This is to certify that the
dissertation entitled

Cortex-inspired Developmental Learning for Vision-based Navigation, Attention and Recognition

presented by

Zhengping Ji

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science

Major Professor's Signature

11/25/2009

Date

MSU is an Affirmative Action/Equal Opportunity Employer

CORTEX-INSPIRED DEVELOPMENTAL LEARNING FOR VISION-BASED NAVIGATION, ATTENTION AND RECOGNITION

By

Zhengping Ji

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Computer Science

2009

ABSTRACT

CORTEX-INSPIRED DEVELOPMENTAL LEARNING FOR VISION-BASED NAVIGATION, ATTENTION AND RECOGNITION

By

Zhengping Ji

A series of conceptual and technical breakthroughs have been made in this dissertation as part of research efforts on autonomous mental development (AMD) [94].

We first designed a developmental learning architecture to develop covert and overt behaviors in the challenging task of vision-based navigation, using reinforcement learning and supervised learning jointly. Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) tree was developed as a cognitive mapping engine to automatically generate internal representations, without a need of human programmers to pre-design task-specific (symbolic) concepts. Its balanced coarse-to-fine tree structure guaranteed real-time retrieval in self-generated high-dimensional state space. K-Nearest Neighbor strategy was adopted in a modified Q-learning to dramatically reduce the training time complexity.

Our further research efforts have been focused on the computational modeling of development in the biological brain. The genomic equivalence principle in biology implied that the learning of every neuron is in-place, indicating that each neuron (cell) is fully responsible for its own development and adaptation while interacting with its cellular environment. Using the in-place learning principle, we introduced a sensorimotor pathway with multiple sensor integration to address a series of challenging limitations that existing neural networks face. The biologically motivated sensorimotor pathway provided attention selection capabilities, sparse-coded representation, and motor association for the temporally dense real-time sequences. Its bottom-up and top-down projections enable unsupervised and supervised learning to occur concurrently.

The in-place learning principle was extended to a new type of cortex-inspired architecture, called Where-What Network (WWN), to model the brain's visual dorsal ("where") and ventral ("what") pathways. This type of general-purpose visuomotor network developed three types of attention: feature-based bottom-up attention, position-based top-down attention, and object-based top-down attention, as three possible information flows through the Y-shaped network. Compared with the previous sensorimotor pathway, the WWN modeled both intra-cortical architecture and inter-cortical wiring (locally and globally) using 6-layer organization scheme in each cortical area. Impressive recognition rate and attention accuracy were reached in a complex dynamic visual environment containing objects in any possible positions.

To My Parents and My Wife.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. Juyang Weng, for his leading efforts and research advice during the past four years. Without his insights, encouragements and supports, the dissertation would not have been completed. I am grateful to my committee members, Dr. Charles Ofria, Dr. Rong Jin, Dr. George Stockman, and Dr. Lalita Udpa, for their helpful suggestions and comments on my research topics and dissertation writing. The gratitude also goes to my mentor and committee member, Dr. Danil Prokhorov at Toyota Technical Center, for his constructive discussions and extensive supports throughout my research.

I would like to acknowledge my special thanks to my colleague, Dr. Shuqing Zeng at General Motor Research, who provided his scientific expertise and invaluable guidance for our collaborated projects.

I also wish to extend thanks to the members of EI Laboratory, past and present, at Michigan State University. My early-stage research benefited from IHDR programs provided by Xiao Huang and Wey-Shiuan Hwang. Matthew Luciw and Mojtaba Solgi invested huge amount of time to help me on the in-place learning algorithm and programming. I also thank Paul Cornwell, Hangqing Zhao, Faliang Chang, Guojin Zhu,

Xiaoan Li, Arash Ashari, Charles Bardel, Kajal Miyan, Feilong Chen, Wei Tong, and Yang Zhou for collaborations and idea exchanges.

Thanks also go to my friends Changqing Chen, Bo Wang, Minghang Cai, Chaopeng Shen, Xiaoguang Lv, Hong Chen, Yi Chen, for their continuous friendship and essential assistance in the past several years.

Last, but not the least, I am grateful to my mom, Rong Hu, and my dad, Chuansheng Ji, as well as my wife Xue Wu for their care, supports and encouragements trough all my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Autonomous Mental Development

The history of machine intelligence followed a well-established engineering paradigm — the task-specific paradigm: human designers are required to explicitly program task-specific representation, perception and behaviors, according to the tasks that the machine is expected to execute. However, AI tasks require capabilities such as vision, speech, language, and motivation which have been proved to be too muddy to program effectively by hand. In this chapter, we present the new task-nonspecific paradigm — autonomous mental development (AMD) (proposed by Weng et. al 2001 [94]) for making man-made machines, aiming to fully automate the mental development process. The approach does not only drastically reduce the programming burden, but also enables machines to develop capabilities or skills that the programmer does not have or are too muddy to be adequately understood.

1.1 Motivation

Existing learning techniques applied to robot learning (e.g., Hexmoor et al. 1997 [31]) differ fundamentally from human mental development. As early as the late 1940s, Alan Turing envisioned a machine that can learn, which he called the "child machine." In his article "Computing Machinery and Intelligence" [82], he wrote:

> "Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child."

In some sense, Turing was among the first few who roughly envisioned a developmental machine, although it was probably too early for him to be adequately aware of the fundamental differences between a program that is tested for its "imitation game" and a "child machine" that mentally develops in the real world, like a human child.

A large amount of work on machine learning was motivated by human learning but not development per se. Soar (Laird, Newell & Rosenbloom 1987 [51]) and ACT-R (Anderson 1993 [2]) are two well-known symbolic systems with an aim to model cognitive processes, although cognitive development was not the goal of these efforts. The finite state machines (FSM) (Gill 1962 [27]), or its probability based variant, Hidden Markov Models (MMM) (Rabiner 1989 [69]) and Markov Decision Process (MDP) (Seneta

1996 [72]), are two general frameworks that have been used for conducting autonomous learning with a given goal in a symbolic world (e.g., Shen 1994 [74]) or reinforcement learning (e.g., Kaelbling, Littman & Moore 1996 [46]). The explanation-based neural network was used by Thrun 1996 [78] to take advantage of domain knowledge that is shared by multiple functions, an approach called "lifelong learning."

Some studies did intend to model cognitive development, but in a symbolic way. BAIRN (a Scottish word for "child") is a symbolic self-modifying information processing system as an implementation for a theory of cognitive development (Wallance, Klahr & Bluff 1987 [85]). Drescher 1991 [21] utilized the schema, a symbolic tripartite structure in the form of "context-action-result," to model some ideas of child sensory-motor learning in Piaget's constructivist theory of cognitive development.

Numeric incremental learning has been a major characteristic of the computational intelligence [55]. Cresceptron (Weng, Ahuja & Huang 1997 [88]) is a system that enables a human teacher to interactively segment natural objects from complex images through which it incrementally grows a network that performs both recognition and segmentation. SHOSLIF for teaching a robot eye-hand system interactively (Hwang & Weng 1997 [37]), the hand-arm system of Coelho, Piater & Grupen 2000 [13], SACOS (Vijayahumar & Schaal 2000 [84]), Cog (Brooks, et al. 1999 [8]) and Kismet (Breazeal 2000 [7]) are motivated by simulating early child skills via learning through interactions with the environment.

All of the above efforts aim at providing a general framework for some cognitive process or simulating some aspects of learning and development. However, these systems do not perform autonomous mental development (AMD). For example, it is a human who manually designs a model representation to a given task. We call this type of mapping as task-to-representation mapping. The way a human manually establishes such a mapping takes several forms:

1. Mapping between real-task concepts and symbols (e.g., soar and ACT-R).
2. Mapping between real-task concepts and model structures (e.g., FSM, HMM and MDP).
3. Mapping between real-task features and input terminals (e.g., skin color and region position as input terminals for detecting human faces).

The AMD is powerful since it does not require a human being to provide any of the above task-to-representation mapping.

1.2 A New Engineering Paradigm

A new paradigm for constructing machines is needed for muddy tasks that humans do well but conventional machines do not. This new paradigm is called autonomous mental development paradigm or AMD paradigm.

1.2.1 AMD Paradigm

In contrast with the manual development paradigm formulated in Sec. 1.1, the AMD paradigm enables a machine to develop its mind autonomously.

The new AMD paradigm (proposed by Weng et. al 2001 [94]) is as follows.

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

2. Designing a developmental program. The human designer designs the developmental program for the robot. Since he does not know what specific tasks the machine will learn, the developmental program does not require any information about the tasks. However, the developmental program may take advantage of the sensors, effectors and the computational resources that have been previously chosen. Prenatal development can be conducted in this stage, using spontaneous (i.e., self-generated) activities.

3. Birth. The human operator turns on the robot whose computer starts to run the developmental program so that the robot starts to interact with the real physical

world autonomously.

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

This AMD paradigm does not start with any specific task and, in fact, the tasks are unknown at the time of machine construction (or programming). The hallmark of the AMD paradigm is that the human engineer does not need to understand or even anticipate the tasks to be learned by the machine. Consequently, the task-specific representation must be generated autonomously by the robot itself, instead of being designed by the human programmer.

1.2.2 Paradigm Comparison

Fig. 1.1 compares the traditional manual development paradigm and the AMD paradigm.

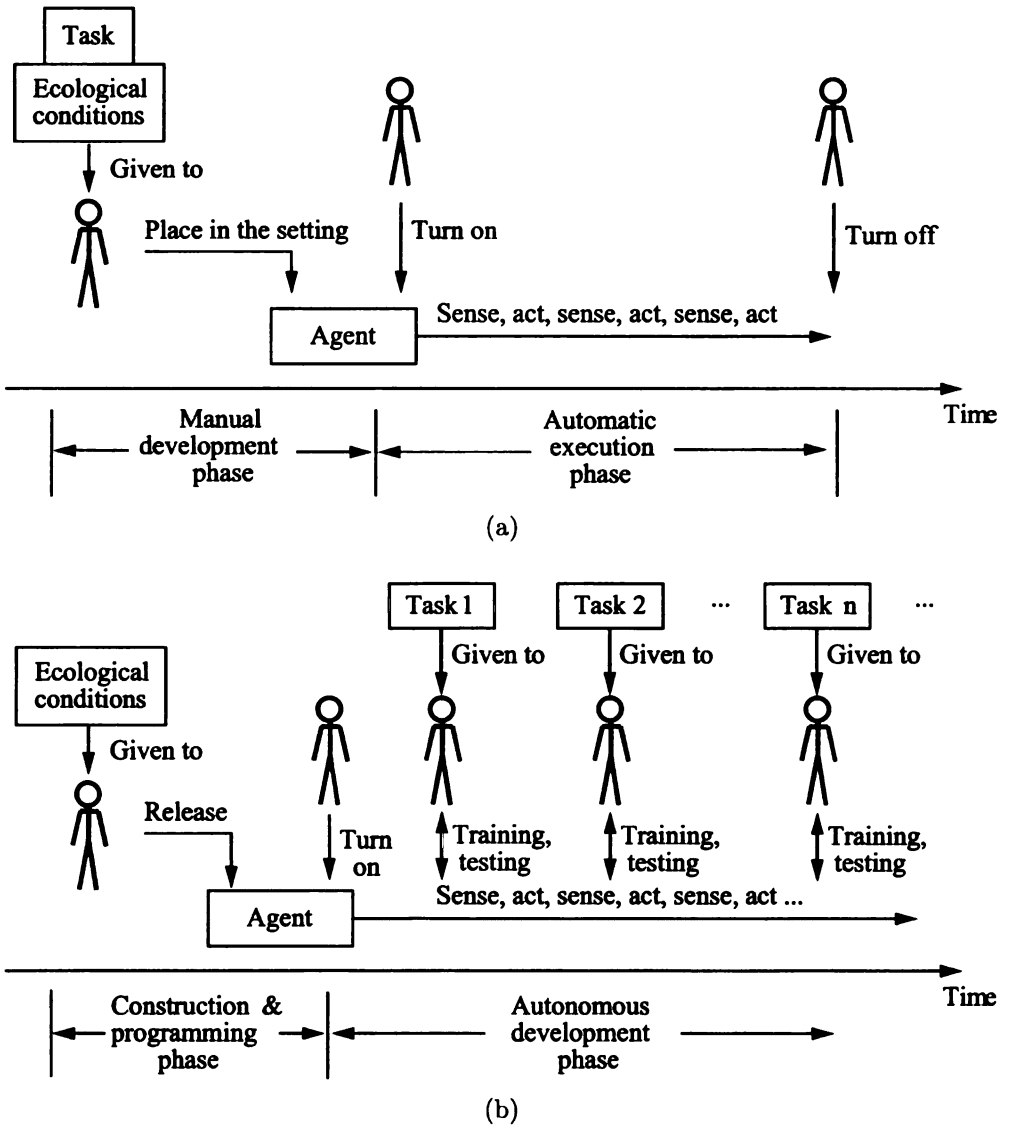


Figure 1.1. Comparison between (a) the manual development paradigm and (b) the autonomous development paradigm. Figure courtesy of Weng et al. 2001 [94]

The term "manual" in the manual development paradigm refers to manually developing task-specific architecture, representation and skills, where two phases are included: the manual development phase and the automatic execution phase. In the first phase, the human developer H is given a specific task T to be performed by the machine and a set of ecological conditions E_c about the operational environment. First, the human developer understands the task. Next, he designs a task-specific architecture and representation and then programs the agent A. In mathematical notation, we consider a human as a (time varying) function that maps the given task T and the set of ecological conditions E_c to agent A:

$$A = H(E_c, T). \tag{1.1}$$

In the second automatic execution phase, the machine is placed in the task-specific setting. It operates by sensing and acting. It may learn using sensory data to change some of its internal parameters. However, it is the human who understands the task and programs its internal representation. The agent only runs the program.

In contrast, the AMD paradigm has different two phases: (1) the construction and programming phase and (2) the autonomous development phase. In the first phase, tasks that the agent will end up learning are unknown to the robot programmer. The programmer might speculate some possible tasks, but writing a task-specific representation is not possible without actually being given a task. The ecological conditions under which the robot will operate, e.g., land-based or underwater, are provided to

the human developer so that he can design the agent body appropriately. He writes a task-nonspecific program called a developmental program, which controls the process of mental development. Thus, the newborn agent $A(t)$ is a function of a set of only ecological conditions, but not the task:

$$A(0) = H(E_c), \tag{1.2}$$

where we added the time variable t to the time varying agent $A(t)$, assuming that the birth time is at $t = 0$.

After the robot is turned on at time $t = 0$, the robot is "born" and starts to interact with the physical environment in real-time by continuously sensing and acting. This phase is called the autonomous development phase. Human teachers can affect the developing robot only as a part of the environment, through the robot's sensors and effectors. After birth, the internal representation is not accessible to the human teacher.

During the autonomous development phase, the tasks are designed by different human teachers according to the mental maturation of the robot as well as its performance. Earlier behaviors are developed first (e.g., moving around guided by vision), before learning new tasks (e.g., delivering objects).

1.3 Computational Challenges

In computational aspects, the natural and engineered AMD must meet the following eight necessary challenges.

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

2. High-dimensional sensors. The dimension of a sensor is the number of receptors. AMD must deal directly with continuous raw signals from high-dimensional sensors, e.g., vision, audition and taction.

3. Completeness in stage representation. Due to the environmental openness and task nonspecificity, it is not desirable for a developmental program to discard, at any processing stage, information that may be useful for future, unknown tasks. The representation should be sufficiently complete for the purpose of generating desired behaviors.

4. Online processing. Online means that the agent must always ready to accept and process sensory information. At each time instant, what the agent will sense next depends on what the agent does now.

5. Real-time speed. The sensory and 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 30Hz for vision). It must be able to deal with one-shot learning , memorizing from a brief experience, e.g., 200 ms.

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. A developmental agent must perform while it "builds" itself "mentally."

8. Scale up to large memory. For large perceptual and cognitive tasks, an AMD agent must be able to deal with large memory and still handle both discrimination and generalization for increasing maturity, all without catastrophic memory loss.

Aforementioned research efforts in Sec. 1.1 mainly aimed at a subsets of the above eight challenges but not all, while the AMD discussed here must deal with all of the eight challenges.

1.4 Contributions

A series of conceptual and computational breakthroughs have been made in this dissertation as part of research efforts on AMD. The major contributions of the dissertation

are summarized as below.

1. Designed a developmental learning architecture with applications to vision-based navigation, such that

   (a) Reinforcement learning of covert behaviors and supervised learning of overt behaviors are integrated into the same interactive learning system.

   (b) Locally Balanced Incremental Hierarchical Discriminant Regression (LBI-HDR) Tree was developed as a cognitive mapping engine to automatically generate internal representations for active vision development, without a need of human programmers to pre-design task-specific (symbolic) representation.

   (c) K-Nearest Neighbor strategy was adopted in a modified Q-learning with short reward delays, dramatically reducing training time complexity in the fast changing perceptions.

2. Introduced a developmental sensorimotor pathway with multiple sensor integration, using what is called in-place learning. The presented work is important and unique in the following senses:

   (a) Multiple sensory modalities were integrated to enable saliency-based attention for the temporally dense real-time sequences.

   (b) Motivated by the early coding scheme in the visual pathway, orientation

selective features were developed in the network to provide a sparse representation of salient areas.

(c) Motor-driven top-down connections enabled sensory invariance to bridge the well-known gap between "semantic symbols" and the network internal representation.

(d) In the aspect of recognition accuracy, computational resource and time efficiency, a comparative study was conducted to explore differences between the in-place learning and LBIHDR, along with other incremental learning methods.

3. Developed a new in-place architecture, called Where-What Network, to model the brains visual dorsal ("where") and ventral ("what") pathways. A series of advances were made along this line of research, including

(a) Computational modeling of attention and recognition as ubiquitous internal actions that take place virtually everywhere in the brain's cortical signal processing and thus sequential decision making.

(b) Computational understanding of saliency-based bottom-up attention, position-based top-down attention and object-based top-down attention as concurrent information stream.

(c) Success in the theory and demonstration of key brain-scale learning mechanisms for intra-cortical architecture and inter-cortical (i.e., cortico-cortical)

wiring.

1.5 Organizations

The rest of this dissertation is organized as follows. Chapter 2 describes a developmental learning architecture to develop covert and overt behaviors in vision-based navigation, using reinforcement learning and supervised learning jointly. Chapter 3 discusses the properties of networked neurons and their representations about how sophisticated mental functions are autonomously developed through extensive embodied living experiences. Chapter 4 presents a context-inspired, developmental pathway, to address a series of challenging limitations that face models of object learning in a complex, natural driving environment. Chapter 5 extends the same learning principle to model the brain's visual dorsal ("where") and ventral ("what") pathways using what is called Where-What Network (WWN). Chapter 6 presents conclusions and identifies directions for future research.

CHAPTER 2

Developmental Learning of Covert and Overt Behaviors in Vision Navigation

Two types of sensorimotor behaviors are involved in a developmental agent: (1) covert behaviors, such as attention, acting on the internal representation with virtual internal effectors; (2) overt behaviors, via external effector, which can be directly imposed (e.g. motors, controllers etc.) from the external environment. In this chapter, we propose a developmental learning architecture for a vehicle-based agent, where covert and overt behaviors are developed to fulfill a challenging task of vision-based navigation. The covert behaviors entail attention to road boundaries, the learning of which is accomplished by a motivational system through reinforcement signals. The overt behaviors are heading directions that can be directly observed by the external environment, including a teacher, where supervised learning is conducted. Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) tree is developed as a cognitive mapping engine to automatically generate internal representations. Its

balanced coarse-to-fine tree structure guarantees real-time retrieval in self-generated high-dimensional state space. K-Nearest Neighbor strategy is adopted in a modified Q-learning to dramatically reduce the training time complexity.

The text of this chapter, in part, is adapted from Ji, Huang and Weng 2008 [40]. Explicit credits will be given in sections.

2.1 Related Work

The covert and overt behaviors have different appropriate learning modes. Supervised learning is regarded as an effective mode to develop a robot's overt behaviors in vision-based navigation (e.g., steering angles and speeds). It is because that overt outputs are often well-defined and can be directly observed from external environment, including a supervisor/teacher. Pomerleau's ALVINN (Autonomous Land Vehicle In a Neural Network) system [68] used a learned neural network to control an autonomous vehicle. The input to the neural network was a 30×32 gray image and the network output was the supervised direction in which the vehicle is steered. Its later versions [43], [44] came with additional capabilities regarding the road detection and confidence estimation. Matsumoto et al., 1996 [98] extended a recognition idea in Horswill's mobile robot Polly [33] by using a sequence of images and a template matching procedure to guide the robot navigation. Gaussier et al. 1997 [65] and Joulian et al. 1997 [10] extracted local views from a panoramic image, where a neural network was used to

learn the association with a direction (supervised signal) to a target goal. More recent vision-based navigation work (e.g., Hong et al. 2002 [32] and Leib et al. 2005 [53]) involved a supervised mapping from color-based histograms to traversability information. However, the assumption that transferable path has similarly colored or textured pixels may not be true in a complex road condition. The DARPA Grand Challenge [15] and Urban Challenge [16] activated many navigation systems (e.g., Thrun et al. 2006 [79] and Urmson at al. 2008 [83]) for autonomous driving, but most of these systems used expensive high-definite LIDAR system, such as Velodyne's HDL-64E sensor [1], to reconstruct the 3-D environment through clouds of geometric data points. The on-going research is more focused on using compact and cheaper sensors, e.g., video cameras, to make autonomous navigation more attractive for commercial usage, such as adaptive cruise control and assistant parking.

Compared to supervised learning, reinforcement learning is indispensable in following aspects of vision-based navigation. (1) Supervised learning is not enough to model sophisticated robotic cognition since it fails to explore the cognitive internal actions. Especially in the vision-based navigation, supervised learning may not allow an autonomous vehicle to "experience" situations that require the correction from a misaligned trajectory. In addition, supervised learning may overtrain the system to straight roads, in the sense that the robotic agent can forget what it learned earlier about driving on curved roads (see detailed discussion in Desouza and Kak 2002 [20]). On the

[1]specifications at http://www.velodyne.com/lidar/vision/default.aspx

other hand, reinforcement learning entails long-term interactions with the exploration of perceptual environment, rather than the imposition of immediate input/output pairs . (2) The progress of supervised learning is too tedious and requires intensive human impositions. The supplying of reinforcement signals, on the other hand, involves much less human efforts, which does not necessarily know the expected outputs. (3) If the robot receives incorrect supervision from a teacher, the wrong action takes an immediate effect, posing difficulties for the robot to make a correction. However, reinforcement learning provides a progressive reward strategy and gives the robot an ability to recover from errors of a sensorimotor behavior.

Quite a few studies applied reinforcement learning for covert vision behaviors, e.g., perceptual attention selection. Whitehead and Ballard's study is one of the earliest attempts to use reinforcement learning as a model of active perception [97]. Balkenius 1999 [5] modeled attention as selection of actions. They trained a robot to pick out the correct focus of attention using reinforcement learning. Experimental results are reported on a simple simulator. Bandera 1996 [9] used reinforcement learning to solve gaze control problem. However, their application was in an artificial environment. Its performance in the real world is unknown. Minut and Mahadevan 2001 [57] proposed a reinforcement learning model of selective visual attention. The goal was to use a fixed pan-tilt-zoom camera to find an object in a cluttered lab environment, i.e., a stationary environment.

Compared to aforementioned studies, the work presented here is unique and important in the following senses. (1) Reinforcement learning of covert behaviors and supervised learning of overt behaviors are integrated into the same interactive learning system for vision-based navigation. (2) The learning of covert behaviors provides perceptual attention on road boundaries, free of dependance on road-specific features, such as textures, edges, colors, etc. (3) All the internal representations (e.g., clusters that represent distributed states) are generated automatically (sensor-driven) without a need for a human programmer to pre-design task-specific (symbolic) representation. (4) Instead of using long delay of rewards in traditional reinforcement learning, we use short reward delays only, in order to solve the fast changing perception problems. (5) To reach the goal of real-time and incremental vision development in non-stationary outdoor environment (vision-based navigation), we introduce LBIHDR (Locally Balanced Incremental Hierarchical Discriminant Regression) as the cognitive mapping engine, which generates continuous states of action space for both internal and external sensations. Its balanced coarse-to-fine structure guarantees real-time retrieval. (6) We investigate the k-Nearest Neighbor algorithm for Q-learning so that at each time instant multiple similar states can be updated, which dramatically reduces the number of rewards that human teachers have to issue.

In what follows, we first describe the covert and overt behaviors in the vision-based outdoor navigation. The detailed architecture of the learning paradigm is presented in Sec. 2.3. The cognitive mapping engine – LBIHDR is described in Sec. 2.4 for

the high-dimensional regression through internal representations. Sec. 2.5 presents a motivational system to modulate the mapping from internal states to internal actions by reinforcement learning. The overall computational steps from sensors to effectors are presented in Sec. 2.6, with an emphasis about how the two types of behaviors are distinguished by their respective learning modes. The experimental results and conclusions are reported in Secs. 2.7 and 2.8, respectively.

2.2 Covert and Overt Behaviors in Vision Navigation

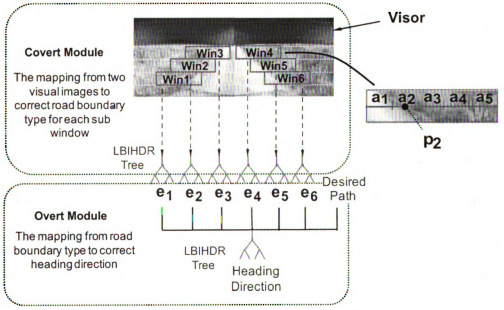


Figure 2.1. The information flow of covert and overt modules in the vision-based outdoor navigation.

Fig. 2.1 shows covert and overt modules of the proposed learning agent in vision-based navigation. An example of visual inputs is shown in Fig. 2.1 (top), with left

and right color images captured by an autonomous driving vehicle. We used a visor on each camera to block the view above road surface, avoiding the distractions from other scenes than road. Part of vehicle body is sensed at bottom area of the image, which is not essential for the robot's navigation. Thus, in Fig. 2.1, we only consider the road region between the top dash line and the bottom dash line. Instead of using the entire road region as input, we further divide two images into 6 sub-windows, specified by rectangles in Fig. 2.1. Given the size of the original image 160(columns)×80(rows)×3 (colors), the dimension of input vector is 38400, which is almost intractable. With six attention windows, the dimension of input vector (for each window) is reduced to $80 \times 20 \times 3 = 4800$. And, it turns out, these six attention windows can cover most of the road boundaries.

Let us look at the extracted window image in Fig. 2.1 (upper right). We divided the horizontal centerline of the window into five ranges, $\{a_1, a_2, a_3, a_4, a_5\}$, corresponding to 5 different road boundary types. In this example, the road boundary intersects the centerline at the point p_2, which falls into the second range. Thus, a_2 is regarded as the expected boundary type for this window. In practice, however, it is possible that the boundary may not stay in the windows. While the road boundary falls on the left side and the right side of an attention window, we choose a_1 and a_5 respectively as the expected boundary type. In these situations, the expected output does not exactly fit the road boundaries. However, due to the incremental effect of frequent adjustment, it will not affect too much on the final decision of heading directions.

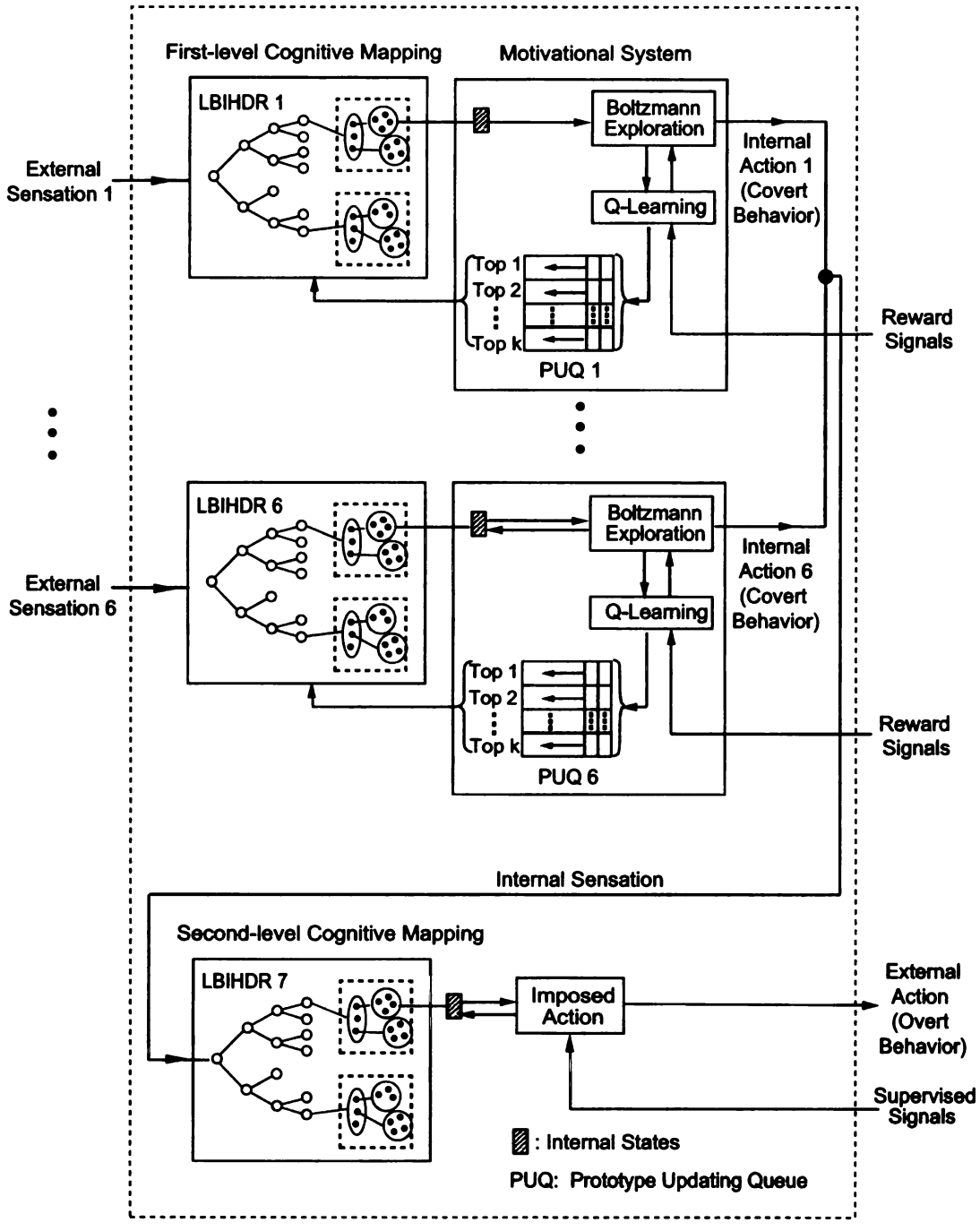


Figure 2.2. The system learning architecture. Two-level cognitive mapping engines provide the internal representations bridging the external sensations and external actions. Given one of six external sensations, i.e., window images, a motivational system with reinforcement learning develops internal actions (covert behaviors) using internal representations from the first-level cognitive mapping. Supervised learning is adopted to develop the external actions (overt behaviors), using internal representations from the second-level cognitive mapping.

A cognitive mapping engine LBIHDR and a motivational system work together to learn the road boundary type (cover behavior) of each window image. A learned road boundary type corresponds to an attentional range, the middle point of which is regarded as the attended boundary location for the current window image. 6 boundary locations are extracted in total to provide a road vector $\mathbf{e} = (e_1, e_2, ..., e_6)$, where e_i is the boundary location for attention window i. This vector, plus a desired path with 10 global coordinates (defined by maps), is mapped to the external effector, where heading directions (overt behaviors) are learned by supervised learning.

2.3 System Architecture

Fig. 2.2 illustrates the system learning architecture for the development of covert and overt behaviors. In our case, the external sensation $\mathbf{r}(t)$ presents one of six window images. Each cognitive mapping LBIHDR i ($i = 1, 2, ..., 6$) clusters the corresponding $\mathbf{r}(t)$ into an internal representation, through which the LBIHDR performs a hierarchical search efficiently to find the best matched internal state. Each internal state is associated with a list of internal actions $A = \{a_1(t), a_2(t), a_3(t), a_4(t), a_5(t)\}$ and corresponding Q-values $Q = \{q_1(t), q_2(t), q_3(t), q_4(t), q_5(t)\}$. The motivational system maps internal state space to internal action space through the reinforcement learning.

Based on the internal actions learnt from six window images, the road boundary vector $\mathbf{e}(t)$ is generated. The supervised learning of LBIHDR 7 maps the road boundary

and its correspondent pre-defined desired path $\mathbf{p}(t)$ to the best matched state associated with a heading direction $h(t)$.

In the next two sections, we will describe major components of the cognitive mapping engine and the motivational system respectively.

2.4 Cognitive Mapping

In the cognitive mapping engine LBIHDR, each internal state s is presented by a pair of $(\mathbf{x}, y)$ in the leaf node, called primitive prototype. $\mathbf{x}$ denotes an internal or external sensation while y denotes an internal or external action. In the proposed navigation system, the prototype of the first-level LBIHDR is a pair of $(\mathbf{r}(t), a^*(t))$, where $a^*(t)$ is the internal action with largest Q-value, and the prototype of the second-level LBIHDR is a pair of $(\mathbf{e}(t) + \mathbf{p}(t), h(t))$.

2.4.1 Incremental Hierarchical Discriminant Regression - IHDR

Fig. 2.3 illustrates the structure of an IHDR tree, originally proposed by Weng and Hwang 2007 [89] . Two types of clusters are incrementally updated at each node of the IHDR tree — y-clusters and x-clusters. The y-clusters are clusters of prototypes in the output space $\mathcal{Y}$ and x-clusters are those in the input space $\mathcal{X}$. Each x-cluster is associated with a unique y-cluster. There are a maximum of q x-clusters at each node

and the centroids of these q x-clusters are denoted by

$$C = \{c_1, c_2, ..., c_q \mid c_i \in \mathcal{X}, i = 1, 2, ..., q\}. \tag{2.1}$$

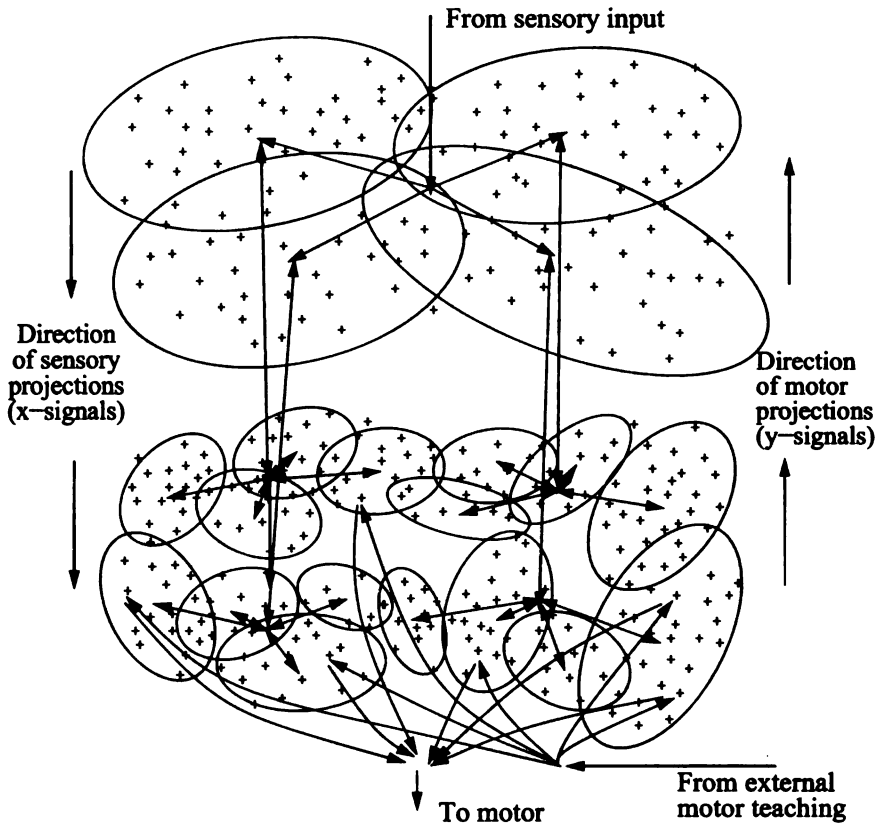


Figure 2.3. Illustration of an IHDR tree. The sensory space is represented by sensory input vectors, denoted by "+". The sensory space is repeatedly partitioned in a coarse-to-fine way. The upper level and lower level represent the same sensory space, but the lower level partitions the space finer than the upper level does. Each node has q feature detectors ($q = 4$ in the figure), which collectively determine to which child node that the current sensory input belongs. An arrow indicates a possible path of the signal flow. In every leaf node, primitive prototypes (marked by "+") are kept, each of which is associated with the desired motor output. Figure courtesy of Weng and Hwang 2007 [89].

Theoretically, the clustering of $\mathcal{X}$ of each node is conditioned on the class of $\mathcal{Y}$ space. The distribution of each x-cluster is the probability distribution of random variable $\mathbf{x} \in \mathcal{X}$ conditioned on random variable $y \in \mathcal{Y}$. Therefore, the conditional probability

density is denoted as $p(\mathbf{x}|y \in c_i)$, where c_i is the i-th y-cluster, $i = 1, 2, ..., q$.

At each node, y part in $(\mathbf{x}, y)$ finds the nearest y-cluster in Euclidean distance and updates (pulling) the centroid of the y-cluster. This y-cluster indicates to which corresponding x-cluster the input $(\mathbf{x}, y)$ belongs. Then, the $\mathbf{x}$ part of $(\mathbf{x}, y)$ is used to update the statistics of the x-cluster. The statistics of every x-cluster are then used to estimate the probability for the current sample $(\mathbf{x}, y)$ to belong to the x-cluster, whose probability distribution is modeled as a multi-dimensional Gaussian at this level. In other words, each node models a region of the input space $\mathcal{X}$ using q Gaussians. Each Gaussian will be modeled by more small Gaussians in the next tree level if the current node is not a leaf node.

The centroids of these x-clusters provide essential information for discriminating subspace. We define the most discriminating feature (MDF) subspace $\mathcal{D}$ as the linear space that passes through the centroids of these x-clusters. A total of q centroids of the q x-clusters give $q - 1$ discriminating features which span $(q - 1)$-dimensional discriminating space $\mathcal{D}$.

IHDR updates the node's statistics (the centroid vector and the covariance matrix) incrementally. The centroid of n input examples $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ is recursively computed from the current input data $\mathbf{x}_n$ and the previous average $\bar{\mathbf{x}}^{(n-1)}$ by Eq. (2.2):

$$\bar{\mathbf{x}}^{(n)} = \frac{n - 1 - \mu(n)}{n} \bar{\mathbf{x}}^{(n-1)} + \frac{1 + \mu(n)}{n} \mathbf{x}_n \tag{2.2}$$

where $\mu(n)$ is an amnesic function defined by Eq. 3.18 in Chaper 3 (see page for details).

The covariance matrix can be updated incrementally by using the amnesic average as well.

$$\begin{aligned} \Gamma_{\mathbf{x}}^{(n)} &= \frac{n-1-\mu(n)}{n}\Gamma_{\mathbf{x}}^{(n-1)} \\ &+ \frac{1+\mu(n)}{n}(\mathbf{x}_n - \overline{\mathbf{x}}^{(n)})(\mathbf{x}_n - \overline{\mathbf{x}}^{(n)})^T \end{aligned} \tag{2.3}$$

where the amnesic function $\mu(n)$ changes with n as we discussed above.

If the node is mature enough (the number of samples hits a certain limitation), it will spawn q children, which means the region of space is modeled by a finer Gaussian mixture. Thus, a coarse to fine approximation of probability distribution of the training samples is achieved.

2.4.2 Local Balance and Time Smoothing

IHDR faces a challenging problem when the number of different action output is too small to enable the span of a sufficient input feature space.

As shown in Fig. 2.4, given two y-clusters (i.e., two classes), specified by "+" and "-", there are 3 input dimensions. $\mathbf{m}_1$ and $\mathbf{m}_2$ are two vectors describing the centroids of the two classes. Thus, the discriminant subspace is determined by a 1-D vector, such that

$$\mathbf{d}_1 = \mathbf{m}_1 - \mathbf{m}_2 \tag{2.4}$$

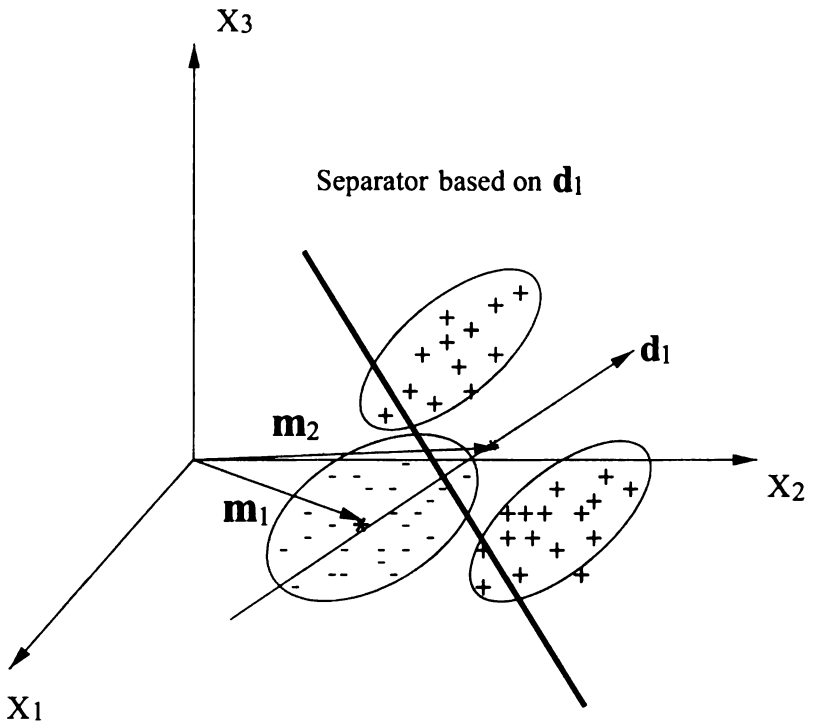


Figure 2.4. A one-dimensional discriminant subspace is formed for two classes. Figure originally created by Xiao Huang in Ji et al. 2008 [40]. Adapted here.

Obviously, $\mathbf{d}_1$ is not enough to partition these two classes.

In order to tackle these problems, we proposed a new node self-organization and spawning strategy in a leaf node. The key idea is that the number of feature detectors is not limited by the number of y-clusters.

Suppose the maximum number of x-clusters $q > 2$ (say $q = 3$) in the illustrated example, we may allocate prototypes of a single class to multiple x-clusters. Let N_t be the total number of prototypes in the leaf node and N_i $(i = 1, 2, ..., c)$ be the number of prototypes for each class, where c is the number of classes. For the ith class, the algorithm will generate $q_i{}'$ x-clusters in the new balanced node.

$$q_i^{'} = q\frac{N_i}{N_t} \tag{2.5}$$

In other words, the number of x-cluster assigned to a class is proportional to its rate of prototypes. In Eq. (2.5), the $q_i^{'}$ is a real fractional number, which can be converted to an integer:

$$\overline{q}_i^{'} = \lfloor q_i^{'} \rfloor \tag{2.6}$$

where $\lfloor\ \rfloor$ is a floor function and $i = 1, 2, ..., c$. In order to guarantees at least one cluster is generated for each old class presented, the balanced number of clusters $q_i^{(\mathrm{b})}$ is defined as follows

$$q_i^{(\mathrm{b})} = \begin{cases} \max\{1,\ \overline{q}_i^{'}\} & \text{if } q_i^{'} \neq 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

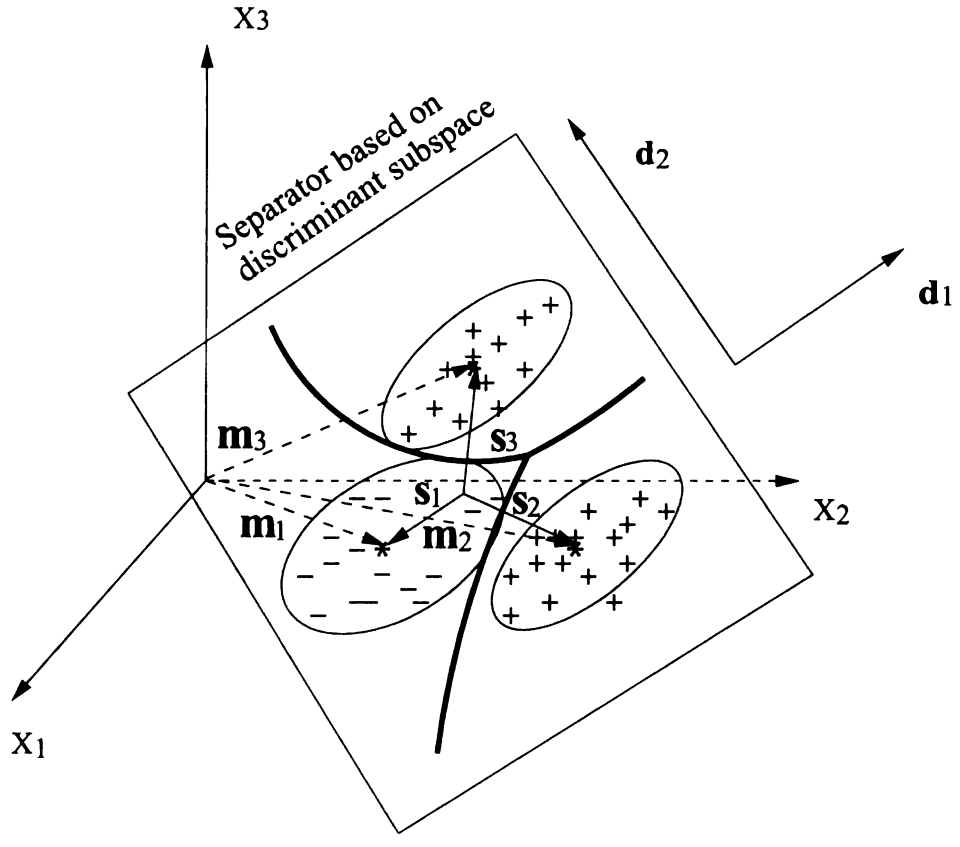


Figure 2.5. Locally balanced node. Two clusters are generated for class 1, such that a more discriminant subspace is formed. Figure originally created by Xiao Huang in Ji et al. 2008 [40]. Adapted here.

Fig. 2.5 shows the balanced node with $q_i^{(b)} = 3$ x-clusters, rather than the original 2 x-clusters. We define the scatter vectors:

$$\mathbf{s}_i = \mathbf{m}_i - \bar{\mathbf{m}} \tag{2.8}$$

where $i = 1, 2, 3$; $\bar{\mathbf{m}}$ is the mean of all centroid vectors. Let S be the set that contains these scatter vectors: $S = \{\mathbf{s}_i | i = 1, 2, 3\}$. The discriminant space is a 2-D space spanned by S, consisting of all possible linear combinations from vectors in S. Obviously, more discriminating features, i.e., $\mathbf{d}_1$ and $\mathbf{d}_2$, are derived from the balanced node. We should note that the algorithm only balances the prototypes in each node, while the entire tree may not be strictly balanced. So this is a local balancing method.

After the node is balanced, the statistics of the new structure must be generated. However, in real-time driving operation, the updating frequency is around 10Hz or higher. If the robot calculates the statistical information in one time slot, it will take a significant amount of time and may pause on the road, which is not allowed. In order to smooth the updating time for each new sample, the algorithm keeps two sets of statistics. While the old statistic is used to build the tree, the new balanced statistics is updated incrementally after the old statistics is half-mature (the number of prototypes in the node hits half of the limitation). When the new statistics is mature, the old one is thrown away and the prototypes are redistributed and updated based on the new structure. This smoothing strategy works very well for real-time navigation.

2.4.3 LBIHDR Learning Procedures

Using the design principles above, we can give the detailed algorithm of LBIHDR, containing three procedures: update-tree, update-node, and balance-node.

Update-tree: Given the root of the tree and sample $(\mathbf{x}, y)$, update the tree using $(\mathbf{x}, y)$.

1. From the root of the tree, update the node by calling **Update-node** and get active clusters.
2. For every active cluster received, check if it points to a child node. If it does, explore the child node by calling **Update-node**.
3. Do the above steps until all leaf nodes are reached.
4. Each leaf node keeps a set of primitive prototypes $(\hat{\mathbf{x}}_i, \hat{y}_i)$. If y is not given, the output is $\hat{y}_i$ where $\hat{\mathbf{x}}_i$ is the nearest neighbor among these prototypes.
5. If y is given, do the following: If $||\mathbf{x} - \hat{\mathbf{x}}_i||$ is less than certain threshold, $(\mathbf{x}, y)$ updates $(\hat{\mathbf{x}}_i, \hat{y}_i)$ only. Otherwise, $(\mathbf{x}, y)$ is a new sample to keep in the leaf node.
6. If the leaf node is half-mature, call **Balance-node**.
7. If the leaf node is mature, i.e., the number of prototypes hits the limitation required for estimating statistics in the new children, the leaf node spawns q children and is frozen. The prototypes are redistributed into the new leaf nodes.

Update-node: Given a node and sample $(\mathbf{x}, y)$, update the node using $(\mathbf{x}, y)$ incrementally.

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

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

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

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

Balance-node: Compare the maximum number of x-clusters (q) and the number of classes in the leaf node (c)

1. If $q \leq c$, still do y-space clustering.

2. Otherwise, ($q > c$), sort x-clusters according to the number of prototypes in increasing order.

3. Calculate the number of clusters for each old class by using Eqs. (2.6) and (2.7).

4. Conduct K-mean algorithm [22] to generate new clusters. Reallocate prototypes to each new cluster.

5. Calculate new statistics for the balanced leaf node.

2.5 Motivational System

A motivational system of the developmental architecture signals the internal states from cognitive mapping, modulates the mapping from internal states to internal action outputs, and evaluates candidate actions.

2.5.1 Q-Learning

Q-learning (Watkins and Dayan 1992 [87]) is one of the most popular reinforcement learning algorithms. The basic idea is as follows: For each state $s(t)$ at time t, keep a Q-value ($Q(s(t), a_i(t))$) for every possible action $a_i(t)$ (here, $i = 1, 2, ..., 5$). The action with the largest Q-value a^* will be selected as output and then a delayed reward $r(t+1)$ will be received. We implemented a modified Q-learning algorithm:

$$\begin{aligned} Q(s(t), a^*(t)) \longleftarrow & (1-\alpha)Q(s(t), a^*(t)) \\ & + \alpha(r(t+1) + \gamma Q(s(t+1), a^*(t+1))) \end{aligned} \tag{2.9}$$

where α and γ are the step-size parameter and the discount-rate parameter, respectively. With this algorithm, Q-values are updated according to the immediate reward $r(t+1)$ and the next Q-value. Thus, a delayed reward can be back-propagated in

time during learning. The formulation of α guarantees that it has a large value at the beginning and converges to a constant smaller value through the robot's experience.

2.5.2 Boltzmann Softmax Exploration

In order to fully explore the action space, Boltzmann Softmax exploration (Sutton et al. 1987 [77]) is implemented. At each state s in our case, the robot has a list of actions $A(s) = \{a_1, a_2, a_3, a_4, a_5\}$ to choose from. The probability for action a_i to be chosen is:

$$p(s, a_i) = \frac{e^{\frac{Q(s,a_i)}{\tau}}}{\Sigma_{a_i \in A(s)} e^{\frac{Q(s,a_i)}{\tau}}} \tag{2.10}$$

where τ is a positive parameter called temperature. With a high temperature, all actions in $A(s)$ have almost the same probability to be chosen for exploration. When $\tau \rightarrow 0$, the Boltzmann Softmax exploration most likely chooses action a^* that has a high Q-value. Now the question is how to determine the value of τ. If we choose a large constant τ, then the robot would keep exploring even it visits one state for several times. If we choose a small τ, the robot would face the local minimal problem and cannot explore enough states. Fortunately a Guassian density model (Eq. (2.11)) for local temperature solves the dilemma.

$$\tau(t) = \frac{c_1}{(2\pi)^{1/2}\sigma} \exp[-\frac{1}{2}(\frac{t-\mu}{\sigma})^2] + c_2 \tag{2.11}$$

where c_1 is a constant to control the maximal value of temperature, c_2 controls the minimal value, t is the age of the state, μ and σ are the mean value and standard

deviation of the Guassian density model, respectively. With this model, τ starts as a small value, then climbs to a large value, and finally converges to a small value.

2.5.3 Prototype Updating Queue

The state space would continue to grow as the robot explores in new environment. In order to reduce the training time, we adopt the k-Nearest Neighbor strategy. For each state, its top-k nearest neighbors are saved in the prototype updating queue (PUQ) (see Fig. 2.2). If a reward is issued, the system not only updates the current state-action pair but also updates the k-NN state-pairs, which dramatically reduces time complexity of the training procedure. Suppose the state space is S and the current state is $s(t)$. Let us define D as the distance between $s(t)$ and the kth nearest neighbor of $s(t)$. The volume of the set of all states whose distance from $s(t)$ is less than D is defined as follows:

$$A(k, S, s(t)) = \frac{2D^n \pi^{n/2}}{n\Gamma(n/2)}. \tag{2.12}$$

Using the k-NN updating rule, the updated space goes from one point $s(t)$ to the described A in Eq. 2.12, which is a tremendous improvement.

We can also look into the training complexity issue. Suppose for each state we need to issue m rewards to get a reliable training result (Q-value converges) and the number of states is N, then we have the time complexity for training, which is $T_{1-NN} = mN$. With top-k updating, we assume each state has the same possibility to be chosen as a

match. The time complexity is $T_{k-NN} = N + \frac{(m-1)N}{K}$, where the first N means that each state has to be visited at least once while $\frac{(m-1)N}{K}$ is the number of training sessions needed using top-k updating. The ratio of time complexity using top-1 updating and top-k updating is

$$\frac{T_{k-NN}}{T_{1-NN}} = \frac{N + \frac{(m-1)N}{K}}{mN} = \frac{k+m-1}{mk} \tag{2.13}$$

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

2.6 Learning Algorithms

The step-by-step computation of the proposed learning architecture (see Fig.2.2) is presented as follows.

1. Capture synchronized left and right images as a new sensory input.

2. Extract 6 attentional window images, each fed into the corresponding first-level cognitive mapping LBIHDR i ($i = 1, 2, ..., 6$).

3. Query each LBIHDR for a matched state $s(t)$, associated with list of internal actions and associated Q-values.

4. Denote the $\mathbf{x}$ part of internal state $s(t)$ as $\mathbf{x_s}(t)$. If $\|\mathbf{x}(t) - \mathbf{x_s}(t)\| < \varepsilon$, use $\mathbf{x}(t)$ to update $s(t)$ through incremental averaging. Otherwise, update the LBIHDR by saving $\mathbf{x}(t)$ directly. ε is considered as a threshold here.

5. Based on the retrieved internal action, give a reward.

6. Update the Q-value of states in PUQ using Q-learning and k-NN updating rule.

7. Based on the covert behaviors $\mathbf{x}'(t)$ generated from 6 window images, query the second-level cognitive mapping and get a matched state $s'(t)$.

8. If $\|\mathbf{x}'(t) - \mathbf{x_s}'(t)\| < \varepsilon$, use $\mathbf{x}'(t)$ to update $\mathbf{x_s}'(t)$ through incremental averaging. Otherwise, update the LBIHDR 7 by saving $\mathbf{x}'(t)$.

9. Execute the y' part (action part) of internal state $s'(t)$.

10. If the retrieval action is not correct, impose the expected action to $s'(t)$ directly.

11. Go to step 1.

2.7 Experiments

The proposed navigation system is evaluated on the autonomous driving vehicle "Crosser", the major components of which are shown in Fig. 2.6. We drove the vehicle to capture 4000 road images using the mounted two cameras, with a time step of 0.25 second and a maximum speed of 10 miles per hour (mph), including both on-road and off-road environment. Five presentative examples are presented in Fig. 2.7.

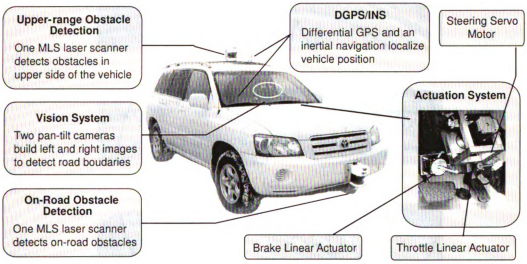


Figure 2.6. The autonomous driving vehicle "Crosser", outfitted by Embodied Intelligence Laboratory at Michigan State University. The vision-based navigation presented in this chapter relies on the local sensation by two pan-tilt cameras, global localization by GPS receivers and maps, and part of actuation system for steering motors. Developed covert and overt behaviors respectively generated capabilities of boundary attention and heading direction controls. The obstacle detection by later scanners and detailed actuation system of the brake and throttle are not discussed in scope of the work.

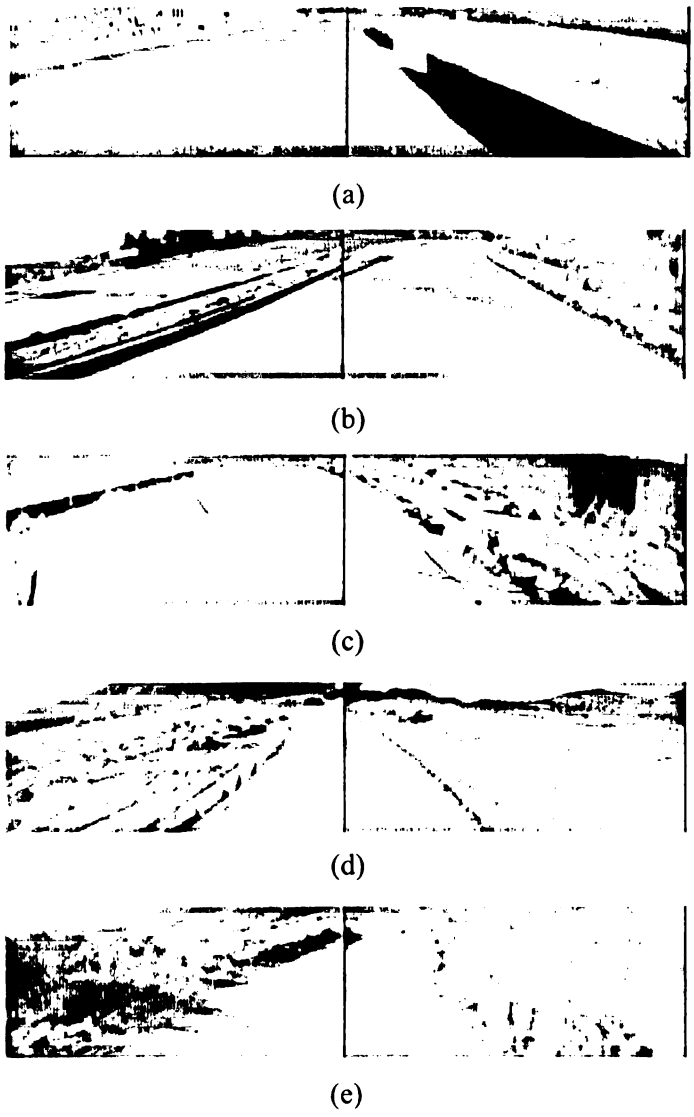


Figure 2.7. Examples of road images captured by "Crosser".

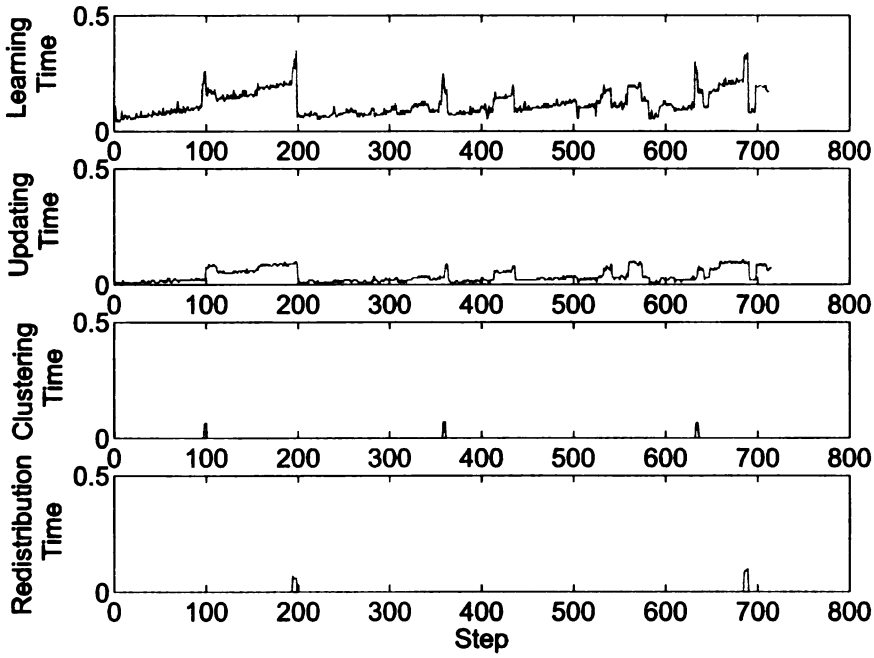


Figure 2.8. The timing recording for the learning of LBIHDR in the covert module.

2.7.1 Learning Efficiency of LBIHDR

In order to show the learning efficiency of the cognitive mapping engine LBIHDR, we recorded the updating time of one LBIHDR tree in the covert module, with high-dimensional inputs of window images. The learning of LBIHDR in the overt module should be much faster due to its low-dimensional inputs. The time profile of 715 samples are reported in Fig. 2.8, where the first row shows the total updating time of each sample and the second row shows the part of updating time without doing local clustering and redistribution. Time records of calculating new statistics (local clustering) and redistribution are shown in third and fourth rows, respectively. When the node is half mature (i.e, the node already has 100 prototypes), the new statistics

are computed through multiple steps; when the node is mature (i.e, the node has 200 prototypes), prototypes are redistributed. The learning time reaches peaks at these moments (see Fig. 2.8). It demonstrates that the time smoothing algorithm (discussed in Sec. 2.4.2) and coarse-to-fine tree structure enable the LBIHDR to learn at 3-4Hz, so that the learning of cognitive mapping is highly efficient.

2.7.2 Q-learning of Covert Behaviors

A motivational system with Q-learning delivers the covert behaviors (internal actions) based on the internal states retrieved from the cognitive mapping. We present the teaching result of internal actions, i.e., road boundary types, for one window image. Given 5 possible internal actions ($a = \{a_i | i = 1...5\}$), we assume that a_2 is the correct action. Fig. 2.9 plots the Q-value of each internal action from visit No. 1 to visit No. 30. After training with reward and punishment signals, the Q-value of each action converges to the expected value. From the figure we can see the Q-value of action 2 (a_2) is the largest while the Q-values of other actions converge to negative values. The learning process takes approximately 17 visits to converge.

In the next experiment, we train 50 consecutive images (which is slowly changed) repeatedly for 20 times. The strategy of top-5 nearest neighbors updating is used here. We should notice that by using the top-k updating mechanism, the number of updates (n_u) is different from the number of visits (n_v). n_v means the number of times that

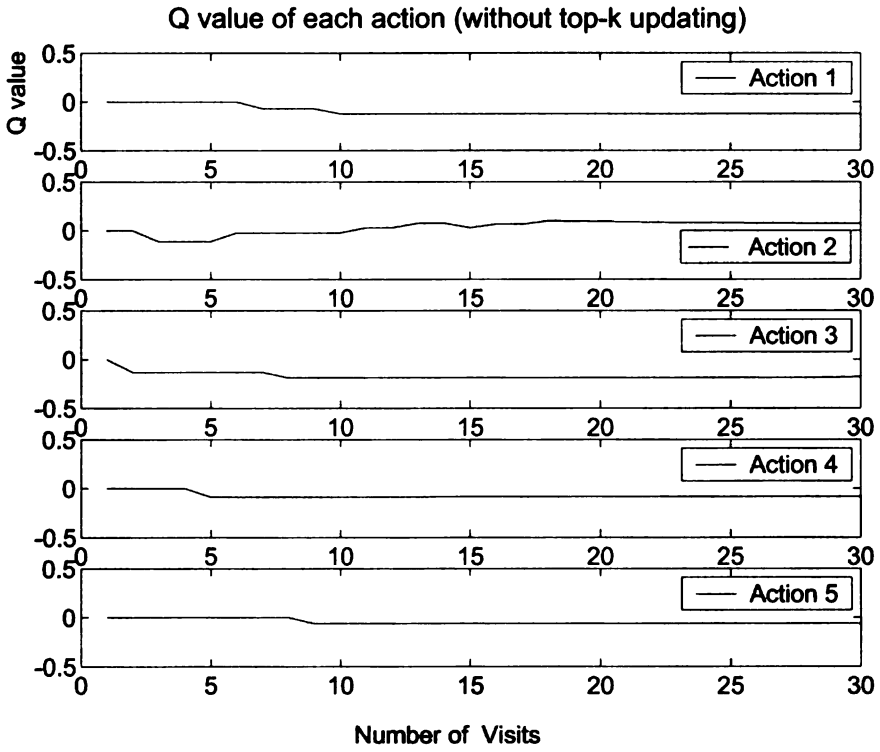


Figure 2.9. Q-value of each internal action over 30 visits for the same image (top-1 Q-learning). Figure plotted by Xiao Huang, from Ji et al. 2008 [40].

state $s(t)$ is retrieved as a top-1 match while n_u means the number of times that state $s(t)$ is retrieved as one of the top-k matches.

The Q-values of each internal action versus the number of updates are shown in Fig. 2.10. Here "+" denotes an action that is associated with the top-1 matched state. Action 2 should be the correct action and presents the largest value after some updates. Because of the trainer's mistake at the beginning, action 1 got several positive rewards (Q-value increases) while action 2 got a couple of negative rewards (Q-value decreases). However, after 30 times of updating, action 2 has the largest value and the system automatically chooses it as the output. The plot thus verifies that k-NN based reinforcement learning is able to tolerate human mistakes.

The Boltzmann exploration is used to generate a probability for each internal action, based on its corresponding Q-value. Fig. 2.11(a) shows the probabilities of actions 1, 2, ..., 5, from bottom to top respectively. The "+" denotes the random value generated by a uniform distribution. If the random value is in one range, say, the bottom range, then action 1 would be taken. Only information from visit No. 1 to visit No. 12 is shown here. As we can see, at the beginning, each action has a similar probability (about 0.2). After training, action 2 is chosen most of the time. The sequence of internal actions is shown in Fig. 2.11 (b). When $n_v > 4$, the system only chose action 2. That is, each attended boundary point takes about 5 visits to converge. This again shows the advantage of k-NN based reinforcement learning.

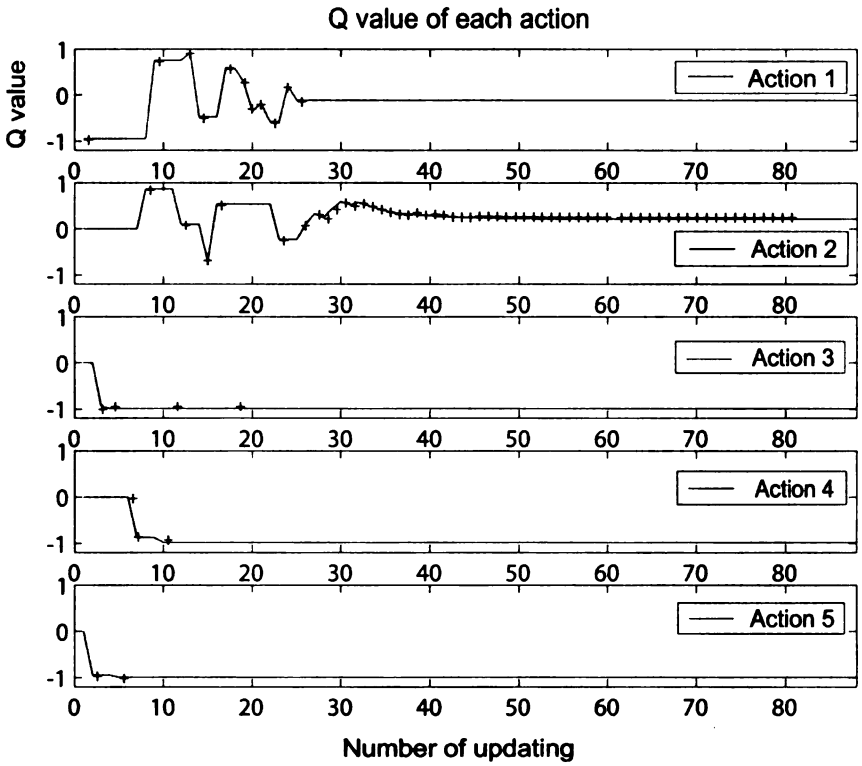


Figure 2.10. Q-value of each internal action with 80 updates for 50 consecutive images (k-NN Q-learning).

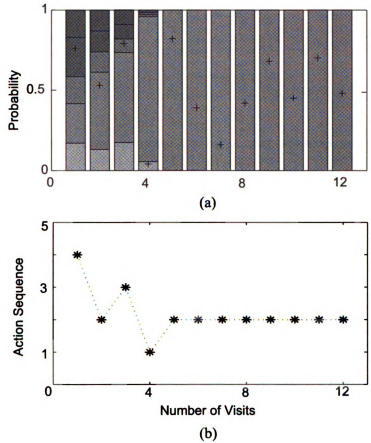


Figure 2.11. (a) The probability of each internal action determined by the Boltzmann Softmax Exploration. (b) The sequence of internal actions. Figure plotted by Xiao Huang, from Ji et al. 2008 [40].

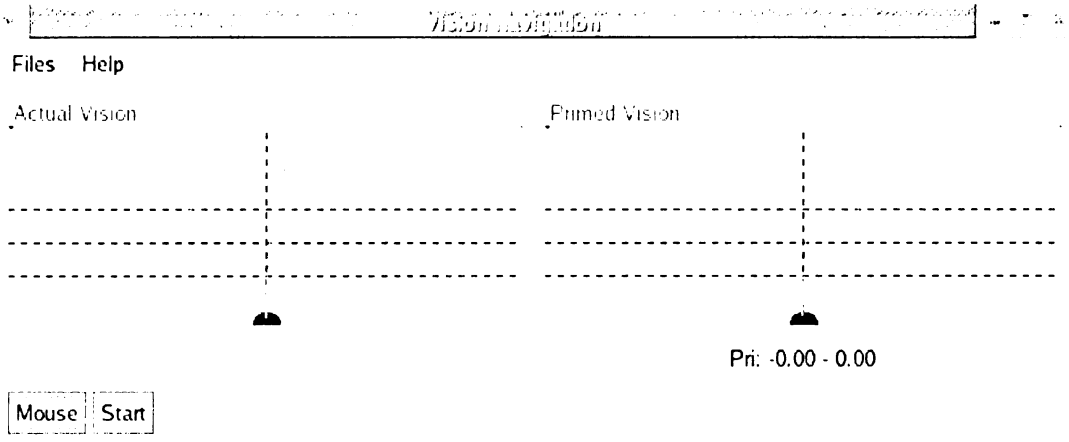


Figure 2.12. The interface for the supervised learning of overt behaviors, i.e., heading directions.

2.7.3 Supervised Learning of Overt Behaviors

Fig. 2.12 shows the interface for the development of overt behaviors. The six edge points provided by covert behaviors are projected onto the actual vision window, along with a desired path manually tagged by 10 global position coordinates (connected by a drawn curve in Fig. 2.12). The supervised learning of heading directions are thus dependant on the joint aspects of local boundary perceptions and global coordinates.

The primitive vision window shows the nearest neighbor in a LBIHDR state, associated with a heading direction displayed under the primitive window. An imposed action is provided by the mouse click to the expected heading direction in the primitive vision window. Because reinforcement learning entails an exploration of internal actions (before convergency) to "experience" all possible local boundary conditions adequately,

providing training situations in misaligned trajectories, the heading direction supervised in the training process, (especially for explored misaligned training conditions) is not identical to actual steering angles operated by the driving vehicle.

Let y_i and $\hat{y}_i$ denote, respectively, the true and estimated heading directions, and $e_i = y_i - \hat{y}_i$ denotes the error for the ith testing sample. We define the mean square error (MSE) as

$$\sigma_e = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{2.14}$$

where n is the number of testing samples. Compared with the existing cognitive mapping engine IHDR [89], the performance of LBIHDR is improved significantly by both re-substitution and disjoint test. As shown in Table. 2.1, the MSE of the heading direction remained at zero by LBIHDR in re-substitution test. For the disjoint test, we selected one sample in each 6 samples for testing. The other 5 samples were used for training. The MSE of heading direction is 0.683^o for LBIHDR and 1.165^o for IHDR. Thanks to the local balance of sample distributions in each node, the retrieval time of LBIHDR is also much less than IHDR. The result shows that LBIHDR performs very well for the non-stationary data, i.e., visual images in a navigation task. It further verifies that the robot is able to drive without a big deviation from the correct path as we expected.

Table 2.1. Mean Square Error of Heading Direction

Method	Resubstitution test	Disjoint test	Testing time (ms)
IHDR	0.252^o	1.165^o	22.4
LBIHDR	0^o	0.683^o	12.8

2.8 Summary

This chapter demonstrated a developmental learning architecture to integrate supervised and reinforcement learning modes for vision-based navigation, where the human teacher took advantage of the two learning modes to teach covert and overt behaviors jointly. Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) tree demonstrated its efficiency in both generation and retrieval of autonomous internal representation for covert and overt learning. A motivational system through reinforcement learning strategy explored the covert attention perception of road boundaries, to facilitate navigation behaviors. Such a general framework performed well in both off-road and on-road navigation together with obstacle detection and avoidance, without a need to any environment-specific constraints.

Since the cognitive mapping LBIHDR stores all the prototypes, it leads to a problem of using too much memory and does not represent information efficiently and selectively (supervised self-organization). In following chapters, rather than use the hierarchical tree, we will discuss a more efficient internal representation using networked neurons. The networked neurons are able to develop and integrate sophisticated mental functions through extensive embodied living experiences — all based on simple computations provided by developmental neurons.

CHAPTER 3

Networked Neurons and Representation

In the perspective of AMD, the human brain is not just an information processor, and more importantly, a processor developer. The ways that the human brain develops functionally are critical to understanding how the mind works, what natural intelligence really is, and what machine intelligence can be. The major blocks of the brain are realized by basic elements — the neurons. Patterns of responses of the networked neurons provide internal representation in the brain areas. In this chapter, we discuss the properties of networked neurons and their representations, in order to provide a deep understanding of mental development by higher animals, and in turn, to entail the brain-inspired computational models for autonomous mental development.

3.1 Neuronal Models

Neurons transmit signals primarily through spikes. The number of spikes per unit time is called firing rate. The higher the firing rate, the stronger the signal. Synapses that link different neurons fall into two categories, excitatory and inhibitory. *Excitatory synapses* produce depolarizing synaptic potential and *inhibitory synapses* produce hyperpolarizing synaptic potential.

The firing rate model allows a moderate temporal sampling rate of about 15 Hz to 30 Hz. This moderate rate is reachable for many high-dimensional large memory applications using current computing technology. Suppose that the sampling rate is f, the time period between two consecutive sampling time is $\Delta t = 1/f$.

The output firing rate of a neuron is directly related to the depolarizing potential minus the hyperpolarizing potential resulting from the excitatory and inhibitory synaptic connections from neighboring neurons. This phenomenon can be expressed in simple mathematics. Suppose that a neuron has n inputs x_i from neighboring neurons, represented by $x = (x_1, x_2, ..., x_n)$, where x_i indicates the firing rate of the corresponding neuron, $i = 1, 2, ..., n$. The synaptic conductances (weights) of the neuron are represented by a vector $w = (w_1, w_2, ..., w_n)$. If the synaptic connection from x_i is excitatory, we use a positive weight w_i whose absolute value is the synaptic strength. If the connection x_i is inhibitory, we use a negative weight w_i. The output firing frequency,

which is considered as the value output y from a neuron, can be modeled as:

$$y = g(x \cdot w - c) = g(\sum_{i=1}^{n} x_i w_i - c) \tag{3.1}$$

where $\cdot$ between two vectors notes the inner product, and c is an offset value representing that the neuron does not fire until the weighted summation reaches this threshold, and g is a nonlinear nondecreasing function.

In summary, the computation by a neuron in Eq. (3.1) is an inner product of the two vectors $\mathbf{x}$ and $\mathbf{w}$, shifted by value c, followed by a function g, as illustrated in Fig. 3.1.

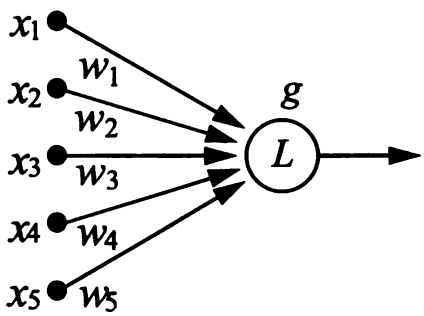


Figure 3.1. Basic operation of a neuron. Figure adapted from Weng and Zhang 2006 [95].

3.2 Competition Among Neurons

Lateral inhibition is a mechanism of competition among networked neurons. The output of neuron i is used to inhibit the output of neuron j which shares the receptive field, totally or partially, with i. Fig. 3.2 illustrates how neighboring neurons share the

same vector of inputs $\mathbf{x}$ and how they are connected through lateral inhibition from the output of neighboring neurons.

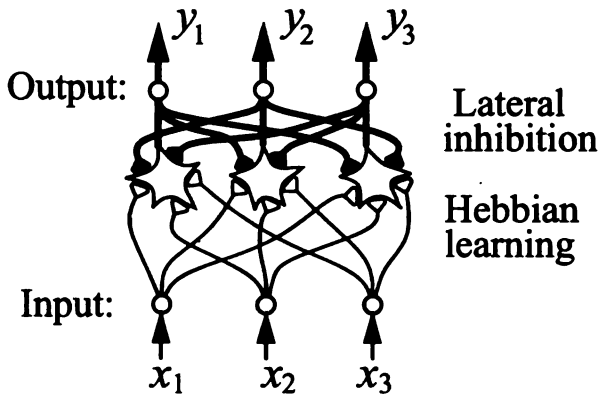


Figure 3.2. Neurons with lateral inhibitions. Hollow triangles indicate excitatory connections, and the solid triangles indicate inhibitory connections. Figure adapted from Weng and Zhang 2006 [95].

According to the above model, the input to the i-th neuron in a layer consists of two parts, the excitatory part $\mathbf{x}$ and the inhibitory part $\mathbf{z}$:

$$z_i = g(\mathbf{w} \cdot \mathbf{x} - \mathbf{h} \cdot \mathbf{z}) \tag{3.2}$$

where $\mathbf{w}$ consists of nonnegative weights for excitatory input $\mathbf{x}$, while $\mathbf{h}$ consists of nonnegative weights for nonnegative inhibitory input $\mathbf{z}$ but its i-th component is zero so that the right side does not require z_i. For biological plausibility, we assume that all the components in $\mathbf{x}$ and $\mathbf{z}$ are nonnegative. The source of $\mathbf{z}$ is the response of neighboring neurons in the same layer.

The nature of inhibition is indicated by the minus sign before $\mathbf{h} \cdot \mathbf{z}$ in Eq. (3.2). In other words, the higher the response from neighboring neurons, the weaker the response

is from this neuron. Vice versa, the higher the response from this neuron, the more effectively this neuron inhibits other neighboring neurons. Given an input vector $\mathbf{x}_t$, such a process of lateral inhibition reaches an equilibrium quickly.

Therefore, the net effect of lateral inhibition is the sparsity of cortical responses from each given input $\mathbf{x}$. If the response of neuron i is the strongest in the neighborhood, it most effectively inhibits its neighboring neurons, which further weakens the inhibition from its neighbors. It is known as the winner-take-all principle — only one neuron is assigned to a region of the input space for the purpose of updating. This is called the one-winner updating principle. This principle of assigning only one neuron to a region is a detailed incarnation of the sparse coding principle. The one-winner principle can be easily extended to the n-winner principle where top $n > 1$ winners update their weights. The winner-take-all mechanism is a computer simulation of the lateral inhibition mechanism in the biological neural networks (see, e.g., Kandel et al. [47]).

3.3 Development of Neural Representations

To enable the networked neurons to develop their neural representations, more specifically, synaptic weights, a biologically inspired learning principle — *Hebb's rule* was proposed by Canadian psychologist Donald Hebb [30, 48].

3.3.1 Hebb's Rule

The basic principle of the Hebbian rule is expressed as: Synapses are strengthened if there is a temporal correlation between their presynaptic and postsynaptic patterns of activity, and weaken otherwise. In a concise phrase, "neurons that fire together will be wired together."

Given the response y of a neuron to its inputs $\mathbf{x}$, Hebb's rule defines the change in presynaptic weights $\mathbf{w}$ to be:

$$\Delta \mathbf{w} = \mathbf{w}(t+1) - \mathbf{w}(t) = \alpha\ \mathbf{x}(t)\ y(t) \tag{3.3}$$

where α is a learning rate which can also change with time t, and

$$y(t) = \mathbf{w}(t) \cdot \mathbf{x}(t) \tag{3.4}$$

such that,

$$\Delta \mathbf{w} = \alpha\ \mathbf{x}(t)\mathbf{x}^T(t)\mathbf{w}(t) \tag{3.5}$$

However, the Hebb's rule has a severe problem: the connections are growing all the time, finally leading to very large values. In that sense, the weight $\mathbf{w(t)}$ will never be able to converge.

3.3.2 Oja's Rule & PCA

In order to constrain the growth of $\mathbf{w(t)}$, the standard Hebb's rule is modified by Oja [60], termed Oja's learinng rule:

$$\mathbf{w}(t+1) = \frac{\mathbf{w}(t) + \alpha \mathbf{x}(t)\ y(t)}{\|\mathbf{w}(t) + \alpha \mathbf{x}(t)\ y(t)\|} \tag{3.6}$$

We expand the Eq. (3.6) into a Taylor series:

$$\mathbf{w}(t+1) \ = \ \frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} + \alpha \ \{\frac{y(t)\mathbf{x}(t)}{\|\mathbf{w}(t)\|} - \frac{y(t)\mathbf{w}(t)(\mathbf{w}(t)\cdot\mathbf{x}(t))}{\|\mathbf{w}(t)\|})\} + O(\alpha^2) \tag{3.7}$$

such that the higher order terms ($O(\alpha^2)$) go to zero for a small learning rate. Then, the Oja's rule can be written as

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha y(t)\ (\mathbf{x}(t) - y(t)\mathbf{w}(t)) \tag{3.8}$$

The squared output guarantees that the larger the output of the neuron becomes, the stronger is this balancing effect.

Combining the Eq. (3.4) and Eq. (3.8), we have

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha \mathbf{x}(t)\cdot\mathbf{w}(t)[\mathbf{x}(t) - \mathbf{x}(t)\cdot\mathbf{w}(t)\mathbf{w}(t)] \tag{3.9}$$

Assuming that $\mathbf{x}(t)$ and $\mathbf{w}(t-1)$ are statically independent, Eq. (3.9) can be writhen as:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha[\mathbf{C}\mathbf{w}(t) - \mathbf{w}^T(t)\mathbf{C}\mathbf{w}(t)\mathbf{w}(t)] \tag{3.10}$$

such that

$$\Delta\mathbf{w} = \alpha[\mathbf{C}\mathbf{w}(t) - \mathbf{w}^T(t)\mathbf{C}\mathbf{w}(t)\mathbf{w}(t)] \tag{3.11}$$

When $\Delta \mathbf{w} = 0$, we have

$$\mathbf{C}\mathbf{w}(t) = \mathbf{w}^T(t)\mathbf{C}\mathbf{w}(t)\mathbf{w}(t) \tag{3.12}$$

Considering that the quadratic form $\mathbf{w}^T(t)\mathbf{C}\mathbf{w}(t)$ is a scalar, this equation is the eigenvalue-eigenvector equation for the covariance matrix. This analysis shows that if the weights converge in the Oja's learning rule, i.e., the weight changes over all the samples are zero, the weight vector becomes one of the eigenvectors $\mathbf{e}$ of the input covariance matrix. When time or number of iterations approaches infinity, the weight vector $\mathbf{w}$ converges to the first principal component $\mathbf{e_1}$ [59]. For this reason, the simple neuron learning by the Oja's rule becomes a principal component analyzer (PCA).

3.3.3 Optimal Hebbian Learning — CCI PCA

Based on the concept of statistical efficiency of Oja's learning rule, the Candid Covariance-free Incremental Principal Component Analysis (CCI PCA) is proposed by Weng and his coworkers 2003 [96], written as a recursive form similar to Eq. (3.8):

$$\mathbf{w}(t+1) = \frac{(t-1)}{t}\mathbf{w}(t) + \frac{1}{t}\mathbf{x}(t)y(t) \tag{3.13}$$

where

$$y(t) = \mathbf{x}(t) \cdot \frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} \tag{3.14}$$

Similar to Oja's learning rule, an eigenvector of covariance matrix $\mathbf{C}$ can be achieved through the incremental way when $\mathbf{w(t+1)} = \mathbf{w}(t)$, such that $\Delta \mathbf{w} = 0$.

In contrast with Oja's learning rule presented in Eq. (3.6), the first term on the right

side of Eq. (3.13) is not normalized. In effect, $\mathbf{w}$ in Eq. (3.13) converges to $\lambda\mathbf{e}$ instead of $\mathbf{e}$ in Eq. (3.8), where λ is the eigenvalue and $\mathbf{e}$ is the eigenvector. In Eq. (3.13), 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. (3.13) to get sample mean), which allows fully use of every observation in terms of statistical efficiency. Note that the coefficient $(t-1)/t$ in Eq. (3.13) is as important as the "learning rate" $1/t$ in the second term to realize sample mean. Although $(t-1)/t$ is close to 1 when t 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 t 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 algorithm.

Amnesic Average

There is a further improvement to procedure Eq. (3.13) proposed by Zhang and Weng [96]. In Eq. (3.13), as $y(t) = \mathbf{w}(t) \cdot \mathbf{x}(t)$, we consider all the "samples" and define

$$\mathbf{z}(t) = \mathbf{x}(t)\mathbf{x}^T(t)\frac{\mathbf{w}(t)}{||\mathbf{w}(t)||} \tag{3.15}$$

to be weighted equally. However, since $\mathbf{z}(t)$ is generated by $\mathbf{x}(t)$ and $\mathbf{x}(t)$ is far away from its real value at a early estimation stage, $\mathbf{z}(t)$ 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. (3.13) into,

$$\mathbf{w}(t+1) = \alpha_1 \mathbf{w}(t) + \alpha_2 y(t)\mathbf{x}(t) \tag{3.16}$$

$t = 2, 3, ...,$ where

$$\alpha_1 = \frac{t - 2 - \mu(t)}{t - 1} \quad \text{and} \quad \alpha_2 = \frac{1 + \mu(t)}{t - 1} \tag{3.17}$$

$\mu(t)$ is called the amnesic parameter. We adopt a three-sectioned profile of $\mu(t)$:

$$\mu(t) = \begin{cases} 0 & \text{if } t \leq t_1, \\ c(t - t_1)/(t_2 - t_1) & \text{if } t_1 < t \leq t_2, \\ c + (t - t_2)/r & \text{if } t_2 < t, \end{cases} \tag{3.18}$$

in which, e.g., $c = 2, r = 10000$. With the presence of $\mu(t)$, larger weight is given to new "samples" and the effect of old "samples" will fade out gradually.

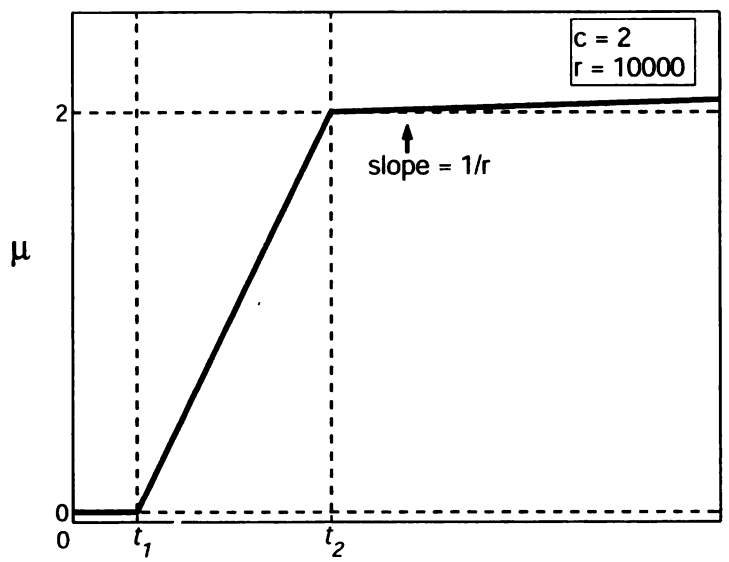


Figure 3.3. Three intervals of $\mu(t)$. Figure courtesy of Weng and Luciw 2006 [92].

The amnesic average also accords with the scheduling of neuronal plasticity which should adaptively change with on-going experience. The neuronal environment in the

brain is not stationary, so neurons must be able to update their representations throughout their lifetime. As seen above and in Fig. (3.3), $\mu(t)$ has three intervals. The first interval, when t is small and the neuron is "immature", computes incremental average. Increasing the learning rate α_2 during this period might be detrimental to estimation. Since learning rate is the smallest during this period, we wish to minimize the effect of any outliers that may occur here. Within the second interval, where $t_1 < t \leq t_2$, learning rate increases linearly from $1/t$ to c/t. The purpose of this interval is that once the representation has stabilized, the learning rate is increased for faster convergence. In the third interval where $t > t_2$, learning rate is prevented from converging to zero as $t \to \infty$ (and $t \to \infty$ since $t \geq t$, always); — instead α_2 converges to $1/r$. So, synapses can update no matter how mature a neuron is, as is important in open-ended development, where each t should become very large.

Higher-order Eigenvectors

The Eq. 3.16 only derived the first principal component, scaled by it eigenvalue λ. However, we know that each estimation $\mathbf{w}_i$ from $i = 2, ..., k$ must be orthogonal to all previous components. Take the derivation of $\mathbf{w}_2$ as an example. In order for $\mathbf{w}_2$ to be orthogonal to $\mathbf{w}_1$, first compute the projection vector of the sample onto $\mathbf{w}_1$. Then subtract that vector from the sample, effectively deriving a new sample in a residual space. In the residual space, every vector will be orthogonal to $\mathbf{w}_1$. Now, the *first* principal component in the residual space is equal to $\mathbf{w}_2$. For any number k of

principal components, $k-1$ residuals are computed at each step.

Denote $\mathbf{x}_i(t)$ as the sample used when updating vector $\mathbf{w}_i$. When $i=1$, $\mathbf{x}_1(t)=\mathbf{x}(t)$ as an original sample. When $i>1$, $\mathbf{x}_i$ is considered as a residual vector. We project $\mathbf{x}_i$ onto $\mathbf{w}_i$

$$\mathbf{y}_i(t)=\mathbf{x}_i(t)\cdot\frac{\mathbf{w}_i(t)}{\|\mathbf{w}_i(t)\|}. \tag{3.19}$$

and subtract the projection vector from $\mathbf{x}_i$ to obtain $\mathbf{w}_{i+1}$.

$$\mathbf{w}_{i+1}(t)=\mathbf{w}_i(t)-\mathbf{y}_i\frac{\mathbf{w}_i(t)}{\|\mathbf{w}_i(t)\|}. \tag{3.20}$$

The procedure 3.19 and 3.20 should iterate $k-1$ times per sample, in order to generate the k principal components.

3.4 Summary

In this chapter, we discussed the properties of networked neurons and their representations in the brain-mind perspective of AMD. Based on the pieces of knowledge in neural models, neural competitions, hebbian mechanisms to develop neural representations and so on, we will present a global cross-scale computational model in the next chapter, integrating pieces of information together, from neurons, to circuits, to cortex, to pathways. The proposed general sensorimotor pathway aims to address a series of challenges that face the autonomous mental development, such as general invariance, completeness of representations, etc..

CHAPTER 4

Developmental Object Learning by an In-place Sensorimotor Pathway

Motivated by neuronal models, lateral inhibition and hebbian learning mechanisms discussed in Chapter 3, we introduce a global, cross-level, sensorimotor pathway, with a design goal to develop a recurrent network, as a function of sensorimotor signals, for open-ended learning of multiple sensorimotor tasks. Rooted in the genomic equivalence principle, a biologically inspired in-place learning represents a new and deeper computational understanding for AMD, meaning that each neuron (cell) is fully responsible for its own development and adaptation while interacting with its cellular environment. Computationally, the biologically inspired in-place learning provides unusually efficient learning algorithms whose simplicity, low computational complexity, and generality are set apart from typical conventional learning algorithms. As a key requirement for such a network, we model bottom-up, lateral, and top-down connections (Felleman and Van Essen 1991 [25], Erisir et al. 1997 [24]) of each cortical area in the network, enabling unsupervised (via ascending connections), competitive (via lateral connections) and su-

pervised learning (via descending projections) to occur concurrently. The biologically motivated model is applied for a challenging application field of object learning in a complex, natural driving environment, with demonstration of hard-coded attention, sparse coding and motor abstraction.

The text of this chapter, in part, is adapted from Ji, Luciw and Weng [42][41]. Explicit credits will be given in sections.

4.1 Biological Visual Pathway

A sensorimotor pathway is a path through which nervous signals are processed in an orderly fashion, starting from a sensory input and ending at a motor output. Our studies are focused on the human's visual pathway as an example.

The visual information is first sensed by the retina which does preliminary signal processing, such as reduction of dimensions from 100 millions receptors in each retina to about 1 million optical nerves sending information to the lateral geniculate nucleus (LGN) which further processes the visual information and communicates (two ways) with the primary visual cortex (V1, also Brodmann area 17).

The primate retina contains the eye's receptor sheet. The output neurons from the retina are the ganglion cells. Most ganglion cells fall into two classes, P or M. Each cell responds to light directed to a specific area of the retina. This area is called the

receptive field of the cell. M cells have a large cell body and large receptive fields while P cells have a small cell body and small receptive fields. M cells send signals that are primarily concerned with motion and gross features. The signals from P cells contribute carry two types of information. One is primarily for detail and form and the other primarily for color. The primate retina also contains ganglion cells that do not fall into the P or M classes. The functions of these cells are largely unknown, but one type is found to carry information for overall ambient light intensity. Such an arrangement may be related to the history of evolution: from lower to higher animals, more visual information is added, from motion and rough forms to detailed shapes and colors. This also indicates the freedom of design choices for developmental robots, do we choose to have the visual information processed in a highly integrated way from the early stages of processing or do we want visual attributes to be extracted separately early and then processed separately? It is up to the experimentalists to find out the pros and cons.

In the primary sensory cortex (V1) – the entry area of the visual sensory modality in the cortex – the nearby neurons receive signals from nearby receptors in the retina pre-processed by LGN. In the other later cortical areas, this orderly fashion is preserved, although the sources of input are different.

4.2 Epigenetic Sensorimotor Pathway

We developed a sensorimotor pathway, as part of efforts toward a global cross-scale computational model for cerebral development at multiple scales, from cells, to cortex, to pathways. The proposed sensorimotor pathway uses multiple layers, each corresponding to a cortical area. Motivated by the studies of primary visual cortex (V1) [34], a range of analogical cells is presented in the first layer, developed from natural images using the same cell-centered algorithm used in the rest of the network. Each of these neurons is responsible for detecting specific features in a local receptive field on the sensory grid. Via the processing of this layer, the sensory input is transformed into a sparse representation. Later layers use the aforementioned in-place learning to self-organize effective and efficient representations for motor output.

The motivation of this work requires a combination of the following challenging properties that are required by development. No existing work we know of can meet all: (1) General internal representation by in-place learning, not constrained for a task-specific learning. (2) Early sparse coding. (3) Without a significant loss of memory problem. (4) High-dimensional information processing. (5) Incremental learning adaptive to the changing of environments. (6) Real-time speed due to low computation complexity. (7) Integration of supervised and unsupervised learning in any order suited for development. All the properties above, coupled with a nurturing and challenging environment, as experienced through sensors and effectors, allow autonomous mental capabilities and

skills to emerge.

The developmental pathway is applied to develop sensorimotor behaviors from sensory information of a video camera and radar system. The information from the video camera and the radar sensor are integrated through the in-place learning scheme for the classification of objects.

4.3 Architecture

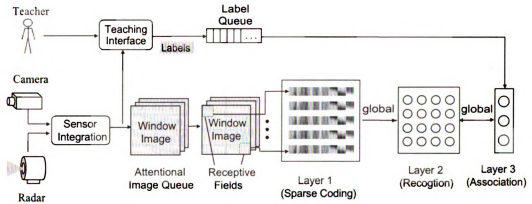


Figure 4.1. Sensorimotor pathway of the vehicle-based agent. Figure partially contributed by Matthew Luciw in Ji et al. [42]. Adapted here.

The architecture of the proposed pathway is shown in Fig. 4.1. The camera and radars work together to generate a set of attended window images, containing nearby objects. A teacher communicates with the system through an interface to train the class labels of objects. A 3-layer sensorimotor pathway provides the processing and learning of the extracted window images. Layer 1 encodes the local receptive fields of each image using orientation-selective filters. Neurons in layer 2 learns the sparse-coded

object representations, associated by layer 3 with teacher's output tokens.

4.4 Sensor Integration

Two external (outward looking) sensors are used in the sensorimotor pathway. The first is the radar modality, utilized to find salient regions (possible nearby objects) within the image. The second senses the vision modality. Information from this sensor (i.e. video camera) is used to develop the capability of object recognition. Table I & II specify the sensor parameters of radar and vision modalities, respectively.

Table 4.1. Sensor Specifications of Radar System

Key parameters	Specification
Refreshing rate	10 Hz
No. of targets	max. of 20 targets
Distance	$2 \sim 150\text{m} \pm \mathbf{max}(5\%, 1.0\text{m})$
Angle	$15^{o} \pm \mathbf{max}(0.3^{o}, \text{range of } 0.1\text{m})$
Speed	$\pm 56\text{m/s} \pm 0.75\text{m/s}$

Table 4.2. Sensor Specifications of Vision System

Key parameters	Specification
Refreshing rate	15 Hz
View of fields	45^{o}
Resolution	320×240

As shown in Fig. 4.2 (right), a group of target points in 3D world coordinates can be detected from the radar system, which includes one long-range radar and four short-range radars, with a detection range up to 150 meters. Each radar point is presented by a triangle, associated with a bar, the length and direction of which indicate the relative speed of an object. As a rudimentary but necessary attention selection mechanism,

we discarded radar returns more than 80 meters in distance ahead or more than eight meters to the right or left outside the vehicle path (see red triangles in Fig. 4.2 (right)).

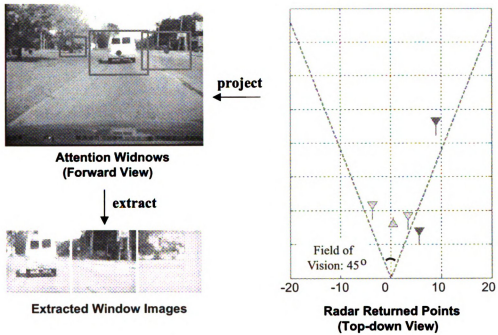


Figure 4.2. A projection of effective radar points (green) onto the image plane, where window images are extracted for further recognition.

The radar-centered coordinates are projected into the image reference system, using a perspective mapping transformation. The transformation is performed using the calibration data that contain the intrinsic and extrinsic parameters of each camera. Given a 3D radar-returned target point, an attention window is created within the image (see Fig. 4.2 (upper left)), based on the parameters of expected maximum height (3.0 meters) and expected maximum width (3.8 meters) of objects. Fig. 4.3 shows the projection from expected maximum size of an object to an window size in the image

plane, based on the longitudinal distance along axis Z_c.

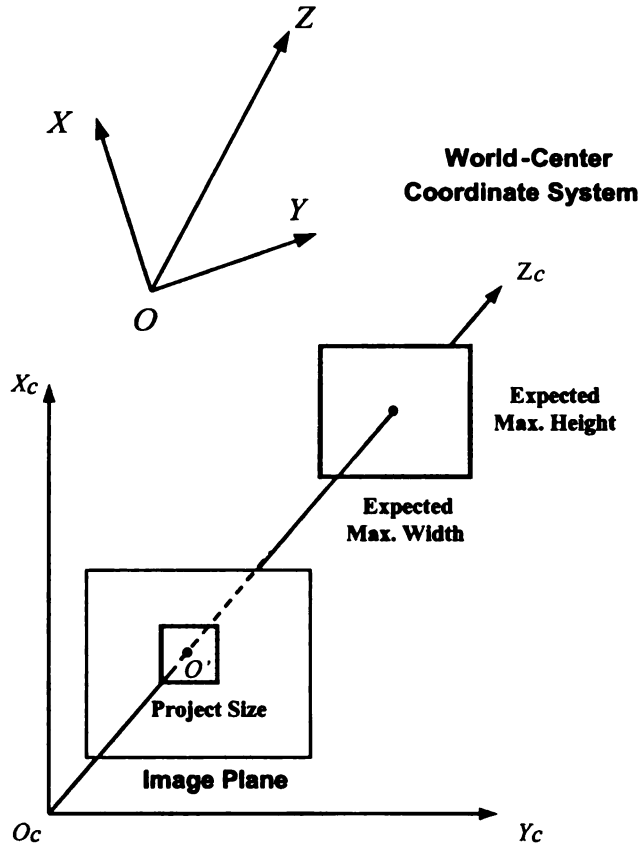


Figure 4.3. The projection from expected object size to the window size in the image plane.

Fig. 4.4 shows examples of the radar-based attention window generation, where most of the non-object pixels have been identified. For each radar window, the attended pixels are extracted as a single image. Each image is normalized in size, in this case to 56 rows and 56 columns as shown in Fig. 4.2 (bottom left). To avoid stretching small images, if the radar window could fit, it was placed in the upper left corner of the size-normalized image, and the other pixels are set to be uniform gray.

Figure 4.4. Examples of images containing radar returned points, which are used to generate attention windows. This figure also shows some examples of the different road environments in the tested dataset.

There may be more than one object in each radar window, but for the purpose of object identification, the radar window is assigned with only one label. The labeled radar windows create a set of selected areas while the rest of the image becomes ignored. This is called *hard-coded* attention selection — finding candidate areas purely based on physical characteristics (radar returns). The attended window images may still contain some information unrelated to the object, such as "leaked-in" background behind the object. However, our object learning scheme does not require the good segmentation of the object itself, but instead depends on the discriminant statical distributions of the scenes in each radar window. The proposed system can thereby learn to detect and recognize multiple objects within the image captured by the video camera, as long as a radar point is returned for each one.

4.5 Sparse Coding

How feasible is the human's visual pathway to represent objects using the raw pixel format in attention windows? Each object appearance could potentially vary quite a bit (the object invariance issue), and "leaked-in" background may pose amount of noises. Many researchers believe that a more abstract internal representation is available in the early cortex of the human visual system. A variety of experimental and theoretical studies indicate that the human visual system efficiently codes the retinal activation using a sparse, overcomplete coding scheme [62]. Theoretically, sparse coding leads to lower mutual information between coded representations than pixel appearances,

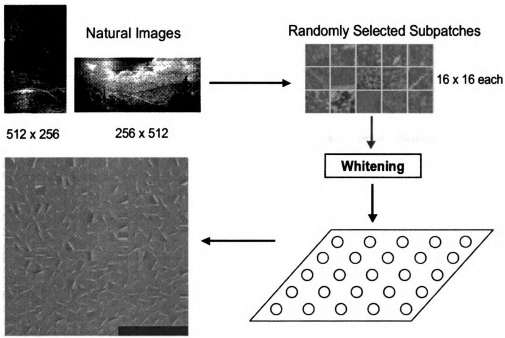


Figure 4.5. The generation of orientation selective filters. Figure partially contributed by Matthew Luciw, from Ji et al. [42].

meaning they will be more independent. This allows object learning and recall (associative learning) to become a compositional problem (i.e., an view of a novel object is decomposed as a composite of a unique set of sparse events). The sparse coding entails a mapping of raw representation from the pixel space to a coded space (in terms of a basis defined by our developed filters), wherein the coded representation of an object can be used for higher-level processing — learning object representations or recognition.

4.5.1 Developing Orientation Selective Filters

What kind of features should be used to code the raw pixel views? Studies in neuroscience (e.g., Srinivasan [58], Atick et al. [3] etc.) have shown that the orientation selectivity of the primary visual cortex (V1) plays an important role in predictive sparse coding on sensory information. Given a network that whitens the input, Weng and Zhang [95] showed how the two biological mechanisms of Hebbian learning and lateral inhibition led to the development of localized, orientation selective filters that form a sparse representation of the input, from a set of natural images. That same procedure was used to generate the orientation selective filters within this system's layer 1. Each individual neuron was developed from real-world natural images, given the receptive field 16×16.

To do so, 16×16 pixels were incrementally selected from random locations in 13 natural images [1] (see examples in Fig. 4.5 (upper left)). Let $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_n\}$ denote

[1] from http://www.cis.hut.fi/projects/ica/data/images/ via Helsinki University of Technology

the set of randomly selected image patches in columns. The whitening matrix $\mathbf{M}$ is generated by

$$\mathbf{M} = \mathbf{VD} \tag{4.1}$$

where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_k\}$ contains principal components of $\mathbf{P}$ in each column, and the matrix $\mathbf{D}$ is a diagonal matrix such that the matrix element at row and column i is $\frac{1}{\sqrt{\lambda_i}}$ and λ_i is the eigenvalue of v_i.

Then, we multiply the image patches by matrix $\mathbf{M}$, such that

$$\mathbf{F} = \mathbf{MP} = \mathbf{VDP} \tag{4.2}$$

where $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, ...\mathbf{f}_n\}$ contains a whitened input in each column. Due to the statistics of the non-white natural image distribution [62], the pre-whitening process is necessary to develop localized filters.

We develop a set of 512 neurons, represented by the column vectors $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{512}$. Given $\mathbf{f}_t$ $(t = 1, 2, ..., n)$, each neuron acts as a learning unit of CCI PCA (see Sec. 3.3.3), competitive with other neurons using winner-take-all principle, such that only the winner neuron

$$\arg \max_{1 \le i \le 512} \{\frac{\mathbf{f}_t \cdot \mathbf{w}_i(t)}{\|\mathbf{w}_i(t)\|}\} \tag{4.3}$$

is updated.

Fig. 4.5 shows the process to develop neuron weights (arranged in a grid) using $n = 1,500,000$ whitened input samples. Each neuron's weight vector is identical in

dimension to the input, thus it can be displayed at a location within the grid, as a 16 row and 16 column image. However, it must first be dewhitened in order to display properly. For example, to restore the original input vector,

$$\mathbf{p} = \mathbf{V}\mathbf{D}^{-1}\hat{\mathbf{p}} \tag{4.4}$$

is the dewhitening procedure.

In the developed filters in Fig. 4.5 (bottom left and arranged in a grid), the component with the highest age (number of updates) is at the top left of the image grid, and it progresses via descending age through each row until the one with the least updates, at the bottom right. We discarded some immature neurons out of the 512, and kept 431 neurons.

Each neuron's weight vector $\mathbf{w}_i$ clearly shows localized orientation patterns, thereby we called them orientation selective filters, i.e. feature detectors, which are functionally similar to those found in V1.

4.5.2 Sparse Representation

How is a radar-returned image coded using the filters developed in the previous subsection? In the human visual cortex, V1 neurons are organized in sets of densely populated columns. Receptive fields for neurons in each column are very similar and neighboring columns overlap significantly [6].

Similarly, we applied square, overlapping, 16 × 16 initial receptive fields over the entire 56 × 56 attention window image plane, where a stagger distance of 8 pixels is used in both horizontal and vertical directions. Given the window image with 56 by 56 dimensions, there were 36 local receptive fields, corresponding 36 neural columns in total. Fig. 4.6 shows the organization of the neural columns.

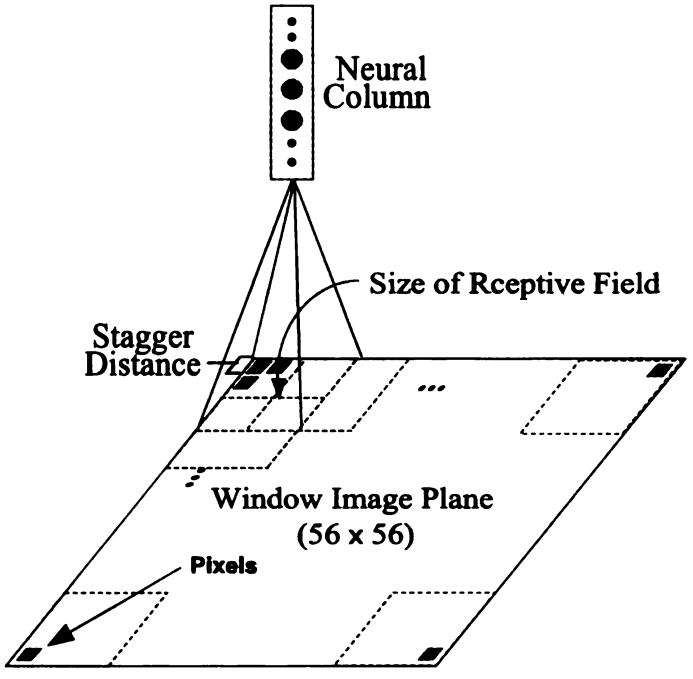


Figure 4.6. Receptive field boundaries and organizations of neural columns. There are 36 total neural columns, neurons in which have initial receptive fields that overlap with neurons in neighboring columns by the stagger distance both horizontally and vertically.

The set of developed orientation selective filters $\mathbf{w}_i$ $(i = 1, 2, ..., 431)$ are copied to 36 neural columns. Each column codes a receptive field $\mathbf{r}_j(t)$ $(j = 1, 2, ..., 36)$ for a sparse representation:

$$s_{i,j} = \frac{\mathbf{w}_i \cdot \mathbf{r}_j(t)}{||\mathbf{w}_i|| ||\mathbf{r}_j(t)||} \tag{4.5}$$

All the coded responses in each neural column are ranked in a descending order, where top-k responding neurons are selected to survival and all the other neurons are set to zero responses. Let $s_{k,j}$ be the k-th responded filter, we have

$$\begin{cases} s_{i,j}(t) = s_{i,j}(t), & \text{if } s_{i,j}(t) \geq s_{k,j}(t) \\ s_{i,j}(t) = 0, & \text{otherwise} \end{cases} \tag{4.6}$$

In our experiments, k is set to 91 for a sparse-coding of 431 filters, largely disregarding less relevant feature detectors. It maps the raw pixel space (56×56) to a sparse, over-complete space (431×36) given layer-1 neurons. The dimensionality is not reduced from the input space to the output space; rather, the redundancy of the input is transformed into the redundancy of the firing pattern of cells. By controlling the sparseness of firing cells, the sparse coding is able to reduce redundant, high-correlated information of inputs and form a representation in such a way that statistical dependency between elements of representation is reduced, while "key" information for later recognition is preserved [26][63].

To verify the functional role of sparse coding, we captured 800 attended window images from our driving sequences and presented them in object-by-object order. Each object possibly appears in several window images with sequential variations, e.g., different scales, illumination and view point variation etc. The correlation matrix of window images is shown in Fig. 4.7 (a), indicating the high statistical dependence among the samples, especially, across different objects. Each image is further coded for a sparse representation. The correlation matrix of generated sparse representations is shown in

Fig. 4.7 (b). It takes the advantage in two aspects: (1) object samples are de-correlated by the coding process, i.e., cross-object correlation is dramatically reduced; (2) object information is maintained, i.e., with-in-object samples keep the high correlation.

4.6 In-place Learning

The sparse representation is processed by the in-place learning via two layers, till the motor output, where each neuron in the output layer corresponds to one object class.

Although various networks have been used to model visual cortical processing and learning [75, 29, 28, 36, 73], there is a lack of computational accounts of cell-centered learning for multiple cortical areas. In-place learning fits to the celled-centered learning scheme, indicating that each neuron is fully responsible for its own development and does not need a dedicated extracellular learner. In-place learning can be compared to non in-place methods in the details of synaptic adaptation. With in-place learning, the input correlation matrix for pre-synaptic activities, commonly used for modeling synaptic adaptation (e.g., [17]) cannot be computed nor stored by any adapting neuron, since there has been no physiological evidence for cell to have so many independently accessible storage units for this type of information (about 0.5 million for an average neuron with 1000 synapses). Instead, the synapses of each neuron, represented as weight vectors, should be adapted incrementally with properly scheduled, experience-dependent plasticity. Each observation along the nonlinear search path of every neuron

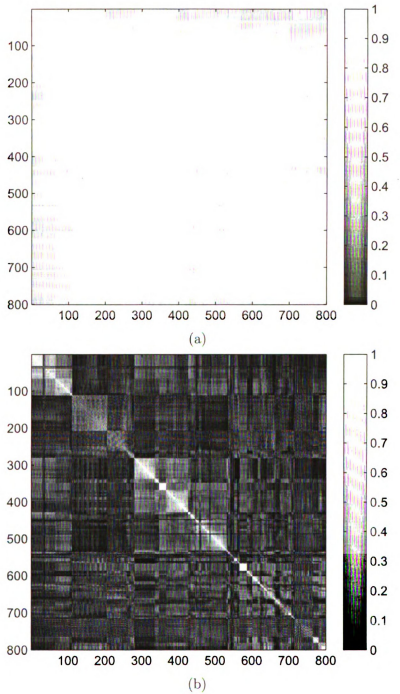


Figure 4.7. Correlation matrix of sampled 800 window images in (a) pixel space and (b) sparse representation space

is taken into account at each update, then discarded. The CCI plasticity rules [93] define the updating strength as a function of the cell's individual age. This method of synaptic adaptation represents a balance, both dealing with the local minima problem (free of local minima) and the loss of memory problem – a neuron must represent an area effectively (being able to withstand perturbations) while at the same time being able to adapt to future change (long term adaptation potential).

4.6.1 Three-way Computations

Fig. 4.8 shows the general structure of in-place learning with three consecutive layers. Each neuron at layer l has three types of weights, modeling the three types of input connections to a neuron:

- bottom-up (excitatory) weight vector $\mathbf{w_b}$ that links input lines from the previous layer $l-1$ to this neuron;
- lateral (excitatory or inhibitory) weight vector $\mathbf{w_h}$ that links other neurons in the same layer to this neuron.
- top-down (excitatory) weight $\mathbf{w_t}$ that links the output from the neurons in the next layer $l+1$ to this neuron.

Assume that this network computes at discrete times, $t = 0, 1, 2, ...$, as a series of open-ended developmental experiences after the birth at time $t = 0$. Every neuron at

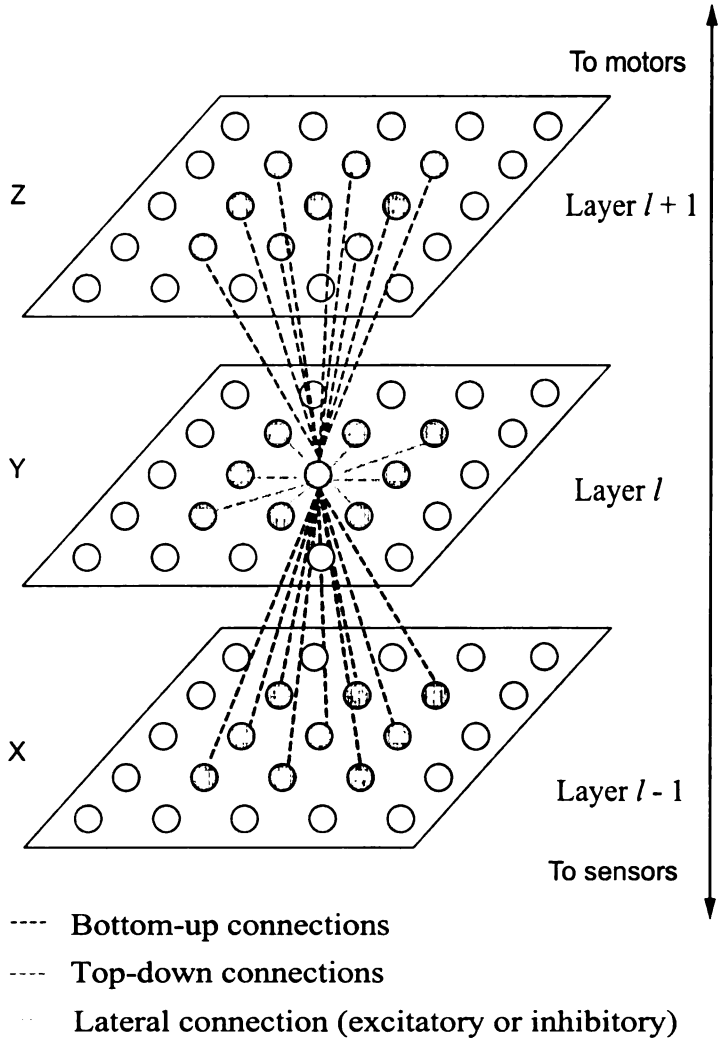


Figure 4.8. In-place learning. Neurons are placed (given a position) on different layers in an end-to-end hierarchy – from sensors to motors. A neuron has feedforward, horizontal, and feedback projections to it. Only the connections from and to a centered cell are shown, but all the other neurons in the feature layer have the same default connections.

the layer l is connected with bottom-up input $\mathbf{x}(t)$, lateral input $\mathbf{y}(t)$ and top-down input $\mathbf{z}(t)$ at time t to give the response $\mathbf{y}(t+1)$ at time $t+1$ and to update the layer to $L(t+1)$, via the layer function modeled as the in-place learning:

$$(\mathbf{y}(t+1), L(t+1)) = \text{In-place}(\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t) \mid L(t)) \tag{4.7}$$

This work does not use explicit lateral connections, but instead uses an approximate method in which the top-k winners (largest responses, i.e., the output of g_i formulated below) and their 3×3 neighborhoods update and fire. The suppressed neurons are considered laterally inhibited and the neighborhood neurons which can fire and update and considered laterally excited.

Here, supervised and unsupervised learning are both possible without explicitly changing the learning mode. The in-place learning in a hierarchical structure allows supervised learning to occur when the motor is imposed (e.g., an object is to be learned by labeling windows) and top-down and bottom-up connections together automatically allow this to occur. When the motor is not imposed, bottom-up connections alone allow unsupervised learning.

4.6.2 Local Density Estimator

Given the limited resource of c neurons, the in-place learning divides the bottom-up space X into c mutually non-overlapping regions, such that

$$X = R_1 \cup R_2 \cup ... \cup R_c \tag{4.8}$$

where $R_i \cap R_j = \phi$, if $i \neq j$. Each region is represented by a single unit feature vector $\mathbf{w}_i$, and all the vectors $\mathbf{w}_i$, $i = 1, 2, ..., c$ are not necessarily orthogonal and not necessarily linearly independent. They span a feature subspace, more specifically, sparse representation space here:

$$S = \text{span}\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_c\}. \tag{4.9}$$

In-place learning utilizes a local approach to estimating the density of X, decomposing a complex global problem of approximation and representation into multiple, simpler and local ones so that lower order statistics (means) are sufficient.

The de-correlated sparse representation (compared to pixel representation) increases the variations along different objects while maintaining low variation within an object, e.g., as shown in Fig. 4.7. Given the same number of neurons, the sparse coding scheme recruits the neuron resource distributed in a subspaces with largest variance, known as *Neuronal Density Theorem* (Weng and Luciw 2008 [90]). It alters the self-organization of neurons along the objects, i.e., information of interest. The sparse-coding scheme (feed-forward processing), together with the top-down connections (feed-back supervision) as described below, obtain the optimality of neuron distribution regarding the "object" information in extracted window images, invariant to irrelevant information like "leaked-in" backgrounds.

4.6.3 Top-down Connections

It is well-known that the feedback, or top-down connections have a crucial impact on the features detected by different neurons, as well as for visual awareness. The top-down connections provide a new subspace where the relevant information subspace (the information that is important to distinguishing between motor outputs) will have a higher variance than the irrelevant subspace. Since higher variance subspace will recruit more neurons due to the *Neuronal Density Theorem*, the representation acuity becomes higher in the relevant subspace, and the representation becomes more suited to the task(s) that were trained.

Consider that the top-down connections are represented by space Z as desired outputs from layer l. Such supervisory signals can be either provided *externally* from a teacher or provided *internally* from the next layer $l+1$.

Thus, Z is a part of the input to the neurons in layer l, along with its bottom-up input space $X = I \times R$:

$$X \times Z = (I \times R) \times Z = I \times (R \times Z)$$

We can see that top-down connections represented by Z in parallel with the relevant subspace R form the new relevant subspace $R \times Z$. This is because relevance is with respect to the motor output space Z.

Fig. 4.9 illustrates the top-down connection roles. As explained in the figure, the

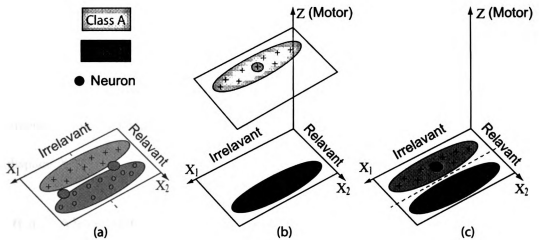


Figure 4.9. Illustration of top-down connection roles. Top-down connections boost the variance of relevant subspace in the neuronal input, resulting in more neurons being recruited along relevant information. The bottom-up input samples contain two classes, indicated by samples "+" and "○" respectively. The regions of the two classes should not overlap if the input information in X is sufficiently rich. The bottom-up input is in the form of $\mathbf{x} = (x_1, x_2)$. To see the effect clearly, assume only two neurons are available in the local region. (a) Class mixed. Using only the bottom-up inputs, the two neurons spread along the direction of larger variation (irrelevant direction). The dashed line is the decision boundary based on the winner of the two neurons, which is a failure (chance). (b) Top-down connections boost recruiting neurons along relevant directions. The relevant subspace is now spanned by bottom-up subspace of x_1 and top-down input space of z during learning. The two neurons spread along the direction of larger variation (relevant direction). (c) Class partitioned. During the testing phase, although the top-down connections do not provide any signal as it is not available, the two learned neurons in the X subspace is still along the relevant direction x_2. The winner of the two neurons using only the bottom-up input subspace X gives the correct classification (dashed line) and the samples are partitioned correctly according to the classes in the feature subspace x_2. Figure courtesy of Weng and Luciw 2008 [90].

top-down connections correspond to relevant information, the variation in top-down signals boosts the total variation of the relevant subspace. This enhanced variation recruits the locally available neurons to spread along the relevant subspace. As shown in Fig. 4.9(c), the neurons spread along the relevant direction in bottom-up subspace X, which are *invariant* to irrelevant information x_1. The classes are partitioned correctly in the subspace (partitioned at the intersection with the dashed line) after top-down connection, but before that, the classes in Fig. 4.9(a) are mixed in the bottom-up subspace X.

4.6.4 Algorithm

Algorithm 1 describes the detailed in-place learning procedure from layer 2 to layer 3 in the proposed system (see Fig.4.1), taking the sparse representations from layer 1 and object class labels from the motor. Each linked weight pair (i, j) shares the same value, i.e., $\mathbf{w}_{\mathbf{t}_i}^{l-1} = \mathbf{w}_{\mathbf{b}_j}^{l}$.

4.6.5 Learning Optimality of Developed Weights

We define $\mathbf{u}_i(t)$ as the *response-weighted input* of a neuron i in an in-place layer, such that

$$\mathbf{u}_i(t) = y_i(t)\ \mathbf{x}_i(t). \tag{4.12}$$

Algorithm 1 In-place Learning

1: Initialize the cell ages: $t = 1$ and the number of neurons: $c_2 = 225$, c_3 =`object categories`.
2: Set learning parameters: $\alpha_2 = 0.3$, $\alpha_3 = 0.0$.
3: **for** $t = 1, 2, ...$ **do**
4: Grab the sparse representation vector $\mathbf{s}(t)$ in layer 1, such that $\mathbf{y}^{(1)}(t) = \mathbf{s}(t)$.
5: Impose the motor vector (class labels) $\mathbf{m}(t)$ to layer 3, such that $\mathbf{y}^{(3)}(t) = \mathbf{m}(t)$.
6: $\mathbf{x}^{(2)}(t) = \mathbf{y}^{(1)}(t)$ and $\mathbf{z}^{(2)}(t) = \mathbf{y}^{(3)}(t)$
7: **for** $1 < l < 3$, i.e., all hidden layers but here only one hidden layer **do**
8: **for** $1 \leq i \leq c_l$ **do**
9: Compute pre-response of neuron i:

$$\begin{aligned} \hat{y}_i^{(l)}(t) &= g_i\Big((1-\alpha_l)\frac{\mathbf{w}_{\mathbf{b}_i}^{(l)}(t)\cdot\mathbf{x}^{(l)}(t)}{\|\mathbf{w}_{\mathbf{b}_i}^{(l)}(t)\|\|\mathbf{x}^{(l)}(t)\|} \\ &+ \alpha_l\frac{\mathbf{w}_{\mathbf{t}_i}^{(l)}(t)\cdot\mathbf{z}^{l}(t)}{\|\mathbf{w}_{\mathbf{t}_i}^{(l)}(t)\|\|\mathbf{z}^{l}(t)\|}\Big) \end{aligned} \tag{4.10}$$

where g_i is a neuron-specific sigmoidal function.
10: Simulating lateral inhibition, decide the winner: $j = \arg\max_{1\leq i\leq c}\{\hat{y}_i^{(l)}(t)\}$;
11: The 3×3 neighboring cells are also added to the winner set η.
12: The response $\mathbf{y}^{(l)}$ is computed from the pre-response. Only the winner neuron(s) have a nonzero response and are copied from $\hat{\mathbf{y}}^{(l)}$.
13: **end for**
14: **end for**
15: **for** $1 < l \leq 3$, i.e., all in-place layers **do**
16: Update the number of hits (cell age) n_j only for the winner neuron(s): $n_j \leftarrow n_j + 1$. Compute $\mu(n_j)$ by the amnesic function defined by Eq. (3.18) in Sec. 3.3.3, where plasticity parameters $t_1 = 20$, $t_2 = 200$, $c = 2$, $r = 2000$ in our implementation.
17: Update winner neuron(s) using its temporally scheduled plasticity:

$$\begin{aligned} \mathbf{w}_{\mathbf{b}_j}^{(l)}(t) &= (1-\Phi(n_j))\mathbf{w}_{\mathbf{b}_j}^{(l)}(t-1) \\ &+ \Phi(n_j)\mathbf{x}^{(l)}(t)y_j^{(l)}(t) \end{aligned} \tag{4.11}$$

where the scheduled plasticity is determined by its age-dependent weight:

$$\Phi(n_j) = (1+\mu(n_j))/n_j,$$

18: All other neurons keep their ages and weight unchanged.
19: **end for**
20: **end for**

Thus, Eq. (4.11) can be expressed as

$$\mathbf{w_{b}}_i(t+1) = (1 - \Phi(n_i))\mathbf{w_{b}}_i(t) + \Phi(n_i)\mathbf{u}_i(t) \tag{4.13}$$

In other words, the weight vector $\mathbf{w_{b}}_i$ is a weighted sum of all the $\mathbf{u}_i(t)$ won by the neuron i:

$$\mathbf{w_{b}}_i = \sum_t \Omega(n_i)\mathbf{u}_i(t), \tag{4.14}$$

where $\Omega(n_i)$ is given by the following expression:

$$\Omega(n_i) = \Phi(p) \prod_{p=q+1}^{n_i} (1 - \Phi(q)). \tag{4.15}$$

Since all the multiplicative factors above are non-negative, we have $\Omega(n_i) \geq 0$. Using the induction on n_i, it can be proven that all the weights $\Omega(n_i)$ sum to one for any $n_i \geq 1$:

$$\sum_{n_i} \Omega(n_i) = 1. \tag{4.16}$$

When $n_i = 1$, we require that $\mu(1) = 0$. Suppose that $\mathbf{u}_i(t)$ are independently and identically distributed (i.i.d.) with the same distribution as a random variable $u_i(t)$,

$$\begin{aligned} \mathrm{E}[w_{b_i}] &= \mathrm{E}\left[\sum_t \Omega(n_i)u_i(t)\right] \\ &= \left(\sum_{n_i} \Omega(n_i)\right) \mathrm{E}\left[u_i(t)\right] \\ &= \mathrm{E}[u_i(t)]. \end{aligned} \tag{4.17}$$

Eq. (4.17) shows that the developed feature value w_{bi} is an unbiased estimator of

$\mathrm{E}[u_i(t)]$. The minimum square error (MSE) of the estimator w_{bi} is then expressed as

$$\begin{aligned}
\mathrm{MSE}(w_{bi}) &= \mathrm{E}\left[\left(w_{bi} - E[u_i(t)]\right)^2\right] \\
&= \mathrm{Var}(w_{bi}) + \mathrm{bias}^2\left(w_{bi}, E[u_i(t)]\right) \\
&= \mathrm{Var}(w_{bi}) \\
&= \left(\sum_{n_i} \Omega^2(n_i)\right) \mathrm{Var}[u_i(t)] \qquad (4.18)
\end{aligned}$$

We defined the *error coefficient*:

$$\varepsilon(n) = \sum_{n_i} \Omega^2(n_i).$$

When $\mu(t) = 0$ for all n, the error coefficient becomes $\varepsilon(n) = 1/n$ and Eq. (4.18) returns to the expected square error of the regular sample mean:

$$\mathrm{MSE}(w_{bi}) = \frac{1}{n}\mathrm{Var}[u_i(t)]. \qquad (4.19)$$

As discussed in Sec. 3.3.3, the multi-sectional function $\mu(n)$ in Eq. (3.18) is more suited for practical signals with unknown *non-stationary statistics*, where the distribution does follow i.i.d assumption in all the temporal phase.

In summary, the in-place learning scheme balances dual optimalities in the aspect of both limited computational resources (spatial) and limited learning experience at any time (temporal) [2], such that

[2]More details and experimental results are available at Weng & Luciw 2009 [91].

1. Given the spatial resource distribution tuned by feed-forward and feed-back computations (see Eq. (4.10)), the developed features (weights) minimize the representational error.

2. The recursive amnesic average formulation enables automatic determination of optimal step sizes in this incremental non-stationary problem.

4.6.6 Learning Complexity

Compared with many Bayesian models (e.g., Emami and Jelinek [23]) and EM methods (e.g., Jornan & Jacobs [45]), the WWN does not require explicit search in high-dimensional model parameter space or compute the second order statistics.

Given each n-dimensional input image $\mathbf{x}$, the system complexity for updating m neurons is $O(mn)$. It is not even a function of the number of inputs t, due to the nature of incremental learning. For the network meant to run in real-time development, this low update complexity is very important.

4.7 Experiments and Results

We used an equipped vehicle to capture real-world images and radar sequences for training and testing purpose. Our dataset is composed from 10 different "environments" – stretches of roads at different looking places and times (see Fig 4.4 for a few examples of different environments). From each environment, multiple different

sequences were extracted. Each sequence contains some similar but not identical images (different scales, illumination and view point variation etc.). The sensorimotor pathway is evaluated in a prototype of two-class problem: vehicles and other objects, which are extendable to learn any types of objects defined by external teachers. There were 927 samples in the vehicle class and 404 samples in the other object class. For all tests, each large image from the camera was 240 rows and 320 columns. Each radar window was size-normalized to 56 by 56 and intensity-normalized to {0 1}.

4.7.1 Learning Efficiency

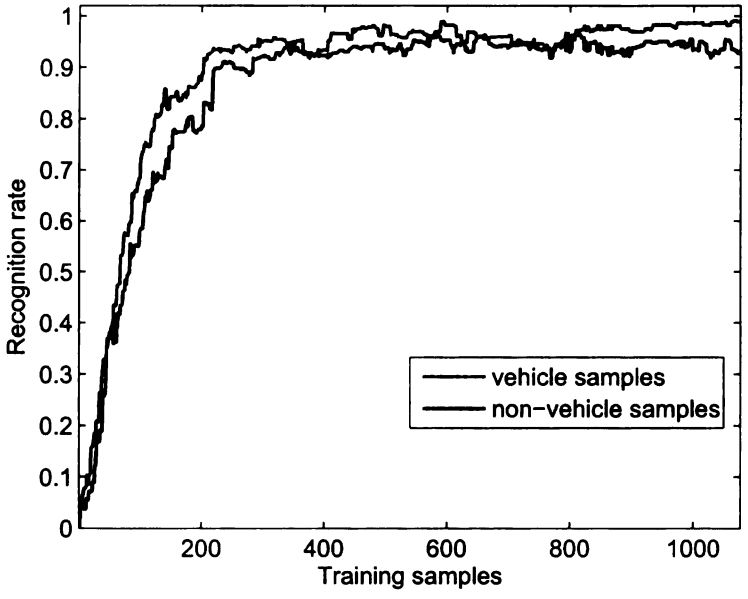


Figure 4.10. System efficiency of incremental learning.

The proposed sensorimotor pathway incrementally updates networks using one piece of training data at a time, and the data is no longer available as soon as it has been

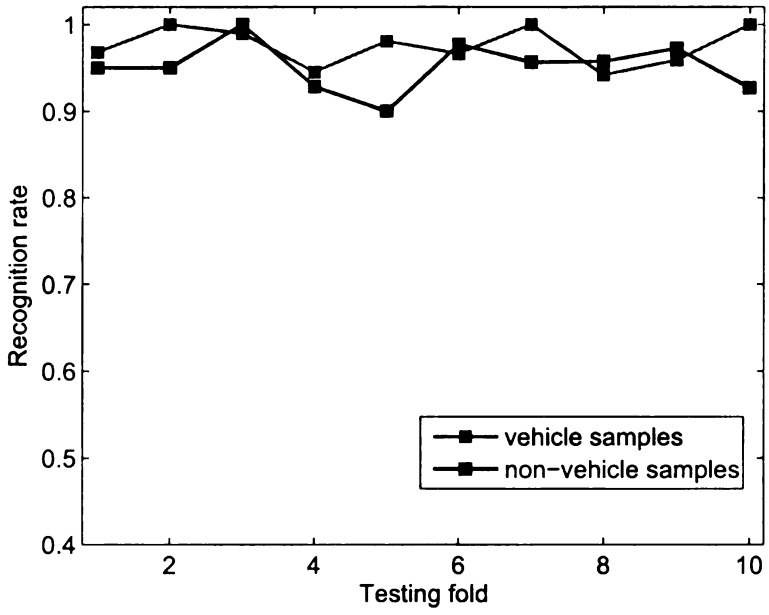
used. After the network initialization with $225 = 15 \times 15$ samples, the rest of data are used for the incremental learning by the network. As shown in Fig. 4.10, the network can be trained very quickly: only 25% of samples are sufficient to achieve around 80% recognition rate on testing all the data samples.

4.7.2 Performance Evaluation

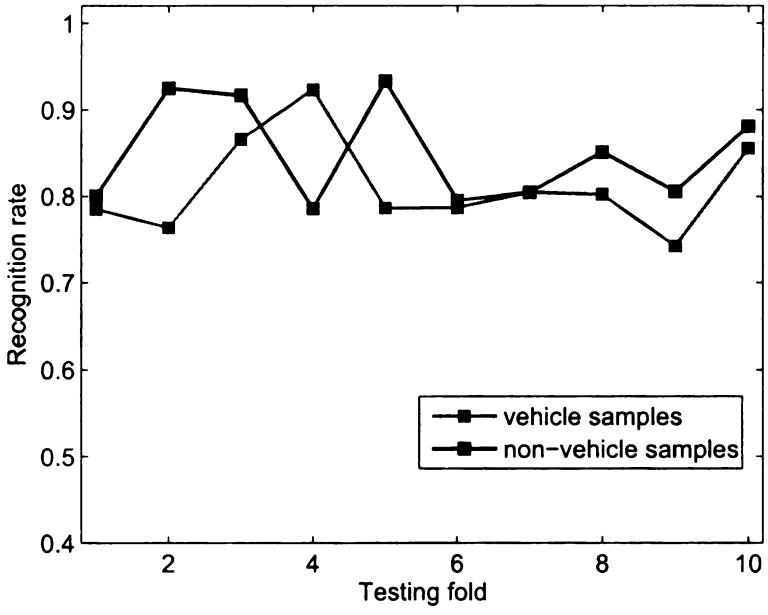
In this experiment, a ten-fold cross-validation is performed to evaluate the system performance. All the samples are shuffled and partitioned to 10 folds/subsets, where 9 folds are used for training and the last fold is used for testing. This process is repeated 10 times, leaving one different fold for evaluation each time. The cross-validation result is shown in Fig. 4.11 (a). The average recognition rate of the vehicle samples is 97.50%, and 95.19% of the other object samples, where the false positive and false negative rates are 2.37% and 5.51%, respectively. Compared to the performance without sparse coding process (see Fig. 4.11 (b)), we found that, averagely, the recognition rate improved 16.35% for positive samples and 10.21% for negative samples, respectively. This shows the power of the sparse-coded representations.

4.7.3 Performance Comparison

In the aspect of open-ended autonomous mental development, an efficient (memory controlled), real-time (incremental and fast), autonomous (cannot turn the system off to change or adjust), and extendable (the number of classes can increase) architecture is



(a)



(b)

Figure 4.11. 10-fold cross validation (a) with and (b) without sparse coding in layer 1

Table 4.3. Average Performance & Comparison of Learning Methods

	K-NN [*]	ISVM [*]	IHDR	Our work
Overall accuracy (%)	80.3 ± 10.4	72.0 ± 9.1	79.5 ± 5.3	**86.39 ± 1.02**
Vehicle accuracy (%)	72.3 ± 15.3	72.5 ± 10.5	71.5 ± 10.9	**87.6 ± 1.05**
Other objects accuracy (%)	**94.5 ± 6.7**	70.8 ± 8.8	90.6 ± 3.7	**84.5 ± 4.2**
Training time / sample (ms)	N/A	128.2 ± 11.7	**2.7 ± 1.1**	110.4 ± 7.9
Testing time / sample (ms)	445.0 ± 7.6	**2.2 ± 0.1**	4.7 ± 1.4	43.0 ± 6.8
Storage elements	1198	**85 ± 5.2**	1198	225

* The performance data was contributed by Luciw, from Ji et al. 2007 [41].

expected. We test and compare the following incremental learning methods to classify the extracted window images to vehicles and other objects: (1) k-nearest neighbor, with k=1, and using a L1 distance metric for baseline performance; (2) Incremental-SVM [12], (3) Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) [89] and (4) the proposed 3-layer network described in this chapter. We used a linear kernel for I-SVM, as is suggested for high-dimensional problems [56]. We did try several settings for a RBF kernel but did not observe as good performance as the linear kernel. Inputs to all compared methods were extracted window images, with input dimension of $56 \times 56 = 3136$.

Instead of randomly selecting samples in cross validation, we used a "true disjoint" test, where the time-organized samples are broken into ten sequential folds. Each fold is used for testing per time. In this case, the problem is more difficult, since sequences of vehicles or objects in the testing fold may have never been trained. This truly tests generalization.

The results are summarized in Tables 4.3. Nearest neighbor performs fairly well, but

is prohibitively slow. LBIHDR combines the advantage of K-NN with an automatically developed overlapping tree structure, which organizes and clusters the data. It is useful for extremely fast retrievals due to logarithmic complexity. LBIHDR is a little worse in performance than K-NN, but is much faster and can be used in real-time for training and testing. However, LBIHDR typically takes a lot of memory. It allows sample merging of prototypes, but in such case it saved every training sample, thereby did not use memory efficiently. I-SVM performed the worst with both types of input, but it uses the least memory, and the number of support vectors is automatically determined by the data. A major problem with I-SVM is lack of extendability – by only saving samples to make the best two-class decision boundary, it throws out information that may be useful in distinguishing other classes that could be added later.

The proposed biologically inspired pathway is able to perform better than all other methods using only 15×15 layer-2 neurons with a top-down supervision parameter $\alpha = 0.3$. It is also fairly fast, and efficient in terms of memory. Overall, the proposed pathway does not fail in any criteria, although it is not always the "best" in any one category, as currently implemented. NN will be too slow and I-SVM is not extendable for open-ended development. LBIHDR has problems using too much memory and does not represent information efficiently and selectively (supervised self-organization).

An incremental teaching interface was developed in MATLAB to experiment with the proposed system. The teacher could move through the collected images in the order

of their sequence, provide a label to each radar window, train the agent with current labels, or test the agent's current knowledge. Even in this non-parallelized version, the speed is close to real-time use. The average time for the entire system (not just the algorithm) to train samples was 6.95 samples/s and the average time for testing was 7.32 samples/s.

4.8 Summary

In this chapter, we demonstrated a general sensorimotor pathway for a developmental object recognition task. The genomic equivalence principle led to an in-place learning framework, in which a cell-centered developmental program would drive the activity of the network. The network autonomously develops its internal representation in reaction to coupled sensor-motor stimuli from the external environment, incrementally, and each cell is a reactive local agent. Hard-coded attention is provided by an efficient integration of multiple sensory modalities. Attended area are transformed into an over-complete, sparse-coded space by early developed filters/features, based on the staggered local receptive fields. The proposed sensorimotor pathway allows incremental and online learning, which is feasible for real-time use of any developmental robot that can sense visual information, radar information, and a teacher's input.

In the next chapter, the in-place learning principle is extended to model the brain's visual dorsal ("where") and ventral ("what") pathways using what is called Where-

What Network (WWN), in which both recognition and attention were integrated and accomplished by a single network. In contrast with the hard-coded attention through sensor integration in this chapter, attention selection (both bottom-up and top-down) will be developed as ubiquitous internal actions that take place virtually everywhere in the brains cortical signal processing and thus sequential decision making.

CHAPTER 5

Where-What Network: Developmental Integration of Attention and Recognition

Rooted in the genomic equivalence principle, this chapter extended the previous in-place sensorimotor pathway to a new developmental architecture, called Where-What Network (WWN), where object location ("where") and object type ("what") are integrated interactively through an experience-based development. This architecture enables three types of attention: feature-based bottom-up attention, position-based top-down attention, and object-based top-down attention, as three possible information flows through the Y-shaped network. The inputs to the network are a sequence of images where a foreground object of size 21×21 pixels may appear anywhere within an unknown, complex, natural background of size 40×40 pixels. The WWN regulates the network to dynamically establish and consolidate position-specified and type-specified connections through a supervised learning mode. The network has reached 95% correct recognition rate for 5 objects (foreground images) and an average of 1.1 pixels in

position error after 30 epochs of training.

5.1 Ventral and Dorsal Pathways

Studies in neuroscience have found two pathways in the primate vision system, the ventral pathway and the dorsal pathway. The ventral pathway takes major part of the signals from the P cells in the retina, via the P channel in LGN, and goes through the cortical regions V1, V2, V4, PIT, CIT, AIT. The dorsal pathway takes major part of the signals from the M cells in the retina, via the M channel in LGN and goes through the cortical regions V1, V2, MT, LIP, MST, VIP, 2a and further on. It was suggested (e.g., Kandel et al. 1994 [47]) that the ventral pathway is mainly responsible for the form and color of objects while the dorsal pathway mainly processes information about motion and position. In other words, the ventral pathway is a "what" pathway and the dorsal pathway a "where" pathway. Fig. 5.1 shows the nature of the processing along the "where" and "what" pathways, both of which are shaped by not only sensory inputs but also the motor outputs.

5.2 Chicken-egg Problem

The connection of the "where" and "what" pathways reveals that attention and recognition happens concurrently, known as a chicken-and-egg problem. Without attention, recognition cannot do well: recognition requires attended areas for the further process-

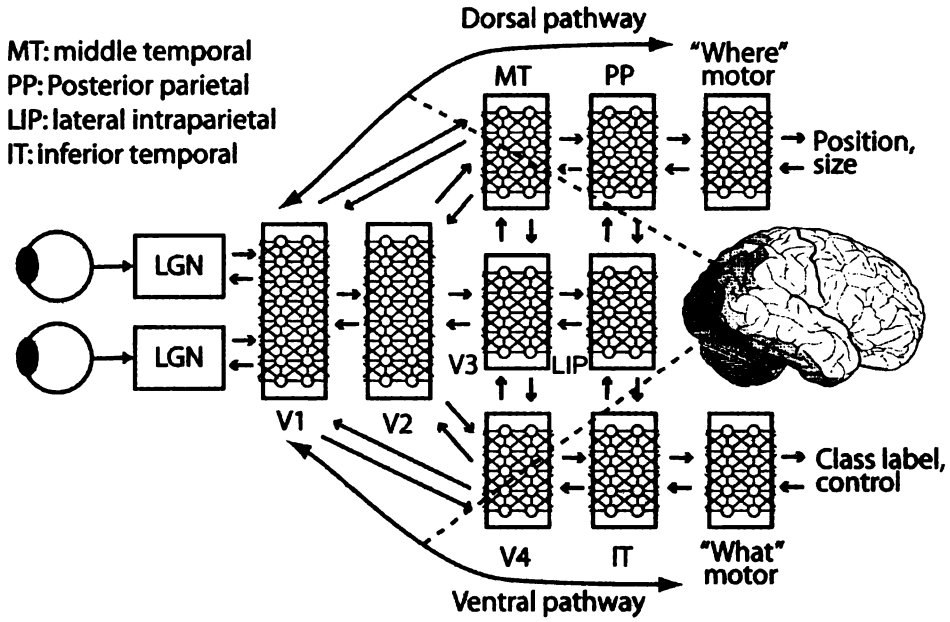


Figure 5.1. The nature of the processing in the "where" and "what" pathways. Figure courtesy of Weng and Luciw 2008 [90]

ing. Without recognition, attention is limited: attention does not only need bottom-up saliency-based cues, but also top-down object-dependant and position-dependant signals.

5.2.1 Bottom-up Attention

Bottom-up attention, also called saliency-based attention, uses different properties of sensory inputs, e.g., color, shape, and illuminance to extract saliency. One of the earliest theoretical studies to address this problem is a psychophysical literature. Treisman et al. [80] proposed that different features are derived from the feature maps in different brain regions. To solve the binding problem, the brain dedicates some processing as a master map to combine the code of the feature maps.

An explicit computational model of bottom-up attention was introduced by Koch & Ullman in 1985 [49], where a "saliency map" was used to encode stimuli saliency at every location in the visual scene. Elementary features, such as color, orientation, direction of movement and disparity are represented parallelly in different topographical maps, corresponding to the feature maps of Treisman's model. A selective mapping transforms these representations into a central representation, where inhibition of returns suppressed the current attended location and enabled the shifting to the next salient location.

More recently, Itti & Koch et al. 1998 [39] integrated color, intensity, and orientation as basic features. Gaussian pyramids were constructed to extract intensity and color features from red-green opponency channels and blue-yellow opponency channels. Four orientation-selective pyramids were generated using Gabor filtering at 0, 45, 90, and 135 degrees. 42 feature maps in total were created therefore: 6 for intensity, 12 for color, and 24 for orientation, normalized and combined to a saliency map.

Backer et al. [4] applied similar strategy to an active vision system, called NAVIS (Neural Active VISion), emphasizing the visual attention selection in a dynamic visual scene. Instead of directly using some low level features like orientation and intensity, they accommodated additional processing to find middle level features, e.g., symmetry and eccentricity, to build the feature map. Fusion of conspicuity maps was conduced using so-called Dynamic Neural Fields [1].

Most of the bottom-up attention selection models share similar ideas to compute a saliency map. The major differences lie in the variety of features, strategies of feature combination, and the rules to find the most salient location.

5.2.2 Top-down Attention

Volitional shifts of attention are also thought to be performed top-down, through space-defined and feature-dependant weighting of the various feature maps.

Early top-down attention models selected the conspicuous positions regardless of being occupied by objects or not, as is called position-based top-down control. Olshausen et al. 1993 [61] proposed a model of how visual attention could solve the object recognition problem of position and scale invariance. A representative top-down attention model was discussed later by Tsotsos et al. 1995 [81], who implemented attention selection using a combination of a bottom-up feature extraction scheme and a top-down selective tuning scheme. A more extreme view was expressed by the "scanpath theory" of Stark & Choi 1996 [14], in which the control of eye movements was almost exclusively under top-down control. Mozer et al. 1996 proposed a model called MORSEL [66], to combine the object recognition and attention, in which attention was shown to help recognition. A top-down, knowledge-based recognition component, presented by a hierarchical knowledge tree, was introduced by Schill et al. 2001 [71], where object classes were defined by several critical points and the corresponding eye movement commands

that maximized the information gain. Rao et al. 2004 [70] described an approach allowing a pair of cooperating neural networks to estimate object identity and object transformations, respectively.

A successful top-down attention system should also accommodate the object recognition abilities to boost the top-down attention control. This integration, in particular, accounts for the increasing efforts on the object-based top-down attention (discussed in Itti & Koch 2001 [38]). An early computational model of object-based attention was proposed by Sun and Fisher 2003 [76], where hierarchical attention selection was obtained by a grouping of location and object features. Orabona et al. 2005 [64] utilized a proto-object selection to guide a humanoid robot's attention in a comparatively simple environment. A spiking neural network is introduced by Lee et al. [52] to model object-based attention to a potential region containing human faces. By modifying Itti's bottom-up attention model [39], Walther et al. 2006 [86] inferred the extent of object information at the attended location to compute the saliency map.

5.2.3 Integration of Attention and Recognition

Aforementioned work provided computational models for attention (both bottom-up and top-down) and its perspective of object recognition capabilities, however, limited work has addressed an issue on the overall interactions and integrations. Especially, what is the computational causality that can account for the concurrent development

of the "where" and "what" pathways by both bottom-up and top-down controls? Deco & Rolls 2004 [18] presented a model to integrate both invariant object recognition and top-down attentional mechanisms on a hierarchically organized set of visual cortical areas. The model displayed position-based and object-based covert visual search by using attentional top-down feedback. Unfortunately the proposed architecture was not demonstrated in a scalable network for an engineering performance of attention and recognition. Moreover, top-down connections were used to propagate top-down signals only, without any internal development through neural computations.

Our recent work (Ji et al. [40]) was a first simple prototype (called WWN-I) for an interactive integration of top-down attention (both position-based and object-based) and recognition. Rather than the simulations of fMRI data, the engineering performance of recognition rate and attended spatial locations are presented in the experiment. The preliminary result showed how one type of information ("where" or "what") assists the network to suppress irrelevant information from background (from "where") or irrelevant object information (from "what"), so as to give the required missing information ("where" or "what") in the motor output. However, since the bottom-up feature-based attention was not modeled (i.e., layer-1 features were off-line developed) in the architecture, the WWN-I cannot recognize or attend to objects when the top-down signals are missing. Fig. 5.2 and Fig. 5.3 showed that top-down connections from one motor layer helps the output from another motor layer in an interactive way, while the performance was dramatically degraded once the top-down signals became unavailable. Moreover,

the limited complexity of "where" and "what" outputs (5 objects and 5 positions) suffer generality and scalability of the network.

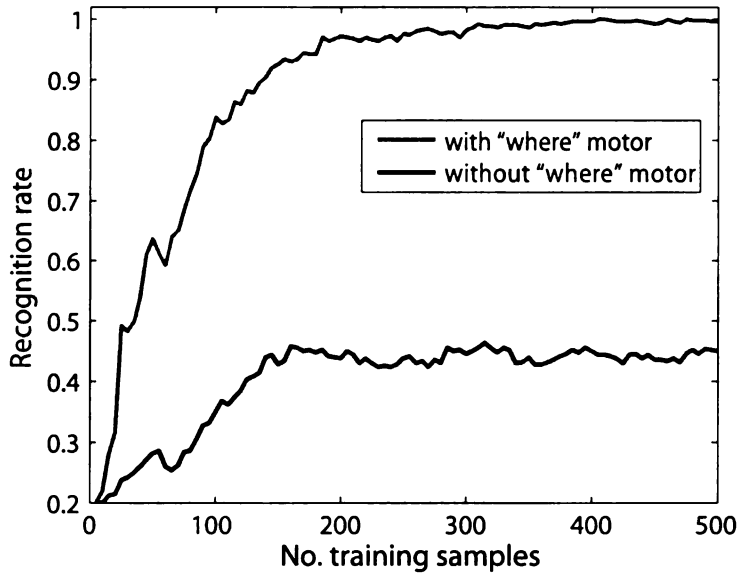


Figure 5.2. Recognition rate with incremental learning, using one frame of training images at a time.

5.3 Where-What Network

To solve the existing limitations in attention and recognition, we reform the WWN-I and propose a general architecture, called Where-What Network (WWN). The proposed work is the more preferable developmental model for a sensorimotor pathway that interactively incorporates bottom-up attention, top-down attention and object recognition as emergent internal actions, solving the where-what problems for naturally unsegmented inputs with low computational complexity.

A series of advances were made in this chapter to make the developmental work

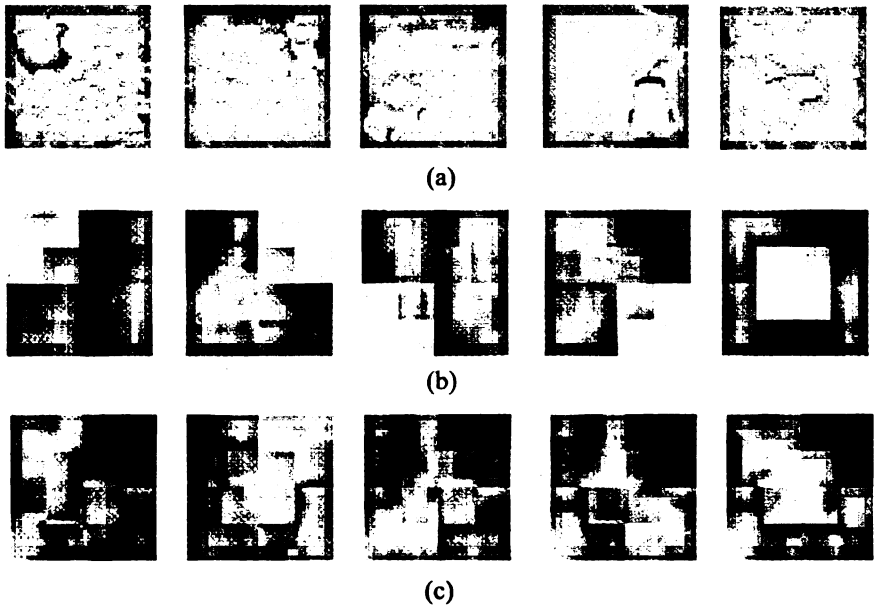


Figure 5.3. (a) Examples of input images; (b) Responses of attention ("where") motors when supervised by "what" motors. (c) Responses of attention ("where") motor when "what" supervision is not available.

novel, important and challenging: (1) Computational modeling of attention and recognition as ubiquitous internal actions that take place virtually everywhere in the brain's cortical signal processing and thus sequential decision making. (2) Computational understanding of saliency-based bottom-up attention, position-based top-down attention and object-based top-down attention as concurrent information stream. (3) General framework on the development of sensory invariance through two interactive top-down abstractions. (4) Success in the theory and demonstration in key brain-scale learning mechanisms – intra-cortical architecture and inter-cortical (i.e., cortico-cortical) wiring. The WWN uses the top-k mechanism to simulate the in-place competition among neurons in the same layer, showing its computational efficiency by avoiding iterations within a layer.

This system is demonstrated in the complex dynamic visual environments, learns on the fly, and produces online internal attention actions and external motor actions. Compared to the limited complexity in our previous WWN-I (Ji et al. [40]), the new developed WWN learned real images including objects on every possible position in complex natural backgrounds. The recognition rate and attention accuracy present very impressive results. Based on years of research towards developmental systems, the proposed work is a significant advance in the intersection of neuroscience, developmental psychology, computer science and robotics.

5.4 Network Structure

An outline of the WWN architecture is illustrated in Fig. 5.4. The network does not mean to duplicate all major biological details, but intends to reveal the computational aspects that possibly contribute to the development of the brain's visual pathways. The neuromorphic network is incrementally updated at a frequency f (e.g., 10 frames per second), taking inputs sequentially from sensors and effectors, computing responses of all neurons, and producing internal and external actions through experience.

Each cortical area in the WWN contains 4 layers, i.e., L2/3, L4, L5 and L6, arranged in a multi-level hierarchy. The same number of neurons (specified under a cortex in Fig. 5.4) are placed in each layer, wired by intra-cortical and inter-cortical connections. The cortex V2 is locally connected with input images and globally connected by the

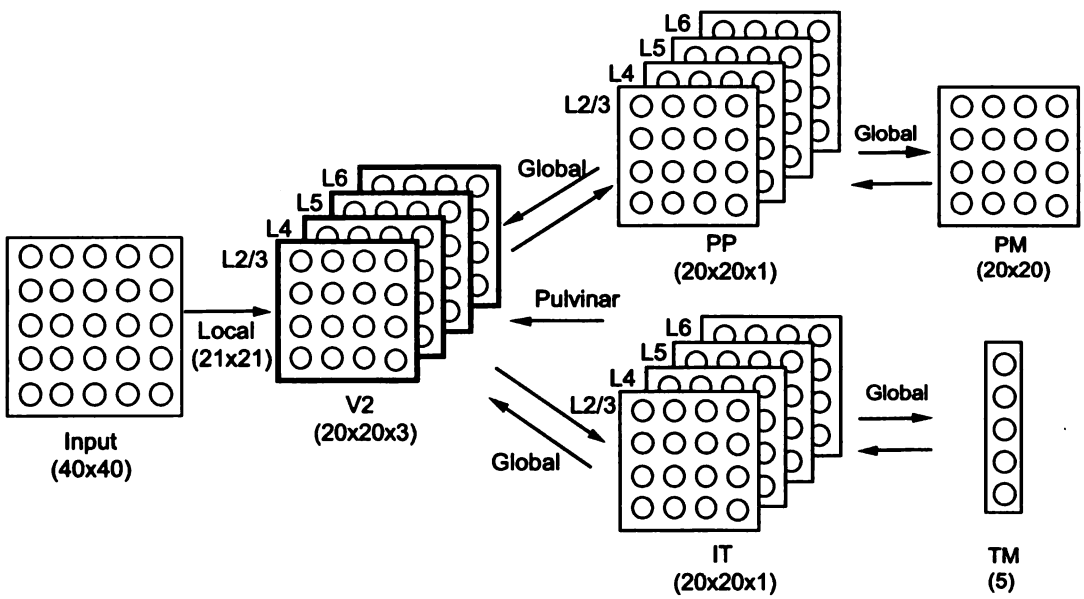


Figure 5.4. Implemented architecture of the Where-What Network II, connected by 3 cortical areas in 2 pathways. The "where" pathway is presented through areas V2, PP (posterior parietal), and PM (position motor), while the "what" pathway is through areas V2, IT (inferior temporal), and TM (type motor). The pulvinar port provides soft supervision of internal attention in V2. Since the size of the object foreground is fixed in the current WWN, the biological LGN and V1 areas are omitted without loss of generalities. The network is possibly extensive to more cortices that match the biological architectures described in Fig. 5.1.

next two pre-motor cortices. The pre-motor cortices PP and IT are globally connected with V2 and their corresponding motor area (PM and TM) respectively. Each neuron is placed at a 3-D position (r rows $\times c$ columns $\times d$ depths) in a layer, connected with two types of information flows, bottom-up and top-down. The pulvinar connection provides retinotopic attention in V2, considered as a port for supervised teaching.

5.4.1 Cortical Connections

Cerebral anatomical studies have shown that the architecture of the cerebral cortex appears to have the same 6-layer organization scheme across different cortical areas in the forebrain. In each layer, there are feedforward as well as recurrent connections between excitatory neurons, and inhibitory neurons provide specific kinds of inhibition, either at somata or at dendrites, depending on the class of cells (e.g., Callaway 1998 [11]). The importance of intra-cortical connectivity for higher brain functions is realized in computational models, where various networks have been used to model the cortical processing and learning [75, 29, 28].

In the WWN, each visual cortex has two functional layers, L2/3 and L4. L6 assists L4 for lateral interactions, and L5 assists L2/3 for lateral interactions as in the 2-level model of Callaway 1998 [11]. Two major simplifications are made here: direct links from L4 to L5, and direct subcortical input to L6 are neglected.

As illustrated in Fig. 5.5, there are 3 types of connections in a cortex V_n, shown

with different colored lines:

1. Intra-cortical connections (red lines) between a functional layer and its corresponding assistant layer, i.e., L4-L6 and L2/3-L5.

2. Intra-cortical connections (green lines) between two functional layers, i.e., L2/3-L4.

3. Inter-cortical connections (blue lines) between the current cortex V_n and the previous cortex V_{n-1} or the next cortex V_{n+1}.

Localized inter-cortical connections are utilized in the cortex V2, coding image features in set of vertical depths. Neurons in later visual cortex (i.e., PP and IT) present global connections to associate abstract features with motors. The increment of cortical areas will entail the WWN to obtain alternative receptive fields at multiple scales and positions, dealing with more "where" and "what" variations.

5.4.2 Local Receptive Fields

The receptive field of a neuron (Hubel and Wiesel [35]) is the area in an input array (image in this case), from which the neuron receives signals, either directly or indirectly. In the current WWN, square, staggered receptive fields are applied to the neurons in V2–L4, perceiving 21 × 21 local areas of the input image plane. Fig. 5.6 shows the organization of the receptive fields, where staggered distance is set to be 1 pixel in both

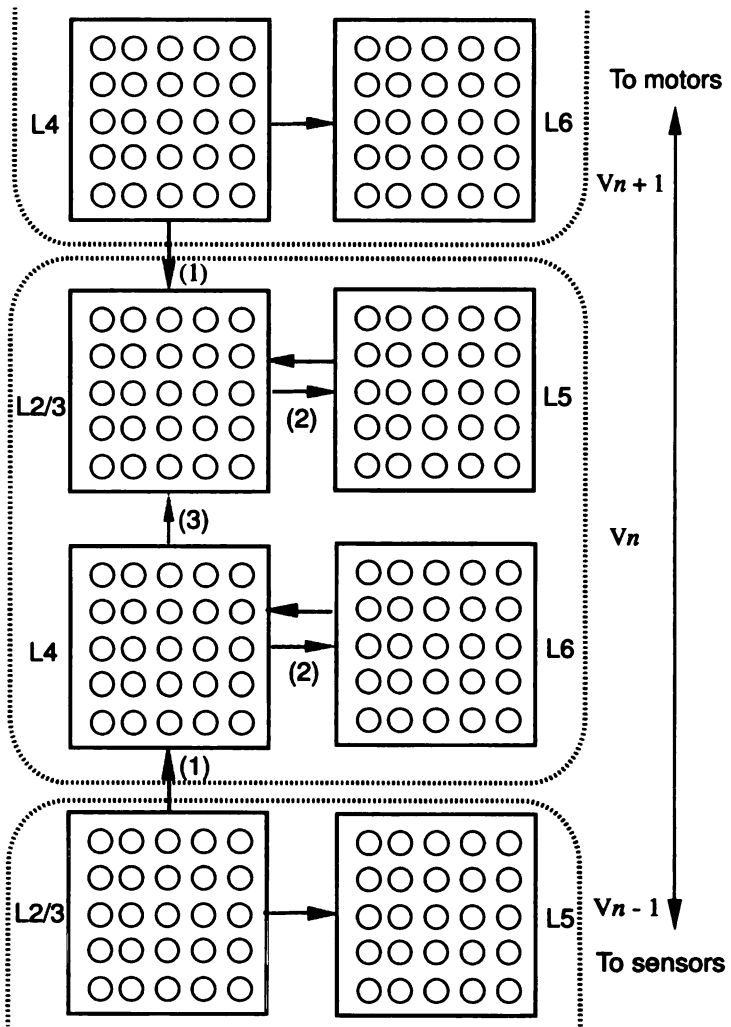


Figure 5.5. A schematic illustration of inter-cortical (from V_{n-1} to V_{n+1}) and intra-cortical connections (from L2/3 to L6) in the WWN. For simplicity, each layer contains the small number of neurons, arranged in a 2-D neuronal plane, where $d = 1$. Numbers in the brackets denote the computational steps from (1) to (3), corresponding to the descriptions from Sec. 5.6.1 to Sec. 5.6.3, respectively. Colored lines indicate different types of projections.

horizontal and vertical directions. A connection falls outside of the neuronal plane is not allowed. Thus, given an input image with dimension 40 × 40, V2–L4 contains 3 depth of $(40-21+1)\times(40-21+1)$ neurons. It also indicates that neighboring neurons receive highly correlated features with 1 pixel shifting in stimuli, providing a structural basis for attending objects in every position.

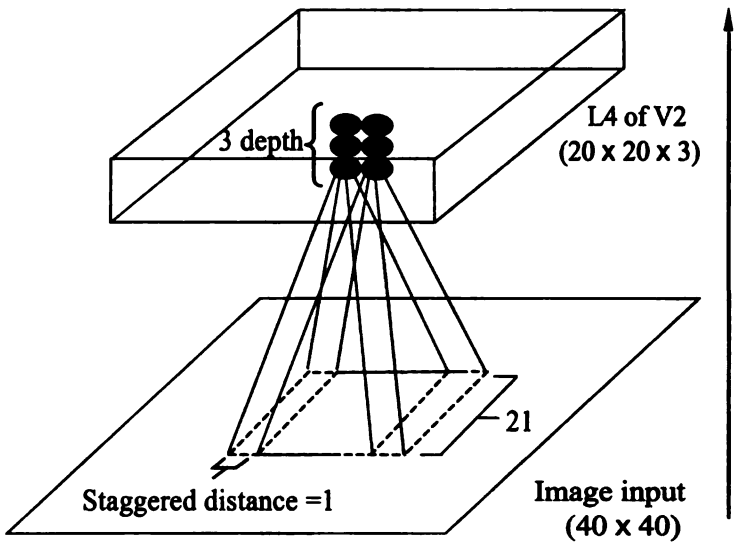


Figure 5.6. Organization scheme for the depths of neurons in V2.

5.4.3 Pulvinar

The pulvinar nucleus of the thalamus is considered as a key structure of the cerebral network, facilitating stimulus-driven attentional control. Early work by Petersen et al., 1985 [67] revealed that pulvinar neurons enabled a spatially selective enhancement of visual responses when the primate was covertly attending to a stimulus location. Amount of positron emission tomography studies (e.g., LaBerge & Buchsbaum 1990

[50]) suggested that the pulvinar become more active when there was a need to filter out irrelevant events. Computationally, Olshausen et al. 1993 [61] described the pulvinar as a strategic function to modulate attentional shifts, and to expand or shrink attentional windows. The attended targets were represented in cortical areas within an object-centered reference frame, which was position and scale invariant, thus suitable for recognition.

Motivated by studies on the pulvinar in stimulus-driven attentional control, the WWN models thalamus complex through the supervision of attended neuronal area. The pulvinar imposes a region of $3 \times 3 \times 3$ neurons in V2-L2/3 (see Fig 5.7), where the center neuron fully "perceives" the current object by its receptive field. The supervised neural representation is given below at time t:

$$\begin{cases} y_i^{(\mathrm{p})}(t) = 1, & \text{if } \ i \in N \\ y_i^{(\mathrm{p})}(t) = 0, & \text{otherwise} \end{cases} \tag{5.1}$$

where N denotes the $3 \times 3 \times 3$ neighborhood.

The soft supervision of the pulvinar assists neural competitions in V2, suppressing neuronal responses whose receptive fields fall out of the attended fields. In contrast, the pre-motor cortex bridges the V2 and the motor area in global connections, the neurons in these two cortices are not organized in the position-specific manner; thus, the pulvinar supervision is not necessary.

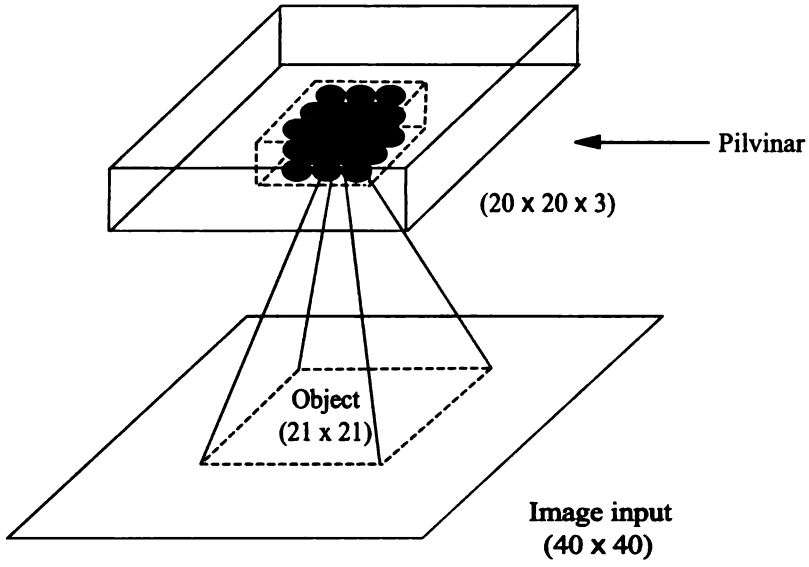


Figure 5.7. The pulvinar supervision in the cortex V2.

5.5 Bottom-up and Top-down Flows

In the WWN, each neuron i at cortex $V_n - L4$ has bottom-up (excitatory) weight vector that links neurons in the earlier cortex. Each neuron i at cortex $V_n - L2/3$ has top-down weight vector that links neurons in the later cortex. Bottom-up connections generate the information flows starting from an input image, going through V2, branching to the ventral and dorsal pathways, and finally reaching the position motor output and the type motor output, respectively. Top-down connections take response patterns that are imposed by one of the two motors. The type motor (TM) represents the object type to be searched in the image and the position motor (PM) represents the retinal position as the focus of attention.

5.5.1 Motor-specific Features

Fig. 5.8 illustrated how cortical neurons develop motor-specific features using the bottom-up and top-down flows. Assisted by the supervision from the pulvinar, the two-way connection enables V2 neurons to learn object features in different positions (e.g., F1/P1, F2/P1, F1/P2, F2/P2 etc. in Fig. 5.8). Due to the lack of support from the pulvinar, background natural textures cannot survive via the neural competitions, i.e., top-1 firing through the lateral inhibition.

As discussed in Sec. 4.6.3, the top-down connections from motor areas coordinate the neural competition through two functional properties of abstraction: (1) Responses from developed neurons are insensitive to irrelevant sensory information (i.e., invariants) but are sensitive to relevant sensory information for discriminants. (2) Neurons form topographic cortical areas according to abstract classes. Both properties transform meaningless (iconic) raw sensory inputs into internal representation with abstract meanings, and lead to the topographic class grouping (TCG) (Luciw & Weng 2008 [54]). That is, based on the availability of neurons, the features represented for the same motor class are grouped together to reduce the complexity of decision boundaries, leading to the better performance.

The neural competition in the IT area leads one or more neurons to fire for a specific type ("what") feature (e.g., F1, F2 or F3 in Fig. 5.8), via the role of motor abstractions. The power of top-down connection causes the winning of one type feature over different

positions, presenting the **position invariance** in IT. Similarly, the PM cortex develops position ("where") features (e.g.,P1, P2 or P3 in Fig. 5.8) over multiple object types, showing the capability of **type invariance**. The effect of the position invariance and type invariance in the IT and PP cortices will be demonstrated in our experiment in Sec. 5.8.

5.5.2 Attention Control

The motor-specific features integrate three types of attention controls in a single network: (a) the **bottom-up saliency-based attention**, varying with an external world change, (b) the **position-based top-down attention**, derived from imposed signals in PM and (c) the **object-based top-down attention**, derived from imposed signals in TM.

Bottom-up attention has been modeled by a variety of hand-designed features, especially in the aspect of saliency properties (as discussed in Sec. 5.2.1). In the WWN, however, bottom-up attention is a natural consequence of experience of interactions (winning in competition), where no master feature maps are required. All the internal representations, i.e., features, are developed through the regulation of cell-centered computations in the two-way information flows.

Position-based top-down attention has been investigated as multi-level shifter circuits that switch the value of feature detectors in an early layer to a standard master map

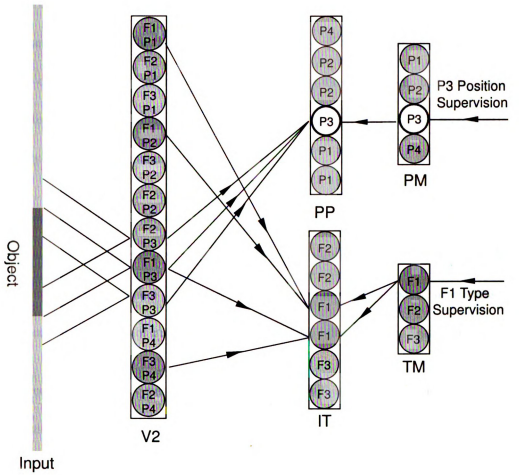


Figure 5.8. Top-down connections supervise the learning of motor-specific features, where the top-1 firing rule is applied. Only input connections to some typical neurons are shown in IT and PP. IT neurons develop position-invariant features since its top-down signals are type-specific. Depending on the availability of neurons in IT, there might be multiple neurons that correspond to a single object type, giving more quantization levels for within-type variation. PP neurons develop type-invariant features since its top-down signals are position-specific. Depending on the availability of neurons in PP, there might be multiple neurons that correspond to a single position, giving more quantization levels for within-position variation.

in a later layer (e.g., aforementioned Olshausen et al. 1993 [61]). Rather than the complicated selection of feature pools, the WWN uses its position motor as a natural source of position-based top-down control, which can be easily accessed and supplied by an external teacher. This tight integration not only allows different alternative sources of top-down attention control to use the same network mechanism, but also opens the door toward self-generated autonomous attention (e.g., through the introduction of novelty-driven mechanisms in motor behaviors).

Object-based top-down attention has been known to exist in primate vision [19][18], but has resisted complete modeling, computer realization and analysis. The WWN presents an interpretation that both position-based and object-based attentions share the same mechanisms of motor-specific controls, as schematically drawn in Fig. 5.8.

Through the three types of attentions, the WWN addresses the general attention recognition problem as follows: presented with an object of interest in an unknown complex background, direct attention to the object, recognize it and tell its retinal position.

5.6 Cortex Computation

Given an image stimulus and two separate motors at discrete time t, the following steps describe the neural competition from layer to layer for a cortex V_n.

5.6.1 Pre-response in L4 and L2/3

The ith neuron at L4 takes its bottom-up input $\mathbf{x}_i(t)$ from the external image stimulus or the previous cortex (refer to Fig. 5.5 step (1)), giving the *pre-response* $y_i^{(4)}(t)$:

$$y_i^{(4)}(t) = \frac{\mathbf{w_{b}}_i(t) \cdot \mathbf{x}_i(t)}{\|\mathbf{w_{b}}_i(t)\|\|\mathbf{x}_i(t)\|} \tag{5.2}$$

Accordingly, the ith neuron at layer L2/3 takes its global top-down input $\mathbf{z}_i(t)$ from the next cortex or the external motors, giving the pre-response $y_i^{(2/3)}(t)$:

$$y_i^{(2/3)}(t) = \frac{\mathbf{w_{t}}_i(t) \cdot \mathbf{z}_i(t)}{\|\mathbf{w_{t}}_i(t)\|\|\mathbf{z}_i(t)\|} \tag{5.3}$$

$\mathbf{w_b}$ and $\mathbf{w_t}$ are bottom-up and top-down weights of neuron i at layers L4 and L2/3, respectively.

5.6.2 Lateral Inhibition

As discussed in Sec. 3.2, lateral inhibition refers to a mechanism of competition among neurons in the same layer. The net effect of lateral inhibition is to suppress weakly responding neurons in the spatial aspect.

The pre-responses in L4 and L2/3 perform the lateral inhibition through the modulatory signals in L5 and L6, respectively, via their point-to-point connections (See Fig. 5.5 step (2)). Coordinated inhibition recruits the right number of neurons for excitatory activities (firing patterns), providing the spatial regulation in the cortical network.

All the pre-responses in a functional layer l (either L4 or L2/3) are sorted in the descending order through its corresponding assistant layer (L6 or L5), such that $q_1^{(l)}(t) \geq q_2^{(l)}(t) \geq ... \geq q_k^{(l)}(t) ... \geq q_m^{(l)}(t)$ $(m = r \times c \times d)$. The top-$k$ pre-responding neurons are selected to survival and all the other neurons are set to zero responses. Thus,

$$\begin{cases} \bar{y}_i^{(l)}(t) = y_i^{(l)}(t), & \text{if } y_i^{(l)}(t) \geq q_k^{(l)}(t) \\ \bar{y}_i^{(l)}(t) = 0, & \text{otherwise} \end{cases} \tag{5.4}$$

where $\bar{y}_i^{(l)}$ presents the *responses* in L4 or L2/3.

The larger k gives more information about the input variation in multiple positions associated with the top-k winning neurons, but the multiple responses do not help when the density of the prototypes is low and the position is highly sensitive. In the current implementation of WWN, there is no with-in class variation and an object shifted with 1 pixel is sensitive to a different "where" motor, therefore, k is set 1 to select the best feature in an accurate position. Such a strong sparse-coding scheme largely disregards less relevant feature detectors in our challenging tasks.

5.6.3 Cortex Representation

Intra-cortical connections (in Fig. 5.5 step (3)) link two functional layers L4 and L2/3 by point-to-point addition. The *post-responses* are computed and stored in L2/3 as follows:

$$\begin{aligned} \hat{y}_i^{(2/3)}(t) = (1 - \gamma^{(\mathrm{V}n)})\big[(1 - \alpha^{(\mathrm{V}n)})\bar{y}_i^{(2/3)}(t) \\ + \alpha^{(\mathrm{V}n)}\bar{y}_i^{(4)}(t)\big] + \gamma^{(\mathrm{V}n)}y_i^{(\mathrm{p})}(t) \end{aligned} \tag{5.5}$$

where $\alpha^{(V_n)}$ and $\gamma^{(V_n)}$ are contributing factors assigned to L4 and the pulvinar respectively at cortex V_n. In our experiment, $\alpha^{(V_2)} = 0.75$ and $\gamma^{(V_2)} = 0.75$. Since the pulvinar supervision is not imposed in the pre-motor cortices, such that $\alpha^{(PP)} = \alpha^{(IT)} = 0.25$ and $\gamma^{(PP)} = \gamma^{(IT)} = 0$.

Lateral inhibition is again applied to the post-responses in L2/3, where top-1 winning selects only one neuron to be fired and updated in the cortex. The inhibited *synthetic response* $\tilde{y}_i^{(2/3)}(t)$ is considered as the cortex representation and then transmitted to L4 for the neural learning of both bottom-up and top-down weights, such that

$$\tilde{y}_i^4(t) = \tilde{y}_i^{2/3}(t) \tag{5.6}$$

5.7 Neuron Learning

Given a neuron i with non-zero *synthetic response* through lateral inhibition, its connection weights are updated using dually optimal in-place learning rule, as discussed in Sec. 4.6. The ith neuron in L4 and L2/3 adapts its bottom-up weight and top-down weight, respectively, according to the same biologically motivated mechanism:

$$\begin{cases} \mathbf{w_{b}}_i(t+1) = (1 - \Phi(n_i))\mathbf{w_{b}}_i(t) + \Phi(n_i)\tilde{y}_i^{(4)}(t)\mathbf{x}_i(t) \\ \mathbf{w_{t}}_i(t+1) = (1 - \Phi(n_i))\mathbf{w_{t}}_i(t) + \Phi(n_i)\tilde{y}_i^{(2/3)}(t)\mathbf{z}_i(t) \end{cases} \tag{5.7}$$

The scheduled plasticity is determined by its age-dependent weight:

$$\Phi(n_i) = (1 + \mu(n_i))/n_i,$$

where n_i is the number of updates that the neuron has gone through. $\mu(n_j)$ is the amnesic function defined by Eq. (3.18) in Sec. 3.3.3, where plasticity parameters $t_1 = 20$, $t_2 = 200$, $c = 2$, $r = 2000$ in our implementation.

Finally, the winning neuron's age n_i is incremented: $n_i \leftarrow n_i + 1$.

5.7.1 Smoothness

The neural resources should be explored to avoid the unbalanced competition among developed (mature) neurons and its undeveloped neighborhood. Since every neuron is initialized with some uniform pattern at the first beginning (e.g., gaussian filter in our case), its competition power is weak, i.e., delivering low pre-responses. Once a neuron wins for development in a neighborhood, the developed neuron increases its competition power to keep winning similar object features associated with different "where" motor classes (e.g., the same object types in neighboring positions). The unbalanced development suffers the discriminative/selective power and idles a large mount of neural resources.

The WWN uses lateral connections to populate $3 \times 3 \times 1$ neighboring neurons around the winner, and update them using the optimal Hebbian-like learning scheme in Eq. (5.7). After the network is developed with one epoch, i.e., trained by all the objects located in all possible positions with different backgrounds, all the network neurons become mature and excitation neighborhood is decreased to be $1 \times 1 \times 1$ to boost the

feature selectivity.

5.8 Experimental Results

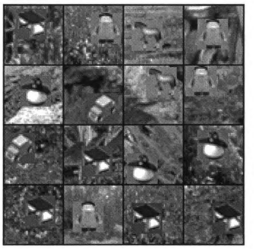


Figure 5.9. 16 sample images used in the experiment, containing 5 different objects, i.e., "penguin", "horse", "car", "person" and "table". Each object is randomly placed in one of 20×20 "where" positions.

Fig. 5.4 shows a specific set-up of cortices, layers, neurons and their connections in the implemented architecture of WWN. The images of objects are normalized in size to 21 rows and 21 columns. One of 5 different object types (i.e., "what" motors) is placed at one of 20×20 different positions (i.e., "where" motors) in a 40×40 background. The background images are randomly selected from a collection of natural images [1]. Each image stimulus is presented with 3 repeated iterations, such that the total number of $20 \times 20 \times 5 \times 3$ times are updated for the network training in an epoch. A set of data

[1] http://www.cis.hut.fi/projects/ica/data/images/ via Helsinki University of Technology

examples are shown in Fig. 5.9.

5.8.1 Salient Features in V2

The network weight of every neuron is initialized by a gaussian function without any specific feature patterns. Running after 30 epochs of development, the network develops 3 depth of salient features (i.e., bottom-up weights) in V2, as shown in Figs. 5.10- 5.12, where each image patch (21×21) presents bottom-up weights of one neuron in the grid plane. All the salient features are developed through the regulation of the two-way information flows, assisted by the pulvinar supervision without any master feature maps.

Every neuron in V2-L4 perceives a local receptive field with size 21×21. Given input stimuli $\mathbf{x}(t)$ over all the training time t ($t = 1, 2, 3, ...$), we define the *cumulative response-weighted stimuli* (CRS) to visualize the attended local features of neuron i:

$$\mathbf{s}_i(t) = \sum_t \tilde{y}_i(t)\ \mathbf{x}(t) \tag{5.8}$$

Fig. 5.13-5.15 visualizes neurons' CRS in all the 3 depths ($d = 1, 2, 3$) of V2-L4, showing that every neuron in one depth detects a specific object ("what") feature in a specific position ("where"), given its local receptive field.

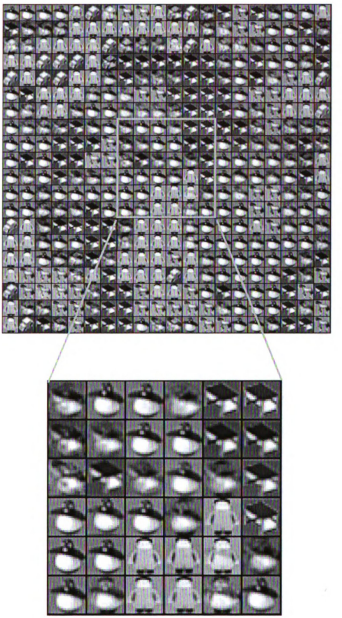


Figure 5.10. Bottom-up weights of 20 × 20 neurons in the first depth of V2-L4.

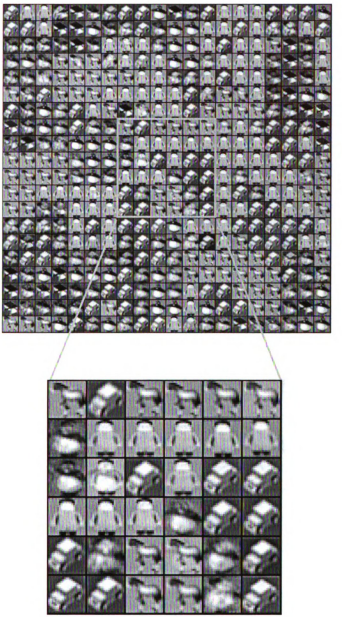


Figure 5.11. Bottom-up weights of 20×20 neurons in the second depth of V2-L4.

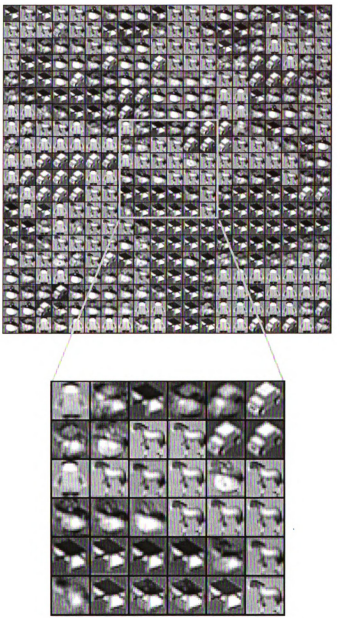


Figure 5.12. Bottom-up weights of 20×20 neurons in the third depth of V2-L4.

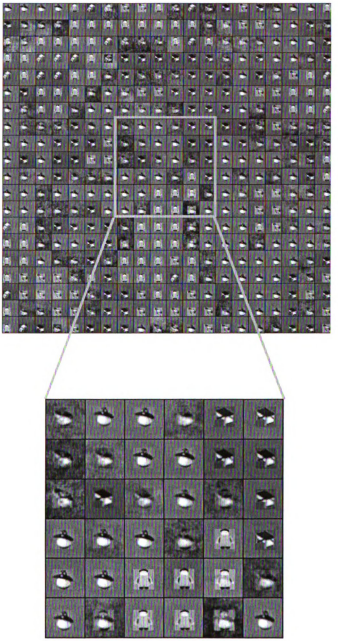


Figure 5.13. Cumulative repones-weighted stimuli of 20×20 neurons in the first depths of V2-L4. Each image patch (40×40) presents the CRS of one neuron in the grid plane.

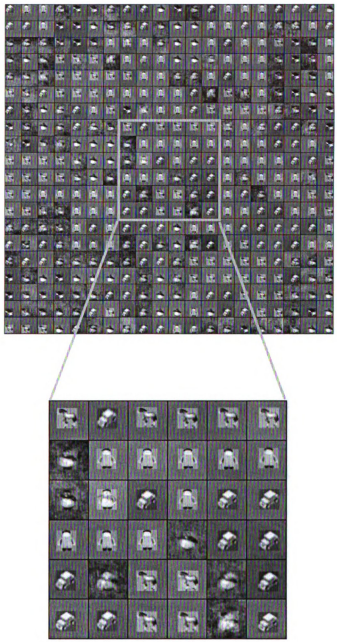


Figure 5.14. Cumulative repones-weighted stimuli of 20 × 20 neurons in the second depths of V2-L4. Each image patch (40 × 40) presents the CRS of one neuron in the grid plane.

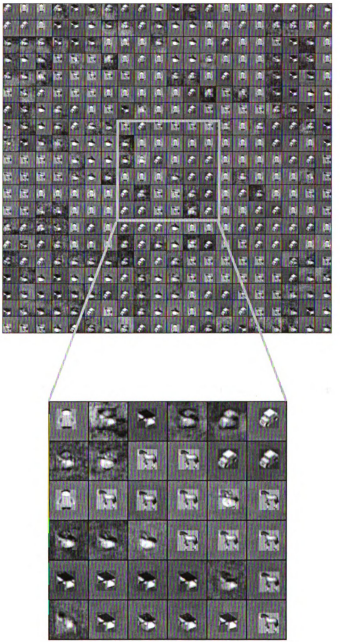


Figure 5.15. Cumulative repones-weighted stimuli of 20×20 neurons in the third depths of V2-L4. Each image patch (40×40) presents the CRS of one neuron in the grid plane.

5.8.2 Top-down Controls From Motor

Top-down connections propagate motor representations through the dorsal and ventral pathways. Fig. 5.16 shows top-down weights in the PP cortex. The PM cortex drives the top-down connections to guide the attention of PP neurons on one or more positions, shown as the white or gray pixel(s) in each image patch of Fig. 5.16. Fig. 5.17 shows top-down weights in the IT cortex. Accordingly, the TM cortex drives the top-down connections to guide the attention of IT neurons on one or more objects, shown as the white or gray pixel(s) in each image patch of Fig. 5.17. The experiment demonstrates that the emergent internal presentations provide position-based and object-based top-down attentions in the pre-motor cortices, as discussed in Sec. 5.5.2.

To evaluate the empirical "probability" of neural firing across classes, we define:

$$p_i = \frac{n(i)}{\sum_1^c n(i)} \qquad i \in 1, 2, ..., c \tag{5.9}$$

where $n(i)$ is the winning age of a neuron fired on a motor class i.

Fig. 5.18 shows class maps of the PP and IT cortex. At each neuron position, a color indicates a motor holding the largest empirical "probability" p_i. There are 5 colors in total for "what" classes and 20×20 for "where" classes. As shown in Fig. 5.18 and discussed in Sec. 5.7.1, neurons that represent the same class tend to group together when the number of available neurons are lager than the number of classes, e.g., IT area has 20×20 neurons for 5 "what" classes. The large number of neurons in IT

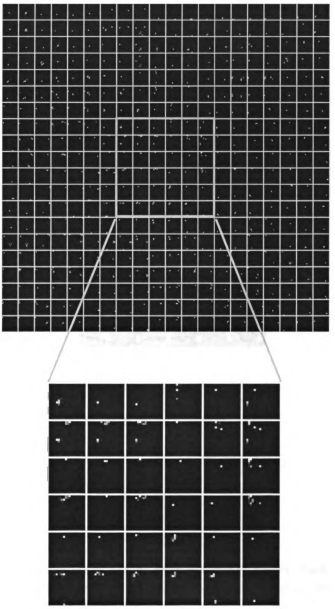


Figure 5.16. Top-down weights of 20×20 neurons in the PP cortex. Each image patch (20×20) in the PP cortex presents a top-down weight from PM input, paying attention to a position (or positions) highlighted by the white or gray pixel(s).

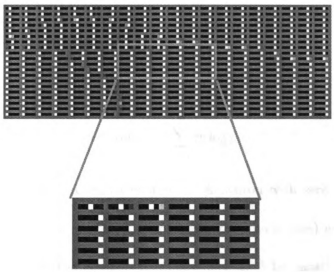


Figure 5.17. Top-down weights of 20×20 neurons in the IT cortex. Each image patch (1×5) in the IT cortex presents a top-down weight from TM input, paying attention to an object (or objects) indexed by the white or gray pixel(s).

area is designed to handle with-in-class variation of objects, which is beyond the scope of this work and being addressed as our on-going research. In principle, the WWN is extendable to handle any type of sensory invariance, using both efficient (not wasting the available resource) and effective (leading to good performance) emergent internal representations through the topographic class grouping.

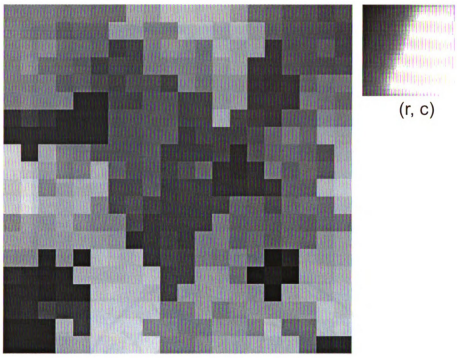
We also define the "purity" of the probability distribution

$$\text{entropy} = -\sum_{i=1}^{n} p_i \log p_i.$$

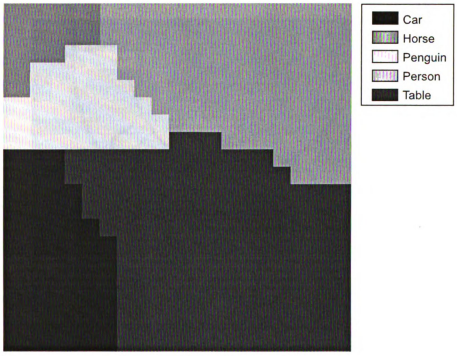
to measures how class-selective the neuron is. A neuron with zero entropy fires for stimuli from the same class, while a neuron with maximum (one) entropy fires with equal probability for all the classes. As shown in Fig. 5.19, most of the developed neurons in the pre-motor cortex (IT and PP area) have **low entropy**, indicating the purity of neurons in their motor perspective. In other words, meaningless pixels values in the images (iconic inputs) are transformed into meaningful abstract regions in the internal cortical layer representation, where each class region represents a meaning of one motor.

5.8.3 Position Invariance & Object Invariance

The CRS in Fig. 5.20 and Fig. 5.21 visualize object invariance in PP cortex ("where" pathway) and position invariance in IT cortex ("what" pathway), respectively. Limited neuron resources are recruited to handle the large variance of input stimuli (20×20

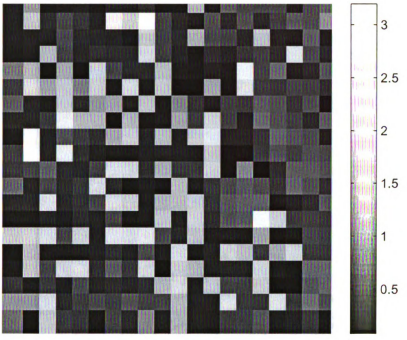


(a)

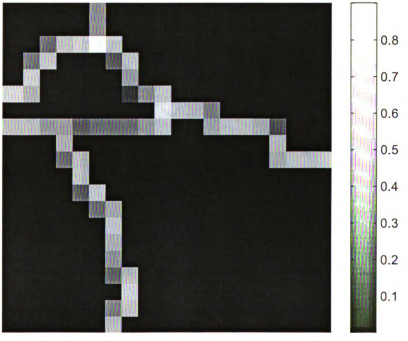


(b)

Figure 5.18. 2D class maps of 20×20 neurons in the (a) **PP cortex and** (b) **IT cortex. Each** neuron is associated with one color, presenting a class (either "where" or "what") with the largest empirical "probability" p_i.



(a)



(b)

Figure 5.19. 2D class entropy of 20×20 neurons in the (a) PP cortex and (b) IT cortex. Each neuron is associated with one color to present its entropy value.

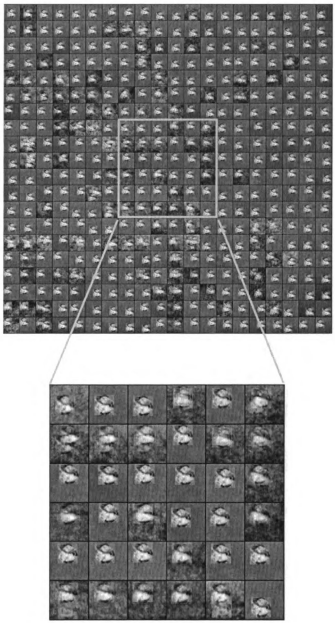


Figure 5.20. Cumulative repones-weighted stimuli of 20×20 neurons in the PP cortex. Each image patch (40×40) presents a CRS showing an object-invariant position pattern, where one position contains 5 overlapped objects with smoothed backgrounds.

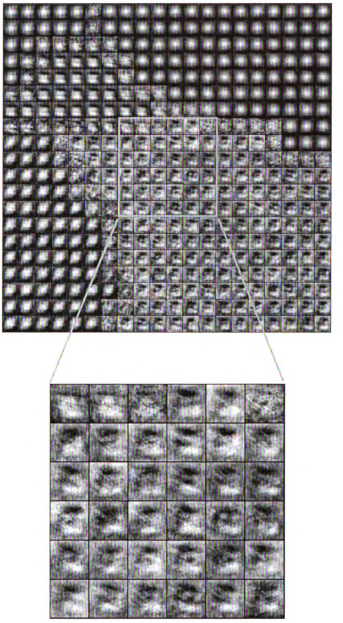


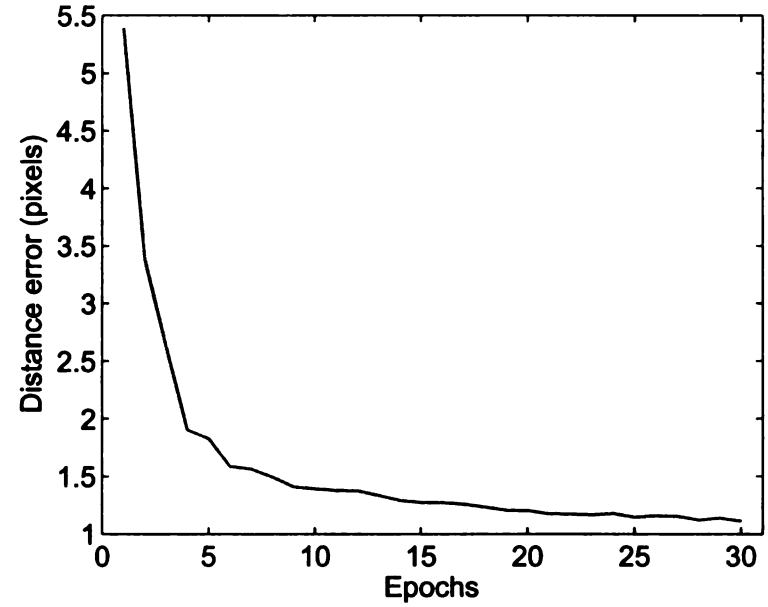
Figure 5.21. Cumulative repones-weighted stimuli of 20×20 neurons in the IT cortex. Each image patch (40×40) presents a CRS showing an position-invariant object pattern, where the same object is averaged over multiple positions.

positions $\times 5$ object types $\times 30$ epochs). As discussed in Sec. 5.5.1, object-invariant PP features are developed to detect one position over multiple object types. The position-invariant IT features are developed to recognize one object over multiple positions.

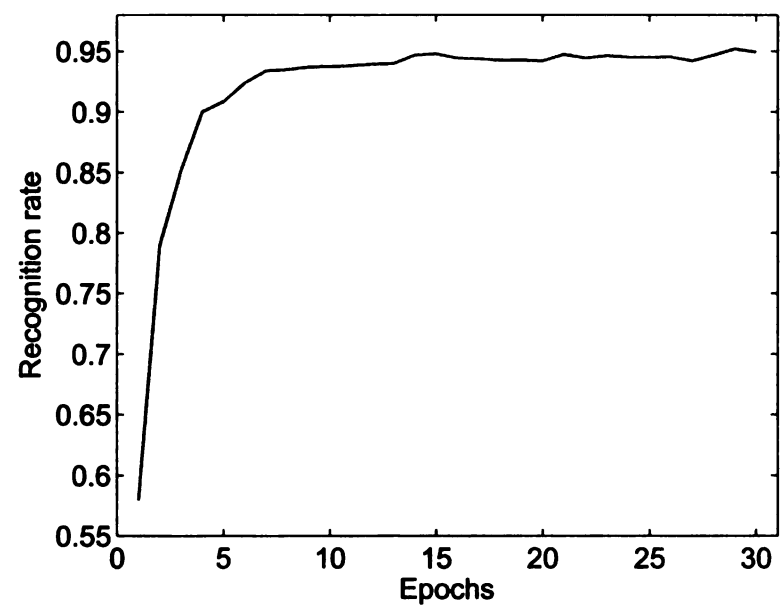
5.8.4 Performance Evaluation

The WWN conducts the described incremental online learning with 30 epochs. After the accomplishment of each epoch, top-down supervision is suspended to "freeze" the network learning, where the network uses bottom-up features to perform with $20 \times 20 \times 5$ image stimuli, covering every object in every position, with different backgrounds from training. The "where" pathway is evaluated by the mean of distance errors for attended positions, while the "what" pathway is measured by the recognition accuracy. The network performance with running epochs is summarized in Fig. 5.22.

Regardless of the large motor variations ($20 \times 20 \times 5 = 2000$), the object recognition accuracy reaches 95% and the average error in position motor outputs decreases to around 1.1 pixels. This is a very impressive result for the challenging task as 75% of the image area of every input image are presented with new natural backgrounds and their locations are unknown. This is the first time that a network learns not only variation within the receptive field of every neuron, but also generalizes across all the receptive fields to reach total positional invariance without a need to hand-program shift-invariance (which cannot handle attention).



(a)



(b)

Figure 5.22. Network performance with running 30 epochs. (a) "Where" motor – PM. (b) "What" motor – TM.

5.9 Summary

The Where-What Network models the brain's visual dorsal ("where") and ventral ("what") pathways using a cortex-inspired architecture, where three types of attention: feature-based bottom-up attention, position-based top-down attention, and object-based top-down attention are integrated together. In contrast with conventional views, the bottom-up saliency in the network is a result of past experience — a combination of image stimuli, pulvinar and top-down flows. The external motors pass top-down enforcement signals to the early cortices, regulating position-based and object-based attentions through the development of invariant "where" and "what" features.

In principle, any type of sensory invariance can be automatically learned by the framework introduced here, including object position, object type, viewing angle, size, lighting, various deformations. The work here only reports preliminary success in 1-pixel sifting of object positions and 5 different objects. Other variations are the subjects of future studies. Computationally, this means that invariant cognition can arise from consistent time proximity between each sensory input and the corresponding motor action. Although sensory inputs are iconic (meaningless pixels), the action is often abstract as along as the action involves the same set of muscles or same cluster of "muxels". This motor-based abstraction is guaranteed in an human society, as any abstract concept that are communicable among humans can be motorized in a human language – verbalized, written, or signed.

CHAPTER 6

Conclusions

This dissertation presented the on-going research on autonomous mental development, aiming at the goal to enable machines to develop internal representation and the associated thinking skills through interactions with the environments.

Based on the concepts and computational aspects of AMD presented in Chapter 1, we proposed a developmental architecture to develop two types of behaviors (covert or overt) in the challenging task of vision navigation. A motivational system is conducted to handle both behaviors through an interactive integration of reinforcement learning and supervised learning. Locally Balanced Incremental Hierarchical Discriminant Regression (LBIHDR) Tree is developed as a cognitive mapping engine to automatically generate internal representations for active vision development, without a need of human programmers to pre-design task-specific (symbolic) representation. Vision-based outdoor navigation is demonstrated as a challenging task example to evaluate the motivational system and cognitive mapping engine. In the experiment, the mean square

error of heading direction is 0^o for re-substitution test and 1.1269^o for disjoint test, which allows the agent to drive without a big deviation from the correct path that we expected.

Due to the limitations that face the above cognitive mapping engine for AMD, we introduced new internal representations based on the biological motivation of lateral inhibition, hebbian learning, and genomic equivalence principles. Through theoretical analysis and computational simulations, the cell-centered in-place sensorimotor pathway shows: (a) how the integration of multiple sensory modalities enables saliency-based attention; (b) how the orientation selective features are developed in the early cortex to provide a sparse coding of salient areas; (c) how the synaptic bottom-up, top-down and lateral connections enable the sensorimotor pathway to discover and recruit cortical neurons to represent optimal features that are invariant to distracters. The reported work was evaluated in the natural driving environments, and cross validation shows the ability to correctly recognize above 95% objects with 10 different city and highway sequences.

In addition to the modeling of recognition capability, the in-place sensorimotor pathway is extended to integrate both attention and recognition through ubiquitous internal actions, which take place virtually everywhere in the brain's cortical signal processing and thus sequential decision making. The general-purpose visuomotor network, called Where-What Network, developed three types of attention: feature-based bottom-up

attention, location-based top-down attention, and object-based top-down attention, where local-to-global invariance from early to later processing, through multi-scale receptive fields. The Where-What Network was evaluated by the images of objects in natural complex backgrounds, reaching 95% correct recognition rate for 5 objects (foreground images) and an average of 1.1 pixels in position error after 30 epochs of training.

The further efforts on this dissertation involve a series of challenges, e.g.,

1. Generalization of in-place where-what pathway to handle object attention and recognition with variable sizes, and with-in-class variations.

2. Computational modeling of the context-based temporal information in order to build the AMD architecture in the time-space domain. The temporal structure of neural activities is extremely important for the functional mental development in the brain. Unfortunately, this key component is missing in our current models.

BIBLIOGRAPHY

[1] S. Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 1977.

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

[3] J. J. Atick and A. N. Redlich. Towards a theory of early visual processing. *Neural Computation*, 2:308–320, 1990.

[4] G. Backer, B. Mertsching, and M. Bollmann. Data- and model-driven gaze control for an active-vision system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(12):1415–1429, December 2001.

[5] C. Balkenius and N. Hulth. Attention as selection-for-action: a scheme for active perception. In *Proceedings of 3rd European Workshop on Advanced Mobile Robots*, Bari, Italy, 1999.

[6] R.G. Vautin B.M. Dow, A.Z. Snyder and R. Bauer. Magnification factor and receptive field size in foveal striate cortex of the monkey. *Exp. Brain Res.*, 44:213, 1981.

[7] C. Breazeal. Social constraints on animate vision. In *Proc. IEEE International Conf. on Humanoid Robots*, Cambridge, Massachusetts, September 7-8 2000.

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

[9] J. Bravo M. Harmon C. Bandera, F. Vico and L. Baird. Residual q-learning applied to visual attention. In *Proc. of the 13th Int'l Conf. Machine Learning*, Bari, Italy, 1996.

[10] A. Revel C. Joulian, P. Gaussier and B. Gas. Learning to build visual categories from perception-action associations. In *Proc. IEEE Intl Conf. Intelligent Robots and Systems*, Sep. 1997.

[11] E. M. Callaway. Local circuits in primary visual cortex of the macaque monkey. *Annual Review of Neuroscience*, 21:47–74, 1998.

[12] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, volume 13, pages 409–415, Cambridge, MA, 2001.

[13] J. Coelho, J. Piater, and R. Grupen. Developing haptic and visual perceptual categories for reading and grasping with a humanoid robot. In *Proc. 1st IEEE International Conf. on Humanoid Robots*, Cambridge, Massachusetts, Sept. 7-8 2000.

[14] T. M. Collection. Learning in pattern recognition. In S. Watanabe, editor, *Methodologies of Pattern Recognition*, pages 111–132. Academic Press, New York, 1968.

[15] DARPA. DARPA grand challenge 2005: Rules. Technical report, DARPA, Oct. 2004.

[16] DARPA. DARPA urban challenge 2007: Rules. Technical report, DARPA, Oct. 2007.

[17] P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* MIT Press, Massachutsetts, 2001.

[18] G. Deco and E. T. Rolls. A neurodynamical cortical model of visual attention and invariant object recognition. *Vision Research*, 40:2845–2859, 2004.

[19] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222, 1995.

[20] G. N. Desouza and A. C. Kak. Vision for mobile robot navigation: a survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

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

[22] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, 2nd edition, 2001.

[23] A. Emami and F. Jelinek. A neural syntactic language model. *Machine Learning*, 60:195–227, 2005.

[24] A. Erisir, S. C. Van Horn, and S. M. Sherman. Relative numbers of cortical and brainstem inputs to the lateral geniculate nucleus. *Proc. of National Academy of Sciences of USA*, 94:1517–1520, 1997.

[25] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[26] P. Foldiak. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

[27] A. Gill. *Introduction to the Theory of Finite-state Machines.* McGraw-Hill, 1962.

[28] S. Grossberg and A. Seitz. Laminar development of receptive fields, maps and columns in visual cortex: the coordinating role of the subplate. *Cerebral cortex*, 13:852–863, 2003.

[29] S. Grossberg and J. R. Williamson. A neural model of how horizontal and interlaminar connections of visual cortex develop into adult circuits that carry out perceptual grouping and learning. *Cerebral cortex*, 11:37–58, 2001.

[30] D. Hebb. *The Organization of Behavior.* Wiley, New York, 1949.

[31] H. Hexmoor, L. Meeden, and R. R. Murphy. Is robot learning a new subfield? The Robolearn-96 Workshop. *AI Magazine*, pages 149–152, Winter 1997.

[32] T. Hong, C. Rasmussen, T. Chang, and M. Shneier. Road detection and tracking for autonomous mobile robots. In *Proc. of SPIE Aeroscience Conference*, Orlando, FL, 2002.

[33] I. Horswill. Polly: Vision-based artificial agent. *Proc. Intl Conf. AAAI*, pages 824–829, 1993.

[34] D. H. Hubel and T. N. Wiesel. Receptive feilds of single neurons in the cat's striate cortex. *Journal of Physiology*, 148:574–591, 1959.

[35] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160(1):107–155, 1962.

[36] C. P. Hung, G. Kreiman, T. Poggio, and J. J. DiCarlo. Fast readout of object identity from macaque inferior temporal cortex. *Science*, 310(5749):863–866, 2005.

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

[38] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2:194–203, 2001.

[39] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 1998.

[40] Z. Ji and J. Weng an D. Prokhorov. Where-what network 1: "Where" and "What" assist each other through top-down connections. In *Proc. IEEE International Conference on Development and Learning*, Monterey, CA, Aug. 9-12 2008.

[41] Z. Ji, M. Luciw, J. Weng, and S. Zeng. A biologically-motivated developmental system for perceptual awareness in vehicle-based robots. In *International Conference on Epigenetic Robotics (EpiRob'07)*, Rutgers, 2007.

[42] Z. Ji, M. D. Luciw, and J. Weng. Epigenetic sensorimotor pathways and its application to developmental object learning. In *IEEE Congress on Evolutionary Computation (CEC'08)*, Hong Kong, 2008.

[43] T. M. Jochem. Using virtual active vision tools to improve autonomous driving tasks. Technical Report Technical Report CMU-RI-TR-94-39, Carnegie Mellon Univ., Oct. 1994.

[44] T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe. Vision-based neural network road and intersection detection and traversal. In *Proc. IEEE Conf. Intelligent Robots and Systems*, volume 3, Aug. 1995.

[45] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

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

[47] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. Appleton & Lange, Norwalk, Connecticut, 3rd edition, 1991.

[48] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, 4th edition, 2000.

[49] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[50] D. LaBerge and M. S. Buchsbaum. Positron emission tomographic measurements of pulvinar activity during an attention task. *Journal of Neuroscience*, 10:613–619, 1990.

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

[52] K. Lee, H. Buxton, and J. Feng. Cue-guided search: a computational model of selective attention. *IEEE transactions on neural networks*, 16:910–924, 2005.

[53] D. Leib, A. Lookingbill, and S. Thrun. Adaptive road following using self-supervised learning and reverse optical flow. In *Proc. of Robotics: Science and Systems*, 2005.

[54] M. Luciw, J. Weng, and S. Zeng. Motor initiated expectation through top-down connections as abstract context in a physical world. In *IEEE International Conference on Development and Learning*, Monterey, CA, Aug. 9-12 2008.

[55] J. L. McCllelland, D. E. Rumelhart, and The PDP Research Group, editors. *Parallel Distributed Processing*, volume 2. MIT Press, Cambridge, Massachusetts, 1986.

[56] B. L. Milenova, J. S. Yarmus, and M. M. Campos. Svm in oracle database 10g: Removing the barriers to widespread adoption of support vector machines. In *Proc. 31st VLDB Conference*, Trondheim, Norway., 2005.

[57] S. Minut and S. Mahadevan. A reinforcement learning model of selective visual attention. In *Proc. of the 5th Int'l Conf. Autonomous agents*, Montreal, Canada, 2001.

[58] S.B. Laughlin M.V. Srinivasan and A. Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proc. R. Soc. Lond. B Biol. Sci.*, 216:427–459, 1982.

[59] E. Oja. *Subspace Methods of Pattern Recognition.* Research Studies Press, Letchworth, UK, 1983.

[60] E. Oja. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 1985.

[61] B. A. Olshausen, C. H. Anderson, and D. C. Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13(11):4700–4719, 1993.

[62] B.A. Olshausen and D.J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–487, 2004.

[63] B. A. Olshaushen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 13 1996.

[64] F. Orabona, G. Metta, and G. Sandini. Object-based visual attention: a model for a behaving robot. In *3rd International Workshop on Attention and Performance in Computational Vision*, San Diego, CA, 2005.

[65] S. Zrehen P. Gaussier, C. Joulain and A. Revel. Visual navigation in an open environment without map. In *IEEE Intl Conf. Intelligent Robots and Systems*, Sep. 1997.

[66] H. Pashler, editor. *Attention.* University College London, London, 1996.

[67] S. Petersen, L. D. Robinson, and W. Keys. Pulvinar nuclei of the behaving rhesus monkey: Visual responses and their modulation. *Journal of Neurophysiologye*, 54:867–885, 1985.

[68] 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, California, 1989.

[69] L. R. Rabiner, L. G. Wilpon, and F. K. Soong. High performance connected digit recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37:1214–1225, Aug. 1989.

[70] R. P. N. Rao and D. H Ballard. Probabilistic models of attention based on iconic representations and predictive coding. In L. Itti, G. Rees, and J. Tsotsos, editors, *Neurobiology of Attention*. Academic Press, New York, 2004.

[71] K. Schill, E. Umkehrer, S. Beinlich, G. Krieger, and C. Zetzsche. Scene analysis with saccadic eye movemens: Top-down and bottom-up modeling. *Journal of Electronic Imaging*, 10(1):152–160, 2001.

[72] E. Seneta. Markov and the birth of chain dependence theory. *International Statistical Review*, 64(3), 1996.

[73] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

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

[75] D. C. Somers, S. B. Nelson, and M. Sur. An emergent model of orientation selectivity in cat visual cortical simple cells. *Journal of Neuroscience*, 15(8):5448–5465, 1995.

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

[77] R.S. Sutton and A.G. Barto. A temporal-difference model of classical conditioning. In *Proc. the Ninth Conference of the Cognitive Science Society*, Hillsdale, New Jersey, 1987. Erlbaum.

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

[79] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, and P. Stang. The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9), 2006.

[80] A. Treisman. Features and objects in visual processing. *Scientific American*, 255(5):114–125, 1986.

[81] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78:507–545, 1995.

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

[83] C. Urmson, J. Anhalt, H. Bae, J. A. Bagnell, C. Baker, R. E. Bittner, T. Brown, M. N. Clark, M. Darms, D. Demitrish, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y. Seo, S. Singh, J. M. Snider, J. C. Struble, A. Stentz, M. Taylor, W. L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar. *Autonomous driving in urban environments: Boss and the Urban Challenge*, 25(1):425–466, June 2008.

[84] S. Vijayahumar and S. Schaal. Real time learning in humanoids: A challenge for scalability of online algorithms. In *Proc. IEEE International Conf. on Humanoid Robots*, Cambridge, Massachusetts, Sept. 7-8 2000.

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

[86] D. Walther and C. Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19:1395–1407, 2006.

[87] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[88] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition and segmentation using the Cresceptron. *International Journal of Computer Vision*, 25(2):109–143, Nov. 1997.

[89] J. Weng and W. Hwang. Incremental hierarchical discriminant regression. *IEEE Transactions on Neural Networks*, 18(2):397–415, 2007.

[90] J. Weng and M. Luciw. Neuromorphic spatiotemporal processing. Technical report, MSU-CSE-08-34, 2008.

[91] J. Weng and M. Luciw. Dually-optimal cortical layers: Lobe component analysis. *IEEE Trans on Autonomous Mental Development*, 1(1), 2009.

[92] J. Weng and M. D. Luciw. Optimal in-place self-organization for cortical development: Limited cells, sparse coding and cortical topography. In *International Conference on Development and Learning (ICDL'06)*, Bloomington, IN, May 31 - June 3 2006.

[93] J. Weng, T. Luwang, H. Lu, and X. Xue. A multilayer in-place learning network for development of general invariances. *International Journal of Humanoid Robotics*, 4(2):281–320, 2007.

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

[95] J. Weng and N. Zhang. Optimal in-place learning and the lobe component analysis. In *Proc. IEEE World Congress on Computational Intelligence*, Vancouver, BC, Canada, July 16-21 2006.

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

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

[98] M. Inaba Y. Matsumoto and H. Inoue. Visual navigation using view-sequenced route representation. *Proc. IEEE Intl Conf. Robotics and Automation*, 1:83–88, 1996.