ape	Hugme	Micky mouse 1	Micky mouse 2	Winnie	Harry Porter
None	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Baby 1	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Baby 2	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Barbie	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Kitty	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Dwarf	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doggy	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Girl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00
Ape	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00
Hugme	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00
Micky 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00
Micky 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00
Winnie	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00
HarryPorter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00
Big	0.00	*93.00*	*93.00*	*91.33*	*93.00*	0.00	1.67	15.50	0.00	*93.00*	2.00	*91.00*	*93.00*	2.00
Small	0.00	0.00	0.00	1.67	0.00	*93.00*	*91.33*	*77.50*	*93.00*	0.00	*91.00*	2.00	0.00	*91.00*

Table 7.3: The correct answer rate 2 (majority correct rate) of SAIL when the questions were aligned with image frame No. 250.

Objects/ Questions(%)	Baby 1	Baby 2	Barbie	Kitty	Dwarf	Doggy	Girl
"Name?"	90.0	90.0	90.0	90.0	100.0	100.0	90.0
"Size?"	100.0	100.0	100.0	100.0	100.0	100.0	80.0
Objects/ Questions(%)	Ape	Hugme	Micky mouse 1	Micky mouse 2	Winnie	Harry Porter	
"Name?"	100.0	70.0	100.0	100.0	90.0	100.0	
"Size?"	100.0	100.0	100.0	100.0	100.0	100.0	

result was that the vision inputs to H-LBE did not change a lot in consecutive frames when the same object was presented. Thus, SAIL was able to answer the question correctly even it was taught when another pose of the object was seen.

In the real-time experiment, the verbal questions ("name?" and "size") were asked followed by the answers imposed through the switch sensors of SAIL. For each object, we usually issued each question five to six times. To make it easy for the trainer to see the response of SAIL, we manually mapped SAIL's action vectors to the names of the objects and used Microsoft text-to-speech software to read out the names. After going through three objects (baby 1, dwarf, and girl), the objects were mounted on the gripper in turn again and the questions were asked without giving answers. We repeated the above process ten times and SAIL responded correctly at about 90% of the time for all the trained three objects.

We also recorded the execution time to have some idea about the speed performance of SAIL. Fig. 7.8 shows the similar pattern as we have seen in Chapter 5 and Chapter 6. The execution time of each step grew at the beginning and it flattened out after about 100 seconds. The short surging period around 100s, 150s and 210s were the times when we changed the objects. Since the visual context changed a lot

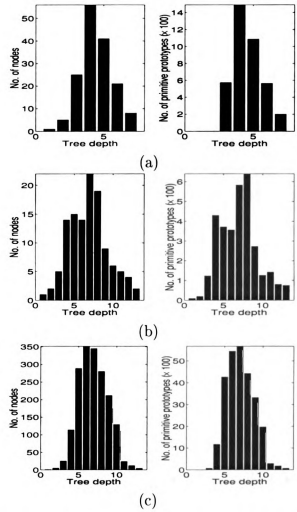


Figure 7.6: The node distribution and primitive prototype distribution in the trees: (a) P-tree of A-LBE; (b) P-tree of V-LBE; (c) R-tree of H-LBE.

at the time, the trees conducted extensive learning and required more time in each execution step. But even in these periods, the execution time of each step is lower than 18.1ms, the required interval of a single speech frame.

The shape the three major trees of the three LBEs are shown in Fig. 7.9. The P-tree of V-LBE is fairly small comparing to the P-tree of A-LBE because SAIL's eye focused on a small field of view covering the object and did not experience very dramatic changes. In contrast, the microphone of SAIL collected the conversation

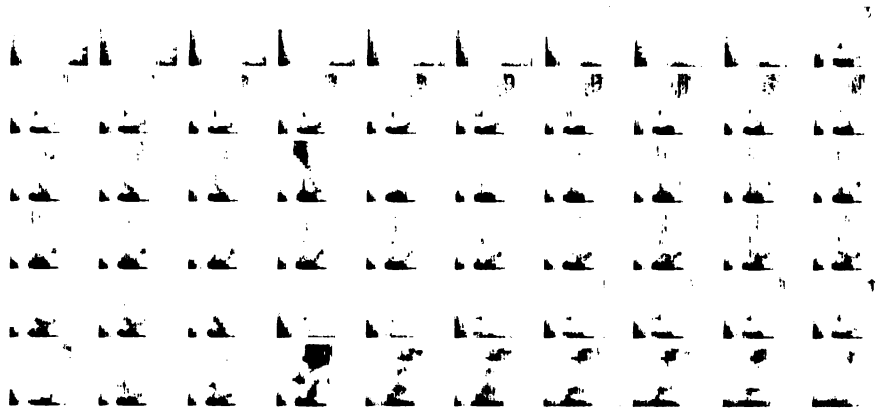


Figure 7.7: A sample sequence of the primed visual sensation.

of the trainer with his lab mate in addition to the verbal questions. The size of the whole "brain" containing three LBEs is about 60MB after the above training process.

7.5 Conclusions

In this chapter, we further extended the architecture of a developmental robot we proposed in the last two chapters in order to handle information from multiple modalities. With this architecture, a robot was able to pursue real-time simple semantics learning. After taught the answers to verbal questions upon the presence of objects, the SAIL robot was able to answer the questions correctly even when the orientation of the objects was changed. This process emulates the way a child learns concepts of the physical world through verbal instructions. The semantics learning system took advantage of the spatiotemporal continuity of the real world. It filtered out the high-frequency components in visual sensation so that the resulted slowly-changing primed visual context was not sensitive to orientation changes, which enabled the system to tolerate the imperfect alignment.

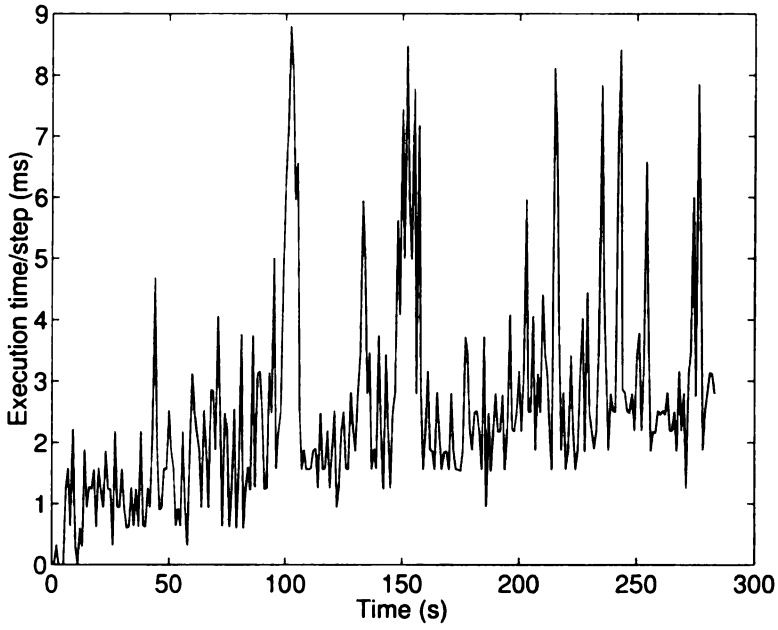


Figure 7.8: The average execution time of the semantics learning system at each time step is much shorter than 18.1ms, the required interval of each speech frame.

With the current implementation, the robot did not discriminate the foreground from the visual background. In other words, the robot did not really have an object concept. It essentially treated the whole image as a pattern, with which the audition signals and behaviors were associated. To achieve object concept learning, among other requirements, the system needs a sophisticated attention mechanism to establish the bound of the objects. While we believe object concepts can be evolved from interactions between the agent and the environment, the detailed mechanism is largely unclear. Another limitation of this implementation is that the action of the system was designed to be the output of one of the three LBEs, namely the H-LBE. This manual design is not practical for an autonomous robot. This semantics learning architecture still needs a good value system to coordinate the behaviors of all the three LBEs.

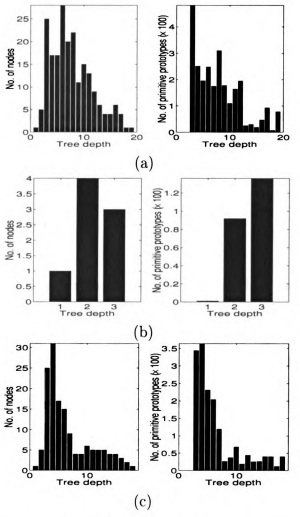


Figure 7.9: The node distribution and primitive prototype distribution in the trees: (a) P-tree of A-LBE; (b) P-tree of V-LBE; (c) R-tree of H-LBE.

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 2.3.3. This work is one of the few works that enable a machine to learn directly from unsegmented and unlabeled sensory streams, a mode in which children learn. A potential benefit of this work is to help us to build machines capable of spoken communication with humans.

At this early stage of research in developmental robots, we are facing an array of challenging technical issues. We have developed three major techniques, based on

Table 8.1: Summary of contributions on developmental robot architecture.

Grounded speech learning	Task transfer	Semantics learning
• Learning from unsegmented auditory stream • Internal action learning through reinforcement learning • AMD mode	• Scaffolding: cognitive behavior scale-up • Communicative learning: language acquisition and behavior-learning using language • AMD mode	• Use of physical world causality • Integration of multimodalities (vision and audition) • AMD mode

which a developmental architecture was implemented on a real robot, SAIL, an early developmental robot prototype. A summary of the contributions is given below. Particularly, the contributions on proposed developmental robot architecture are listed in Table 8.1.

1. The proposed complementary candid incremental principal component analysis (CCIPCA) algorithm computes principal components of a sequence of samples incrementally without estimating the covariance matrix. Motivated by the concept of statistical efficiency (the estimate has the smallest variance given the observed data), it keeps the scale of observations and calculates the mean of observations incrementally, which is the most efficient estimate for some well known distributions (e.g., Gaussian). Although the efficiency is not guaranteed in our case because of the unknown sample distribution, empirical studies of this method show very fast convergence rate compared to existing IPCA algorithms, especially for high dimensional image vectors. A mathematical proof of the convergence of CCIPCA is given.

2. Hierarchical discriminant regression is a hierarchical statistical modeling method introduced by Hwang and Weng [35]. It automatically derives discriminant features and thus automatically generates internal representations. The coarse-to-fine memory self-organization of HDR ensures a logarithmic time complexity and plays an important role in reaching real-time speed when processing high dimensional input. HDR requires the sample distributions in the input and output spaces to have the same topology, which is usually not fulfilled in audition-driven behavior development. To comply with the requirement of the HDR method, we set the output part as the concatenation of the auditory signal and the motor control signal. This combination gave a much better performance when the input and output have different distribution topology.

3. While supervised learning is efficient, it is not practical to constantly provide input-output pairs for an open-ended learning robot and it is impossible when a robot learns internal behaviors. We unify supervised learning and reinforcement learning in a single framework.

4. We have realized a grounded speech learning system that develops its audition-driven behaviors through physical interactions with the environment. Our experiments showed that after a short process of grounded simple language acquisition, the robot could produce desired behaviors, external behaviors, such as body movement, and internal behaviors, such as selective attention, upon hearing verbal commands. A special study was provided on the establishment of a particular internal behavior, selective attention, through reinforcement learning.

5. The capability of learning new skills is crucial for a robot to learn complex cognitive behaviors. With an add-on priming capability over the grounded speech learning system, the SAIL robot successfully developed complex behaviors (procedures) upon the acquisition of simple ones, which we call a *scaffolding* process. Our experiments showed that, the robot's behavior could be shaped by the trainers through verbal instructions (communicative learning). The learning process of the agent was conducted through online real-time interactions thanks to the above grounded speech learning system and the missing context problem was resolved by a prediction model modified from Q-learning.

6. A further extended architecture was able to pursue real-time simple semantics learning through perception from multiple modalities. After taught the answers to verbal questions upon the presence of objects, the SAIL robot was able to answer the questions correctly even when the orientation of the objects was changed. This process emulates the way a child learns concepts of the physical world through verbal instructions. The semantics learning system took advantage of the spatiotemporal continuity of the real world and filtered the high-frequency components in visual sensation. The resulting slowly-changing primed visual context was not sensitive to orientation changes and could tolerate the imperfect alignment.

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 one of the 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.

Internal attention. We proposed a simple attention mechanism in Chapter 5 to select context with appropriate length. The same mechanism can be used in visual perception in order to select appropriate portion in the field view of a robot instead of treating the whole view monolithically. However, such an attention behavior could be very complicated. How to effectively learn this behavior is an open question.

A sophisticated value system. One of the limitations of the semantics learning system is that the action of the system was designed to be the output of H-LBE. The coordination of the multiple LBEs' behavior requires a sophisticated value system, which takes the contextual information into consideration when making decisions on action selection. 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? What is the mechanism for it to cooperate with other modules, such as sensory mapping and cognitive mapping?

High-level symbolic concepts. We have discussed the limitations of using

symbolic representation in Chapter 1. However, it is obvious that human beings have symbolic concepts such as language. The low level numerical sensory input is converted to high level symbolic concepts somewhere in the brain. While avoiding a manual design of symbolic representation to get across this gap, we need to investigate the underlying mechanism for this transition, which will be essential for implementing a system with more complicated cognitive behaviors.

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 video indexing and intelligent vehicles, and apply the principles and techniques of autonomous mental development.

Appendix A

Convergence Analysis of CCIPCA

We are going to prove that, with algorithm given by

$$v_i(n) = \frac{n-1}{n} v_i(n-1) + \frac{1}{n} u_i(n) u_i^T(n) \frac{v_i(n-1)}{||v_i(n-1)||} \tag{A.1}$$

$$u_i(n) = u_{i-1}(n) - u_{i-1}^T(n) \frac{v_{i-1}(n)}{||v_{i-1}(n)||} \frac{v_{i-1}(n)}{||v_{i-1}(n)||} \tag{A.2}$$

$v_i(n) \to \lambda_i e_i$ when $n \to \infty$, where λ_i is the i-th largest eigenvalue of the covariance matrix of $\{u(n)\}$, e_i is the corresponding eigenvector, and $u_1(n) = u(n)$, under the following assumptions,

1. $A(n) = u(n)u^T(n)$ are mutually statistically independent with $E\{A(n)\} = A$ for all n.

2. A is of full rank, which means λ_d, the smallest eigenvalue of A, is not zero.

In the following sections, starting from the case of $i = 1$, we first prove that, under certain assumptions, the above algorithm converges to the asymptotically stable

solutions of a related differential equation with probability 1 (*w.p.*1). Then, we show the asymptotically stable solutions are $\pm\lambda_1 e_1$ and all the assumptions are satisfied. We prove $v_i(n) \to \lambda_i e_i$ for $i > 1$ using mathematical induction.

A.1 Relation to a differential equation

When $i = 1$, the algorithm is,

$$v_1(n) = v_1(n-1) + \frac{1}{n}\left(\frac{A(n)}{||v_1(n-1)||} - I\right) v_1(n-1) \tag{A.3}$$

where $A(n) = u_1(n)u_1^T(n)$. It may be rewritten as,

$$v_1(n) = v_1(n-1) + \frac{1}{n}\left(\frac{A}{||v_1(n-1)||} - I\right) v_1(n-1) + \frac{1}{n}\frac{A(n)-A}{||v_1(n-1)||} v_1(n-1) \tag{A.4}$$

where $A = E\{A(n)\}$ for all n.

Lemma A.1.1 $\lim_{n\to\infty} P\{\sup ||\frac{A(n)-A}{||v_1(n-1)||} v_1(n-1)|| \geq \varepsilon\} = 0$.

Proof. Since $||\frac{A(n)-A}{||v_1(n-1)||} v_1(n-1)|| = ||A(n) - A||$ and $A = E\{A(n)\}$ for all n, it is simple to conclude

$$\lim_{n\to\infty} P\{\sup ||\frac{A(n)-A}{||v_1(n-1)||} v_1(n-1)|| \geq \varepsilon\} = 0$$

◁

Lemma A.1.2 $v_1(n)$ *is bounded w.p.*1.

Proof. According to Eq. A.3, we have,

$$\begin{aligned} ||v_1(n)||^2 &= ||v_1(n-1)||^2 + \frac{2}{n}\frac{v_1^T(n-1)A(n)v_1(n-1)}{||v_1(n-1)||} - \frac{2}{n}v_1^T(n-1)v_1(n-1) \\ &+ \frac{1}{n^2}\frac{v_1^T(n-1)A^2(n)v_1(n-1)}{||v_1(n-1)||^2} + \frac{1}{n^2}v_1^T(n-1)v_1(n-1) \\ &- \frac{2}{n^2}\frac{v_1^T(n-1)A(n)v_1(n-1)}{||v_1(n-1)||} \end{aligned} \tag{A.5}$$

We know that $v_1^T(n-1)A(n)v_1(n-1) \leq \lambda_{max}(A(n))v_1^T(n-1)v_1(n-1)$, where $\lambda_{max}(A(n))$ is the largest eigenvalue of $A(n)$. If $||v_1(n-1)|| \geq 2\lambda_{max}(A(n))$, we have,

$$\begin{aligned} \frac{2}{n}\frac{v_1^T(n-1)A(n)v_1(n-1)}{||v_1(n-1)||} &\leq \frac{2}{n}\frac{\lambda_{max}(A(n))v_1^T(n-1)A(n)v_1(n-1)}{||v_1(n-1)||} \\ &\leq \frac{1}{n}v_1^T(n-1)v_1(n-1). \end{aligned} \tag{A.6}$$

Similarly, since $v_1^T(n-1)A^2(n)v_1(n-1) \leq \lambda_{max}^2(A(n))v_1^T(n-1)v_1(n-1)$, if $||v_1(n-1)|| \geq \lambda_{max}(A(n))$ for certain $n > 2$, we have,

$$\begin{aligned} \frac{1}{n^2}\frac{v_1^T(n-1)A^2(n)v_1(n-1)}{||v_1(n-1)||^2} &\leq \frac{1}{n^2}\frac{\lambda_{max}^2(A(n))v_1^T(n-1)v_1(n-1)}{||v_1(n-1)||^2} \\ &\leq \frac{1}{n^2}v_1^T(n-1)v_1(n-1) \\ &< \frac{1}{2n}v_1^T(n-1)v_1(n-1) \end{aligned} \tag{A.7}$$

Further, when n is large enough ($n > 2$), we have,

$$\frac{1}{n^2}v_1^T(n-1)v_1(n-1) < \frac{1}{2n}v_1^T(n-1)v_1(n-1). \tag{A.8}$$

Considering Eqs. A.6-Eq. A.8, we have, for large enough n, if $||v_1(n-1)|| \geq 2\lambda_{max}(A(n))$,

$$||v_1(n)|| < ||v_1(n-1)||.$$

Otherwise,

$$||v_1(n-1)|| < 2\lambda_{max}(A(n)).$$

Since $\lambda_{max}(A(n))$ is bounded $w.p.1$ because $A = E\{A(n)\}$ for all n, we have $||v_1(n)||$ is bounded w.p. 1. ◁

Theorem A.1.1 *Let v_{10} be a locally asymptotically stable (in the sense of Liapunov) solution to*

$$\dot{v}_1 = \left(\frac{A}{||v_1||} - I\right) v_1 \tag{A.9}$$

with domain of attraction $\mathcal{D}(v_{10})$. If there is a compact set $\mathcal{A} \subset \mathcal{D}(v_{10})$ such that $v_1(n) \in \mathcal{A}$ infinitely often, $v_1(n)$ tends to v_{10} almost surely.

Proof. We prove it using Theorem 2.3.1 in Kushner and Clark [48]. Assumptions A.2.2.1, A.2.2.2, and A.2.2.3 in [48] are trivial. Lemma A.1.1 fulfills the assumption of A.2.2.4. Together with Lemma A.1.2, all the assumptions are satisfied and, thus, Theorem 2.3.1 implies Theorem A.1.1 here. ◁

A.2 Prove $v_1(n) \to \pm\lambda_1 e_1$

In this section, we are going to find the asymptotically stable solutions of Eq. A.9.

Expending v_i in terms of the eigenvectors, which is a base of d-dimensional space,

we have, $v_1 = \sum_{j=1}^{d} \alpha_j e_j$, where $\alpha_j = v_1^T e_j$. Considering $Ae_1 = \lambda_1 e_1$, we can write Eq. A.9 as,

$$\dot{\alpha} = \left(\frac{A_\lambda}{\sqrt{\sum_{k=1}^{d} \alpha_k^2}} - I \right) \alpha \tag{A.10}$$

where, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)^T$, $||v_1|| = \sqrt{\sum_{k=1}^{d} \alpha_k^2}$, and $A_\lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$.

A.2.1 $||v_1||$ is bounded

Let $r = ||v_1|| = (v_1^T v_1)^{1/2}$, we have,

$$\begin{aligned}
\dot{r} &= \frac{1}{r} v_1^T \dot{v}_1 \\
&= \frac{1}{r} v_1^T \left(\frac{A}{||v_1||} - I \right) v_1 \\
&= \frac{1}{r} \left(\frac{\sum_{k=1}^{d} \lambda_k r^2 \cos^2 \theta_k}{r} - r^2 \right) \\
&= \sum_{k=1}^{d} \lambda_k \cos^2 \theta_k - r
\end{aligned}$$

where θ_k is the angle between v_1 and e_k. Because $\sum_{k=1}^{d} \cos^2 \theta_k = 1$ and $\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0$, we have,

$$\dot{r} = \lambda_d + \sum_{k=1}^{d-1} \cos^2 \theta_k (\lambda_k - \lambda_d) - r \tag{A.11}$$

where $0 \leq \sum_{k=1}^{d-1} \cos^2 \theta_k (\lambda_k - \lambda_d) \leq \sum_{k=1}^{d-1} (\lambda_k - \lambda_d) < \sum_{k=1}^{d-1} \lambda_k$. Hence, $||v_1||$ is bounded between λ_d and $\sum_{k=1}^{d} \lambda_k$ when $t \to \infty$.

A.2.2 $\alpha_j/\alpha_1 \to 0$

Let $\theta_j = \alpha_j/\alpha_1$ for $j > 1$ when $\alpha_1 \neq 0$, and we have,

$$\begin{aligned}
\dot{\theta}_j &= \frac{1}{\alpha_1^2}(\alpha_1\dot{\alpha}_j - \alpha_j\dot{\alpha}_1) \\
&= \frac{1}{\alpha_1^2 \sum_{k=1}^{d} \alpha_k^2}(\lambda_j\alpha_1\alpha_j - \lambda_1\alpha_1\alpha_j) \\
&= \frac{\theta_j}{\sqrt{\sum_{k=1}^{d} \alpha_k^2}}(\lambda_j - \lambda_1) \qquad \text{(A.12)}
\end{aligned}$$

Since $\lambda_d \leq ||v_1|| = \sqrt{\sum_{k=1}^{d} \alpha_k^2} < \sum_{k=1}^{d} \lambda_k$ and $\lambda_j < \lambda_1$, $\theta_j \to 0$ as $t \to \infty$. Again because of the upper bound of $||v_1||$, α_1 is bounded and, consequently, $\alpha_j \to 0$ for $j > 1$ when $t \to 0$.

A.2.3 $\alpha_1 \to \pm\lambda_1$

From Eq. A.10, we have,

$$\dot{\alpha}_1 = \left(\frac{\lambda_1}{\sqrt{\sum_{k=1}^{d} \alpha_k^2}} - 1 \right) \alpha_1.$$

We may drop α_j for $j > 1$ when $t \to \infty$ and get,

$$\dot{\alpha}_1 = \pm\lambda_1 - \alpha_1,$$

which means $\alpha_1 \to \pm\lambda_1$ when $t \to \infty$.

A.2.4 Summary

Above we assume $\alpha_1(t) \neq 0$. When $\alpha_1(t) = 0$, $v_1(t)$ is in a $(d-1)$dimensional space, $R_1^{d-1} = span\{e_i, i = 2, \ldots, d\}$. In other words, $\mathcal{D}(\{\pm\lambda_1 e_1\})$ is $R^d - R_1^{d-1}$. Since it is very unlikely that we choose $v_1(0)$ in space $R^d - R_1^{d-1}$, $v_1(n)$ enters $\mathcal{D}(\{\pm\lambda_1 e_1\})$ $w.p.1$. Applying Theorem A.1.1, we have $v_1(n) \to \pm\lambda_1 e_1$ $w.p.1$ when $n \to \infty$.

A.3 Prove $v_i(n) \to \pm\lambda_i e_i$ with induction

We want to prove that $v_i(n) \to \pm\lambda_i e_i$ under induction assumption that $v_j(n) \to \pm\lambda_j e_j$ $(j < i)$.

From Eq. A.2, we have,

$$\begin{aligned} u_i(n) &= u_{i-1}(n) - \frac{v_{i-1}(n)}{||v_{i-1}(n)||}\left[\frac{v_{i-1}^T(n)}{||v_{i-1}(n)||}u_{i-1}(n)\right] \\ &= \Pi_{j=1}^{i-1}\left[I - \frac{v_j(n)v_j^T(n)}{||v_j(n)||^2}\right]u_1(n) \end{aligned} \tag{A.15}$$

To simplify notation, we define,

$$\Pi_i(n) = \Pi_{j=1}^{i-1}\left[I - \frac{v_j(n)v_j^T(n)}{||v_j(n)||^2}\right],$$

where $\Pi_i(n) = I$ if the superscript and the subscript cross over. So, Eq. A.1 can be written as,

$$v_i(n) = v_i(n-1) + \frac{1}{n}\left(\frac{\Pi_i(n)A(n)\Pi_i(n)}{||v_i(n-1)||} - I\right)v_i(n-1) \tag{A.17}$$

where $A(n) = u_1(n)u_1^T(n)$.

Similar to the case of $v_1(n)$, we may find the related differential equation of Eq. A.17 as,

$$\dot{v}_i = \left(\frac{\Pi_i A \Pi_i}{||v_i||} - I \right) v_i \tag{A.18}$$

where $A = E\{A(n)\}$ and $\Pi_i = \Pi_{j=1}^{i-1} \left[I - \frac{v_j v_j^T}{||v_j||^2} \right]$.

With the induction assumption, we write, for $j < i$,

$$\frac{v_j}{||v_j||} = e_j + \varepsilon_j w_j \tag{A.19}$$

where w_j is a time-variable unit-length vector, and for $j < i$, $\varepsilon_j(t) \to 0$ as $t \to \infty$. (Following analysis will be similar if we write, $\frac{v_j}{||v_j||} = -e_j + \varepsilon_j w_j$.)

From the definition of Π_i we have,

$$\begin{aligned}
\Pi_i(t) &= \Pi_{j=1}^{i-1}[I - (e_j + \varepsilon_j(t)w_j(t))(e_j + \varepsilon_j(t)w_j(t))^T] \\
&= \Pi_{j=1}^{i-1}[I - e_j e_j^T - \varepsilon_j(t)(w_j(t)e_j^T + e_j w_j(t)^T + w_j(t)w_j(t)^T] \\
&= I - \sum_{j=1}^{i-1} e_j e_j^T - O(\varepsilon(t))
\end{aligned} \tag{A.20}$$

where $\varepsilon(t) = \arg\max_{j<i} \varepsilon_j(t)$. Further, we have,

$$\begin{aligned}
\Pi_i(t) A \Pi_i^T(t) &= \Pi_i(t)[A - \sum_{j=1}^{i-1} \lambda_j e_j e_j^T - O(\varepsilon(t)] \\
&= A - \sum_{j=1}^{i-1} \lambda_j e_j e_j^T - O(\varepsilon(t)
\end{aligned} \tag{A.21}$$

Expending v_i in terms of the eigenvectors, we have,

$$v_i(t) = \sum_{k=1}^{d} \alpha_k(t) e_k, \tag{A.22}$$

where $\alpha_k(t) = v_i^T(t)e_k$. Substituting Eq. A.21 and Eq. A.22 into Eq. A.18, and ignoring $O(\varepsilon(t))$ when t is large, we have,

$$\dot{\alpha} = \left(\frac{A_{\lambda i}}{\sqrt{\sum_{k=1}^{d} \alpha_k^2}} - I \right) \alpha$$

where, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)^T$, $||v_i|| = \sqrt{\sum_{k=1}^{d} \alpha_k^2}$, and $A_{\lambda i} = \text{diag}(0, 0, \ldots, 0, \lambda_i, \lambda_{i+1}, \ldots, \lambda_d)$. Since $\dot{\alpha}_j = -\alpha_j$ for $j < i$, we have $\alpha_j \to 0$ when $t \to \infty$. Dropping α_j $(j < i)$, we have very similar differential equations as in Eq. A.10. Following the proof in Section A.2.1– A.2.3, we can conclude with $\alpha_i \to \pm\lambda_i$ when $t \to \infty$.

Similar to the analysis in Section A.2.4, we have $v_i(n) \to \pm\lambda_i e_i$ $w.p.1$ when $n \to \infty$.

A.4 Conclusions

As a conclusion of above analysis, with the algorithm given by Eq. A.1 and Eq. A.2, $v_i(n) \to \pm\lambda_i e_i$ when $n \to \infty$, where λ_i is the i-th largest eigenvalue of the covariance matrix of $\{u(n)\}$, and e_i is the corresponding eigenvector.

Bibliography

[1] *Proc. of the 2nd International Conference on Development and Learning.* Cambridge, MA, June 12-15, 2002.

[2] K. Allan. *Natural Language Semantics.* Blackwell Publishers Ltd, Malden, MA, 2001.

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

[4] C. Balkenius. Generalization in instrumental learning. In P. Maes, M.J. Mataric, J.A. Meyer, et al., editor, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* The MIT Press/Bradford Books, Cambridge, MA, 1996.

[5] C. Balkenius and J. Moren. Computational models of classical conditioning: a comparative study. *Lund University Cognitive Studies*, 62, 1998.

[6] B.I. Bertenthal, J.J Campos, and K.C. Barrett. Self-produced locomotions: an organizer of emotional, cognitive, and social development in infancy. In

R. Emde and R. Harmon, editors, *Continuities and Discontinities in Development.* Plenum Press, New York, NY, 1984.

[7] H. Bourland and N. Morgan. Neural networks for statistical inference generalizations with applications to speech recognition. In *Proc. International Joint Conference on Neural Networks*, pages 242–247, Singapore, 1991.

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

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

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

[11] 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.

[12] H.H. Bulthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceeding of the National Academy of Sciences*, 89:60–64, January 1992.

[13] J.P. Byrnes. *Cognitive Development and Learning in Instructional Contexts.* Boston: Allyn and Bacon, 1995.

[14] D.J. Chalmers. Subsymbolic computation and the chinese room. In J. Dinsmore, editor, *The Symbolic and Connectionist Paradigms: Closing the Gap*, pages 25–48. Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[15] J.A. Clouse. Learning from an automated training agent. In G. Weis and S. Sen, editors, *Adaptation and learning in multiagent systems.* Springer Verlag, Berlin, 1996.

[16] M. Cole and S. R. Cole. *The Development of Children.* Freeman, New York, third edition, 1996.

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

[18] V.R. de Sa and D. Ballard. Category learning through multimodality sensing. *Neural Communication*, 10:1097–1117, 1998.

[19] J. R. Deller, Jone G. Proakis, and John H. L. Hansen. *Discrete-Time Processing of Speech Signals.* IEEE Press, New York, NY, 2nd edition, 2000.

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

[21] R.O. Duta, P.E. Hart, and D.G. Stork. *Pattern classification.* Wiley, New York, second edition, 2001.

[22] J.L. Elman, E.A. Bates, M.H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness: A connectionist perspective on development.* MIT Press, Cambridge, MA, 1997.

[23] J. Weng et al. Autonomous mental development by robots and animals. *http://www.cse.msu.edu/dl/whitepaper.pdf.*

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

[25] M. Fisz. *Probability theory and mathematical statistics.* John Wiley & Sons, Inc., New York, third edition, 1963.

[26] M. Franzini, M. Witbrock, and K.F. Lee. Speaker-independent recognition of connected utterances using recurrent and non-recurrent neural networks. In *Proc. International Joint Conference on Neural Networks*, 1989.

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

[28] S. Harnard. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

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

[30] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction To the Theory of Neural Computation.* Addison-Wesley, Reading, MA, 1991.

[31] H.V.B. Hirsch and D.N. Spinelli. Modification of the distribution of receptive field orientation in cats by selective visual exposure during development,. *Experimental brain research*, 13:509–527, 1971.

[32] T.S. Huang, L.S. Chen, and H. Tao. Bimodal emotion recognition by man and machine. In *Proc. ATR Workshop on Virtual Communication Environments*, Kyoto, Japan, April, 1998.

[33] W.M. Huang and R. Lippmann. Neural net and traditional classifiers,. In D. Anderson, editor, *Neural Information Processing Systems,*, pages 387–396. American Institute of Physics, New York, NY, 1988.

[34] X.D. Huang. Phoneme classification using semicontinuous hidden markov models. *IEEE Trans. Signal Processing*, 40(5), 1992.

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

[36] 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.

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

[38] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, 23:67–72, February 1975.

[39] G. Zavaliagkos, J. McDonough, D. Miller, et al. The bbn byblos 1997 large vocabulary conversational speech recognition system. In *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, pages 905–908, Seattle, Washington, May, 1998.

[40] M. Johnson. *The body in the mind: the bodily basis of meaning, imagination, and reason.* The University of Chicago, Chicago and London, 1974.

[41] D. Jurafsky and J.H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2000.

[42] E.R. Kandel, J.H. Schwartz, and T.M. Jessell, editors. *Principles of Neural Science.* Appleton and Lance, Norwalk, Connecticut, third edition, 1991.

[43] A.H. Kloph. A neuronal model of classical conditioning. *Psychobiology*, 16:85–125, 1988.

[44] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, second edition, 1988.

[45] T. Kohonen. *Self-Organization and Associative Memory.* Springer, Berlin, third edition, 1989.

[46] T. Kohonen. The self-organizing map. *Proc. of the IEEE*, 78:1464–1480, 1990.

[47] E. Kreyszig. *Advanced engineering mathematics.* Wiley, New York, 1988.

[48] H.J. Kushner and D.S. Clark. *Stochastic approximation methods for constrained and unconstrained systems.* Springer-Verlag, New York, 1978.

[49] L. Lamel and R. Cole. Language resources: spoken language corpora. In G. B. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, Cambridge, UK, 1997.

[50] C.H. Lee. On stochastic feature and model compensation approaches to robust speech recognition. *Speech Communication*, 25:29–47, 1998.

[51] Q. Liu, S. Levinson, Y. Wu, and T. Huang. Robot speech learning via entropy guided LVQ and memory association. In *Proc. INNS-IEEE International Joint Conference on Neural Networks*, pages 2176–2181, Washington, D.C., July 14-19, 2001.

[52] N Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5(5):552–563, 1995.

[53] N.J. Mackintosh. *Conditioning and Associative Learning.* Clarendon Press/Oxford University Press, Oxford/New York, 1983.

[54] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 200:269–294, 1978.

[55] N. Mercer. *Words and Minds: How We Use Language to Think Together.* Routledge, London, UK, 2000.

[56] J. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, pages 281–294, 1989.

[57] R.D. Mori and F. Brugnara. Hmm methods in speech recognition. In G. B. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, Cambridge, UK, 1997.

[58] E. Ochs and B Schieffelin. Language acquisition and socialization: three developmental stoires and their implications. In Shweder R. and LeVine R., editors, *Culture Theory: Essays on Mind, Self, and Emotion.* Cambridge University Press, Cambridge, UK, 1984.

[59] E. Oja. *Subspace Methods of Pattern Recognition.* Research Studies Press, Letchworth, UK, 1983.

[60] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Application*, 106:69–84, 1985.

[61] N.L. Owsley. Adaptive data orthogonalization. In *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, pages 109–112, Tulsa, Oklahoma, April 10-12 1978.

[62] J. Pauls, E. Bricolo, and N Logothetis. View invariant representations in monkey temporal cortex: position, scale, and rotational invariance. In Shree K.

Nayar and Tomaso Poggio, editors, *Early Visual Learning*, pages 9–41. Oxford University Press, New York Oxford, 1996.

[63] I.P. Pavlov. *Conditioned Reflexes: an Investigation of the Physiological Activity of the Cerebral Cortex.* Oxford University Press, London, 1927.

[64] B. Peskin, L. Gillick, and N. Liberman. Progress in recognizing conversational telephone speech. In *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, pages 1811–1814, Munich, Germany, April, 1997.

[65] P. J. Phillips, H. Moon, P. Rauss, and S. A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 137–143, Puerto Rico, June 1997.

[66] J. Piaget. *The origins of intelligence in children.* International Universities Press, New York, 1956.

[67] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing*, volume 1, pages 305–313. Morgran-Kaufmann Publishers, San Mateo, CA, 1989.

[68] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, New York, 2nd edition, 1986.

[69] P. Price. Spoken language input: spoken language understanding. In G. B. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, Cambridge, UK, 1997.

[70] S.E. Levinson, Q. Liu, et al. The role of sensorimotor function, associative memory and reinforcement learning in automatic acquisition of spoken language by an autonomous robot. In *Proc. Workshop on Development and Learning*, East Lansing, Michigan, April 5-7, 2000.

[71] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ, 1993.

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

[73] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[74] K. Richardson. *Models of Cognitive Development.* Psychology Press Ltd, East Sussex, UK, 1998.

[75] G. Rigoll. Maximum mutual information neural networks for hybrid connectionist-hmm speech recognition systems. *IEEE Trans. on Speech and Audio Processing*, 2(1), January 1994.

[76] A. Robinson and F. Fallside. Static and dynamic error propapgation networks with application to speech coding. In D. Anderson, editor, *Neural Information Processing Systems*, pages 632–641. American Institute of Physics, New York, NY, 1988.

[77] D. Roy. Learning from multimodal observations. In *Proc. IEEE Int. Conf. Multimedia and Expo*, New York, NY, July, 2000.

[78] D. Roy and A. Pentland. Learning words from sights and sounds: A computational model. *Cognitive Science*, 26(1):113–146, 2002.

[79] D. Roy, B. Schiele, and A. Pentland. Learning audio-visual associations using mutual information. In *Workshop on Integrating Speech and Image Understanding, Proc. Int'l Conf. Comp. Vision*, Corfu, Greece, September, 1999.

[80] J. Rubner and K. Schulten. Development of feature detectors by self-organization. *Biological Cybernetics*, 62:193–199, 1990.

[81] S. Russell and R. Norvig. *Artificial Intelligence – A Modern Approach.* Prentice-Hall, Inc., New Jersey, 1995.

[82] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, 26:43–49, February 1978.

[83] L.M. Saksida, S.M. Raymond, and D.S. Touretzky. Shaping robot behavior using principles from instrumental conditioning. *Adaptive Behavior*, 22(3/4):231–249, 1998.

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

[85] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *IEEE Trans. Neural Networks*, 2:459–473, 1989.

[86] I. Sirovich and M. Kirby. Low-dimensional procedure for the caracterization of human faces. *Journal of Optical Society of America A*, 4(3):519–524, March 1987.

[87] O. Sporns, N. Almassy, and G.M. Edelman. Plasticity in value systems and its role in adaptive behavior. *Adaptive Behavior*, 7(3), 1999.

[88] R.S. Sutton and A.G. Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological Review*, 88:135–170, 1981.

[89] R.S. Sutton and A.G. Barto. A temporal-difference model of classical conditioning. In *Proc. the Ninth Conference of the Cognitive Science Society*, Erlbaum, 1987.

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

[91] Zeppenfeld T., Finke M., Ries K., Westphal M., and Waibel A. Recognition of conversational telephone speech using the janus speech engine. In *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, pages 1815–1818, Munich, Germany, April, 1997.

[92] P.A. Thompson. An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise. In *Proc. 13th Asilomar Conf. on Circuits, System and Computers*, pages 529–533, Pacific Grove, CA, 1979.

[93] K. Torkkola and M. Kokkonen. Using the topology-preserving properties of sofms in speech recognition. In *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, pages 261–264, Toronto, Canada, 1991.

[94] D.S. Touretzky and L.M. Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5(3/4):219–247, 1999.

[95] T. Vintsyuk. Element-wise recognition of continuous speech composed of words from a specified dictionary. *Kibernetika*, 7:133–143, March-April 1971.

[96] P. Violi. *Meaning and Experience.* Indiana University Press, Bloomington, IN, 2001.

[97] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37, March 1989.

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

[99] R. Watrous. *Speech Recognition using Connectionist Networks.* PhD thesis, University of Pennsylvania, 1988.

[100] J. Weng. The living machine initiative. Technical Report CPS 96-60, Department of Computer Science, Michigan State University, East Lansing, MI, Dec. 1996.

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

[102] J. Weng and I. Stockman (eds). *Proceedings of NSF/DARPA Workshop on Development and Learning.* East Lansing, Michigan, April 5-7, 2000.

[103] J. Weng, T. S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences.* Springer-Verlag, New York, 1993.

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

[105] J. Weng, W.S. Hwang, Y. Zhang, and C. Evans. Developmental robots: theory, method and experimental results. In *Proc. 2nd International Symposium on Humanoid Robots*, pages 57–64, Tokyo, Japan, October 8-9, 1999.

[106] J. Weng, W.S. Hwang, Y. Zhang, C. Yang, and R. Smith. Developmental humanoids: humanoids that develop skills automatically. In *Proc. The First IEEE-RAS International Conference on Humanoid Robots*, Boston, MA, September 7-8, 2000.

[107] J. Weng, Y. B. Lee, and C. H. Evans. The developmental approach to multimedia speech learning. In *Proc. IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing*, pages 3093–3096, Phoenix, Arizona, March 15-19, 1999.

[108] J. Weng, J McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291:599–600, January 26, 2001.

[109] J. Weng, Y. Zhang, and W.S. Hwang. Teaching a learning vehicle - a developmental perspective. In *Proc. Robotics and Mechatronics Congress (RMC2001)*, Singapore, June 6-8, 2001.

[110] S.L. Zeger and B. Qaqish. Markov regression models for time series: a quasi-likelihood approach. *Biometrics*, 44(4):1019–1031, December 1988.

[111] Y. Zhang and J. Weng. Grounded auditory development of a developmental robot. In *Proc. INNS-IEEE International Joint Conference on Neural Networks*, pages 1059–1064, Washington, DC, July 14-19, 2001.

[112] V. Zue, R. Cole, and W. Ward. Spoken language input: speech recognition. In G. B. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology*. Cambridge University Press, Cambridge, UK, 1997.