# Visual Map-Making and Guided Reinforcement Learning for Mobile Robot Navigation: A Neurocomputational Approach

by

Ivan Andrew Bachelder

B.S., Electrical Engineering, Cornell University (1989)
M.S., Computer Science & Engineering, MIT (1991)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

February 1996



Author_____
Department of Electrical Engineering and Computer Science
October 1, 1995

Certified by_____
Allen M. Waxman
Senior Staff, MIT Lincoln Laboratory
Thesis Co-Supervisor

Certified by_____
Nathaniel I. Durlach
Senior Scientist, Electrical Engineering and Computer Science
Thesis Co-Supervisor

Accepted by_____
Frederic R. Morgenthaler
Chairman, Committee on Graduate Students

# Visual Map-Making and Guided Reinforcement Learning for Mobile Robot Navigation: A Neurocomputational Approach

by

Ivan Andrew Bachelder

Submitted to the Department of Electrical Engineering and Computer Science
on November 13, 1995, in partial fulfillment of the
requirements of the degree of
Doctor of Philosophy

## Abstract

A mobile robot that quickly constructs and adapts an environment map from visual input, and uses this map to evaluate the utility of places and routes from rewards and penalties received over time, offers distinct advantages for navigation. It can not only recognize its current place and predict the outcome of locomotive actions, but also learn to avoid penalizing, and favor rewarding, places and actions. A map and an associated evaluation constitute a navigation strategy, which provides a substrate for dynamically planning safe, efficient routes to desirable places, or for generating behavioral policies.

This thesis develops a neurocomputational sub-system for learning navigation strategies within 3D environments defined by spatially distributed sets of visual landmarks. The development exemplifies a cognitivist emulation of cognitive mapping and reinforcement learning in the rat hippocampus. The map-making component, inspired by analogy between learning 3D objects and environment layouts, forms maps similar to aspect graphs using hierarchical, relational, view-based learning principals. It comprises two architectures. The first performs unsupervised place (local region) learning by combining "What" with "Where", namely through conjunctions of landmark identity, pose, and egocentric gaze direction within local, restricted views of the environment. The second associatively learns action consequences by incorporating "When", namely through conjunctions of learned places and coarsely-coded motions. The map-evaluation component of the sub-system consists of a reinforcement learning architecture, which associatively evaluates the utility of each place and action in the learned map. Together, the map and an evaluation constitute a navigation strategy reminiscent of a partially observable Markov decision process, and provide a substrate for intra-place localization, place prediction, environment recognition, route planning, and exploration.

The proposed sub-system and supporting sub-systems for featural grouping, gaze control, object vision, and path planning constitute a fully integrated navigation system. Through a framework called teletraining, wherein an operator selectively guides the exploratory behavior of the robot and issues reinforcement, this system may be trained in an entirely unsupervised fashion to a level of competence appropriate for supervisory control. Results from implementing most of these sub-

systems on the mobile robot called MAVIN (the Mobile Adaptive VIsual Navigator) demonstrate the potential for these capabilities.

Thesis Co-Supervisor: Allen M. Waxman
Title: Senior Staff, MIT Lincoln Laboratory

Thesis Co-Supervisor: Nathaniel I. Durlach
Title: Senior Scientist, Electrical Engineering and Computer Science

## Acknowledgments

I thank Allen Waxman for advising and supporting me and my research at Lincoln Laboratory for the past five years. I have learned a great deal as a scientist and engineer under Allen's direction, have admired his technical ability, his drive, his attention to detail, and the quality of his work, and have greatly appreciated his sincere concern for my professional and personal well-being.

I also thank the other members of my committee, Nat Durlach, Rod Brooks, and Tom Sheridan, for being so accommodating with my somewhat special situation here at Lincoln Laboratory. I especially thank Nat for agreeing to serve as my on-campus advisor.

Much gratitude goes to various members of the Machine Intelligence Technology Group at Lincoln Laboratory. Over the past six years I have been particularly lucky to have befriended many talented, knowledgeable, and intellectually engaging people in a comfortable research environment. I am especially indebted to Richard Lacoss for recruiting me to the group as a Summer Staff member straight out of college, for supporting both me and the mobile robotics effort here at Lincoln Laboratory ever since, and for offering me a number of very generous employment opportunities.

I thank Mike Seibert for entertaining many of my half-baked ideas, for career advice, and for proofing drafts of my papers and this thesis. I thank Rob Cunningham for moral support (as the only other graduate student in the group), for his expert help with countless computer-related problems, for providing imagery, and, of course, for introducing me to ultimate frisbee! I thank Dave Fay for being a great friend both in and outside of work, for moral support, for technical aid with documenting the mobile robot results, and for putting up with my blabbering on from time to time. I thank Alan Gove, Rob Baxter, and Rob Steele for listening to my ideas and presentations during numerous group meetings, and for occasional technical support. I thank many other members of the group for their occasional technical aid, including Kathy Smith, Jim Carrick, John Delaney, Paul Harmon, and Carol Lazott. I especially thank Joe Racamato for being such a whiz with the hardware, and for being so generous with his time when the robot needed repairs or an upgrade. And I would be hard pressed not to include both former and current group secretaries Lori Benjamin and Caryn Cassidy for taking care of all the paperwork. Overall, I have greatly enjoyed and will miss being a part of Group 21 (now group 401).

Of course, these four years have not been easy, and I certainly would not have stuck it out had it not been for the encouragement, love, patience, and support of Robin, my wife. I celebrate her own graduation this year, and look forward to pursing our personal dreams more vigorously in the absence of the Ph.D. time sink.

This work was sponsored in part by the U. S. Air Force of Scientific Research, and the Advanced Research Projects Agency. Opinions, interpretations, conclusions, and recommendations are those of the author, and are not necessarily endorsed by the United States Air Force.

# Dedication

I dedicate this thesis to the memory of my father, Philip I. Bachelder,
a graduate of MIT, an inspirational perfectionist and overachiever,
and a truly great Dad.

While he never imposed his own interests on my career decisions,
he taught me that true happiness and wealth arise naturally
from pursuing the work one enjoys most,
regardless of its outlook for fortune or prestige.

# Biography

## Education

B.S. with distinction, Electrical Engineering 1989
College of Engineering
Cornell University
Ithaca, NY

M.S., Computer Science and Engineering 1991
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
Thesis under Prof. S. Ullman: "Matching Contours Using Local Affine Transformations"

## Career History

Research Assistant/Summer Staff 1989-1996
Machine Intelligence Group
MIT Lincoln Laboratory
Lexington, MA

Teaching/Research Assistant 1989-1991
MIT Artificial Intelligence Laboratory
Cambridge, MA

Cooperative Intern 1987-1988
Digital Systems Laboratory
Raytheon Company
Bedford, MA

## Publications

Bachelder, I. A., & Waxman, A. M. (1995). A View-Based Neurocomputational System for Relational Map-Making and Navigation in Visual Environments. *Robotics and Autonomous Systems*, Special Issue on Perception and Action, in press.

Waxman, A. M., Seibert, M., & Bachelder, I. A. (1995). Visual Processing of Object Form and Environment Layout. In *The Handbook of Brain Theory and Neural Networks* (eds. M. Arbib and P. Arbib), pp. 1021-1024, MIT Press: Cambridge, MA.

Bachelder, I. A. & Waxman, A. M. (1994). Mobile Robot Visual Mapping and Localization: A View-Based Neurocomputational Architecture that Emulates Hippocampal Place Learning. *Neural Networks*, Special Issue: Models of Neurodynamics and Behavior, Volume 7 (6/7), pp. 1083-1100.

Waxman, A. M., Seibert, M., Gove, N., Fay, D. A., Cunningham, R. K., & Bachelder, I. A. (1994). Visual Learning of Objects: Neural Models of Shape, Color, Motion and Space. In *Computational Intelligence: Imitating Life* (eds. J. M. Zurada, R. J. Marks II, and C. J. Robinson), pp. 237-251, IEEE Press: New York, NY.

Bachelder, I. A., Waxman, A. M., Seibert, M., & Gove, A. N. (1994). From Learning Objects to Learning Environments: Biological and Computational Neural Systems. *Proceedings of the ARPA Image Understanding Workshop*, November 13-18, Monterey, CA, pp. 871-883.

Bachelder, I. A. & Waxman, A. M. (1994). A Neural System for Qualitative Mapping and Navigation in Visual Environments. *Proceedings of 'From Perception to Action' (PerAc'94)*, September 7-9, Lausanne, Switzerland, pp. 266-277.

Bachelder, I. A. Waxman, A. M., & Seibert, M. (1993). A Neural System for Mobile Robot Visual Place Learning and Recognition. *Proceedings of 'Artificial Neural Networks In Engineering' (ANNIE'93)*, November 14-17, St. Louis, MO, pp. 343-351.

Bachelder, I. A. Waxman, A. M., & Seibert, M. (1993). A Neural System for Mobile Robot Visual Place Learning and Recognition. *Proceedings of the 1993 World Congress on Neural Networks (WCNN'93)*, July 11-15, Portland, OR, pp. 512-518.

Bachelder, I. A. & Waxman, A. M. (1992). Neural Networks for Mobile Robot Visual Exploration. *SPIE Proceedings 1831, Mobile Robots VII*, November 15-20, Boston, MA, pp. 107-119.

Bachelder, I. A., and Ullman, S. (1992). Contour Matching Using Local Affine Transformations. *Proceedings of the 1992 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'92)*, June 15-18, Champaign, IL, pp. 798-801.

Bachelder, I. A., & Ullman, S. (1992). Contour Matching Using Local Affine Transformations. *Proceedings of the Image Understanding Workshop*, January 26-29, San Diego, CA, pp. 299-310.

## Honors & Awards

Novel Engineering Applications, First Runner Up
Artificial Neural Networks In Engineering (ANNIE'93) Conference
November 14-17
St. Louis, MO

Dean McMullen Scholar
Cornell University

# *Contents*

# *List of Figures*

# *List of Tables*

# CHAPTER 1 *Introduction*

Autonomous systems, whether biological creatures or mobile robots, benefit from the ability to learn internal representations of their environment by integrating the sensory information gathered incrementally from the world around them. Environment representations allow autonomous systems to reason about the future effects of their actions, to evaluate the relative utility of situations and actions, and to plan appropriate courses of action accordingly in order to achieve their goals as efficiently as possible.

The information embodied by an internal representation of an environment and its evaluation can be thought of as a *strategy* for performing a particular task. As defined in this thesis, a strategy consists of a *task-independent model* of an environment, together with a *task-dependent evaluation* over that model (see Figure 1). One task-independent model is a representation which defines the perceptual states (situations) of the environment and the relationships between these states with respect to actions and events. Note that the term *task-independent* implies that the representation is independent of the particular tasks (e.g. delivering papers), though not necessarily the *types* of tasks (e.g. navigation), that it might be used to perform. One task-dependent evaluation is a relative utility assessment of each of the states and actions with respect to some task performance criteria, such as survival. Under these definitions, a strategy provides the computational framework for achieving a variety of goals optimally through the generation of an *action policy*, a mapping from situations to actions, which, when followed, produces a goal-seeking *behavior*. The act of learning a strategy might be accomplished either by exploring and directly interacting with the environment, or through the guidance of a teacher.

This very general notion of *strategy learning* (SL) might apply to many different types of environments, tasks, and goals. This thesis specifically describes the development of a system that learns and applies strategies for *visual navigation*; that is, a system that learns, perhaps under the general guidance of a trainer, how to perform navigation tasks in a large-scale environment defined by a spatial arrangement of visual landmarks.

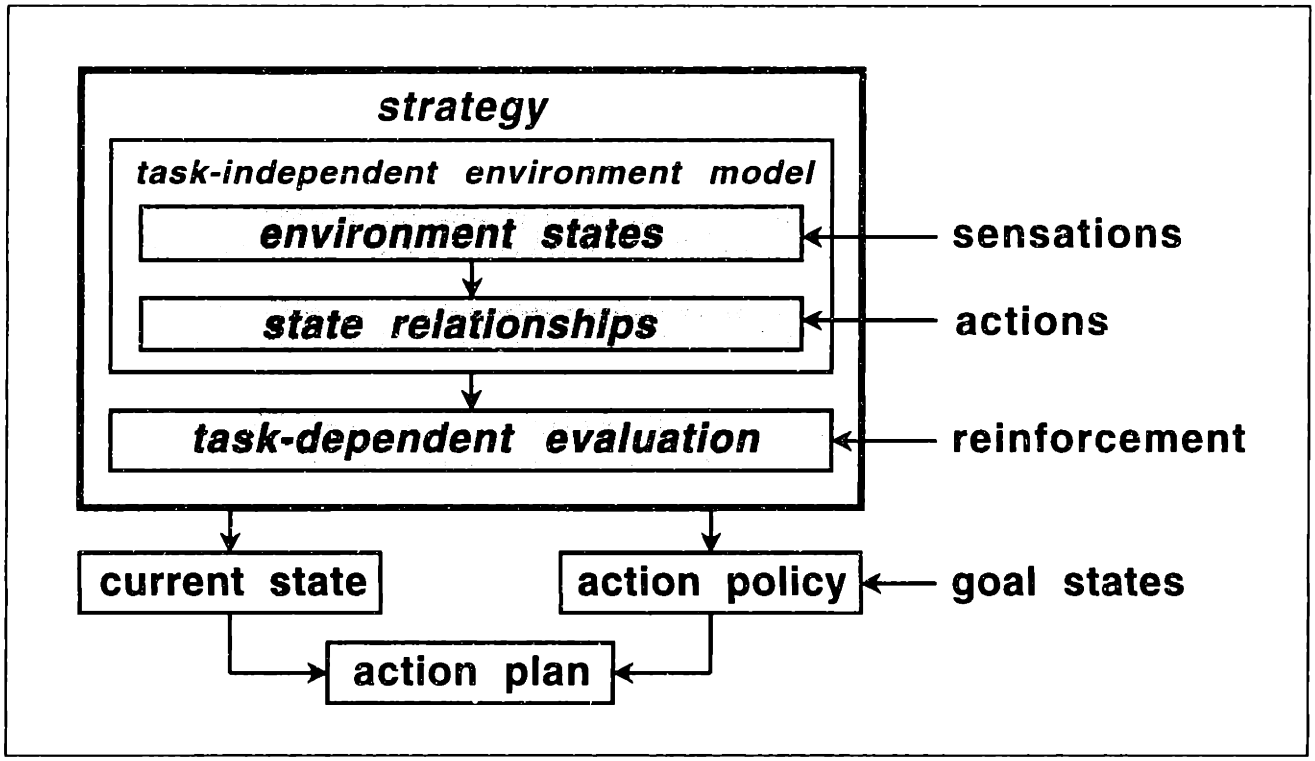


**FIGURE 1: THE CONCEPT OF LEARNING AND UTILIZING A STRATEGY.**
*A strategy consists of a task-independent model of an environment, composed of a set of learned environment states and their relationships, and a learned task-dependent evaluation over that model. Given a set of goals, a strategy might be used to compute an action policy. Given also the current situation, as defined by the model, the action policy might then be used to compute an action plan. (Note: Shaded boxes denote where learning occurs.)*

# 1.1 Strategy Learning for Visual Navigation

## 1.1.1 Statement of the Problem

The problem of learning strategies for visual navigation concerns the processes of visually assessing the positions and appearances of individual landmarks, incrementally learning and adapting a task-independent *map* of the environment from sensory (landmark) and motor perceptions, and evaluating the utility of places and actions defined by the map with respect to task-dependent feedback received either from the environment or from a teacher (see Figure 2).

A creature or mobile robot that can perform these processes can recognize its current *place* in the environment and predict the outcome of its *locomotive actions*. It can also learn to avoid certain places and actions because they are penalizing, or favor them because they are rewarding. The *navigation strategy* defined by this map and its associated utility values provides the substrate for dynamically planning safe, efficient *routes* to a variety of desirable places, and for generating appropriate behavioral policies for navigating to those places in the future. The following example illustrates this basic idea.

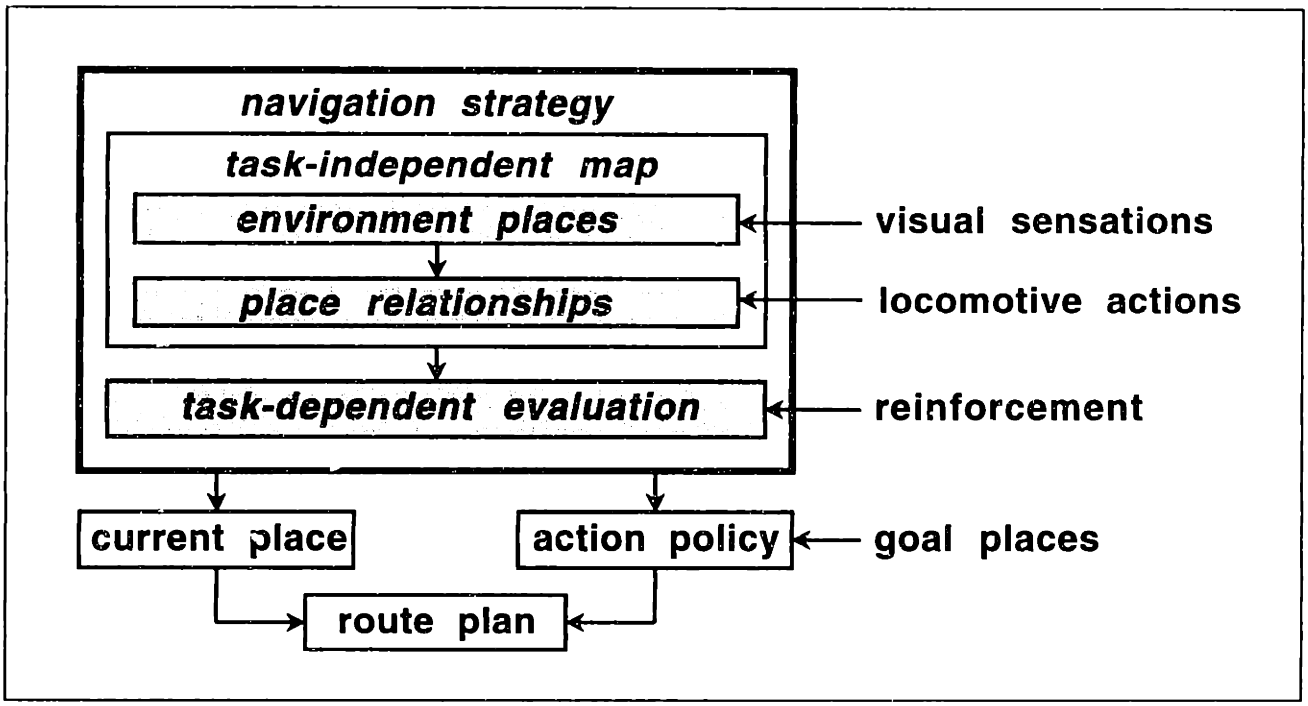


**FIGURE 2: THE CONCEPT OF LEARNING A STRATEGY FOR NAVIGATION.**
*A navigation strategy consists of a task-independent map of a visual environment, consisting of a set of places and their relationships, and a task-dependent evaluation over that map. Given a set of goal places, a navigation strategy might be used to compute a locomotive action policy. Given also the current place, as recognized using the map, the locomotive action policy might then be used to compute a route plan (cf Figure 1). (Note: Shaded boxes denote where learning occurs.)*

### 1.1.2 An Example: The "Paperboy" Scenario

Consider the task of a paperboy delivering both the daily and Sunday newspapers on his bike to various customers in a suburban neighborhood. Every day he must travel to all of the customers' houses and deposit a newspaper, while at the same time avoiding unfriendly dogs and keeping away from busy roads. He usually wishes to perform this task as quickly as possible, especially on weekdays when his friends are available after school. In the winter, he often has difficulty seeing because the sun sets before he has finished delivering. In the summer, however, he sometimes has spare time to explore new areas and features of the neighborhood, and occasionally finds a short-cut or two. Over the long term, customers periodically move in and out of the neighborhood, new roads, houses, and street signs are constructed, traffic patterns change, and unfriendly dogs move in or out with their owners.

Under these circumstances, it might be best for the paperboy to draw a map of the territory as he encounters it on his delivery route or during his explorations. On this map he could indicate the locations of the roads and landmarks (houses, trees, street signs, etc.), and could correct for gradual changes in these locations as he discovers them. He could then use the map to figure out his current position in the neighborhood, simply by matching the landmarks he sees while riding down the street with what the map tells him he might expect to see. When the sun sets, he could

keep from getting lost by following the map using a flashlight. He could also pencil in the places of unfriendly dogs and indicate the traffic level on each of the roads. This record would then allow him to chart and follow a relatively safe, efficient delivery route each day, and update this route as new information arrives.

At first, the paperboy would have difficulties delivering because his map would contain very little information. If he has only the addresses of each of his customers, he would have to explore extensively before he finds their homes. He would also run into many dogs and have to cross many a busy road. Note, however, that he may be able to convince the previous paperboy to hand over his map, and to perhaps guide the new paperboy to the customers' houses and warn him about the dogs and busy roads along the way.

Note that the paperboy's map is independent of any of the particular navigation tasks that he might use it to perform. In this particular case, his task is to deliver all the papers while avoiding nasty dogs and busy roads. But if he instead wanted to drive his car to a friend's house in the neighborhood, then nasty dogs and busy roads make little or no impact, and the task might become one of, say, travelling to a particular house while avoiding dirt roads and traffic lights. In either case, the task-independent map remains the same, while the task-dependent evaluation of each of the places and roads in the map changes. For any particular task, however, the map and its evaluation constitutes a strategy, which is independent of the goals it might be used to achieve. The task of paper delivery constitutes a delivery strategy, while the task of driving to a friends house constitutes a driving strategy. For the paper delivery task, the goals on the weekdays and on Sunday differ, since the Sunday paper customers differ from the daily customers. Similarly, the goals might differ for the task of driving to a friend's house, depending on which friend the paperboy wishes to see. In all of these cases, however, the goal-independent strategy remains the same, even though the goals change.

Of course, most paperboys do not usually employ maps of their neighborhoods for the purpose of every day delivery, at least not beyond the early days of their service, presumably because they can *learn* and remember what they have seen. However, it is conceivable that what they learn resembles such a map and its evaluation. The acts of drawing the map and indicating task-specific locations of the neighborhood are analogous to learning an internal task-independent model of an environment and its task-dependent evaluation, respectively, while the act of correcting (erasing and redrawing) this map is analogous to *adapting* the learned internal model and its evaluation.

## 1.2 Technical Requirements and Objectives

As previously mentioned, the objective of this thesis is to develop an integrated system for learning and using navigation strategies under the guidance of a teacher. The general requirements of this system pertain to *landmark vision*, *map construction and application*, and *map evaluation*.

### 1.2.1 Landmark Vision

Landmark vision refers to the sensory aspect of visual navigation. It entails finding, tracking, and extracting features from visual landmarks in the environment. It also entails processing the extracted visual features to create an input suitable for robust map-making. The paperboy in the scenario described above must, for example, be capable of visually locating houses, signposts, and so on, and tracking these landmarks long enough to assess their appearance and relative spatial positions. The proposed system must utilize landmark features that exhibit invariance (or at least an insensitivity) to viewing conditions, such as lighting, orientation, size, and perspective distortion. For example, the paperboy must assess the relative positions and appearances of houses and street signs regardless of whether it is high noon or twilight, or if it is raining or snowing, and whether they are a few yards away, or a few football fields away.

### 1.2.2 Map Construction and Application

Map-making entails both the use of sensory (landmark) information to learn and recognize places in the environment, and the use of motor (locomotive action) information to learn the spatial relations between places. The paperboy, for example, can visually recognize the street or intersection he sees from the spatial arrangement of houses and trees, and can learn to get from one place to the next by riding his bike between them. The learned map must also be amenable to environment recognition and route planning. The paperboy, for example, is capable of recognizing the neighborhood through which he current rides by matching expectations derived from the map with what he sees, and can also use the map to plan a route to a particular house within that neighborhood.

### 1.2.3 Map Evaluation

Map evaluation refers to assessing the utility of each of the places and locomotive actions with respect to the rewards and punishments administered by an external teacher, supervisor, society, or the environment itself, and using this learned utility to plan efficient routes. The paperboy can, for example, learn through experience just how dangerous each of the intersections and roads are by witnessing the occurrence of nasty dogs or heavy traffic, and can plan better routes to avoid these situations in the future.

## 1.3 Technical Approach

The approach taken in *designing* the proposed system is driven by arguments for *biological emulation*. The approach taken in *evaluating* the system is driven by arguments for *real-time demonstration on a mobile robot.*

### 1.3.1 Emulating Visual Learning in Biological Organisms

Much evidence suggests that biological organisms construct strategies for performing tasks in visual domains using *hierarchical*, *relational*, *view-based* (HRV) learning principals. These principals are characterized by the following mechanisms:

1. A *self-organization* (automatic categorization) of the visual entity (e.g. object, environment, etc.) into perceptual *categories*. Each category is view-based in that it is represented by a *viewer-centered* (egocentric) 2D spatial pattern called a *characteristic view*, which is invariant to a variety of visual transformations.
2. A learning of the qualitative relationships (i.e., *transitions*) between each of the learned view categories. These relations might be "tagged" by (associated with) categorized operative actions or visual events.
3. A pooling of view categories with similar properties and their transitions to form more general *object-centered* (allocentric) categories, such as object and environment categories. This pooling defines a hierarchical structure, and completes the task-independent model of the environment.
4. A learning of the utility of each of the view, action, and hierarchical categories through direct association. These associations are learned during reinforcing events such as the issuance of rewards and punishment by external agents.

Most of these general mechanisms are particularly prevalent for visual object learning and recognition in the temporal lobe of the macaque monkey, and also for visual navigation in the hippocampus of rats. Arguably, autonomous robotic systems stand to gain much by emulating the neural structures and principals that might be responsible for these HRV processes, since they are adaptive, consistent, and efficient (Perrett & Oram, 1994). *The main objective of this thesis is to demonstrate that such biological emulation at the neurocomputational level can achieve a viable artificial system for visual navigation.* Here, neurocomputational emulation refers to employing the computational principles that underlie biological learning, primarily at the neural network (neurophysiological) and architectural (anatomical) levels (as opposed to the behavioral or psychophysical levels), in order to demonstrate potential engineering advantages over purely computational approaches. It does not necessarily imply that the proposed system will accurately model biological systems, nor that its performance will exceed that of existing navigation systems.

The concept of HRV learning for navigation is illustrated in Figure 3. In realizing this HRV learning, the proposed navigation system (see Figure 4) is motivated by a vast body of scientific evidence for HRV learning in biological organisms, and by studies of the effects of guidance on task training in humans. The main component of this system is a *navigation strategy learning sub-system* (NSLS). In accordance with the HRV learning principals noted above, this sub-system comprises a *place learning architecture* (PLA) that self organizes invariant environment views into *place categories*, an *action consequence learning architecture* (ACLA) that learns the transitions between places in terms of actions and forms an environment hierarchy, and a *reinforcement learning architecture* (RLA) that associates utility with each of the places and actions in the map based on reinforcement signals.

### 1.3.2 Real-time Demonstration on a Mobile Robot

In order to verify that the proposed system is viable for autonomous navigation, one must show that it actually works. Its main sub-systems are therefore implemented, and their navigation capabilities demonstrated, on the mobile robot MAVIN (Mobile Adaptive VIsual Navigator — see Figure 5 and Appendix A) in a simple laboratory environment. These sub-systems include the

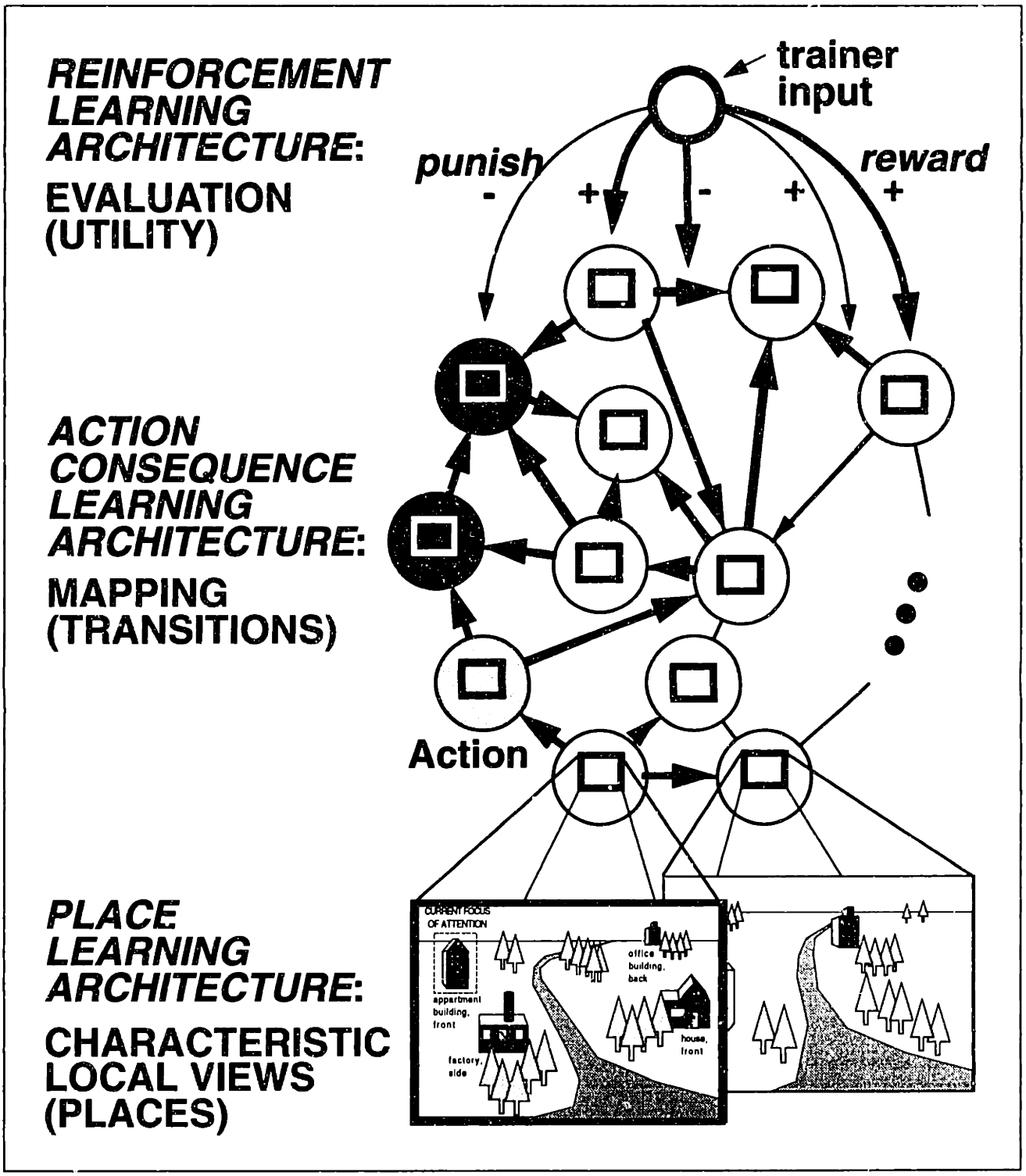


**FIGURE 3: HIERARCHICAL, RELATIONAL, VIEW-BASED STRATEGY LEARNING FOR NAVIGATION.** *The proposed navigation system employs three modular neural architectures: the place learning architecture (PLA), action consequence learning architecture (ACLA), and reinforcement learning architecture (RLA). These architectures respectively learn characteristic views of the environment (places), transitions between views associated with locomotive actions (to create a map), and a direct evaluation of each of the places, actions, and transitions in the map (utility) based on punishments and rewards received from a trainer.*

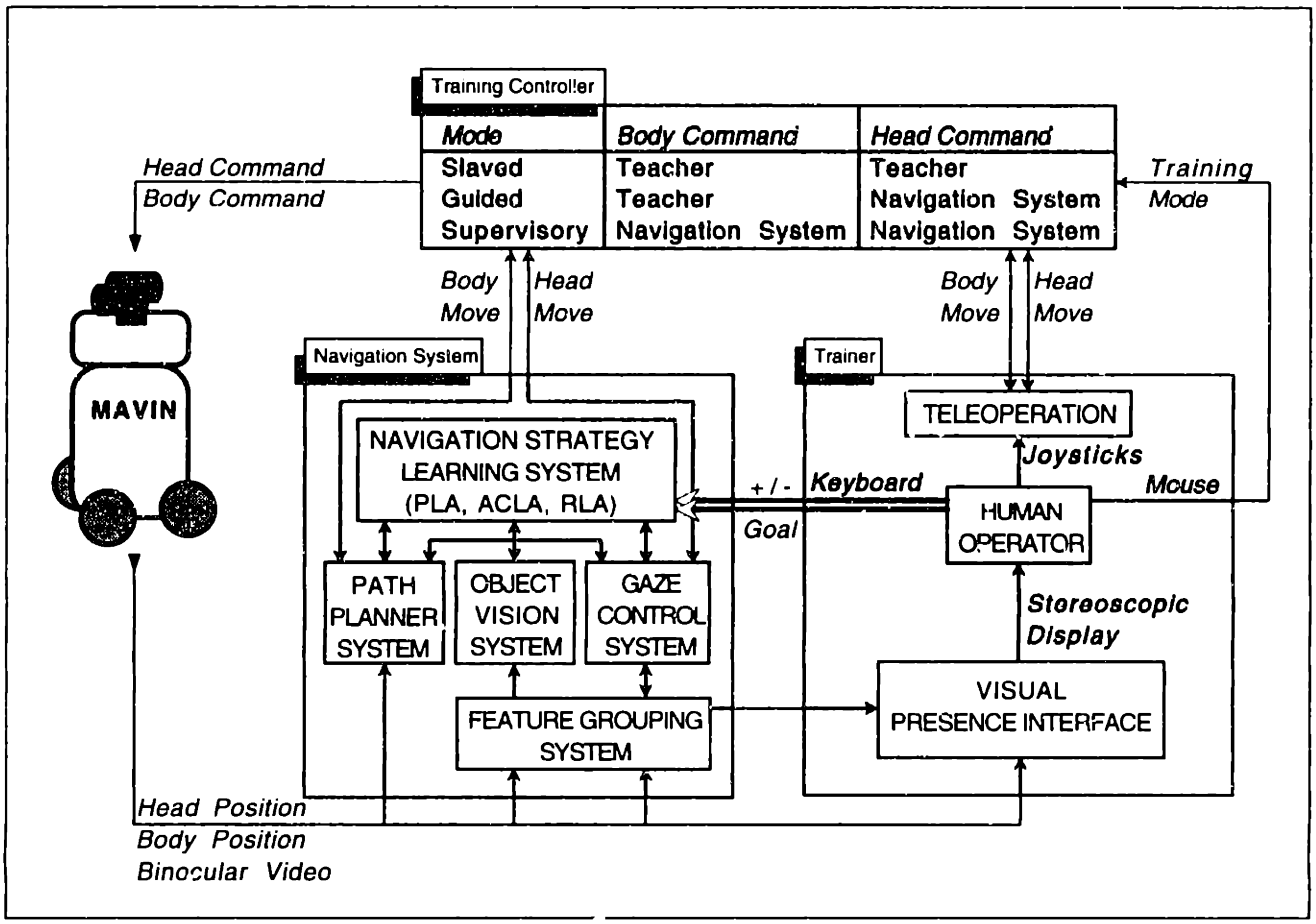


**FIGURE 4: PROPOSED FRAMEWORK FOR TEACHING NAVIGATION STRATEGIES VIA TELETRAINING.**

NSLS (described in the previous section), a *gaze control sub-system* (GCS), and an *object vision sub-system* (OVS). (Though not implemented, a *featural grouping sub-system* (FGS) and a *path planning sub-system* (PPS) are also discussed for the sake of completeness). Using a technique called *teletraining*, the fully integrated system can then be trained to perform various navigation tasks in this environment by a human operator. In several phases of this training procedure, the operator analogically guides MAVIN through the task environment via direct *teleoperation* (remote control) with the aid of *telepresence* (the subjective feeling of presence at the robot's location), and indicates the places of rewards and penalties while the robot learns a map in an unsupervised fashion. Subsequently, the operator merely looks on and issues reinforcement in order to make corrections.

This empirical verification (testing) of a neurocomputational system on a robot is more convincing than simply making theoretical arguments or simulating the system on a computer terminal. It illuminates the problems associated with applying the proposed system in real-time, on a real mobile robot, and in the (simplified) physical world, thereby preventing the designer from subconsciously making implicit assumptions. The designer must either knowingly simplify the implementation to deal with difficult problems and explicitly document these simplifications, or expand the system to effectively handle such problems.

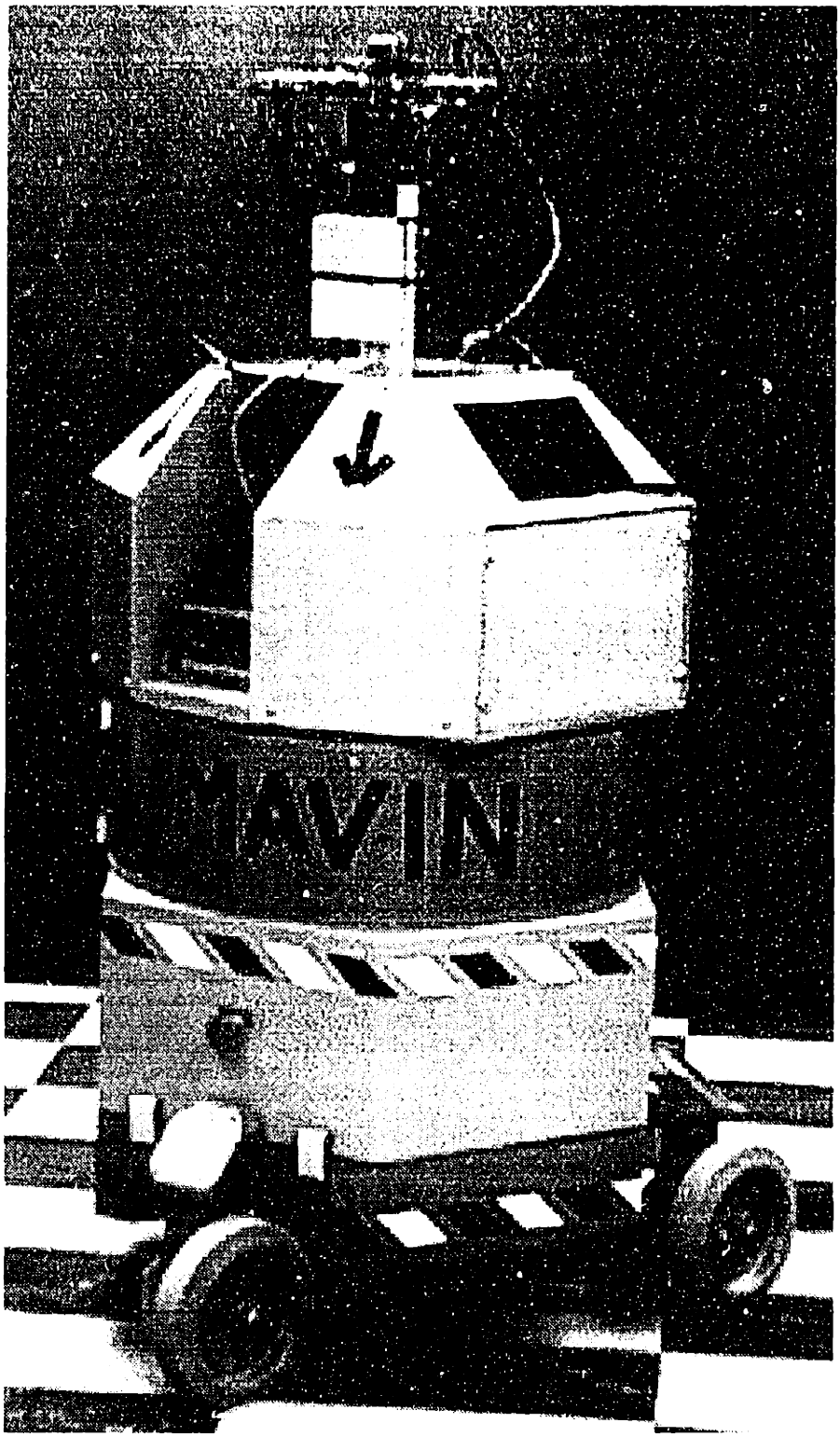


**FIGURE 5: THE MOBILE ADAPTIVE VISUAL NAVIGATOR (MAVIN).**

*MAVIN (the Mobile Adaptive VIsual Navigator) consists of a mobile platform, upon which sits a pan-tilt unit serving as a neck for a binocular vergence apparatus for two CCD cameras.*

## 1.4 Applications and Contributions

On a pragmatic level, the proposed system has a number of possible applications. In addition to environment map-making and navigation, general application areas for the HRV approach to strategy learning include visual object recognition and other higher level tasks like visually-guided manipulation and assembly. Specific applications for visual navigation include exploration, construction, and resource extraction in space, undersea, subterranean, and military environments. In particular, the proposed system might be used to learn and perform navigation-intensive competencies for semi-autonomous mobile robots in complex and hazardous scenarios like reconnaissance operations and nuclear waste cleanup.

On a scientific level, the proposed system makes significant contributions in the areas of biological modelling, visual learning and recognition, mobile robotics, and teleoperation and supervisory control. Primary contributions include a faithful neurocomputational emulation of rat navigation and its demonstration on a mobile robot, and the use of telepresence in teaching robots via reinforcement learning. Secondary contributions include the direct extension of HRV learning principals from object vision to environment navigation, and a fast mechanism for learning new behaviors via reinforcement learning using task-independent models. These contributions are elaborated in the Conclusion (Chapter 11).

## 1.5 Thesis Overview

The remainder of this thesis is presented in two parts. Part I consists of Chapters 2-5, which together describe the biological motivations, technical background, and previous work related to strategy learning and navigation; Part II consists of Chapters 6-10, which describe the design and implementation of the various architectures and sub-systems in the proposed neurocomputational system, and report the results of using the system to perform several navigation tasks on the mobile robot MAVIN in a simple laboratory environment. The experienced reader more interested in the actual systems developed here may wish to start with Part II, perhaps after skimming some of the motivational issues in Chapter 2, since ample cross references are made throughout the thesis back to the relevant background chapters in Part I.

Part I (Technical Background and Related Work):

Chapter 2: This chapter begins by presenting a detailed review of the biological motivations for the HRV approach to strategy learning. In particular, it argues for a cognitivist theory of intelligence and learning, and discusses the evidence for HRV cognitive maps for visual navigation. It also illuminates an analogy between object learning and environment map-making, and discusses an existing object vision system that motivates the computational form of the proposed system. Finally, it discusses the psychological data pertaining to the training of strategies via guidance by a teacher.

Chapter 3: This chapter details various symbolic, statistical, and neural network methods for learning task-independent models or representations of environments consisting of states and transitions between states. This chapter not only forms the basis for the rest of the background chapters, but also provides the framework for analyzing the biological model of map-making discussed later in the thesis.

Chapter 4: This chapter discusses various computational issues and physical techniques for task learning, including direct and indirect methods for reinforcement learning, the relationship between reinforcement learning and control learning, and teleprogramming techniques for training robots. This chapter in effect provides computational arguments for cognitivist theories of biological learning.

Chapter 5: This chapter reviews previous work in the realm of visual navigation, including that related to active search, tracking, and feature extraction for visual landmarks, and the use of landmarks to form maps for visual navigation. The latter discussion evaluates both AI approaches to forming maps based on distinctive places, and neurocomputational architectures based on biological models of place learning and cognitive mapping. The systems and models discussed in this chapter provide a framework in which to directly evaluate the map-making system proposed in Part II of the thesis.

Part II (Neurocomputational Architectures: Design and Implementation):

Chapter 6: This chapter describes in mathematical detail some of the supporting architectures for landmark-based visual navigation, including the featural grouping sub-system (FGS), gaze control sub-system (GCS), object vision sub-system (OVS), and path planning sub-system (PPS). It also describes the simplifications made to each of these sub-systems that are currently necessary in order to implement each of the navigation architectures described in subsequent chapters in real-time on MAVIN.

Chapter 7: This chapter presents the evolution of the place learning architecture (PLA). It describes in mathematical detail the implementation of several different versions of this architecture on the mobile robot MAVIN, and presents the results of these implementations graphically.

Chapter 8: This chapter describes the neural dynamics of the action consequence learning architecture (ACLA), and presents in mathematical detail a probabilistic interpretation of this architecture. It also describes how the ACLA might be used to predict the consequences of actions, and reports on the results of simulating the architecture in this capacity using visual data gathered by MAVIN in a structured laboratory environment.

Chapter 9: This chapter presents the implementation of the reinforcement learning architecture (RLA). It describes how the relative utility of places and actions are learned by the robot during exploration, and how this learned utility interacts with the ACLA in order to perform route planning. Again, the chapter reports on the results of simulating the architecture in this capacity using data gathered by MAVIN.

Chapter 10: This chapter describes how each of the neural architectures are integrated on MAVIN to form a trainable navigation system. It presents at the conceptual level the process of teletraining, and relates the results of using this training methodology to teach MAVIN to solve a particular navigation task in a simple laboratory environment.

In conclusion, Chapter 11 discusses the results, details the primary and secondary contributions of the thesis, and presents avenues for future work.

The appendices contain the following subject matter:

Appendix A: This appendix describes the mobile robot MAVIN, and the computer systems it employs to run the various sub-systems described throughout the thesis.

Appendix B: This appendix defines the mathematical notation used to describe the various computations performed by the sub-systems, architectures, networks, and frameworks introduced in Part II and in subsequent appendices.

Appendix C: This appendix defines the operations used to perform coarse population coding in the various neurocomputational architectures described throughout Part II.

Appendix D: This appendix defines the dynamics used to implement the associative artificial neural networks employed in Part II.

Appendix E: This appendix reviews the pertinent elements of *Markov decision processes*, which are used to interpret various portions of the architectures proposed in Part II.

# Part I Technical Background and Related Work

As outlined in the introduction, the computational system proposed in this thesis is inspired by theories of cognitive mapping in biological organisms. In particular, it is inspired by the apparent formation of cognitive maps in the visual domain using *hierarchical*, *relational*, *view-based* (HRV) learning principals. It is important to emphasize here that the proposed system is not intended as a *model* for brain structures or computations; it merely *emulates* the process of cognitive map formation in biological organisms in order to realize performance goals.

The following four chapters detail the technical background related to both the biological interpretations and computational methods employed throughout the thesis. Most of the biological data concerning cognitive mapping and task learning is described in Chapter 2, while the computational issues are presented in Chapters 3-5. Computational issues concern not only the more abstract problems pertaining to learning task-independent representations of environments (Chapter 3) and learning new tasks (Chapter 4), but also the more specific problems of visual map-making and navigation for a mobile robot (Chapter 5). All of these issues are discussed in the context of existing biological models, neurocomputational networks, computational systems, and mathematical techniques.

# CHAPTER 2 *Biological Motivations*

The navigation strategy learning system proposed in this thesis is motivated by a *cognitivist* account of behavioral learning in higher animals. In particular, it is motivated by evidence for the formation of *cognitive maps* — internal, task-independent representations of environments — as an integral step in learning to perform new tasks. In the visual domain, the formation of cognitive maps seems to take place using *hierarchical*, *relational*, *view-based* (HRV) learning principals (see Section 1.3.1). Presumably, task learning occurs as a result of evaluating the utility of situations and actions defined by the cognitive map with respect to external reward and punishment, perhaps issued by a trainer or the environment itself. In humans, the effectiveness of this task training is often improved by physically guiding the trainee through the task.

This chapter briefly describes the basis for cognitivist theory in the context of behavioral learning, then goes on to present specific anatomical, neurophysiological, and behavioral evidence for two examples of HRV cognitive mapping: map-making and navigation by rodents, and object learning and recognition by primates. It also reviews an existing HRV neurocomputational system for object vision, which provides not only the computational tools for visually processing landmark objects later in this thesis, but also the conceptual basis for how HRV spatial cognitive maps might be formed. This conceptual basis is facilitated by a direct analogy between visual environment learning and navigation, and visual object learning and recognition. Finally, the chapter reviews some psychological evidence for the effectiveness of physical guidance in training humans to perform new tasks.

# 2.1 Cognitivist Theories of Behavioral Learning

## 2.1.1 Classical and Operant Conditioning

The study of behavioral learning in biological organisms has traditionally encompassed two forms of conditioning, referred to as *classical conditioning* and *operant conditioning* (for a review, see Staddon & Ettinger, 1989). Classical conditioning describes a scenario wherein an organism is repeatedly presented with a conditioned stimulus (CS), an initially neutral stimulus which only elicits an orienting reflex response, closely coordinated in time with an unconditioned stimulus (US), which initially elicits an unconditioned reflex response (UR). This presentation eventually causes the organism to anticipate the US and elicit a conditioned response (CR), similar to the UR, when the CS is presented alone. The repeated pairing between the CS and the US is referred to as *reinforcement*. For example, in Pavlov's (1927) classic demonstration, a dog is conditioned to salivate (CR/UR) in response to ringing a bell (CS) as a result of repeatedly ringing the bell just prior to presenting the dog with meat (US).

In contrast, operant conditioning (often called *instrumental conditioning* or *trial-and-error learning*) involves an increased occurrence of a particular behavior (called an *operant*), not as a consequence of any external stimuli, but instead as a result of rewards and punishments delivered to the organism following the behavior (Thorndike, 1911; Skinner, 1932). In this case, reinforcement between the operant and the reward occurs only as a result of a correct or incorrect response. This process is called *reinforcement learning* (RL). The fact that both types of conditioning involve reinforcing events and are affected by timing, in addition to the fact that they both occur mainly in the context of survival, suggests that they emerge from a common neural mechanism (Kupferman, 1985), which is thought to reside in an area of the brain called the *hippocampus* (see Schmajuk & DiCarlo, 1991).

## 2.1.1 Cognitivist vs. Behaviorist Accounts

Most models of behavioral conditioning involve some sort of associative learning scheme (e.g. Grossberg & Schmajuk, 1988; Schmajuk & DiCarlo, 1991; see Staddon & Ettinger, 1989). One such model (based on Grossberg & Schmajuk, 1988) has been demonstrated successfully using the mobile robot MAVIN in the laboratory here at Lincoln Laboratory (Baloch & Waxman, 1991a,b). According to *behaviorist* theories, these sorts of mechanistic stimulus-response-reward RL rules form the entire basis for behavior (Pavlov, 1927; Thorndike, 1911; Skinner, 1932). But another way of looking at these associative models is in terms of the greater context provided by task-independent models or representations of environments. Such models can be thought of as embodying the interrelationships between concepts, situations, and actions, regardless of external reward or penalty. This representationalist view of reinforcement learning forms the basis for *cognitivist* theories of learning (Kohler, 1925; Tolman, 1932), which posit that an animal uses inferential learning to acquire knowledge (called *cognitions*) despite the absence of reinforcement, and uses this knowledge to execute *purposive*, as opposed to *reflexive*, behavior.

According to cognitivist theories, both classical and operant conditioning become a matter of directly associating rewards and penalties with concepts in the model. For example, the dog in

Pavlov's (1927) experiment might simply learn a model of the world that says that meat follows bell, and associate the meat with a reward. Salivation is then not a direct consequence of the bell ringing, but instead an indirect result of predicting meat. Thus, assuming that the activation of a particular learned category (US) in a model initially elicits (e.g. due to direct association) a particular reflex action (UR), classical conditioning can be thought of as learning direct relations between one concept (US) and another (CS), and a subsequent *prediction* of a response (UR) as an indirect result of these associations. Similarly, operant conditioning can be thought of as a direct association of reinforcing events with situations and actions in the model, and a subsequent *planning* and execution of the sequence of actions that predict the greatest reward and the smallest penalty, thereby producing a behavior.

It should be noted that cognitivist theories do not completely preclude stimulus-response learning. Certain situations may call for quick, reflexive responses that bypass the internal computations required to predict or plan. In fact, cognitivist theory provides a way to learn such responses in an indirect fashion. For example, direct CS-UR relations might be learned simply by associating concepts with predicted outcomes. Similarly, entire sequences of planned actions might simply be stored (e.g. using time-delay neural networks like avalanche networks — see Simpson, 1990), and elicited in response to some internal drive, such as hunger. Cognitivist theories cannot, however, explain the phenomena of conditioned "excitation" or "inhibition," nor do they completely explain extinction phenomena (see Baloch & Waxman, 1991a,b; Grossberg & Schmajuk, 1988).

Such cognitivist theories are supported by evidence that animals can learn without actually performing certain tasks, particularly by accounts of *latent learning*, where an animal uses knowledge previously acquired without reinforcement (e.g. during *exploration*) to immediately improve its responses when reinforcement is suddenly introduced. For example, a rat that is allowed to explore a maze prior to the issuance of food at a particular location learns to reach the food much more quickly than it would if it had not explored the maze beforehand. Cognitivist theories are also supported by accounts of *detour behavior*, wherein a rat in a maze seems to integrate previously travelled routes in order to travel a completely new path to a particular location. Perhaps the greatest support for cognitivist theories, however, are neurophysiological accounts of *cognitive mapping*.

## 2.2 Cognitive Maps

Cognitivist thinking formed the basis for Tolman's (1948) early theory of cognitive maps (see Staddon & Ettinger, 1989; see also Section 2.2.1), which can be considered as internalized task-independent representations of the external world. Tolman (1948) postulated that such maps could be formed by combining local *expectancies*, each of which code the situation that results after making a specific response in a specific situation.

The direct evidence for cognitive mapping has mainly been gathered from laboratory rat experiments in the context of visual map-making and navigation. It is this evidence that primarily motivates the HRV approach to map-making developed herein. Note, however, that substantial evidence also exists in primates for the learning of task-independent HRV representations of visual objects. Due to an analogy between object and environment learning, a neurocomputational

system for visual object learning and recognition provides the conceptual motivation for *how* one might learn spatial cognitive maps.

### 2.2.1 Visual Map-Making by Rodents

Laboratory rat experiments have certainly achieved stereotypical status in the field of experimental psychology, and have long been used to study phenomena like maze learning and detour behavior (for a review, see Staddon & Ettinger, 1989). The hypothesis that animals employ spatial cognitive maps to perform these tasks dates back to Tolman (1948; see also Kaplan, 1973), who described such maps as embodying a knowledge of "what objects [i.e. landmarks] are where, and what situations [i.e. places] lead to what situations."

#### Place Cells

More recent neurophysiological investigations into cognitive mapping have mainly focussed on so called *place cells* in areas CA1 and CA3 of the rat hippocampus (for a review of the literature, see O'Keefe & Nadel, 1978; McNaughton, 1989; Muller et. al., 1991; Burgess et. al., 1995). Each of these cells fires when the rat enters a local region of a visual environment (i.e., place field — see Figure 6), independently of goal-driven behavior. Often, but not always, place cells are described to be broadly tuned to a particular heading of the rat (see Figure 7). Amazingly, place cells exhibit one-shot or fast learning (learning in a single trial), and rapidly establish their place fields (Wilson & McNaughton, 1993).

Studies indicate that place cells actually respond to the *identity, pose* and *spatial distribution* of visible landmarks within a *local sensory view* of the environment. This combined representation seems consistent with the fact that *object vision* and *spatial vision* streams (i.e. "What" with "Where" pathways) in the primate visual system respectively culminate in the temporal and posterior parietal cortices (see Section 2.2.2; see Figure 10), and project to the hippocampal formation via the entorhinal cortex (McNaughton, 1989; Rolls, 1991; Perrett & Oram, 1994). Perhaps similar streams exist in the rat brain.

The importance of landmark identity and pose is indicated by studies where modifying environment boundaries (e.g. walls) causes changes in place fields only at the boundaries themselves, while removing or modifying the appearance of a substantial portion of landmarks results in significant changes in the shape and position of multiple place fields (see Muller et. al., 1991). Appearance here refers to the qualitative percept of landmark form, since raw sensory details such as color, texture, and size seem to play no major role in place cell firing. The firing of place cells is affected by the spatial constellation of landmarks, since a change in the relative position of one of the landmarks in the environment also causes changes in place fields. Note, however, that place fields exhibit invariance to the absolute position and distance of landmarks, in that they qualitatively scale, rotate, and translate with the landmark constellation, even when only one landmark defines the environment. Such invariances are typical in *object recognition*, and are reflected here in the mapping of visual environments (more on this later).

Together, these findings support a view-based representation of spatial relations, such as the set of visual directions to recognized landmarks. The existence of coarsely tuned head direction cells in

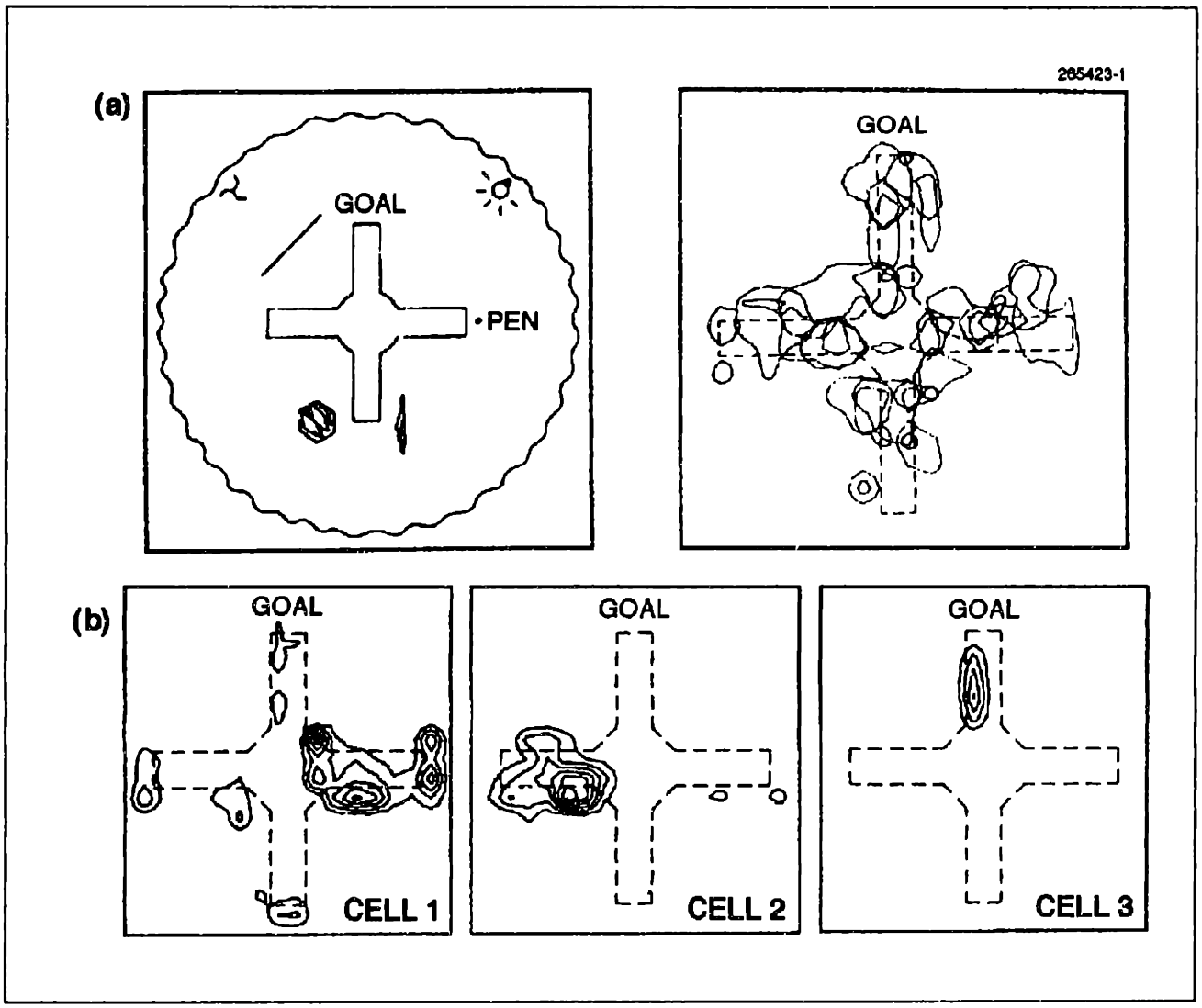


**FIGURE 6: SPATIAL FIRING PATTERNS OF PLACE CELLS IN THE RAT HIPPOCAMPUS.**
*Place field firing patterns measured in the hippocampus of freely moving rats: (a) A four-arm maze environment consisting of several visual landmarks, parcelled into overlapping regions formed by contours (drawn at twice the mean firing rate) of the spatial firing patterns measured for 8 hippocampal place cells. (b) Separate spatial firing pattern contours for 3 place fields. (Adapted from O'Keefe & Speakman, 1987, with permission from Springer-Verlag)*

the postsubiculum (the dorsal portion of the hippocampal formation — see Figure 8) have been hypothesized as the source of egocentric directional information for visual landmarks (see McNaughton, 1989). Each of these cells has roughly a Gaussian receptive field over the space of possible head directions, with a standard deviation ranging between 20° and 60° (again, see McNaughton, 1989).

Because spatial firing patterns of place cells overlap, several can together coarsely map an entire environment, and a small ensemble of place cells can code for location (Wilson & McNaughton, 1993). Note, however, that some place cells have firing rate profiles that peak very close to one

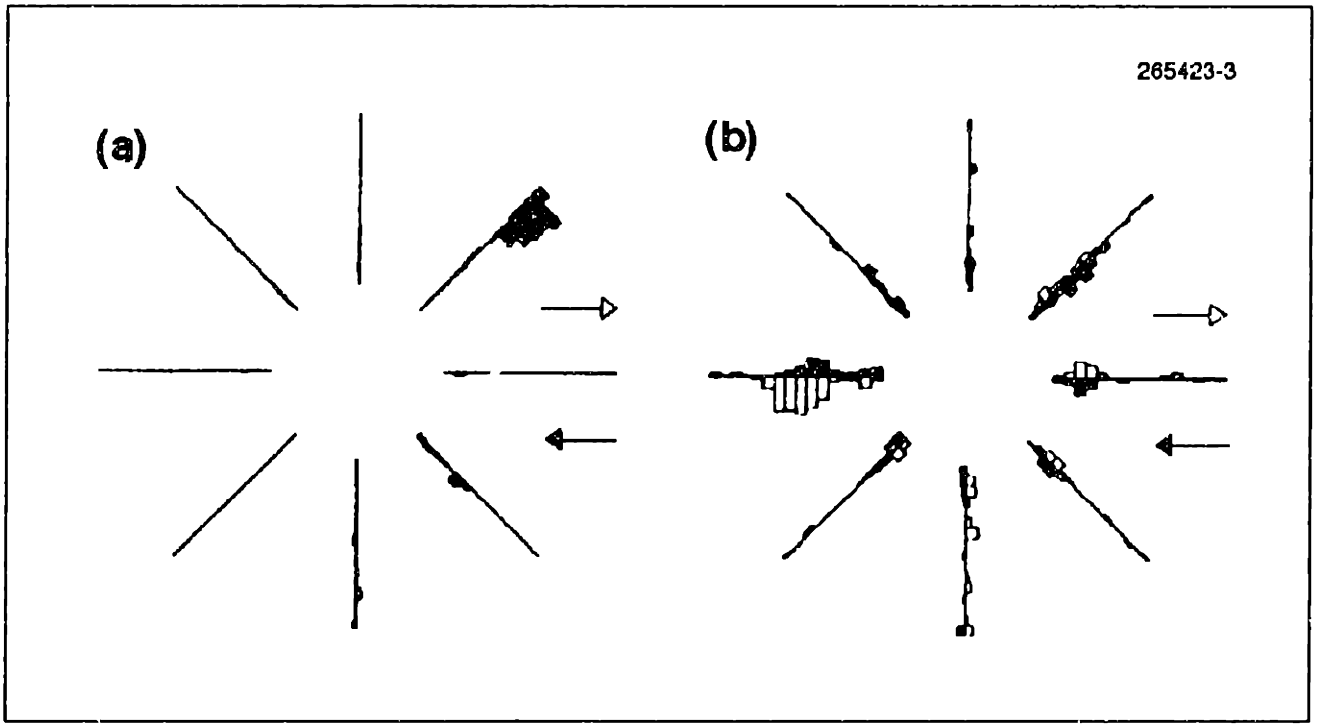


**FIGURE 7: FIRING RATE HISTOGRAMS FOR HEADING TUNED PLACE CELLS IN RATS.**
*The positional firing rate histograms of 2 heading tuned place cells (A and B) recorded in an 8-arm radial arm maze (adapted from McNaughton et al., 1983, with permission from Springer-Verlag). For each arm, the black histogram corresponds to inward movement, and the white histogram corresponds to outward movement.*

another and overlap to an immense degree (see, for example, the firing rate contours in Figure 6a). This observation suggests that a large number of overlapping place cells code an environment, and that experimenters typically find and measure only a small proportion of these cells for any single environment. This "redundant" population coding might simply allow the rat to localize itself more accurately by providing a measure of robustness to localization in the presence of noise and dying place cells.

### Evidence for Relational Coding

Despite the fact that a large degree of overlap between place fields allows them to code individual locations in an environment, several other facts indicate that the hippocampus actually employs relational principals to perform this environment coding. First, a broad class of "displace cells" respond to various combinations of place fields and body motions (O'Keefe & Nadel, 1978). Some of these cells signal the movement of the rat between place fields, possibly coding for transitions between place cells. Others (about 40%) signal movements of the rat through particular place fields with specified directions and velocities, perhaps facilitated by cells in the parietal neocortex that signal equivalence classes (classes with similar consequences) of left turn, right turn, and forward motions (see McNaughton et. al., 1991; see also Figure 8).

Second, some place cells actually predict the rat's arrival within their place fields with a time scale on the order of 100 ms (see Muller et. al., 1991). In fact, established place cells often fire in

the correct locations in the absence of visual input (i.e., in complete darkness), provided that the rat has had time to orient itself beforehand (i.e., with the lights on). Interestingly, the time scale for these predictions corresponds to the 5-8 Hz rhythm (EEG) exhibited by a set of *theta cells* in the hippocampus, which increases in frequency when the rat performs translational movements (see McNaughton, 1989).

Finally, anatomical studies of area CA3 have revealed a lateral meshwork of synapses forming sparse but extensive interconnections between pyramidal place cells via a set of interneurons (see McNaughton, 1989; Muller et. al., 1991).

### Anatomy of the Hippocampal Formation

In interpreting these neurophysiological observations, it is important to keep in mind that many cell types in the hippocampal formation exhibit spatial specificity. Each of these cell types responds to its own set of visual, motor, and temporal characteristics. For example, a group of so-called "misplace" cells seem to fire when visual expectations of the presence or absence of a visual landmark are not met (O'Keefe & Nadel, 1978). There are even a variety place and displace cells, each selective to slightly different conditions (e.g. sensory views, landmarks, body motions, head direction, etc.). Anatomical data indicate cells with place-like spatial properties all along the main hippocampal processing pathway extending from the medial entorhinal cortex (MEC), to the dentate gyrus, to areas CA3 and CA1 respectively, and finally to the subiculum (see McNaughton, 1989; Muller et. al., 1991; see also Figures 8 and 9). Other minor pathways also exist between these areas.

More specifically, a set of highly active granule cells with place-like properties in the dentate gyrus receive divergent input from the superior MEC (as well as mossy, or multipolar, cells). These cells feed (again divergently) a set of place cells in area CA3, which in turn contact place cells in area CA1. Both granule and place cells are powerfully inhibited by a relatively small set of basket cells, which have been implicated in the shunting inhibition often associated with self-organizing competitive networks, and both place and basket cells are modulated by the theta cells. Finally, place cells in CA1 contact place-like cells in the subiculum that seem to indicate combinations of location and head direction (perhaps modulated by the head direction cells in the post-subiculum), and these cells make feedback connections via the deep layers of the MEC to the superior MEC.

The system proposed in this thesis does not attempt to emulate each cell type and its possible role in spatial navigation. Rather, it emulates the general concepts of spatial cognitive mapping in biological systems in order to achieve a certain level of functionality.

### The Hippocampus as a Working Memory for Spatial Information

It is also important to note that the hippocampus does not encompass all the necessary mechanisms for learning and using cognitive maps, nor is it exclusively involved in the modelling of spatial information. Studies indicate that other brain structures represent local space around the animal to facilitate motor skills like reaching, and perhaps local navigation (for a review, see Soechting & Flanders, 1992). Studies also indicate that the hippocampus serves as a short-term or

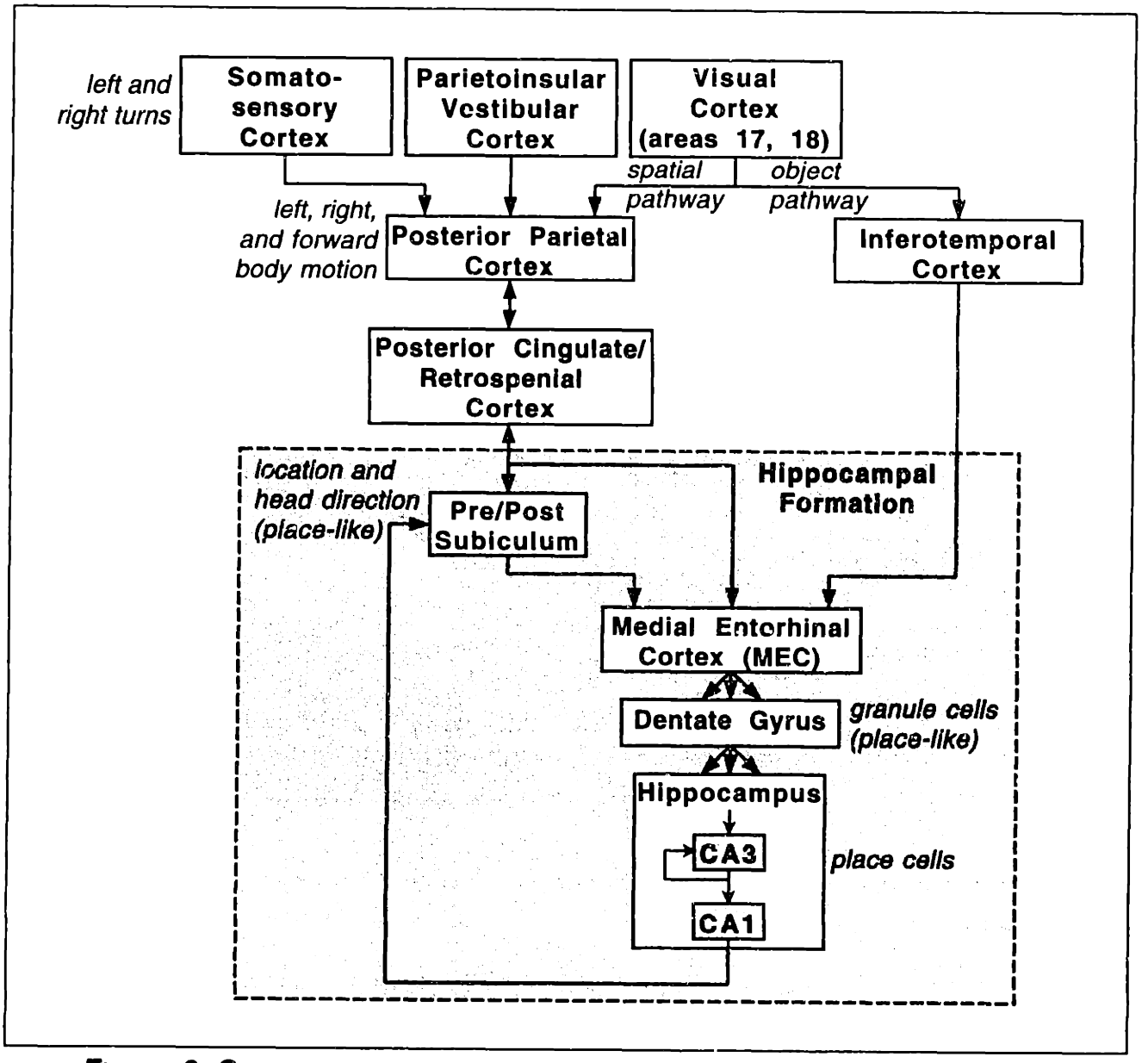


**FIGURE 8: GROSS NEURO ANATOMY OF THE RODENT HIPPOCAMPAL FORMATION.**
*Sensory inputs originating from the somatosensory, parietoinsular vestibular, and visual cortices converge on the hippocampal formation (Adapted from McNaughton et. al., 1991). Visual information makes its way to the hippocampal formation via both the spatial and object vision pathways (see also Figures 9 and 10).*

*working memory* store for general relational learning on a short time scale, and that it somehow modulates the more permanent storage of cognitive maps in higher brain centers (i.e. the neocortex) by playing back information to a slower learning mechanism (see Minai and Levy, 1993a).

This working memory role is supported by the apparent re-use of place cells across multiple environments with greatly differing visual characteristics (summarized by McNaughton, 1989). While individual place cells very rarely code for more than one place field in a single environment (i.e. *split place* fields — see, for example, Cell 1 in Figure 6), they frequently code for more than one

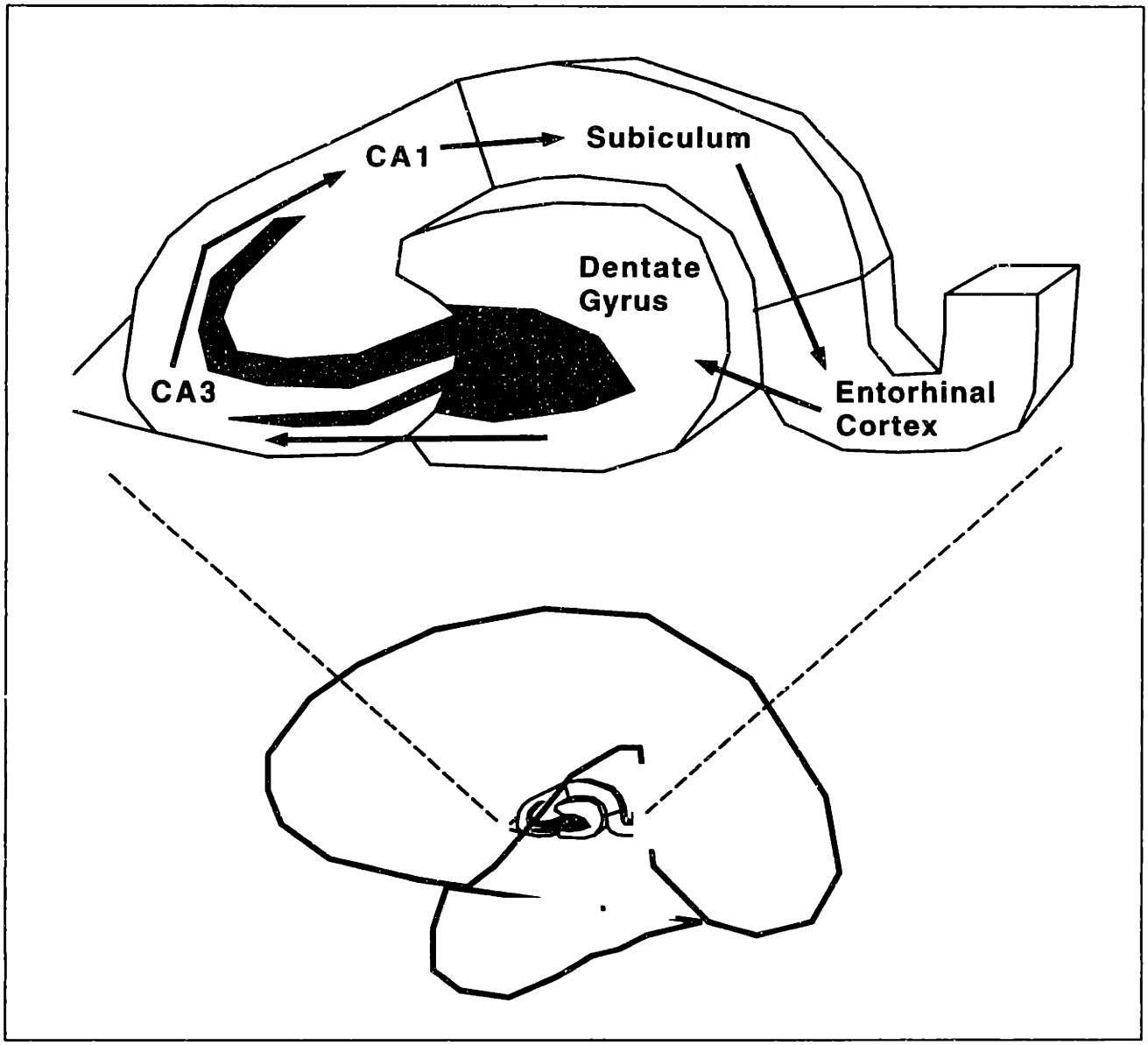


**FIGURE 9: ANATOMY OF THE MAMMALIAN HIPPOCAMPAL FORMATION.**
*A rough depiction of the mammalian hippocampal formation (single hemisphere). (adapted from McNaughton, 1989).*

place across several environments. In fact, each place cell has a 70% chance of coding for a place field for any particular environment. Finally, the working memory role is supported by the fact that lesions to the hippocampus do not completely destroy the rat's ability to navigate (though it does impair the rat's ability to learn new place fields).

### HRV Spatial Cognitive Maps in Other Organisms

The evidence for cognitive mapping in rats is augmented by studies that indicate the use of spatial cognitive maps in other organisms. The notion that biological cognitive maps universally take the form of relations between distinctive places does not yet have sufficient evidence (probing individual neurons in the human brain, for example, is a bit questionable from a moral standpoint!).

However, studies have shown that insects (e.g. bees, ants) navigate by continuously comparing retinal images to stored "snapshots" of landmark panoramas (Gould, 1986; Cartwright & Collett, 1987; see Staddon & Ettinger, 1989; Wehner & Menzel, 1990).

Other studies have established a role of the primate hippocampus in object-place localization associations (in addition to other non-spatial roles). In fact, some pyramidal cells in the primate hippocampus exhibit spatial characteristics similar to those of place cells in the rat hippocampus, and form a recurrent network with Hebbian synapses in area CA3 (Rolls, 1991). Furthermore, damage to or removal of portions of the hippocampal formation (area 7) causes disorders of orientation, egocentric localization, spatial perception, and route behavior in humans (see De Renzi, 1982; Rolls, 1991).

At the psychological level, studies have shown that, while driving, humans only perform visual searches for conspicuous targets in a small azimuth range centered about the vehicles aiming point (Cole & Hughes, 1990). Furthermore, children have difficulty following routes in reverse directions, and in predicting landmarks while doing so (Piaget & Inhelder, 1967). These studies suggest that humans might form relational maps using restricted local sensory views.

### A Computational Hypothesis

Despite the complexity of spatial processing in biological organisms, a consistent hypothesis maintains that granule cells compete to learn combinations of landmark form and spatial direction represented in the entorhinal cortex; that place cells in hippocampal area CA3 compete to learn this conjunction; that body translations reset this learning process; that an associative network learns relations between CA3 place cells via a set of displace cells responsive to locomotive actions; that both sensory and predictive activity of CA3 place cells feed a set of CA1 place cells; and that CA1 place cells impose top down expectations for landmark shape and direction via directional cells in the subiculum.

This hypothesis resembles McNaughton's (1989) egocentric conceptual model of hippocampal function (for more detail, see Section 5.2.2), which maintains that *a place cell responds to a restricted local sensory view of an environment, and that relations between these heading tuned place cells are learned by associations conditional on specific turning and forward body motions.* This model forms the conceptual basis for map-making in the proposed computational system. The embedding of McNaughton's basic ideas into the system is facilitated by an analogy between 3D environment learning and navigation and 3D object learning and recognition.

## 2.2.2 Object Learning and Recognition by Primates

### View-Based Representations of Faces, Heads, and Body Parts

Cognitive mapping in the rat hippocampus bares a strong resemblance to HRV object representations in the shape processing ("What") pathway of the primate visual cortex, which runs from the primary visual cortex to the superior temporal sulcus (area STPa) via the infero-temporal cortex (IT) along the path V1-V2/V3-V4-PIT-CIT-AIT-STPa (Mishkin et. al., 1983; DeYoe & Van Essen, 1988; Zeki & Shipp, 1988; Perrett & Oram, 1993; see Figure 10). Cells in IT code increasingly complex spatial patterns of features (e.g. star-shaped stimuli) in a hierarchical fashion

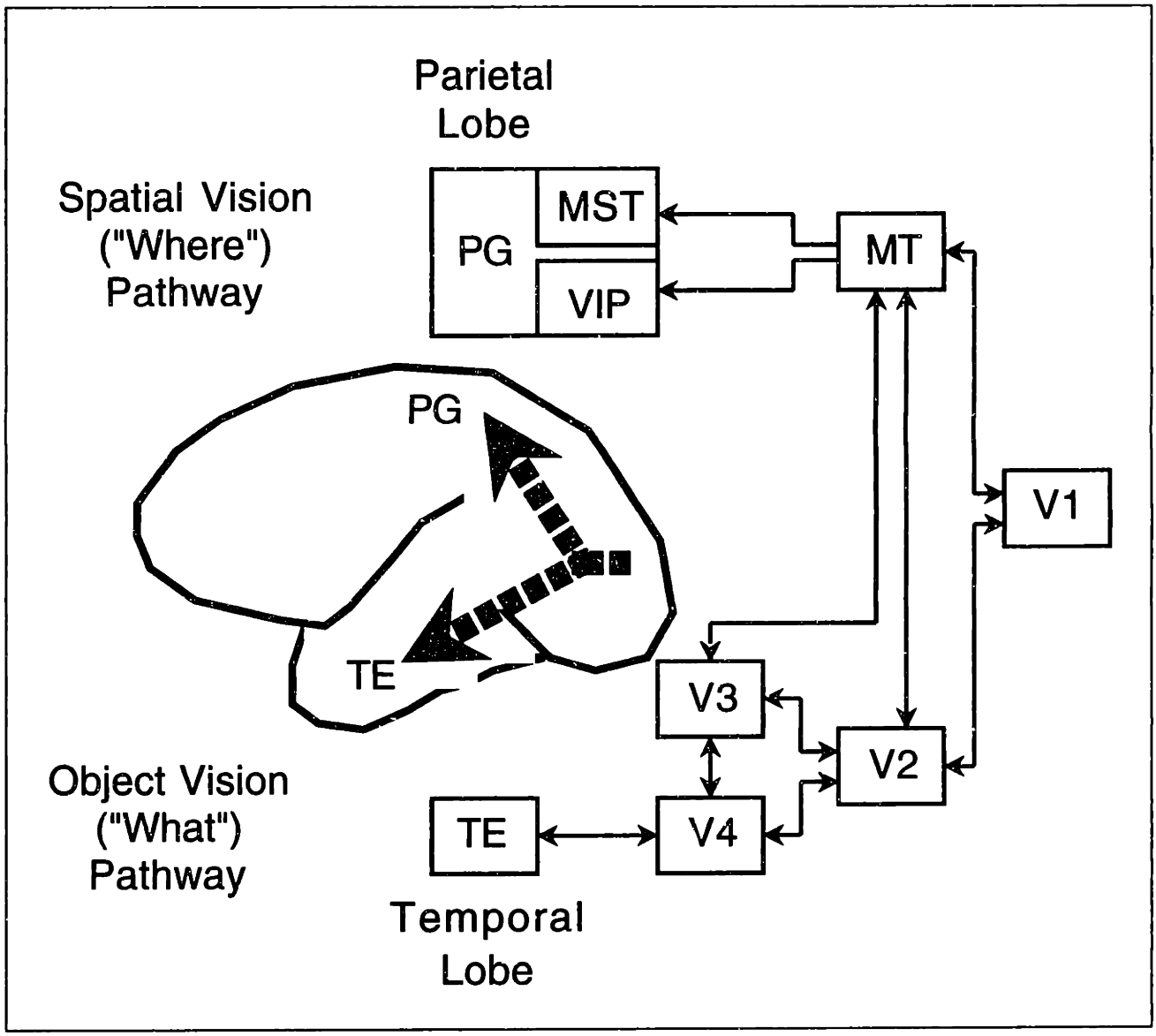


**FIGURE 10: GROSS NEURO ANATOMY OF OBJECT AND SPATIAL VISION STREAMS IN PRIMATES.**
*Adapted from Mishkin et. al. (1993).*

(Tanaka, 1993; see Desimone, 1992), while cells in STPa code view-based representations of faces, heads, and other body parts, invariant to illumination, object position, scale, and foreshortening on the visual field (see Perrett & Oram, 1993, 1994; see Figure 11). Further evidence supports the learning of temporal associations between shape coding cells (Miyashita, 1988), which is consistent with yet another set of cells that fire during rotations between views of heads (Perrett & Oram, 1994). Finally, a third set of cells respond to particular faces/heads regardless of view. These observations suggest that single "grandmother cells" form object-centered representations by hierarchically pooling a set of cells that code, invariant to a host of visual transformations, complex, view-specific, spatial patterns (Seibert & Waxman, 1989; Perrett & Oram, 1994). They

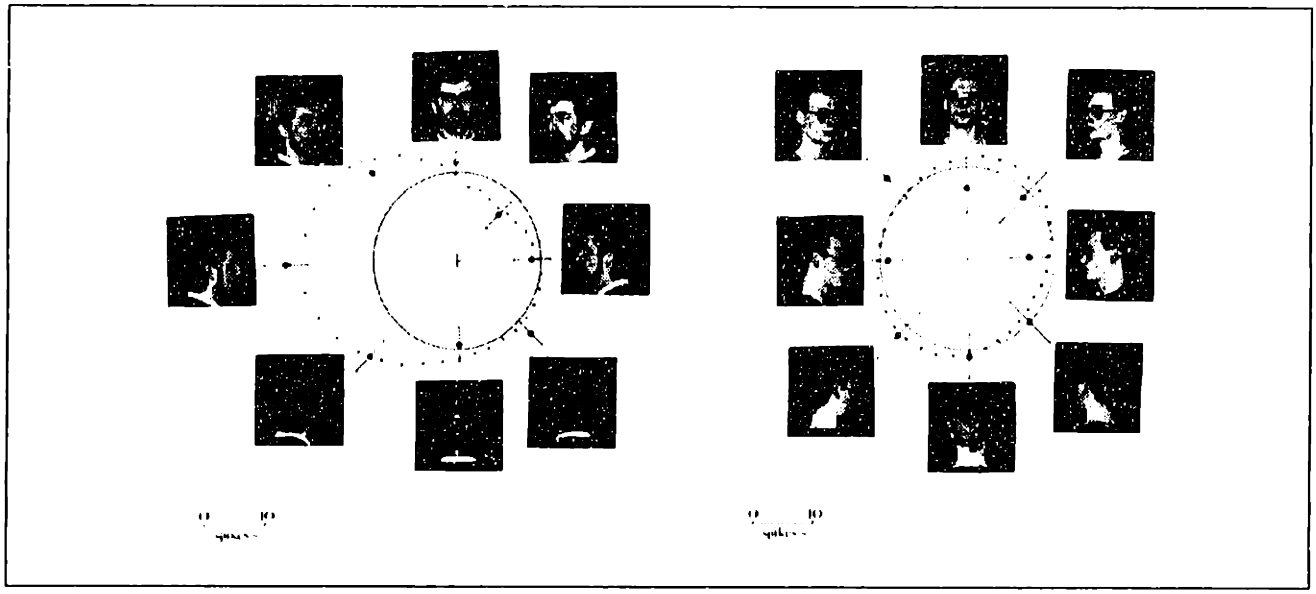

**FIGURE 11: OBJECT VIEW CELLS IN THE MACAQUE MONKEY.**
*The activity level of a single cell in the superior temporal sulcus of the macaque monkey exhibits a broad tuning to a particular view of the face on the left (adapted from Perrett et. al., 1989). Firing rate is plotted radially as a function of viewing direction (solid circles show the spontaneous firing rate, i.e. without a visual stimulus).*

also suggest that view transitions play an important role in object representation and recognition (Seibert & Waxman, 1989). All of these representations are independent of any reinforcement events, and can therefore be considered cognitive maps of 3D objects.

### Neurocomputational Emulation of Biological Object Vision

Given this evidence, it is not surprising that the HRV learning approach to adaptive perception was first demonstrated by a biologically-inspired adaptive neural system for learning and recognizing 3D objects (Seibert & Waxman, 1989, 1992; Waxman et. al., 1993). Not unlike map-making in the proposed system, this object vision system models an object as a collection of learned *characteristic invariant views* and possible *transitions* between these views (see Figures 12), resulting in a structure reminiscent of an aspect graph representation (Koenderink & van Doorn, 1979). The system takes as input image view sequences of an object in relative motion. From each of these view frames it extracts features and performs a number of invariance transformations. It then categorizes each of the resulting invariant view patterns into *aspect categories* that parcel the explored portion of the *viewing sphere* (whose coordinates correspond to possible viewing directions) into "islands" called *aspect regions*. Finally, it learns the possible transitions between aspect categories over multiple view frames, and subsequently gathers evidence for an object based on both aspect category and transition matches. (See Chapter 6 for a more detailed account of this system.)

The Seibert-Waxman object vision system can learn and recognize moving model aircraft from their visual silhouettes in real time (Seibert & Waxman, 1989, 1992; see Figure 13). It can also learn and recognize tactical targets from their appearance in synthetic aperture radar (SAR) spotlight sequences, and spinning reentry vehicles in the inverse SAR (ISAR) domain (Waxman et.

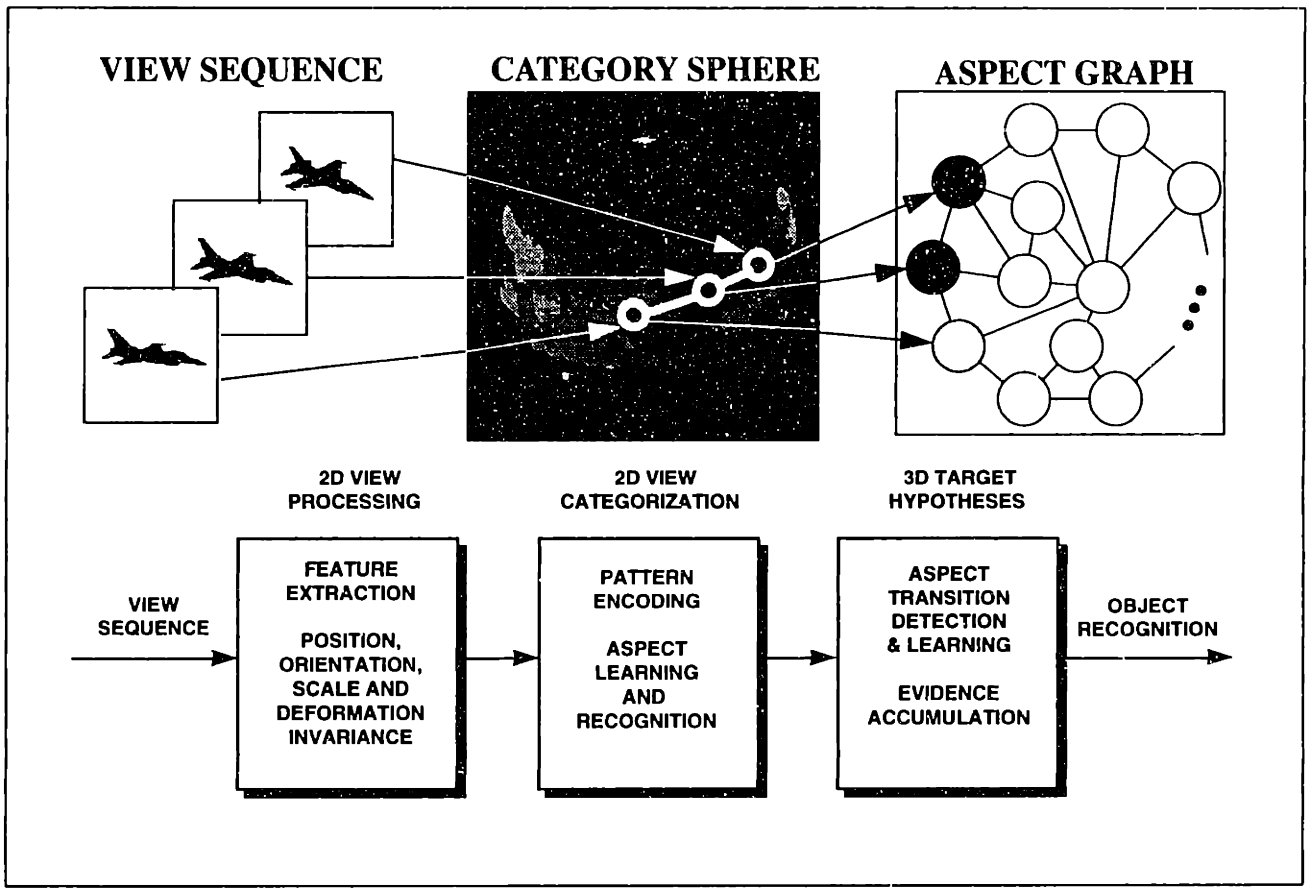


**FIGURE 12: A NEUROCOMPUTATIONAL SYSTEM FOR OBJECT LEARNING AND RECOGNITION.**

*The Seibert-Waxman object vision system parcels an object into prototypical invariant view categories called aspects (Seibert & Waxman, 1989, 1992). These categories form aspect regions on a category sphere, the surface of which represents the set of possible viewing directions. The system also learns the possible transitions between these regions, and thereby forms a representation of an object similar to an aspect graph.*

al., 1994; Waxman et. al., 1995; Bernardon & Carrick, 1995). Finally, it has been used on a mobile robot to learn and recognize simple objects and visual landmarks composed of 3D light patterns (Baloch & Waxman, 1991a,b; Bachelder et. al., 1993; Bachelder & Waxman, 1994; see Chapter 7).

### 2.2.3 The Object-Form/Environment-Layout Analogy

A number of recent modifications to the Seibert-Waxman recognition system serve to illuminate a useful analogy between learning an object and learning an environment. A first modification preserves some degree of orientation specificity of the incoming view patterns, which produces aspect categories broadly tuned (as opposed to invariant) to orientation (Bachelder et. al., 1994; see also Figure 14). This modification was motivated by recent physiological investigations (Perrett & Oram, 1993, 1994; Tanaka, 1993) suggesting that cells along the pathway in area IT

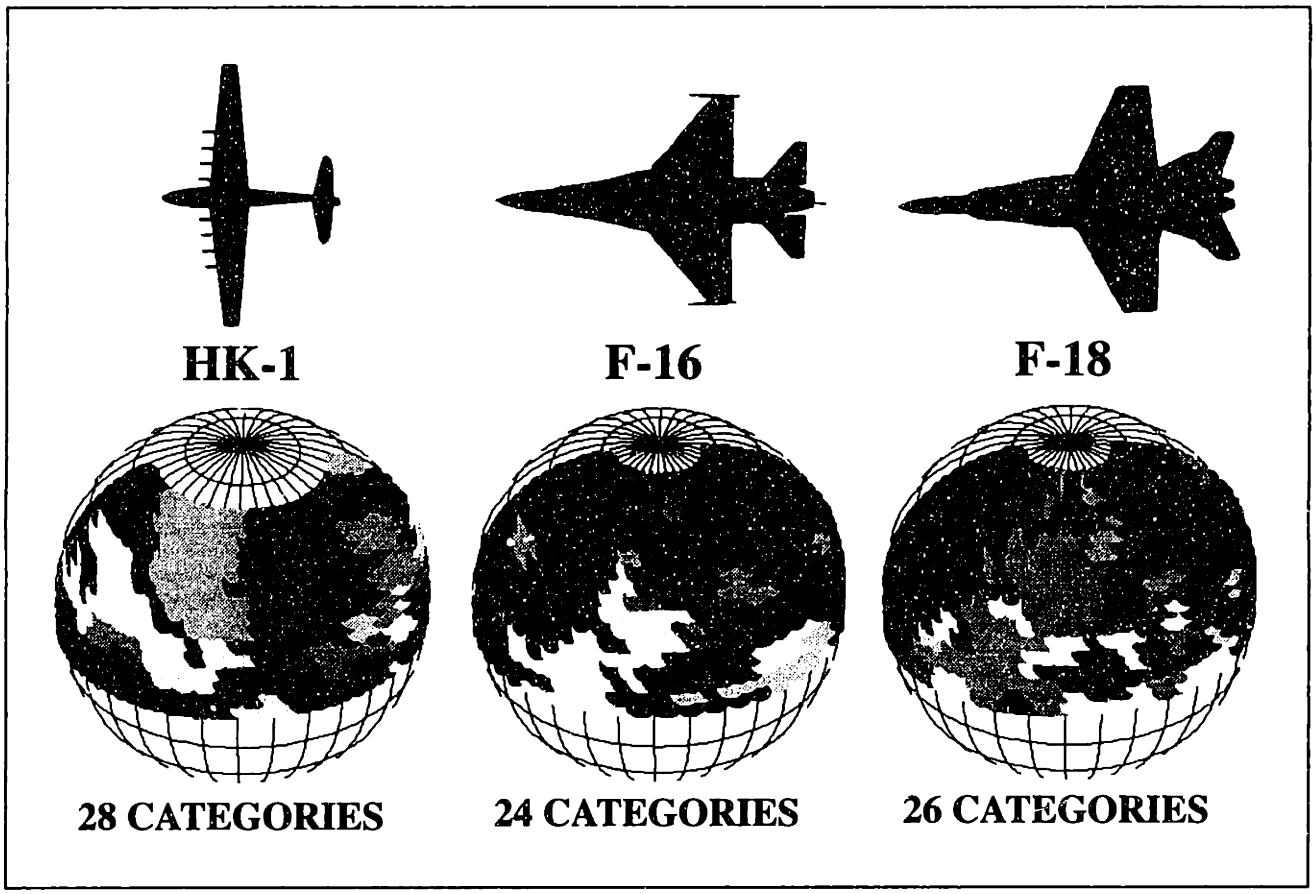


**FIGURE 13: LEARNED ORIENTATION INVARIANT ASPECT CATEGORIES.**
*Learned aspect categories for the F-18, F-16, and HK-1 aircraft. Typically 500 views are compressed to 25 categories and associated transitions (adapted from Bachelder et al., 1994).*

broadly preserve orientational selectivity to feature patterns, and that cells in area STPa finally achieve orientational invariance via hierarchical pooling. (Interestingly, this modification also overcomes several computational instabilities.)

Another recent extension supports learning class-object-part relations, and distinguishes between similar objects of a class on the bases of learned saliency maps in hierarchy (Seibert & Waxman, 1993; Waxman et al., 1993; Seibert et. al., 1995). In line with the separate processing of object parts by biological systems (see Treisman et. al., 1990), one might also modify the approach to first learn the aspects of object parts, and subsequently learn the aspects of entire objects in terms of the spatial relations among of part aspects (a combination of "What" and "Where"). Here, a "part" constitutes a complex feature pattern rather than a point feature. Finally, other work in progress relates object view quantization to learned image motion patterns (both within and between aspects) in order to develop a mechanism for learning aspect transitions in terms of visual motion events (Cunningham & Waxman, 1994b), and may provide a means for improved object recognition, inspection, and manipulation. This work is motivated by evidence for faces in motion (see Perrett & Oram, 1994), and cells in area MST that code complex patterns of motion (see Cunningham & Waxman, 1994b).

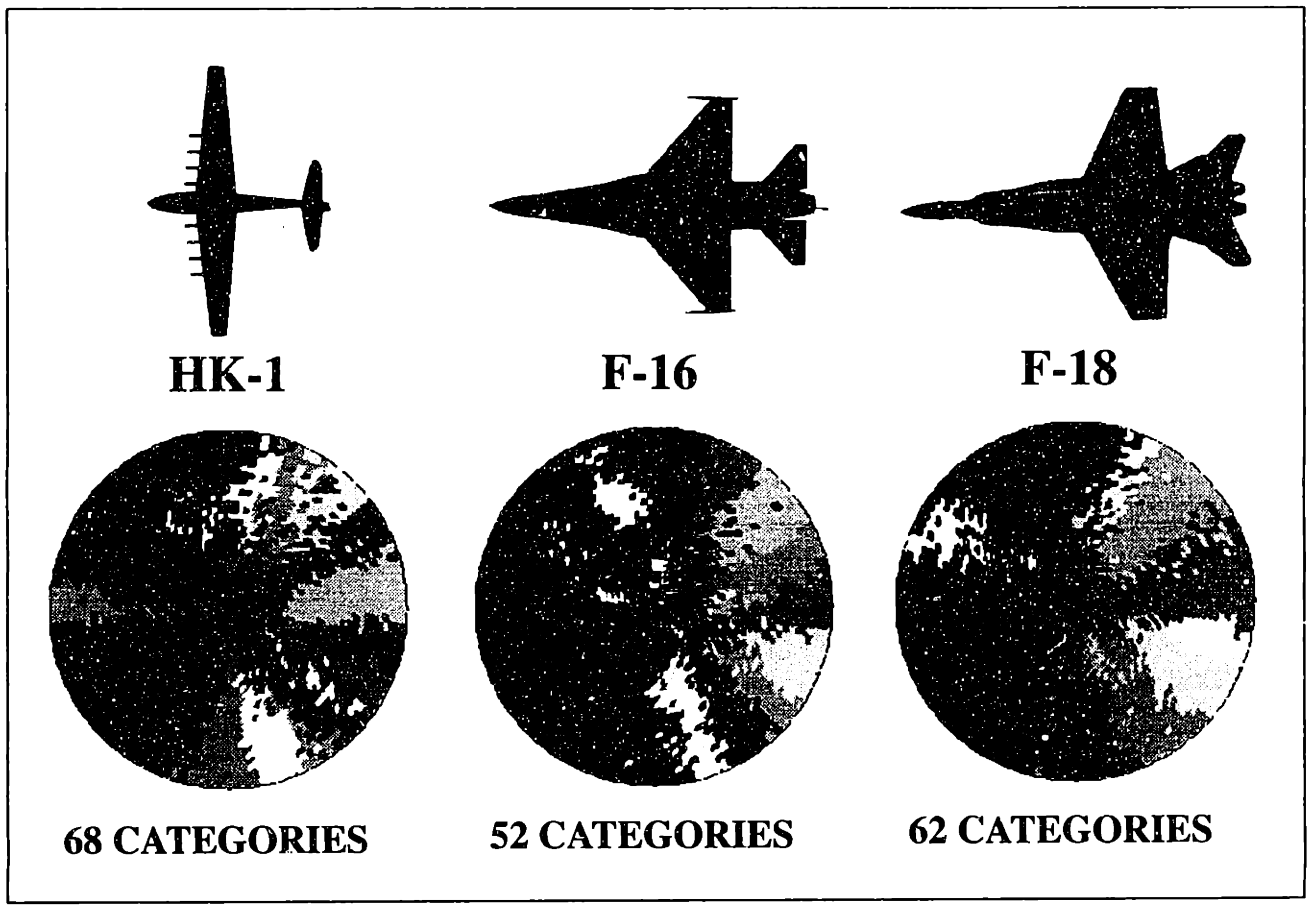


**FIGURE 14: LEARNED ORIENTATION TUNED ASPECT CATEGORIES.**
*Aspect spheres for three aircraft, preserving orientation information. Each category sphere, shown from the north pole, plots the learned aspect categories as a function of viewing direction for a single orientation. Typically 3600 views are compressed to approximately 60 categories preserving broad tuning to orientation. (adapted from Bachelder et al., 1994).*

In light of these recent modifications, the Seibert-Waxman object vision system might be thought of as a cognitive map of an object that defines expectancies of object aspects based on visual motion events. It therefore offers not only the tools for landmark shape processing during place learning, but also the conceptual framework for emulating spatial cognitive mapping. Places can be thought of as the viewer-centered spatial distribution and appearance of landmarks from a particular location and heading, just as object aspects can be thought of as the viewer-centered spatial distribution and appearance of parts from a particular viewing direction and orientation. Furthermore, action induced transitions between places are comparable to motion induced transitions between aspects. In short, *learning a 3D environment defined by a distributed constellation of visual landmarks can be thought of as learning an object composed of its constituent parts from the inside* (Benedikt, 1979).

## 2.3 Training Humans to Perform New Tasks

Beyond the learning of cognitive maps as part of a cognitivist theory for learning new tasks, is the issue of how new tasks might be *trained*. On an intuitive level, training encompasses much more than simply issuing rewards and penalties. More often, a trainer also actively *instructs* and *demonstrates*. Methods of instruction include verbal descriptions ranging from hints and suggestions to simply telling the trainee how to perform the task, while methods of demonstration range from having the trainee watch the trainer ("learning-by-watching") to having the trainer physically guide the trainee ("learning-by-doing"). For the most part, the training of animals is, of course, limited to demonstration techniques.

The notion of "teletraining" a mobile robot proposed in this thesis, is motivated by psychological studies of the effectiveness of various methods for training humans to perform new tasks (see Holding, 1987, for a review). These studies have shown that trainees often have difficulty performing perceptual-motor actions based on verbal instructions, and may actually be hindered by such instructions. For the most part, verbal instructions are effective only when they are short and spaced throughout the task (like hints, suggestions, or possible sub-goals), or when they are used to impart general principals or rules. On the other hand, training by demonstration (through imitation, films, visual cues, and physical guidance) can be quite effective in teaching entire segments of behavior, especially when general goals and procedures are taught rather than specific actions. In many tasks, like maze following, training through demonstration is particularly effective when both the correct and incorrect alternatives are clearly indicated by the trainer. This knowledge of alternatives is implicit in the case of physical guidance, especially *restrictive* physical guidance (as opposed to *forced-response* guidance). For example, a training harness restricts the movement of a student gymnast, thereby giving her a general idea of the correct and incorrect movements she can make. An example of forced-response guidance, on the other hand, is when an instructor physically guides the gynmnast's leg in order to demonstrate a particular acrobatic move.

Restrictive physical guidance can be thought of as a *directing of exploratory behavior* to those parts of the environment that are specific to the task at hand. Many tasks (like navigation) concern only small portions of the environment (e.g. the area around the direct path between the current location and the goal), and are affected by only a few important situations within those portions (e.g. dangerous places to avoid along the way). In order to quickly train a person to perform a given task, a trainer typically exploits the trainee's ability to learn an environment incrementally by preferentially directing exploration to germane portions and indicating important situations via rewards and penalties.

## 2.4 Summary

This chapter has reviewed the basis for cognitivist theories of behavioral learning in higher animals, and has argued for the role of physical guidance in training behavior for new tasks. In particular, it has discussed the learning of HRV cognitive maps for both map-making and navigation, and object learning and recognition. Evidence for map-making and navigation has been gathered from studies of *place cells* in the hippocampus of laboratory rats. These place cells seem to combine information from object and spatial vision streams (the "What" and "Where") in order to

learn and represent local areas called *place fields* in terms of the identities, poses, and spatial positions of visual landmarks within a local restricted sensory view of the environment. By learning relations between place cells with adjacent and overlapping place fields in terms of locomotive actions, the hippocampus then seems to form spatial cognitive maps. Similar mechanisms seem to be at work for object learning and recognition in the shape processing stream within the temporal lobe of the macaque monkey. These mechanisms have been modelled by a neurocomputational object vision system, which forms representations of objects similar to aspect graphs. A few minor extensions of this system illuminate an analogy between HRV approaches to learning object form and learning environment layout, namely that learning an environment consisting of a set of visual landmarks is like learning an object consisting of its visual parts, seen from the inside. Finally, the chapter has briefly argued, based on psychological studies on humans, for the use of physical guidance in addition to the issuance of rewards and penalties during the training of new visuo-motor tasks. This guidance can be viewed as a directing of exploratory behavior to the most relevant parts of the environment.

In the remainder of this thesis, biological evidence for cognitivist learning and physical guidance motivates both the analysis of existing schemes for learning representations, tasks, and maps, and the development of a new neurocomputational system for learning navigation strategies during teleoperation.

CHAPTER 3

# *Representation Learning*

As discussed in the previous chapter, strategy learning embraces a cognitivist view of behavioral learning. Thus, it relates to computational techniques for *learning* and *applying* task-independent models of environments (see Shen, 1994, for a review). This chapter discusses some of the basic computational issues associated with learning abstract representations of environments at multiple scales for the purpose of learning new tasks, concentrating mainly on the two interacting problems of learning *states* and *transitions* between states. For both problems, the chapter critically reviews symbolic, statistical, and neural approaches with the basic requirements of task learning in mind.

## 3.1 General Issues

### 3.1.1 Model Application

The process of applying a learned task-independent model typically refers to the computations involved in *state prediction*, *state estimation*, and *planning* (see Shen, 1994). State prediction entails assessing the consequences of an action sequence, without necessarily executing it, by applying the state transitions in the forward direction. For example, a paperboy (see Section 1.1.2) might use his map to determine where he would end up if he rode down the road a mile or so.

State estimation regards the process of disambiguating between states following the execution of an action sequence using both the instantaneous sensory evidence for each of the states, and the contextual information embodied by state prediction. For example, upon actually riding down the road some distance, the paperboy might use his map to determine where he is by comparing the predicted place with what he sees.

Finally, planning entails using the learned state transitions in the backwards direction in order to determine the best sequence of actions, with respect to some external performance criteria, that

leads to a desired state. The paperboy might, for example, wish to calculate the shortest safe route to a particular house by tracing back through the map to his current position.

Both the reliability and the storage and time efficiency of model application depend on the form of the representation. For instance, the paperboy does not want to represent the information in the map as a list of places and roads with cross references. Such a map would be much too tedious to apply. A better solution might be a metrical map (i.e. a typical road map) or relational map (see Section 5.2.1).

### 3.1.2 Model Construction

In an unknown environment, model learning entails incrementally *constructing* both internal state and the causal relationships between those states. This construction refers to both learning the environment and adapting the learned model to account for changes in the environment and correct for inaccuracies. It entails both *exploring* and *experimenting*. Experimentation refers to correcting errors in the learned model by taking a selected sequence of actions and comparing experiences with expected consequences, while exploration refers to the problem of expanding the learned model by taking a sequence of actions with unknown consequences (see Shen, 1994).

On the one hand, model construction must be rapid in order to assimilate information from only a few passes through the environment. On the other, it must be robust to noise. A paperboy does not, for instance, want to forget an instance where taking a particular road lead to a place in which a dog attacked him, but also does not want to avoid a road altogether simply because it seemed busy on one particular day (perhaps only due to a funeral procession!).

### 3.1.3 On-Line Learning

Ideally, model learning and application must be interleaved. Even partial information about an environment might be useful, and it may otherwise take a long time to gather all the information about an environment and insure the model has been completely learned. As a case in point, the paperboy might use (apply) his neighborhood map effectively even though it may only be approximate and incomplete. Concurrently, he can update (construct) the map with the information he gathers during his delivery, thereby improving the approximation and expanding the map. This concurrent application and construction, often referred to as *on-line* learning (see Shen, 1994), becomes even more important when modelling *dynamic* (changing, or non-static) environments, which require constant model updates. The paperboy must, for example, be able to add new roads and houses to the map as they are built.

Unfortunately, on-line learning also introduces the problem of deciding when to explore and experiment, and when to apply the model to solve problems, often referred to as the "exploration/ exploitation" trade-off (see Singh, 1994). The paperboy gives up the prospect of finishing his route early by actively exploring a new road, but might also discover a shortcut that saves him time in the future — one that he would not have found if he had exploited the information in his current map to get through with the route quickly.

## 3.2 Learning Internal States

### 3.2.1 Generalization vs. Discrimination in Concept Learning

The problem of converting continuous and often uncertain sensations into internal state, often called the *signal-to-symbol problem*, falls under the general AI paradigm of *concept learning*. In the context of model construction, an efficient concept learning system must trade off *generalization* and *discrimination*. On the one hand, it must generalize over inputs with similar characteristics so that states are insensitive to irrelevant attributes (like noise) in the input (see Lippmann, 1989a) and memory requirements are kept to a minimum. The set of possible inputs belonging to a particular state is often called an *equivalence class* (in the sense that the state defines an equivalence relation between each of the elements in the set). On the other hand, it must discriminate between inputs with different characteristics so that states are sensitive to important attributes in the input. The paperboy, for example, must be capable of recognizing his current place in the neighborhood despite the visibility conditions (e.g. lighting), yet also distinguish between places based on the presence or absence of various landmarks.

Discrimination often becomes a problem when the sensory input contains only partial information about the environment (e.g. in *translucent*, as opposed to *transparent*, environments), causing a many-to-one mapping from environment state to internal state (called *perceptual aliasing*). In such cases, the environment is said to contain *hidden state*, differentiable only using contextual (top-down) information (Shen, 1994). An example of this is when the paperboy enters an intersection with very few distinguishing landmarks and cannot distinguish it from another intersection in another part of the neighborhood. But knowing that he came from a given place in the neighborhood along a previously travelled route, he has little difficulty locating himself.

### 3.2.2 General Approaches

Most existing methods for state learning address the generalization/discrimination trade-off by *splitting* states based on relevance (Kaelbling, 1990) or when expectations are not met (Chrisman, 1992), *merging* similar states (Mahadevan & Connell, 1990), or automatically selecting features that define states using genetic algorithms (see Section 4.1.2).

Though continuous state representations are possible (e.g. in adaptive control theory — see Isermann et. al., 1992), most methods for concept learning categorize inputs into discrete states. *Supervised* methods are said to classify sensory input using the correct answers or error feedback provided by an external teacher, while *unsupervised* methods cluster (i.e. self-organize) sensory input into discrete categories automatically. This categorization process can also be on-line and incrementally or adaptively form state representations (see Section 3.1.3), or off-line and form representations using a separate training session. Finally, categorization techniques can either take binary inputs, or continuous-valued inputs. In general, learning and adapting state for task-independent models of real-world environments requires unsupervised, on-line mechanisms with continuous-valued inputs. The paperboy, for example, must be capable of learning a map of his environment based on continuous-valued visual sensations (time varying retinal imagery) without relying on an external teacher.

Existing concept learning systems can be grouped roughly into *symbolic*, *statistical*, and *artificial neural network* approaches, each more expressive (general), but also more complex, than the previous.

### 3.2.3 Symbolic Approaches

Symbolic categorization techniques, by far the most common in the AI literature, form concepts by learning rules from boolean inputs using propositional logic (e.g. k-CNF, k-DNF formulas). Even when these propositions are well-grounded in the sensations of the learner (e.g. using geometric rules), symbolic techniques largely ignore the signal-to-symbol problem. *Fuzzy logic* generalizes logical operations by allowing propositions to exhibit graded (rather then binary) values between TRUE and FALSE, and replacing the AND and OR operators with MIN and MAX, respectively (see Luo, 1989). However, it still does not address how such graded values are initially obtained.

### 3.2.4 Statistical Approaches

Statistical approaches attempt to form categorical distributions of inputs in some $n$-dimensional space. Supervised statistical classifiers include optimum classifiers (which take binary inputs), and k-nearest neighbor classifiers (which take continuous valued inputs). They also include the popular probabilistic Bayesian classification techniques that assume a general form for input distributions (typically Gaussian), learn the parameters of these distributions off-line, and form decision boundaries using Bayesian decision theory. Generally, probabilistic techniques yield a *likelihood* for each state (i.e. the probability of seeing the data, given the state). To determine the current state from the input, one might take the state with the *maximum likelihood* (ML); alternatively, one might assume a prior probability distribution over the states, and use Bayesian inference to determine the *maximum a posteriori probability* (MAP) (see Luo, 1989). *Coarse coding* methods (e.g. Watkins, 1989) are similar, but essentially pre-design the statistics (no learning is involved) and assume a uniform prior distribution.

Unsupervised statistical clustering techniques have been successfully employed to learn the states of real environments (Mahadevan & Connell, 1990). They include the leader clustering algorithm (which takes binary inputs), and the k-means clustering algorithm (which takes continuous valued inputs).

### 3.2.5 Artificial Neural Networks

Artificial neural networks (ANNs) can learn states by using connectionist principals to generalize across inputs, and often achieve low error rates in comparison to traditional Bayesian approaches (Anderson, 1987; for a review, see Lippmann, 1989a,b; Simpson, 1990). In fact, they can not only implement ML classifiers and represent statistical decision boundaries (Lippmann, 1989b), but can also learn and approximate continuous valued functions. *Distributed* ANNs (e.g. backpropagation networks) represent concepts like states and actions with patterns of activity across multiple units, while *localist* ANNs (e.g. ART networks) represent concepts with single units (Elman,

1991). The latter correspond to "grandmother cell" theories of neural representation in the brain (see, for example, Section 2.2.2). Regardless of coding strategy, however, most ANNs require a pre-processing of some sort to deal with the signal-to-symbol problem.

Well-known supervised ANNs with binary valued inputs include Hopfield nets and hamming nets, while supervised ANNs with continuous valued inputs include perceptrons, multi-layer perceptrons (backpropagation networks), adalines (adaptive linear elements), madalines (multiple adalines), multilayer madalines, radial basis function (RBF) classifiers, restricted coulomb energy (RCE) networks, Boltzman machines, and CMAC networks.

Unsupervised ANNs with binary valued inputs include ART1, while unsupervised ANNs with continuous valued inputs include ART2, ART2A, fuzzy-ART, and Kohonen self-organizing feature maps. (Note: ART stands for Adaptive Resonance Theory — see Carpenter & Grossberg, 1991). Of these ANNs, only the ART networks operate on-line, and only the ART2 network operates in continuous time.

## 3.3 State Transition Learning

### 3.3.1 Inductive Learning

State transition learning falls under the AI paradigm of *inductive learning*, which refers to the process of learning causal relationships between concepts. With respect to task-independent model learning, inductive learning pertains to learning the consequences or outcomes of actions and events from predictive failures (Shen, 1994). Note that in a *semi-controllable* environment, such events include the actions of other systems.

Due to the complexity of continuous models, most inductive learning schemes work in discrete time, define discrete actions, and learn discrete state transitions. In general, the same principals that apply to defining states also apply to defining discrete actions (i.e. the generalization/discrimination trade-off). The paperboy, for example, must be capable of discriminating between locomotive actions the lead him to different places, but generalize over actions that lead him to the same place. Note, however, that previous work has largely ignored the problem of learning or representing continuous actions analogically, and symbolic representations have consisted mainly of coarse descretizations. In contrast, many schemes exist for learning discrete state transitions. As with concept learning, these schemes tend to fall into three categories: symbolic, statistical, and neural.

### 3.3.2 Symbolic Approaches

Symbolic methods for state transition learning typically employ highly theoretic algorithms and data structures for learning *deterministic* environments (environments in which action consequences are always the same). One of the most popular frameworks for representing symbolic relational information, especially visual information, is the *semantic network* (see Ballard &

Brown, 1982), a graph-like data structure that represents inferential and analogical relations between world entities. Note, however, that semantic networks provide no mechanisms for learning, nor do they incorporate causation (e.g. actions or events) into relations.

Based on Piaget's constructivist theory of mind, Dresher (1991) has presented another symbolic mechanism which represents an environment using items (binary state elements), actions (which cause transitions between states), and schemas (which encapsulate the consequences of actions in states). The mechanism can construct "spin-off" schemas by learning reliable relations between actions and items, composite actions defined by particular goal states, and synthetic items undefinable in terms of simple conjunctions between other items and actions.

Other symbolic work includes off-line, provably correct algorithms (like L* and CDL+) for learning *finite state machines* (FSMs) like finite state automata (FSAs), push-down automata, and Turing machines (see Shen, 1994). Much work has particularly focussed on FSAs, which basically consist of a finite number of discrete states, a finite number of discrete actions, and a deterministic mapping from state-action pairs to states (transitions). Learning methods for FSAs primarily construct simple graph models of environments like "grid worlds" (Rivest & Schapire, 1987; Betke et. al., 1994). Some of these theoretical methods have even been cast in connectionist architectures (Porat & Feldman, 1991).

Symbolic methods for learning relations rely primarily on classical AI planning techniques like A* and STRIPS (for a review of symbolic planning methods, see Ballard & Brown, 1982; McDermott, 1992). These deterministic symbolic algorithms generally ignore the continuous and uncertain nature of the real world.

### 3.3.3 Statistical Approaches

Statistical methods can deal with probabilistic state inputs, and can represent uncertainty by learning the *stochastic* (non-deterministic) properties of state transitions. Such methods include *Bayesian networks* and *Markov models.*

#### Bayesian Networks

The ever popular Bayesian networks (see Pearl, 1988) represent knowledge with directed acyclic graphs (DAGs), the edges of which are annotated by conditional probabilities. Such networks can be used to perform MAP state estimation. Note, however, that the acyclic nature of these networks makes them unsuitable for representing most environments. An offshoot of Bayesian networks is the Markov network, whose underlying structure is an undirected graph.

#### Markov Models

Many of the more flexible probabilistic methods for state transition learning employ some variant of *first order* Markov models. Like FSAs, these models characterize an environment by discrete states and transitions between states; unlike FSAs, they represent transitions using conditional probabilities. Thus, first order Markov models are representationally more powerful than FSAs.

They do assume, however, that the probability of achieving a particular state at the next time step depends only on the current state and the action performed in the current state (the *Markov assumption*). (Note that higher order models are also possible.)

One particular first order Markov paradigm, the hidden Markov model (HMM), has received much attention in the context of speech modelling, learning, and recognition (see Rabiner & Juang, 1986; Morgan & Scofield, 1991), and has more recently become promising for spatio-temporal pattern recognition (Fielding et. al., 1993) and for controlling selective fixation (Rimey & Brown, 1991; see Section 5.1.1). HMMs can be used to learn, recognize, and estimate the state of environments from temporal sequences using powerful, efficient algorithms for *optimization* (determining the optimal model parameters, given the state sequence and the number of states in the model), *evaluation* (determining the probability of the observed input sequence, given the model), and *estimation* (uncovering the hidden state sequence, given the input sequence and the model). These algorithms are referred to as Baum-Welch reestimation, the forward-backward procedure, and the Viterbi algorithm, respectively.

Note, however, that the Baum-Welch algorithm is off-line; it requires that the number of states be determined a priori, and cannot adapt incrementally to new inputs after training. While HMM learning can be extended to learn the number of states (Chrisman, 1992; Shen, 1994), it nevertheless requires that the model re-learn the training data following the addition of any new states.

### Markov Decision Processes

More general first order Markov models like *Markov decision processes* (MDPs), also called *controlled Markov processes*, have long been popular in the operations research community (see White, 1993; see Appendix E). MDPs model the environment using transition probabilities conditional on both discrete, deterministic states and actions, and define immediate rewards over the states and action transitions. In the context of this thesis, MDPs represent an entire strategy, composed of a task-independent model (states and transitions) and a utility evaluation over that model (rewards). (See Section 4.1.3 for a more detailed discussion of how this relates to generating policies for new behaviors.)

Partially observable MDPs (POMDPs) additionally relax the assumption that the current state is known at all times; rather, the current state of the system is indicated as a probability distribution (White, 1993). One can perform state estimation in a POMDP using Bayesian updating principals to combine the incoming probability distribution, obtained from the current observation, and the expected probability distribution, computed using the previous state distribution and the current actions. Using an MDP, one can also evaluate and compare the expected *value function* (the total or average reward received up to some specified time horizon as a function of state) of various action policies, and plan by determining the *optimal policy*, the action policy that maximizes the value function at the current state. This determination of the value function generally requires either *value iteration* or *policy iteration*, both of which are dynamic programming techniques for solving Bellman's equation (see White, 1993; see Appendix E).

Unfortunately, finding the optimal policy requires exponential time with respect to the number of states (more on this later in Section 4.1.3). Furthermore, the learning of MDPs has proven quite

difficult (Lin, 1993). Sutton's (1990) Dyna architecture learns an environment model incrementally and quickly in order to aid reinforcement learning (see Section 4.1.3), but assumes the world to be deterministic (see also Lin, 1993). Other systems estimate transition probabilities using the classical ML (maximum likelihood) approach (see Section 3.2.4), which is similar to the stochastic approximation methods typically employed for estimating Markov chain probabilities in the operations research community (Sato et. al., 1982; Moore & Atkeson, 1993; Singh, 1994). But these statistical methods require many iterations in order to calculate statistically accurate running averages.

In addition to problems with learning MDPs, many real-world scenarios are not easily modelled using the first order Markov assumption (or, for that matter, FSAs). For instance, actions may have delayed effects or "momentum", and transitions between states may be caused by external events rather than controllable actions (Materic, 1994). Even nearly Markovian environments might cause an MDP to perform badly due to just a single failure to distinguish between two states (Whitehead, 1992; Singh, 1994).

On the other hand, the conditions for Markov modelling have long been studied in the operations research community, and it is unclear whether an appropriate Markovian state and action representation does not always exist for any given real-world scenario (perhaps at the expense of computational and storage efficiency). Non-Markovian effects (e.g. the delayed effects of actions) can often be considered part of the state space, while external events can often be considered part of the action space. Furthermore, the incorporation of partially observable states diminishes the effects of non-Markovian environments by avoiding strict decisions.

### 3.3.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are simply ANNs with adaptable feedback connections between layers that can be viewed as context or state. Unlike time-delay neural networks (TDNNs), which can only learn isolated state sequences, RNNs can implement FSMs (see Sontag, 1995). They can also approximate complex dynamical systems, which generalize FSMs by allowing continuous and graded states, actions, events, and time (Shen, 1994). Dynamical systems minimally specify a state space and state transformation equations (either stochastic or deterministic) as a function of time. RNNs most often approximate dynamical systems with discrete but graded states, actions, and time. (Dynamical RNNs that operate in continuous time have also been employed, but these systems nevertheless have characteristic time constants — see, for example, Beer & Gallagher, 1992; Yamauchi & Beer, 1994.) RNNs have especially been successful for speech processing (see Morgan & Scofield, 1991, for a review). As with any connectionist approach, RNNs come in two varieties: distributed and localist.

#### Distributed RNNs

Because distributed RNNs naturally develop novel representations and generalize to new inputs, they develop their own dynamic state. Intuitively, they envelope the feedforward, supervised neural mechanisms for state estimation mentioned in Section 3.2.5. However, they in general require much training and are very difficult to design and analyze (Elman, 1991).

Jordan & Rumelhart (1992) have presented a general RNN architecture for learning *forward models* (internal representations of states and action consequences) based on prediction errors (differences between model predictions and environment sensations). This architecture consists of an input layer with state and action units, several hidden layers, and an output layer of predicted sensation units. The architecture represents state using one of the hidden layers, which is fed back to the state units in the input layer. In principal this architecture can employ localist principals, but has to date only been demonstrated with distributed networks using backpropagation (i.e., distributed principals) in the context of learning control problems (see Section 4.2).

In an effort to separate visible output states from hidden recurrent states and avoid the unstable problem of backpropagation through time, Elman (1991) has proposed an extension of this architecture, called a *simple* RNN (SRNN). This distributed network consists of an input layer, an output layer, and a single hidden layer, which feeds and receives input from an additional context layer. Several researchers argue that SRNNs can represent an even broader class of machines than FSAs, and have shown that SRNNs can learn simple finite state grammars (FSGs) by incrementally encoding temporal context (Elman, 1991; Servan-Schreiber et. al., 1991). In fact, SRNNs come to represent FSG states with individual hidden units, provided that they initially contain at least as many units as there are states (Servan-Schreiber et. al., 1991). These units exhibit graded responses reminiscent of conditional probabilities. However, SRNNs require much training, and generally fail to learn more complex grammars (Servan-Schreiber et. al., 1991; Mozer & Bachrach, 1991).

In response to this inability, several researchers have proposed algorithms for learning with *fully connected* RNNs (FCRNNs). FCRNNs impose constraints on neither the connections between units, nor the receiving of inputs. They have been shown to learn even complex FSGs; however, such learning generally requires an immense amount of computation time (Smith & Zipser, 1989).

**Localist RNNs**

Localist RNNs generally permit faster learning and allow easy implementation and analysis, but require greater structural design (Elman, 1991). Mozer & Bachrach (1991) have proposed a localist RNN for learning FSAs using an update graph (Rivest & Schapire, 1989). The network uses a set of learned weights to represent state transitions, and multiplicatively gates (modulates) these transitions by mutually exclusive actions. By satisfying a number of weight constraints using a combination of gradient descent and normalization, the network insures that the weights converge to mutually exclusive 0-1 transitions for each state-action pair. This learning algorithm degrades gradually with increasing noise, and outperforms the corresponding theoretical learner for update graphs (i.e. that of Rivest & Schapire, 1989) in simple environments, provided that the network initially contains enough units. Furthermore, the learned network can make predictions, which may be fed back and averaged with the incoming sensory input. However, the network's performance declines rapidly for complex FSAs with many states. Attempts at backpropagating errors over multiple time steps to improve learning have enjoyed very limited success.

Localist RNNs have also been used to implement first order Markov models (see Lippmann, 1989a,b; Morgan & Scofield, 1991; Bourlard & Morgan, 1995). Sun et. al. (1990) have shown an HMM to be a special case of linear, second-order RNN, and have proposed using the Baum-

Welch reestimation formula to learn such networks. In related work, Bourlard & Wellekens (1988) have proven that RNNs like the ones proposed by Jordan & Rumelhart (1992) can calculate probabilities for HMMs (see Lippmann, 1989b). Finally, Santini & Del Bimbo (1995) have shown that RNNs can be trained to implement MAP classifiers.

## 3.4 Hierarchical Representations

Together, state and state transition learning constitute a task-independent model of an environment at a single *scale*. In terms of state learning, this scale is determined by the degree to which the model generalizes over the sensory input. In terms of state transition learning, the scale is determined by the degree to which the model generalizes over the actions. In both cases, the appropriate scale for the model depends on the task. Some tasks require precise control over behavior, and therefore a greater amount of distinguishability between states and actions (small scale). Others require only coarse control over behavior, and therefore warrant more generalization over states and actions (large scale). For example, a paperboy must employ a detailed map in order to perform his route within, say, a city apartment complex, but only requires a coarse map to deliver papers in a rural neighborhood. Thus, the ability to perform a variety of different tasks efficiently often requires learning several task-independent models at different scales.

One way to learn task-independent models at multiple scales is to define states and transitions hierarchically. That is, equivalence classes of states at one level of the hierarchy define states at the next level up. Hierarchical representations not only suggest a relatively simple method to learn states at multiple scales, namely by pooling less general states into more general states, but also make way for the ability to use *hierarchical planning* techniques (Albus, 1985; see also Luo, 1989; Ballard & Brown, 1982). In this paradigm (also called *hierarchical control* or *hierarchical problem solving*), planning is performed first within a large scale representation of the environment, the results of which constitute a set of goals for planning in a smaller scale representation, and so on. At any scale, the goals are close enough to the current state to solve the problem relatively quickly. For example, the paperboy might employ several more detailed maps of particularly densely populated areas in the neighborhood, as well as a more coarse map representing the relationships between these different areas. He could then plan first within the coarse map, thereby determining a qualitative route from one area to the next, and within each area plan a more accurate path using the appropriate detailed map. Though this processes does not guarantee that the resulting plan will be optimal, it often yields a *satisficing* (i.e. adequate and useful) plan that is nearly optimal.

## 3.5 Summary

This chapter has presented the general issues of learning and applying abstract, task-independent representations of environments on-line as part of the process of learning new tasks. Model construction refers to building and updating a model based on experiences during exploration and experimentation. This process must trade off the speed of learning with robustness to noise. Model application refers to the processes of predicting consequences of actions, estimating cur-

rent state, and planning optimal action policies. The effectiveness of model application greatly depends on the form of representation employed. Interleaved (on-line) construction and application implies a trade-off between exploration and exploitation.

The problems of learning states and transitions between states have been discussed in the general contexts of concept learning and inductive learning, respectively. Concept learning algorithms in real-world environments, in addition to adhering to the general requirements for model learning, must trade off generalization with discrimination, and in general call for unsupervised techniques that can handle continuous-valued sensory inputs. Inductive learning algorithms must employ these same principals to represent actions. For the most part, existing techniques fall short of these requirements. Symbolic approaches are theoretically sound, but ignore the signal-to-symbol problem and the continuous and noisy characteristics of realistic environments. Statistical approaches (particularly Markov decision processes) offer powerful techniques for model application, but lack adequate methods for on-line model learning. Finally, unsupervised neural approaches show much promise for learning concepts and transitions due to their ability to approximate dynamical systems, but are not yet capable of quickly learning action-induced transitions between concept categories. Regardless of the approach, learning environment states and transitions can be performed on multiple scales, which can aid the process of planning satisficing policies through the use of hierarchical planning techniques.

In the next chapter, we use the principals put forth in this chapter in order to computationally assess both cognitivist and behaviorist approaches to task-learning. These principals also come into play later in Chapter 5 when discussing various AI methods and biological models for map-making and navigation.

CHAPTER 4

# *Task Learning and Teleprogramming*

Strategy learning entails learning new tasks through the guidance of an external teacher based on a cognitivist theory of behavioral learning. A learned strategy can be used in conjunction with on-line planning methods to compute an *optimal action policy* (sometimes called an *action map* — see Kaelbling, 1990) for performing a particular task, given a set of goals. The act of following this policy produces an optimal behavior for the task.

Recent interest in task learning has primarily resulted from a proliferation of behavior-based approaches to AI. In contrast to *deliberative* (planning-based) systems, which use task-independent models of environments to plan actions, behavior-based systems generate actions directly using local, distributed calculations involving shallow structures such as lookup tables and combinational logic. Examples include animat approaches, reactive planning paradigms, motor-schemas, the subsumption architecture (Brooks, 1991), situated action approaches (Kaelbling, 1990), and other hybrid approaches combining purely reactive (no internal state) and deliberative methods (Payton et. al., 1990).

Brooks (1991) suggests four types of learning for behavior-based systems, addressing the premise that learning should only be utilized to deduce information that is not readily available to the programmer a priori: *calibration learning*, *representation learning*, *coordination learning*, and *new behavior learning*. Task learning primarily addresses the learning of new behaviors through the generation of an optimal action policy (lookup table). Strong motivations for automatic policy generation stem from a desire to spawn complex yet useful behaviors and modify them as goals change, without having to program or hard wire them by trial-and-error through ad hoc design.

This chapter discusses and evaluates the existing methods for task learning on computational grounds. It compares *direct* and *indirect* methods of reinforcement learning, respectively corresponding to cognitivist and behaviorist models in biology, and to deliberative and behavior-based approaches to AI. The chapter also illuminates important qualitative relationships between reinforcement learning and learning optimal control. This discussion leads to a review of teleprogramming techniques for guidance-based task learning.

## 4.1 Reinforcement Learning

In computational terms, the reinforcement learning (RL) paradigm specifies a broad class of algorithms for automatically generating optimal action selection policies with respect to maximizing some scalar reward over a period of time. This reward can originate either externally (e.g. from a teacher or the environment) or internally (e.g. as a function of state, action, or combination thereof). Because no explicit solutions are provided, reinforcement learning algorithms are generally considered unsupervised.

### 4.1.1 General Issues

In addition to the representational issues associated with defining internal states and representing actions and events (see Section 3.2), the major technical issues associated with RL concern *exploration* and *credit assignment*. Exploration addresses the fact that a variety of actions must be executed in a number of states (perhaps multiple times) before a task can be learned with any amount of competence. To be practical, RL requires efficient exploration methods, which can be thought of as a directing of attention. For example, without an previous knowledge, a paperboy must actually traverse the neighborhood several times before he can learn how to complete his route competently.

Credit assignment refers to the problem of rating actions in states based on the receipt of some reinforcement signal. The *temporal credit assignment* problem (Sutton, 1988) addresses the fact that reinforcement might be delayed in time with respect to actions (e.g. when presented only when a goal is reached). For example, a paperboy does not know that riding down a new road leads him to a desired place until he actually reaches that place a few minutes later. *Structural credit assignment* refers to the need to generalize reinforcement over similar actions across similar states (see Section 3.2). For example, a paperboy must be capable of deducing that riding down the left side of the road will take him to the same place as riding down the right side.

Possible criteria for evaluating RL techniques include noise immunity, convergence rate, incrementality, computational tractability in time and space, and groundedness with respect to incoming sensory data (Mahadevan & Connell, 1992). In evaluating RL techniques it is important, but often neglected, to conceptually divide and understand the difference between indirect (sometimes called *model-based*) RL paradigms, which learn models (referred to here as a strategy) as an intermediate step to optimal policy generation, and direct (sometimes called *model-free*) RL methods, which learn only the states of the environment and infer and optimal action policy directly (Moore & Atkeson, 1993; see Singh, 1994; see Figure 15). This distinction parallels the difference between cognitivist (i.e. internal representation) and behaviorist (i.e. stimulus-response-reward) theories of behavior, respectively (see Section 2.1). To determine which of these approaches is computationally more appropriate for any given learning problem, one must especially consider the trade-off between *goal generalization* (the ability to use learned information to achieve multiple goals) and *planning efficiency* (the time necessary to determine a course of action to achieve a particular goal).

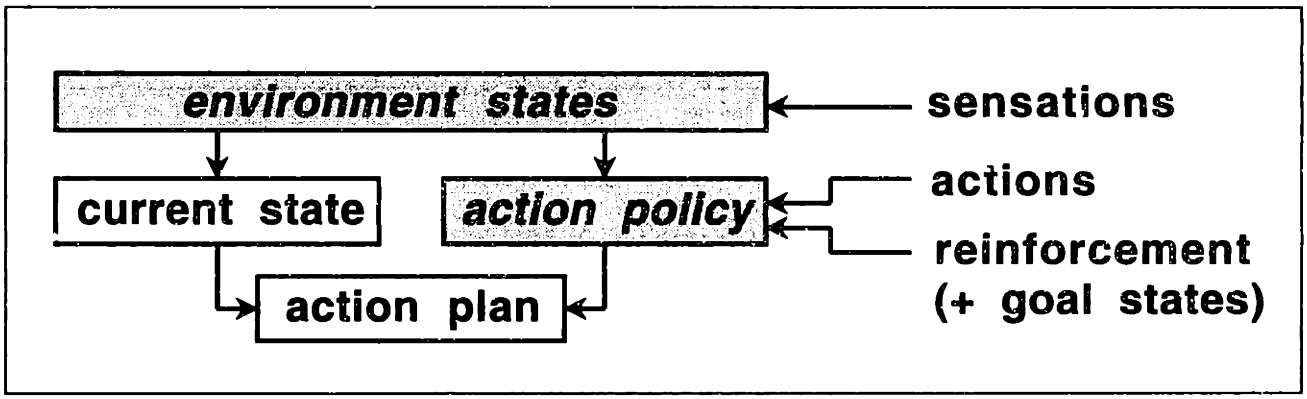


**FIGURE 15: THE CONCEPT OF DIRECT REINFORCEMENT LEARNING.**
*Direct reinforcement learning entails learning only the environment states as an intermediate step to learning the action policy, as opposed to learning a detailed model of the environment (cf Figure 1). Given the current state, determined using the learned environment states, the action policy might then be used to compute an action plan. (Note: Shaded boxes denote where learning occurs.)*

### 4.1.2 Direct Reinforcement Learning

Direct RL has recently become very popular for learning new policies for behavior, mainly because it requires no on-line planning, which can be time-consuming and impractical for many situations (e.g. optimal policy determination for POMDPs). The planning operation is instead embedded in a learning rule. This advantage makes direct methods particularly attractive for tasks that are well-suited for reactive approaches, such as obstacle avoidance (see Kaelbling, 1990), and for tasks with fixed goals and immense state spaces, such as games.

However, because the learned policy is static, stationary, and global (i.e. can be affected by any portion of the environment), it must be re-learned when the environment, performance criteria, or goals change. Some researchers claim that incorporating goals into state definitions can solve this problem. But goal-specific states only help a system to remember the policies for certain goals; they do not address the problem of having to completely learn policies for new goals from scratch. This drawback makes direct RL quite unattractive for learning domains like visual navigation. For example, a paperboy using a direct approach would have to re-learn his locomotive action policy every time he wanted to ride to a different house in the neighborhood. Thus, while direct RL techniques have a high planning efficiency, they do not generalize over goals.

Direct RL methods are also particularly susceptible to perceptual aliasing (see Section 3.2.1). Because they do not explicitly learn action consequences, they cannot use history to disambiguate between two similar hidden states, and must instead perform further actions (Whitehead, 1992) or keep a short-term history (Cassandra et al., 1994) to resolve ambiguity. To prevent this dilemma, direct RL methods have no choice but to increase the size of the state space in order to make as many states distinguishable as possible. Of course, as states quantize an environment more finely, the structural credit assignment becomes more prevalent.

Direct methods can be classified as *global*, *local*, or *behavior-based*.

### Global Methods: Genetic Algorithms

Policies can be thought of as rules or programs for acting in environments. Several global methods for learning policies capitalize on this idea by using *genetic algorithms* (GAs). Instead of incrementally constructing programs, GAs attempt to automatically "breed" or "evolve" programs for tasks via selective search through a multidimensional parameter space, specifically by generating an initial random population of programs and improving it using an iterative procedure (see Koza et. al., 1992). This procedure executes and evaluates the fitness of each program in the current population with respect to some performance criteria, and replaces this population with copies of well fit programs (called *reproduction*) as well as new programs created through a genetic recombination of two programs (called *crossover*). To be practical, GAs must keep the search space small by specifying a general structure for programs. However, they do not necessarily require a prior definition of states and actions (Yamauchi & Beer, 1994). GAs have been applied to various problems in optimal control, game theory, planning, artificial life, and empirical discovery.

With regard to learning action policies, GAs solve the exploration problem by searching using *mutation*, and address the credit assignment problem (both temporal and structural) through a global competition between programs with respect to an overall performance measure. One particularly popular algorithm for this is the *bucket brigade* algorithm for classifier systems (Holland, 1985). Unfortunately, neither symbolic nor neural forms of GAs scale well to larger problems such as those encountered in a real environment (Grefenstette, 1992; Koza et. al., 1992; Beer & Gallagher, 1992), mainly because the search space becomes prohibitively large.

### Local Methods: Temporal Difference Learning

By far, the most popular class of direct RL methods is *temporal difference learning* (TD), due to Sutton (1988; see Singh, 1994, for a review). TD methods address the temporal credit assignment problem by incrementally learning action policies using an error signal computed from temporally successive predictions. Popular TD methods include Q-learning (Watkins, 1989), which directly learns the expected payoff over time for each state-action pair (called Q-values), and Adaptive Heuristic Critic (AHC) (Barto et. al., 1983; see also Simpson, 1990), a neural algorithm. Both of these methods have been successfully implemented, mainly for environments represented by discrete states and actions.

Techniques for solving the structural credit assignment problem using TD include ANNs (Anderson, 1987), coarse coding (Watkins, 1989), decision-tree coding (Chapman & Kaelbling, 1991), variable resolution state maps (Moore, 1991), and the use of GAs to better organize states (Grefenstette et. al., 1990). "Soft" algorithms also generalize over similar actions by adding Gaussian noise to discrete actions (Singh, 1994).

Techniques for exploring the state-action space for TD include probabilistic distribution methods (Watkins, 1989; Sutton, 1990), which direct exploration by a non-stationary policy defined by Gibb's distribution over Q-values; optimistic initial value methods (Sutton, 1990; Kaelbling, 1990), which give initially high estimates to all the Q-values; interval estimates (Kaelbling, 1990), which estimate confidence for each action and directs exploration to maximize the upper bound on expected payoff; perturbation methods (Whitehead, 1992), which specify actions that

deviate slightly from the current best guess at the optimal policy; and statistical counting methods (Singh, 1994), which keep track of the number of times each action has been executed in each state. In addition to these unsupervised exploration methods, Lin (1990) has suggested the role of external teacher in directing exploration, which would speed up the learning process in large state spaces, and keep the system out of local minima. This idea relates to guidance-based training of humans (see Section 2.3).

TD methods have been influenced by Learning Automata Theory (Narendra & Thathacher, 1989). In particular, TD has been likened to estimating the optimal policy for an MDP through *asynchronous dynamic programming* reminiscent of value iteration (Watkins, 1989; Werbos, 1987). Some methods even extend TD methods to POMDPs by letting the sensory input define the state vector (Singh, 1994), though it remains unclear whether raw sensory input can give a good indication of Markovian state, since algorithms of this type have only been applied to relatively simple environments.

Note, however, that TD learning converges asymptotically, and in general requires an infinite number of learning trials (Watkins, 1989). Moreover, the order of state visitation can drastically affect the rate of convergence, and there exist no sound metrics for optimizing this order, or for determining when learning has converged to the optimal policy (Singh, 1994). Other disadvantages of TD include space complexity and parameter sensitivity (Mataric, 1991). Furthermore, most TD methods assume a finite environment, and that the number of states are known in advance (Singh, 1994). Finally, TD cannot accurately propagate credit over a large number of states unless the reinforcement signal is error-free (Lin, 1993). Lin (1993) suggests using hierarchical control techniques to solve this problem (see Section 3.4). In any event, even RL methods with partially observable state have not yet been implemented in complex, real-world environments.

**Behavior-Based Methods: Coordination Learning**

Aside from TD and genetic algorithms, a few behavior-based approaches to RL simply bypass the temporal credit assignment problem altogether, namely by assuming that some performance feedback (e.g. "progress estimators") is always available to the learner (Maes & Brooks, 1990; Mataric, 1994; Colombetti & Dorigo, 1994). However, these statistical methods have been developed and implemented in the context of generating compound behaviors by learning to coordinate designed policies for simple behaviors, rather than learning new policies for behaviors from scratch (see distinction made in Section 2.1). This formulation of the problem (conditions and behaviors rather than states and actions) keeps the state space manageable (Mataric, 1994), and can be likened to hierarchical control (see Section 3.4).

### 4.1.3 Indirect Reinforcement Learning

There are many advantages of learning strategies (i.e. with task-independent environment models) as an intermediate step to action policy generation. Indirect forms of RL have a simpler form of reinforcement (Kaelbling, 1990). In particular, the temporal credit assignment problem is trivially solved, since reinforcement can be directly associated with the states and actions defined by the task-independent model. The strategy can also disambiguate between similar states using his-

tory, thereby solving the perceptual aliasing problem. Perhaps the greatest advantage, however, regards the ability to generalize over possible goals and tolerate or adapt to environmental changes. Goal changes require no new learning, only a re-planning. Furthermore, small changes to the environment require only the adaptation of the local parts of the strategy that have been affected, rather than re-learning the entire strategy (see, for example, the discussion in Section 1.1.2). Therefore, the computational expense of building the task-independent model can be "amortized" across the set of tasks (Singh, 1994).

Of course, the trade-off between goal generalization and planning efficiency remains. A strategy has much larger storage requirements than a policy, since it must encompass the consequences of taking each of the actions in each of the states. The need to perform planning is also cumbersome and time-consuming, especially in large environments. For this reason, such deliberative approaches have been extensively argued against by proponents of behavior-based intelligence (Brooks, 1991; Mataric, 1994).

Fortunately, policies need not be optimal in practice, and useful (i.e. satisficing) policies often demand very little time to compute (Shen, 1994). For example, the determination of satisficing policies for POMDPs seldom requires many value iterations (see White, 1993; Shen, 1994; see Appendix E). Satisficing solutions can also be determined using hierarchical planning techniques (see Section 3.4). Singh (1994), for example, has implemented a simple hierarchical scheme wherein all the possible goal states of a primitive (low) level define the states at an abstract (high) level. For tasks with more stringent demands, the use of learned models does not preclude the off-line "compilation" of action policies, often called *universal plans* (Kaelbling, 1990). Finally, it may be possible to interleave planning with incremental learning and control, since incremental changes to the model seldom affect the optimal action policy greatly.

Unfortunately, very few existing methods for learning environment models easily apply to real environments (see Chapter 3). The use of learned models for RL has primarily been restricted to aiding or accelerating direct RL methods. In the GA literature, it has merely been suggested that reactive and deliberative style approaches could be combined by including the agent's current goal as one of its current sensors, and by changing this goal sensor with a planning module (Grefenstette, 1992). In the TD literature, task-independent models are referred to as *world models* (Sutton, 1990), and strategies are referred to as *action-models* (Lin, 1993). Lin (1992) suggests that action-models are useful for experiencing the consequences of actions without actually participating in the real world. Sutton (1990) uses a deterministic world model for relaxation planning, a shallow look-ahead similar to dynamic programming that can perform limited hypothetical searches over possible situations, thereby speeding up exploration (see also Singh, 1994). Neither of these methods attempts to generalize across actions or states. Moore & Atkeson (1993) have presented a related technique, called "prioritized sweeping", and generalize this technique using memory-based function approximators that assume local smoothness of states (Moore, 1991). None of these methods provide practical mechanisms for learning the full, accurate description of the underlying POMDP (see Section 3.3.3).

Autonomous exploration techniques for model-based reinforcement include competence maps (Thrun & Moller, 1992), which estimate the errors in the policy over the state space (similar to the update graph of Rivest & Schapire, 1989; see Section 3.3.2), and methods that estimate inverse

models (Moore & Atkeson, 1993), which are used to maintain a priority queue of states ordered by the expected magnitude of change during future updates. Several researchers have also devised methods that dramatically speed up the learning process by allowing a human teacher to bias exploration, either by directly changing strategic actions in the current action policy (Clouse & Utgoff, 1992), or by allowing the system to learn by "watching" a human operator act in a part of the state space likely to be part of the solution (Whitehead, 1992). This latter technique is similar to the idea of teletraining proposed in this thesis.

## 4.2 Relationships to Optimal Control Learning

Optimal control learning refers to a broad class of methods for learning control laws for physical tasks from experience rather than designing them a priori. Most of these methods employ ANN controllers (see Miller et. al., 1990), but others have also been proposed. Like methods for adaptive control theory (Isermann et. al., 1992), all methods for learning control require some sort of correct answer or error feedback rather than an overall evaluation; therefore, unlike RL, it is considered a form of supervised learning. Intuitively, optimal control learning is learning with an external teacher, while RL is learning with an external "critic" (Kaelbling, 1990). However, like RL, the learning of control laws entails finding a mapping from sensations to actions (called an *inverse model* of an environment), and the feedback required to learn such mappings might arise from the environment or an external teacher. In fact, RL has been linked to optimal control theory (Watkins, 1989), and RL algorithms can be used to solve supervised learning problems, and vice versa, albeit at the expense of computational efficiency (Jordan & Rumelhart, 1992). Therefore, many of the issues for control law learning are analogous to those of RL.

### Direct vs. Indirect Learning

One analogous issue concerns the learning of task-independent models versus direct learning. Unlike traditional adaptive control techniques that require a model of the plant (called the *forward model*), or at least an estimate of this model (see Isermann et. al., 1992), methods for control learning have the option of directly learning the inverse model.

To date, most studies have shown indirect methods to be superior. Atkeson et. al. (1988) have demonstrated the importance of forward models for performance. Jordan & Rumelhart (1992) have compared *distal* (using forward models) to direct methods, and cite difficulties with direct methods due to possible many-to-one mappings from actions to sensations. In some cases, an infinite number of inverse models exist, which presents problems for corresponding learning algorithms. They also note that direct inverse modelling is not goal directed — there is no direct way to find an action that produces a desired sensation without interpolating. On the other hand, the distal approach entails concurrently learning a forward model of the environment, predicting the sensations for executed actions using the current model, and learning the inverse model using the prediction error. This approach guarantees the generation of a single, particular inverse model, perhaps with certain desired properties. In accordance with these arguments, Jordan & Rumelhart (1992) found the distal method more effective than the direct method for learning manipulator control problems using RNNs trained using backpropagation (see Section 3.3.4).

**Control Learning vs. Coordination**

Another related issue for control learning concerns the difference between learning complex control laws from scratch and learning how to switch between simple control laws. Like RL methods that learn to coordinate behaviors, complex control problems can be solved by switching between or selecting several simple control laws, namely by linearizing the control plant in regions and gain scheduling. Jacobs & Jordan (1993), for example, have proposed a probabilistic neural controller for partitioning the parameter space into such regions using gradient descent and competitive learning. Again, these processes can be likened to hierarchical control techniques.

**Learning-by-Doing**

A final parallel between control learning and RL problems, is that both can be performed using a "learning-by-doing" methodology. In RL, learning-by-doing takes the form of a directing of exploration by an external teacher. In control learning, it entails direct guidance. Jordan & Rumelhart (1992) call this guidance *imitation*, referring to the underlying inference of a teacher's intentions, as opposed to *envisioning*, which involves converting abstract goals into sensations using deductive and inductive reasoning without considering specific actions.

Learning-by-doing, though certainly not new to the robotics community, has recently become prevalent for teaching control skills to robots through the guidance of an operator. For example, Asada & Liu (1990) have taught a manipulator to perform deburring tasks through repeated demonstration. Their system uses an ANN to cluster features like forces, positions, sounds, and grinding wheel speed and torque, and directly associates appropriate actions with each cluster. Another example is the work of Pomerleau (1993), who uses ANNs to teach a computer to drive a car by demonstration. When this form of teaching is performed from a distance, it is sometimes referred to as *teleprogramming*.

## 4.3 Teleprogramming for Supervisory Control

Strategy learning provides a framework for teaching representations and tasks. As discussed in previous sections (see Sections 2.3 and 4.2), these processes can benefit from human intervention, both in the guidance of exploratory behavior, and in the issuance of rewards and penalties. In the robotics community, this idea has been put forth as *teleprogramming* (term due to Alami et al, 1990), wherein a human supervisor "programs" or "teaches" a robot by guiding it through a task via direct *teleoperation*, the extension of a human operator's low level sensing and manipulation abilities to a remote site (Sheridan, 1992).

### 4.3.1 The Supervisory Control Paradigm

Many researchers have pondered (without necessarily naming) the idea of teleprogramming in the context of *supervisory control* (Freedy & Weltman, 1973; Thompson, 1976; Parker & Pin, 1988; Sheridan, 1988). In this paradigm, a human supervisor sends his intent to a semi-autonomous system via high-level, low-bandwidth commands (e.g. goals, constraints, plans, and suggestions), and receives back low-bandwidth status information (e.g. accomplishments, difficulties, concerns, and other high level information) (Sheridan, 1992). Both of these communication channels may

be bi-directional in order to provide direct feedback. The form of communication may be *analogic* (e.g. signals from joysticks, master-slave mechanisms, knobs, and dials), *symbolic* (e.g. syntactic constructs from key, textural, or language input with distinguishable meaning), or a combination of the two. Finally, the semi-autonomous system might be either deliberative or behavior-based (see Bellingham & Humphrey, 1990).

Supervisory control contrasts with a relatively high-bandwidth continuous control via direct teleoperation. Both supervisory control and direct teleoperation are appropriate for tasks that are unsuitable for autonomous systems due to their complexity, yet also unsuitable for humans because they present safety hazards (e.g. undersea, space, terrestrial mining, and hazardous waste cleanup), exhibit information overload (e.g. flight control systems, nuclear power plants), or both (e.g. aids for the handicapped, firefighting, policing, military operations, transportation systems, flexible manufacturing systems). But supervisory control is especially appropriate for tasks characterized by impoverished communications (e.g. low bandwidth, time delays, or reliability). In such environments, teleoperated systems perform poorly; they become unstable unless the operator uses a *move-and-wait* technique, which is time consuming, or receives feedback from a *predictive display*, which requires an accurate, detailed world model (Sheridan, 1992).

Sheridan (1992) describes five duties for the human supervisor within a typical supervisory control system: *high-level planning*, *monitoring*, *intervening*, *learning*, and *teaching*. Teleprogramming addresses the *teaching* duty of the human operator. The semi-autonomous system learns tasks by observation based on the sensations experienced during teleoperation, which establishes the low level competencies responsible for carrying out high level commands. (Alternative methods for teaching in supervisory control are mainly rule-based. For example, Buharali & Sheridan (1982) successfully taught a system to drive a car by repeatedly giving the system rules based on fuzzy logic.)

### 4.3.2 Skill Acquisition for Telerobotics

Recently, teleprogramming has become a particularly popular method for guidance-based skill acquisition for the supervisory control of robots, often referred to as *telerobotics*. For example, Yoerger (1982) has used a hierarchical, teleoperative control structure to teach desired trajectories to robot manipulators. The manipulator could be taught by moving it along the boundaries of the desired workspace, by moving it to a series of positions and entering curve fitting parameters, or by guiding it along the desired trajectory itself.

Hirzinger & Landzettel (1985) have used an associative memory architecture to learn and adapt both the force and rate controls for a manipulator. Brooks (1989) has taught a telerobot by demonstration with a joystick and a sensor ball using a FSA controller with pre-defined states and action transitions (see Section 3.3.2). The current state was determined by matching the pattern of sensory input with stored templates determined using a correlation-based method.

Cannon (1992) has presented a controller interface for teaching a mobile robot object manipulation tasks. Following a specification of the task, and the manual pointing of the robot's cameras at an object by a teacher, the robot learns through demonstration how to move its base and 6-DOF

arm, without previous knowledge of the object or its position. Finally, Funda & Paul (1992) teach trajectories to a manipulator in a simulated polyhedral model of an environment, created using a graphic simulator.

Note that all of these efforts employ teleprogramming as a means for learning control. That is, they use teleprogramming to provide error feedback to a supervised learning system. This strategy contrasts with the notion of *teletraining* put forth later in this thesis, wherein teleprogramming is used as a means for exploration in the context of reinforcement learning.

### 4.3.3 Teaching Interfaces for Teleprogramming

The appropriate mechanisms for teleprogramming largely depend on the targeted task. Considerations include both the *form* and *fidelity* of the command interface for teaching.

#### Form of the Command Interface

In general, it has been suggested that a combination of analogic and symbolic commands should be used to control teleoperators, since complex geometrical and temporal configurations of objects are not readily described symbolically for many tasks (Ferrell, 1973). This observation can be likened to the signal-to-symbol problem described in Section 3.2. Along these lines, Grossman (1977) clearly showed that a combination of guiding and symbolic programming to be faster than a purely symbolic programming system.

Note, however, that there is the distinction between *analogic commands* and *analogic teaching*. Analogic teaching involves a physical guidance of the robot, which could be performed using either analogic or symbolic commands (Sheridan, 1992). On a macroscopic scale, analogic and symbolic teaching respectively correspond to communicating to a learner indirectly through *sensory channels* or directly through *internal channels*. Sensory channels require analogic teaching because they preprocess the input in order to convert sensations into categorized internal states and actions. Analogic teaching is more difficult to realize in an artificial system, since it requires solving the signal-to-symbol problem, but it makes teaching much easier when the space of the task is continuous (as is true for motor skills).

In contrast, internal channels require symbolic interfacing because they directly interface to the conceptual categories like states, actions, pain, and pleasure. Symbolic teaching is easier to implement, since inputs to the learning system need no pre-processing, but it requires that the teacher know the internal structure of the learner. Note that spoken language is one way in which to bridge the gap between symbolic and analogic teaching. Speech recognition can be viewed as a preprocessing step that converts analogic sounds into appropriate semantic categories. For example, a reinforcement learning system might take as input spoken words like "No!" and "Yes!", "Good!" and "Bad!".

Ideally, a natural, responsive communications interface for guidance control should be used for teleprogramming. Just as detailed verbal instruction is not particularly suited for task learning in humans (see Section 2.3), one would especially like to avoid the use of tedious symbolic control languages for robots. Arguably, physical guidance is easier when the control interface is a natural

extension of the operator's own sensing and manipulation capabilities. Unfortunately, the difficulty of programming decisions using analogic teaching has led to the development of sophisticated symbolic teaching interfaces ranging from teach boxes to programming language systems.

### Fidelity of the Command Interface

A related consideration in teleprogramming concerns the quality and amount of sensory (e.g. visual) feedback provided to the human operator. For teleoperation in general, it is important that the robot and human operator accurately perceive the actions, sensations, and abilities of one another (Parker & Pin, 1988). With respect to teleprogramming, humans can teach a robot only what it is capable of sensing (Lin, 1993), and the effectiveness of such teaching can only be as good as what the human is capable of sensing from feedback. The first of these requirements can be addressed by providing enough sensors to the robot, while the second can be addressed by displaying the output of those sensors effectively to the human operator.

Some researchers have used graphical displays to present sensor output to the human operator. For example, Wang et. al. (1993) use a mouse to control a submersible by dragging its computer generated image, which is overlaid upon a fish-eye view of the world. However, human factors studies have indicated that it is important to arrange communications (control and feedback) in a physically isomorphic manner (see Sheridan, 1988). This implies not only a combination of analogic and symbolic control and teaching, but also an information rich mapping from robotic sensors (e.g. cameras) directly to human senses (e.g. vision).

## 4.3.4 Telepresence for Teaching

### The Role of Presence in Teleprogramming

Both the need for a natural control interface and the need to enhance communications for teleprogramming argue for a role for a sense of presence in teleprogramming. In particular, they argue for *telepresence*, an ideal achieved by providing sufficient and realistic sensory feedback from a remote teleoperator to a human operator such that he or she feels physically present at the remote site (Sheridan, 1992). This ideal is intimately related to *virtual reality* (VR), the extension of a person's sensing and manipulation capabilities to a virtual environment (Sheridan, 1992; see Rheingold, 1991; Heeter, 1992). (Note: the term "telepresence" is sometimes used to mean "teleoperator system." In this thesis, it merely refers to the subjective feeling of presence.)

Though the feeling of presence has not definitively been shown to enhance teleoperation (Sheridan, 1988), it does have the potential to provide isomorphic analogic control over, and high fidelity sensory feedback from, remote systems. To date, telepresence has mainly been studied in the context of direct teleoperation. With respect to teleprogramming, however, telepresence has the potential to provide not only the high fidelity sensory feedback necessary to guide a robot safely through a remote visual environment, but also the simple control interface necessary to teach a robot without a complicated control language. In other words, it provides a relatively simple, cheap, transparent interface with which an expert in a particular task (in this case visual navigation) can teach the robot without having to learn how to control or communicate with it. The trainer simply drives the robot through the environment and "points out" important situations.

### Achieving Presence

A whole host of issues, most of which have yet to be sorted out by researchers, affect the degree to which an operator might feel present at a remote site (see Schloerb, 1995). The purpose here is not to study such issues in order to achieve an optimal sense of presence (which is, in any case, often questionable, especially in hazardous environments), or even to study the best way to present information to a robot in order to aid the training process, but rather to choose the appropriate techniques for providing an adequate degree of presence to an operator such that a teacher can train a robot using a transparent interface.

One way to achieve telepresence with decent fidelity is to establish an anthropomorphic control interface (Held & Durlach, 1992). Indeed, telepresence is usually achieved by slaving remote manipulators to human movements using fairly anthropomorphic analogic devices (like data gloves and head trackers) and providing visual (sometimes binocular) and auditory feedback using head-mounted displays (HMDs). Of course, the appropriate degree and fidelity of isomorphism does depend on the desired task (Sheridan, 1992; Zeltzer, 1992). Complex, uncertain tasks probably require higher fidelity than simple tasks. For example, the remote geographical exploration of the rocks on Mars requires detailed, isomorphic tactile and visual feedback (McGreevy, 1992).

In the realm of visual tasks, the operator's ability to perceive depth greatly influences the feeling of presence. Depth perception allows an operator to manipulate or steer the robot around objects in the remote environment, for example. Certainly, depth perception in human operators arises even when viewing monocular video imagery (e.g. from occlusion, perspective, texture gradient, image size, linear perspective, areal perspective, shadowing, and image motion), and a potent depth perception arises when the operator views, using an HMD, the video of a monocular camera that is slaved to his head (e.g. from motion parallax, and oculomotor cues). But stereoscopic displays are in general much better at persuasively imparting a sense of depth than monocular displays (Kim et. al., 1985). They allow faster perception of scene layout, visual noise filtering and enhanced effective image quality, enhanced slope and depression detection, wider field-of-view (FOV), enhanced object recognition and image interpretation, increased user satisfaction, and fewer casual errors (Drascic, 1991). Tachi (1988) has qualitatively demonstrated these advantages in experiments using a three-wheeled remotely driven vehicle with a pan-tilt stereo video system, which have proven binocular HMDs superior to conventional video displays for the task of avoiding collisions while controlling the robot with a joystick.

### The Use of Augmenting Graphics

For navigation in visual environments, presence helps mainly to orient the operator, provide a description of the visual scene from the robot's perspective, and naturally provide a sense of depth for driving the robot with a joystick. To bolster the depth and orientation cues provided by a sense of presence, several methods for providing orientation cues during navigation have been borrowed from *augmented reality*, an offshoot of VR which concerns the overlaying of computer generated graphics onto real visual input. Alexander (1990), for example, prevents disorientation of human operators in navigation tasks by overlaying a computer generated view of the inside of a

car (i.e., edges of the windows, etc.) onto the video input, and by transforming this view appropriately to reflect head and body motions. Presumably, a computer can generate such views using the internally measured relations between a robot's eyes, head, neck, and body.

Note, however, that the usefulness of augmented reality for telepresence is limited. First of all, studies have determined that overlaying an analog proximity detector on the video display gives very poor depth perception in object manipulation tasks (Winey, 1981), and that overlaying graphical grids onto either monocular or binocular displays does not appreciably improve depth perception (Kim et al, 1985). Artificial shadowing on the floor and walls gives the operator a good perception of the object's position in the environment (Winey, 1981). Unfortunately, none of this depth information is generally available to a remote system in real environments without resorting to a detailed model or brittle depth measuring techniques (see Section 5.1). Thus, the types of information that might be displayed graphically to convey depth in VR systems (e.g. occlusion, image size, linear perspective, texture gradient, areal perspective, and shading) cannot generally be used to augment depth perception for telepresence.

## 4.4 Summary

This chapter has presented some of the computational issues related to training an autonomous agent to perform a new task. First, it discussed direct and indirect methods for reinforcement learning (RL), the unsupervised learning of optimal action policies from rewards and punishments received through time. Direct RL methods differ primarily with respect to their treatment of structural and temporal credit assignment, but also with respect to environment exploration. They include "global" genetic algorithms, "local" temporal difference learning, and behavior-based approaches, and correspond to behaviorist theories of learning in animals, since they learn direct "stimulus-response" mappings from internal states to actions. Though they avoid on-line planning, they do require re-learning the entire policy every time the task, goals, or environment changes, which makes them unsuitable for learning to navigate in large scale, dynamic environments. Indirect methods, on the other hand, are analogous to cognitivist theories of learning, since they learn task-independent representations as a intermediate step in learning new tasks, and use on-line planning to compute the optimal policy. In effect, they bypass the temporal credit assignment problem altogether. They are also advantageous because changes to the goal require only a re-planning (which might be performed off-line), changes to the task require only a re-learning of the reinforcement signal, and changes to the environment are likely to involve only local modifications to the learned representation. Unfortunately, indirect methods lack an efficient technique for learning task-independent representations, and often require an inordinate amount of time to plan.

The same sort of analysis holds for optimal control learning; that is, learning forward models of the plant using a distal approach seems to be more flexible than directly learning an inverse model. RL and optimal control learning can also be used interchangeably, though RL is better suited for the unsupervised learning of new tasks using a reinforcement signal, while control learning paradigms are better suited for the supervised learning of tasks with predefined input/output relations. Finally, both methods benefit (though in different ways) from a learning-by-doing, or guidance, paradigm. In RL, guidance directs exploratory behavior, while in control learning it

provides the correct response. When such guidance is performed from a distance using teleoperation, it is referred to as teleprogramming. For effective teleprogramming, task learning requirements argue for a combination of analogic and symbolic forms of teaching, and a sensory interface with adequate fidelity to insure that the teacher and robot share task-specific perceptions. For the case of visual navigation, both of these requirements are satisfied simply by presenting an operator with binocular video to create a sense of presence, and augmenting this video with orienting graphics.

In the next chapter, we begin to ground the discussion of task learning in the realm of visual navigation, using principals from both this and the last chapter to evaluate existing AI and biological modelling approaches to map-making and route planning.

CHAPTER 5

# *Visual Navigation*

This chapter discusses the problem of learning to navigate using visual sensors. In particular, it discusses the problems associated with *map-making* and *route planning* for a mobile robot in a visual environment defined by a distributed set of visual landmarks. This discussion parallels the more abstract discussions of representation and task learning in the previous two chapters. Map-making involves the on-line, unsupervised construction of environment state and transitions between states to form from visual input a task-independent model of the visual environment appropriate for navigation. In addition to the general principals of concept and inductive learning discussed in Chapter 3, it entails a domain-specific solution to the signal-to-symbol problem, discussed here in the context of extracting and processing visual landmark features using active vision techniques. Also discussed are both AI and biological modelling approaches to map-making and navigation using visual landmarks.

## 5.1 Active Vision for Landmark Perception

Active vision (sometimes called *animate vision* or *purposive vision*) is a general area of vision research concerned with developing task-specific algorithms that dynamically control the physical characteristics of visual sensors. Proponents of this line of research argue that such algorithms have computational benefits. Much in line with the HRV learning methodology described herein, they argue against the general *reconstructive* approach to vision, which treats vision as a general module for constructing a detailed internal 3D metrical model of the environment using notoriously time-intensive and error-prone techniques like stereopsis, shape-from-shading, and depth-from-motion. Instead, they favor simple, fast (typically parallel) algorithms that operate in sensor-based coordinate systems and selectively direct sensors in order to extract information relevant to the task at hand (see Horswill, 1993).

*Attentive vision* is that part of active vision specifically concerned with focussing attention on interesting visual objects. (Other domains of active vision address the computational advantages of moving the visual sensors.) At the core of attentive vision research is the problem of *gaze con-*

*trol*, which encompasses the two complimentary processes of *gaze change* and *gaze stabilization* (Abbott, 1992; Swain & Stricker, 1993). Gaze change, sometimes called *selective fixation*, refers to the problem of "where to look next," and generally requires the ability to direct visual sensors quickly and discontinuously to new targets (Abbott, 1992). Gaze stabilization, sometimes called *gaze holding* or *visual tracking*, refers to the problem of keeping the image of a visual object steady on the imaging sensor despite possible movements of the object or the sensor.

With respect to map-making, active vision provides a framework for searching for landmarks, tracking them, and assessing their appearance. Each of these tasks requires different kinds of visual features.

### 5.1.1 Visual Search

Visual search refers to the process of finding particular objects in a visual scene by sequentially attending to areas in that scene where an object might reside. With respect to map-making, it addresses the problem of finding landmarks in the visual panorama. Implicitly, visual search requires a physical movement of visual sensors (i.e. selective fixation) to scan the scene for possible object locations, as well as the computational shifting of visual processing operations to test prospective areas within the field-of-view (FOV). Both of these processes, respectively referred to as *overt* and *covert* visual search, require mechanisms for *selecting* new locations in a scene, as well as *attending* to and *tracking* those locations.

#### Preattentive Selection

Active vision researchers argue that *preattentive* processing can ease the computational demands of visual search. Relevant visual data tends to be grouped in space, and preattentive processing allows a visual system to attend directly to possible groups without searching serially through clutter (Swain & Stricker, 1993). Preattentive processing also has merit with biological systems. In humans, visual search is driven by "pop-out" cues that arise from a considerable amount of bottom-up parallel processing (Treisman et. al., 1990; see Grossberg et. al., 1994). These cues effectively eliminate a large number of visual distractors, and allow a person to search sequentially through the remaining possibilities by executing approximately 3 or 4 discontinuous eye movements every second (see Abbott, 1992). Pop-out cues seem to indicate, in a topographic fashion, combinations of low-level visual stimuli (e.g. color, high contrast, and high spatial frequency), higher level features (e.g. vertices and axes of symmetry), sudden changes or motion, and stimulus proximity and direction. Note, however, that they also depend on high-level goals and the task at hand (Yarbus, 1967). For example, when searching for visual landmarks while driving, most people quickly attend to targets, regardless of size, with outstanding eccentricity, background complexity, contrast, color, boldness of internal structure, and azimuth (Cole & Hughes, 1990).

*Biological Theories*

An early theory of preattentive vision maintained that pop-out cues signal conjunctions of simple features like color, brightness, orientation, disparity, and movement (Treisman et. al., 1990). These conjunctions theoretically formed a retinotopic *saliency map*, the peaks of which determine

the locations of possible visual targets (Koch & Ullman, 1985; Treisman et. al., 1990). Another theory (Julesz & Bergen, 1985), partially motivated by a competitive group of neurons in the visual cortex with elongated receptive fields, postulated that the visual system used local operations to extract *textons* (textural elements), a set of elongated blob features (e.g. line terminators, line crossings, rectangles, ellipses, and line segments) with specific properties (e.g. color, angular orientation, width, length, binocular and movement disparity, and flicker rate). Here, pop-out cues arise from sharp spatial variation in the spatial statistics of these textons. Based on this theory, several heuristic computational methods (e.g. Voorhees, 1987) can pick out textons in natural imagery and measure the spatial differences between them. Correlation-based machine vision techniques also exist for detecting lines, circles, ellipses, and polygons in imagery (Ballard & Brown, 1982).

Evidence now indicates that local features and textons are simply the beginnings of preattentive vision in humans, and that pop-out properties actually occur as a result of more sophisticated hierarchical linking or *perceptual grouping* processes that cluster features into regions (Beck & Prazdney, 1983; Caelli, 1985; see Grossberg et. al., 1994). Not unlike the argument made by active vision proponents, this perceptual grouping process is thought to reduce combinatorial explosion associated with directly searching for objects in a visual scene (Desimone, 1992). Theories and models of perceptual grouping processes in biological organisms date back to the Gestalt psychologists. One current model, consisting of the boundary contour and feature contour systems (BCS/FCS) of Grossberg and his colleges, performs perceptual grouping through a combination of contour enhancement and "filling in" (see Grossberg, 1994). The BCS sets up a local cooperation between topographically arranged units with similar oriented local filters (receptive fields) and a competition between units with different filters, and propagates activity along units with dipole receptive fields. The FCS concurrently performs a spreading of activation within a topographically arranged set of units, and bounds this spreading by the contours in the BCS. Thus, contours and regions form the basis for complimentary and interacting grouping processes. This system has been successfully used at Lincoln Laboratory to contrast-enhance noisy imagery in the SAR, IR, and light amplified domains (Waxman et al., 1993).

*Computational Approaches*

In the machine vision community, perceptual grouping has been explored primarily as a means for accelerating the search for objects and removing clutter in the context of model-based object recognition (see Grimson, 1990). Ullman (1987) refers to this process as *indexing*, the bottom-up computation of "odd-man-out" locations. Early grouping methods attempted to segregate images into figure and background regions or segments using largely ad hoc techniques like thresholding the image by the values corresponding to local minima in the image histogram, and performing spatial clustering with morphological operations like region growing, shrinking, merging, and splitting using various algorithms like connected components analysis (see Ballard & Brown, 1982; Haralick & Shapiro, 1985). The utter failure of such approaches has contributed to the infamy of the *segmentation* problem as one of the largest stumbling blocks for machine vision, and many researchers avoid addressing it by abstracting the problem away (i.e. working on higher level vision modules), simplifying the image domain, or posing the recognition problem as an exhaustive exponential search through all possible matchings of features (Grimson, 1990).

More recent methods for perceptual grouping have relaxed the segmentation assumption in favor of more heuristic paradigms aimed at decreasing the search time for object recognition (see Grimson, 1990). For example, Brooks (1981) has proposed a recognition system that groups edges likely to arise from the same generalized cylinder. Similarly, Lowe (1985) has proposed a system that groups edges with common properties invariant to camera view point (e.g. parallel edges, co-termination, co-linearity, clustering in space, connectivity, and repetitive textures), and Jacobs (1989) has extended this paradigm to a system that handles arbitrarily large groups with various properties, specifically by maximizing the likelihood that a single object has produced each set of grouped features. This system has achieved surprisingly good performance.

Based on Ullman's (1987) visual routines paradigm, Ullman & Sha'ashua (1988) have implemented an iterative computational scheme for boundary tracking that propagates local saliency (e.g. color, contrast, and orientation) along contours using a network of locally connected processing elements that works to minimize a global energy function. This scheme can smooth contours and fill in gaps, and can pick salient object contours out of clutter. Brunnstrom (1993) has employed a similar iterative local propagation technique for active vision, but also considers the independent propagation of information at different depth planes using stereopsis or accommodation.

Finally, Seibert & Waxman (1989) have presented a neural mechanism, called the diffusion-enhancement bilayer (DEB), for the simultaneous perceptual grouping and centroid determination of proximal features (see Waxman et. al., 1989; Cunningham & Waxman, 1994a). This network creates long-range attractive forces between features (edges or high curvature points) by setting up local feedforward and shunted feedback (on center/off surround) connections between processing elements. Through the processes of passive decay, a diffusion of activity (cooperation), contrast enhancement (competition), and local maximum picking, the DEB produces stable centroids of groups of features at multiple spatial scales as a function of time. It has been implemented in real-time on a PIPE video-rate processor in the context of generating attentional cues for the object vision system described in Section 2.2.2.

Despite that fact that these systems drastically reduce the number of hypothesized object locations in certain controlled situations, real-time perceptual grouping and segmentation in general remains an unsolved problem. Thus, existing methods for active search rely primarily on simple indexing schemes based on saliency maps computed using a combination of featural and task-dependent inputs (Clark & Ferrier, 1992; see Swain & Stricker, 1993). Note, however, that visual search can also be driven by contextual information. In fact, the search trajectories of human subjects differ greatly depending on context (Yarbus, 1967). In the active vision community, contextual inputs influence visual search by priming attention with top-down expectations derived from internal models. For example, Rimey & Brown (1993) describe a framework for attentional priming that uses Bayes nets (see Pearl, 1988; see Section 3.3.3) to indicate the expected distributions of parts in a scene, and computes the best sequence of eye movements to answer a particular question about that scene.

### Selective Attention and Marking

*Selective* or *focal attention* refers to the process of filtering out unwanted clutter in order to exclusively process a location that has been selected by the search process. (Note that this visual definition of attention differs from other attentional phenomena like alertness, arousal, and vigilance.) Selective attention can be thought of as a selective processing in space, time, and resolution (Swain and Stricker, 1993). In the case of overt visual search, visual attention is accomplished by holding the center of the FOV on the visual target, called a *fixation*. Shifts in attention are accomplished by directing the FOV discontinuously, called a *saccade*. In humans, saccades are effected by a topographic map in the superior coliculus that represents possible eye positions. The duration of fixation depends on the amount of information contained at the fixation point (Yarbus, 1967).

In the case of covert visual search, visual attention takes the form of a spatially restricted processing within the FOV of each visual sensor. In humans, restricted processing encompasses non-contiguous regions in depth. This phenomenon occurs as a result of topographical, modulating neural structures, located in areas V4 and IT along the cortical pathway for shape processing (the "What" pathway — see Section 2.2.2), which effectively filter out or ignore those stimuli outside the locus of attention (Desimone, 1992). The modulation signal arises from a competitive interaction between featural cells in areas V4 (e.g. orientation, length, width, spatial frequency, and color) and MT (orientation, direction and speed of motion, and binocular disparity) for all locations in the visual field. This processes occurs not only as a result of intrinsic (bottom-up) properties of stimuli, but also as a result of task-dependent (top-down) instructions, perhaps due to reciprocal connections between successive areas in the shape processing pathway, using principals similar to those of Adaptive Resonance Theory (Carpenter & Grossberg, 1991).

Restricted processing in the active vision community has mainly resembled some of the more simple "spotlight" or "zoom lens" models of visual attention (see Treisman et. al., 1990), which qualitatively mimic the modulatory process of biological attention. Some attentional schemes extend this spotlight model to depth. For example, the "zero-disparity" filter maximizes the autocorrelation of enhanced binocular images (see Brown et. al., 1992). Within the filter, similar features are correlated across binocular images at different resolutions based on disparity with respect to the horopter, orientation, magnitude, and contrast direction, effectively creating a window of attention in disparity space.

In addition to attentional mechanisms for processing possible target locations, some *marking* process must keep track of the locations that have already been searched (Ullman, 1987). In biological organisms, this marking process occurs in topographical maps of the visual scene. This process has been mimicked by computational methods that use multi-resolution maps to support coarse-to-fine visual searches on a number of spatial scales (see Abbott, 1992).

## 5.1.2 Visual Tracking

Visual tracking refers to the process of keeping fixated objects stable on the visual FOV. This process can be quite complex due to varying speeds, sizes, visibilities, and distributed natures of targets (Swain & Stricker, 1993). With respect to map-making, visual tracking addresses the

problem of attending to hypothesized groups of features that might contain landmarks long enough to either reject them as distractors, or perform shape processing operations, despite possible motions of sensor.

Like visual search, visual tracking requires moving the visual sensors. But in order to reduce motion-induced blur, this movement must precisely match that of the visual target. In humans, this process is referred to as *static fixation* or *smooth pursuit*, which is augmented by discontinuous corrective saccades when the tracking error grows too large (see Yarbus, 1967). In the active vision community, it usually takes the form of some sort of velocity control with predictive output, again augmented by quick changes in position when the error accumulates above some threshold (see Brown et. al., 1992).

Tracking, like search, also requires grouping operations, but on a more purposive level in order to exclusively segment targets out from the background. Existing active vision systems employ some sort of segmentation in depth (see Brown et. al., 1992; see Section 5.1.1). Beyond purposive grouping, most tracking methods stabilize the target with respect to its centroid (e.g. Brown et. al., 1992). Though centroid calculations are rarely accurate when the object is occluded, they nevertheless yield stable points that can be used to keep the target stabilized in the image. In general, the more features, the more stable the centroid. Thus, it is best to use edges rather then points features for tracking. Interestingly, the DEB network described in Section 5.1.1 computes the centroid of targets when applied at the appropriate scale (Waxman et. al., 1989), and has been used to perform tracking operations in the context of object recognition (Seibert & Waxman, 1989; see Section 2.2.2).

### 5.1.3 Feature Extraction

#### General Issues

In addition to visual search and tracking abilities, an active vision system must be capable of extracting more complex visual features in order to code object shape and identify objects (if nothing else than to serve as the verification step for visual search). Like visual tracking, feature extraction requires purposive grouping operations to segment object features out from the background and from each other, and to remove extraneous clutter. In Gestalt terminology, this grouping process can be thought of as assembling the object from its constituent segments or parts. Without such grouping operations, extracted features are largely sensitive to viewing conditions. For instance, Horswill (1993) presents a system that can recognize landmarks directly from camera imagery based entirely on correlation. A similar strategy, based on simulated annealing, is employed by Betke & Makris (1994). While indeed fast, these techniques can only be used in situations where such conditions can be controlled (e.g. office environments), and require that the repertoire of objects to be recognized is small and pre-determined by the programmer.

As described in Section 2.2.2, biological organisms seem to create features for object recognition in a hierarchical fashion. They seem to learn and recognize the viewer-centered spatial arrangement of oriented part features. Cells along the shape processing pathway code for progressively more complex features, and have receptive fields with progressively larger area (for example, some cells in IT cover the entire FOV). From a computational standpoint, the spatial arrangement

of such features must differ sufficiently for a large number of views, so that different objects can be easily distinguished from one another. On the other hand, they must also exhibit invariance to viewing conditions, such as lighting, and stability to small changes in viewing direction. This trade-off directly relates to the discrimination/generalization trade-off for concept learning (see Section 3.2).

The most relevant artificial methods for feature extraction are those designed for object recognition systems. Existing methods primarily extract high contrast edges invariant to lighting conditions, and process these edges to form contours. Some methods extract even higher level features from contours, such as high curvature points (e.g. Kitchen & Rosenfeld, 1982; Seibert & Waxman, 1989, 1992) and inflection points (e.g. Huttenlocher & Ullman, 1990), both of which can often sufficiently disambiguate objects. Note, however, that while inflection points exhibit invariance to viewing direction (Huttenlocher & Ullman, 1990), they are highly sensitive to noise. High curvature points (corners) change gradually with viewing direction, but are relatively insensitive to noise, provided that the Gaussian curvature operators involving second derivatives are applied only to a smoothed version of the image. For these reasons, Seibert & Waxman (1989, 1992) have employed the DEB (see Section 5.1.1) to pick out high curvature points on multiple spatial scales in real-time, namely by applying it to segmented contour imagery, propagating regions with positive Gaussian curvature and negative mean curvature, and picking out the local maxima.

**Landmark Feature Extraction**

In the context of map-making, feature extraction provides inputs to landmark shape processing mechanisms. Unfortunately, the segmentation and purposive grouping problem remains one of the most outstanding problems in object recognition, and attempting to solve this problem could (and has) entailed several theses in and of itself. Luckily, however, the task of finding and processing landmarks differs in one very important way from the task of object recognition. Namely, it allows the designer to define the attributes of landmarks, and thereby avoid having to segregate arbitrary visual objects with arbitrary motions from arbitrary visual scenes. In fact, a landmark might best be defined abstractly as a 2D spatial pattern of visual features with some local properties, rather than the intuitive notion of a physical object. Using this definition of landmark, problems like partial occlusion cease to exist, since feature extraction does not necessarily entail the exclusive and complete extraction of features belonging to a single physical object. The robot need only scan the visual panorama, pick out salient points based on coarse, heuristic local measurements over the scene (i.e. preattentive pop-out cues — see Section 5.1.1), and attend to each one of them sequentially to test more local properties. For example, the map-making system proposed by Levitt & Lawton (1990; see Section 5.2.1) extracts landmark features by forming spatio-temporal groupings, based on the more abstract notion of a landmark as a locally distinctive, spatially and temporally stable "visual event" that defines a single direction. This definition of landmark fits in very well with the proposed HRV learning methodology for map-making, since it does not require that landmarks look the same or be recognizable from different viewing directions.

There are, however, some caveats of this approach. First of all, it requires that pop-out cues be stable — that is, that they be uniquely re-acquired in multiple fields-of-view (i.e. from any viewing position), insensitive to movements of the robot or the landmarks, and constant over time (Levitt

& Lawton, 1990). Second, it requires some perceptual (not purposive) grouping of features for shape processing, even though this grouping does not necessarily reflect exclusive features from a physical object. Finally, it requires non-hierarchical methods for defining landmark shape — that is, place recognition must not require a prior learning of landmarks.

## 5.2 Map-Making and Navigation

The landmark processing operations discussed in the previous section can be thought of as domain-specific methods for solving the signal-to-symbol problem of concept learning (see Section 3.2). Note, however, that visual landmark processing is only the beginning of the computational process of forming maps for navigation. Existing schemes for map-making using landmarks include both traditional AI localization techniques for mobile robots, and neurocomputational architectures based on biological models of cognitive mapping in rats.

### 5.2.1 AI Approaches to Mobile Robot Localization

#### Reconstructive vs. Distinctive Place Approaches

Existing methods for mobile robot map-making and navigation have largely emerged from mainstream AI and machine vision techniques (e.g. Moravec, 1981; Thorpe et. al., 1988; Zhang & Faugerhaus, 1992; for a review, see Luo, 1989). Though many of these techniques can accurately represent and reason about physical environments in sophisticated ways, they primarily take a reconstructive approach to vision (see Section 5.1), and exhibit somewhat brittle performance (see Brooks, 1985). Reconstructive (metrical) representations include configuration spaces, generalized cones, voronio diagrams, grid models, segment models, vertex models, convex polygons, and polygon region models (see Levitt & Lawton, 1990; Hwang & Ahuja, 1992). Construction techniques for map-making include stereopsis (e.g. Braunegg, 1993), which is sensitive to matching errors, ultrasonic range-finding (e.g. Drumheller, 1987; Elfes, 1987), which is prone to missing or multiple returns, and triangulation, which requires at least three non-collinear landmarks in greatly separated directions. The resulting errors require extra machinery in order to represent uncertainty (e.g. Elfes, 1987; Leonard & Cox, 1994). Furthermore, planning safe paths over long distances using such models typically requires a time-consuming, geometrical search over an immense space of possibilities (see Hwang & Ahuja, 1992). Therefore, reconstructive approaches do not form representations well-suited to the task of navigation in large-scale environments (see Section 3.1 for a brief discussion of the general affects of representation on model application).

In reaction to this brittle performance, a number of *relational* map-making schemes have suggested the alternative of learning adjacency relations between *distinctive places* (Kuipers, 1978, 1979; Brooks, 1985; Davis, 1986; Levitt et. al., 1987; Kuipers & Levitt, 1988; Sarachik, 1989; Levitt & Lawton, 1990; Mataric, 1992). Implicit in this approach is the separation of long-term, global, qualitative map-making in large scale space for purposes of route planning *between* distinctive places, from short-term, local, metrical map-making in the robot's immediate surroundings to facilitate motion planning for obstacle avoidance *within* distinctive places. This separation yields computational advantages, such as insensitivity to noisy and drifting dead-reckoning and compass readings, efficient storage of spatial information, and straightforward route planning via

graph search. Existing schemes for relational map-making differ both in their learning and recognition of places, and in their learning and application of relations. These differences parallel those between existing methods for state and transition learning for task-independent models. In particular, methods for place learning and recognition can be thought of as concept learning systems that incorporate domain-specific knowledge (see Sections 3.2 and 5.1).

### Methods for Place Learning and Recognition

A variety of computational methods exist for visually defining and recognizing distinctive places. Several define places as positions within global 3D maps (e.g. Chatila & Laumond, 1985), or represent places as local 3D maps (Yeap, 1988; Asada et. al., 1990; Malkin & Addanki, 1990; Braunegg, 1993). Though useful for obstacle avoidance as well as high level planning, such paradigms defeat the main purpose of qualitative topological maps by posing place recognition as a matter of 3D reconstruction.

Other more qualitative methods can, for the most part, be exclusively typed as either *landmark-* or *image-based* methods. Learning places for either type usually refers to storing the input as a new place when it is not recognized. Keep in mind, however, that this learning does not encompass adaptation to dynamic environments. Existing methods of both type can be evaluated based on their ability to distinguish between (discriminate) various locations in the environment, called *localization*, yet exhibit invariance (generalize over rather than be sensitive to) various imaging conditions like lighting (see Section 3.2).

*Landmark-based Methods*

Existing landmark-based methods for defining distinctive places in general require the accurate recognition of distinguishable landmarks, often by attempting to extract 3D models (e.g. Davis, 1986). This implies that the performance of place recognition is sensitive to the performance of object recognition. The most primitive of such methods simply uses the landmarks themselves to define places, as in Kuipers' Tour model (1978, 1979). This method, also used by Mataric (1992), provides no localization information over the majority of the environment, therefore requiring dead reckoning to navigate between places, and it cannot disambiguate (distinguish) between places defined by indistinguishable landmarks without contextual information.

Greater localization ability can be achieved by representing a place as a list of landmarks in the environment, either ordered or labelled by absolute azimuth direction, as in the Qualnav model proposed by Levitt & Lawton (1990). However, this method provides only limited localization at the expense of sensitivity to landmark occlusion, and requires an accurate compass. Furthermore, the number of places increases quadratically with the number of landmarks, since boundaries between places are implicitly formed (as a consequence of the ordered lists) by the lines passing through pairwise sets of landmarks.

*Image-based Methods*

Image-based methods for defining places are advantageous in that they are easily mapped onto parallel hardware. The most primitive form of image-based place definition is by unprocessed camera imagery. Zheng & Tsuji (1992) have proposed a system that represents places by pan-

oramic imagery, which must be explicitly matched to input imagery using a time-consuming dynamic programming method. Horswill's (1993) mobile robot represents directional places by blurred camera imagery that slightly mitigates sensitivity to changes in viewing conditions. These methods are only applicable to indoor environments with precise lighting control. Furthermore, they are sensitive to extraneous visual input, rotation of the robot, and small variations in the pose and positions of visual features.

More involved methods for image-based place definition incorporate invariance to lighting conditions and small changes in the visual input. Nelson (1989) describes a pattern matching system for visual homing that represents places by dominant edge orientation within coarse "receptive field" neighborhoods. Engleson & McDermott (1991) have extended this methodology to "image signatures" defined by combinations of quickly computable neighborhood quantities such as intensity and edge strength. The inexact nature of the visual measurements causes similarity profiles (degree of resemblance between input and stored patterns as a function of location) to vary smoothly over "local recognition neighborhoods" that cover the environment. The methodology trades off the size of stored patterns with the distinguishability of learned places by varying the number of neighborhoods. It also trades off distinguishability with the number of stored patterns (number of places) by varying the recognition threshold. Thus, environments can be mapped into places at multiple spatial resolutions with differing storage capacity. Nelson demonstrates that such methods are computationally tractable for environments with 3 or fewer dimensions, and shows that the probability of interference between patterns is small for non-rotational motions. However, the sensitivity of (poor generalization over) input patterns to robot orientation requires an explicit search following rotation in order to align the input with possible matches. Nelson vaguely suggests using 3D object recognition in conjunction with image-based place recognition to deal with this problem. (This notion is also realized by the sub-system proposed herein.)

Finally, Kuipers & Byun (1990) have proposed that places be defined by a number of metrical or image-based "distinctiveness measures," depending on the characteristics of the environment. In theory, both types of measure allow one to perform fine localization, since they tend to produce "similarity profiles" that indicate the distances to places (used here to perform gradient ascent). Examples of metrical measures include distances to nearby objects, and the number of open spaces (e.g. hallway routes) around the robot. Aside from the brittleness of such measurements alluded to earlier, it is not likely that they can produce distinguishable places over large-scale environments, since ambiguity results from characterizing places by a handful of parameters. An example of an image-based measure is temporal discontinuity of sensor readings over small steps. In the case of visual sensors, such measures are clearly highly sensitive to lighting conditions and orientation of the robot.

### Methods for Constructing Relations Between Places

The learning of relations between distinctive places has primarily been limited to primitive, highly-symbolic, graph-like topological structures, sometimes augmented by distances and directions to form *diktiometric* representations (Engelson & McDermott, 1992). A few schemes use more sophisticated action-based methods, such as associative links labelled by appropriate actions (Kuipers, 1983), probabilistic knowledge embodied by symbolic fuzzy maps (Davis, 1986; Kuipers & Byun, 1990), or action consequences in terms of conditional probabilities (Kuipers,

1983). Of particular note is the probabilistic Markovian model proposed by Dean et. al. (1990a), which embodies action consequences using temporal belief networks (a specialization of Bayesian networks). This system explicitly models time by allocating nodes for each time step.

The drawbacks of existing systems are many. First of all, none provide solid methods for learning uncertain relations by assimilating information incrementally (on-line) during exploration (see Section 3.1). Second, only the system proposed by Engelson & McDermott (1992) can incrementally refine maps and adapt them to dynamic environments (where the consequences of one's actions change over time) during exploration. Based on perceived errors in the learned map (geometric or transitional), this system splits single place nodes that code for multiple locations, merges multiple places that code for single locations, and updates the relations between places (this technique relates to those described in Section 3.2 for dealing with the discrimination/generalization trade-off). Third, most systems require complicated reasoning algorithms such as rehearsal procedures (Rivest & Schapire, 1987; Kuipers & Byun, 1990) in order to perform state (place) estimation. Finally, planning in these systems is usually limited to symbolic AI graph search techniques like A* (e.g. Levitt & Lawton, 1990; see Section 3.3.2).

### 5.2.2 Biological Modelling and Emulation of Cognitive Mapping

Many researchers have proposed conceptual biological models of place cell learning and cognitive map construction (see Section 2.2.1 for a review of this phenomenon; see Burgess et. al., 1995, for a partial review of existing models). These models have inspired a number of neurocomputational architectures. Though some of these architectures have been simulated in simple computer worlds, none have been implemented in realistic environments, and most offer no detailed neurocomputational framework for possible real-time implementation. Furthermore, very few directly support the interpretation or generation of locomotive actions.

Existing neural architectures differ in their representation of places, in their use of learned places to form maps, and in their use of learned maps to navigate. One can evaluate them with respect to each of these facets on both biological and computational grounds.

#### Neurocomputational Architectures for Place Cell Learning

Existing architectures for place cell learning are primarily based on one of three types of biological models: *allocentric*, *egocentric*, or a combination thereof (*semi-egocentric*). Allocentric (i.e. object-centered) models learn places in a coordinate system defined entirely by the external environment, using landmarks and/or dead-reckoning. Egocentric (i.e. viewer-centered) models learn places in a coordinate system defined entirely by the rat's midline location and orientation. Finally, semi-egocentric models learn places in a coordinate system centered on the robot, but oriented with respect to the environment (e.g. "North"). As with AI methods for learning distinctive places, the architectures that these models inspire can be computationally evaluated with respect to generalization and discrimination.

*Architectures Based on O'Keefe's Allocentric Model*

The first and most widely known model for biological cognitive mapping and place learning is credited to O'Keefe & Nadel (1978), who proposed that place cells learn distance relations between visible landmarks relative to an allocentric frame of reference. These distance relations are represented by a 2D layout of the landmarks, centered about their collective centroid, and oriented with respect to the landmark constellation.

The major computational problem with this model is the sensitivity of a centroid or other global reference point to the occlusion of even a single landmark. In general, architectures based on such allocentric models suffer from these occlusion problems, as well as problems associated with odometric and compass drift.

*Architectures Based on Zipser's Semi-Egocentric Model*

Semi-egocentric models are by far the most popular in the modelling literature. These models can generally deal with limited landmark occlusion, and achieve an independence of odometric drift, but still have difficulty with compass drift.

Existing architectures for the most part follow the general principals of Zipser's semi-egocentric model (1986), a general 2 layer network that takes as input the activity levels of "feature detectors" tuned to various attributes of visual landmarks (such as distance, retinal size, direction, etc.). This network can, for example, represent the retinal sizes of and distances to landmarks (measured in an egocentric coordinate system), explicit visual angles between adjacent landmarks, or the absolute angles to landmarks (measured in an allocentric coordinate system).

Motivated by Zipser's model, Schmajuk and his colleges (Schmajuk & Blair, 1993) have developed a neural architecture that tunes detectors to absolute visual angles of landmarks, as driven by reinforcement learning based on classical conditioning theories (Schmajuk & DiCarlo, 1991). This system ignores evidence for the independence of place learning and goal behavior. Practically speaking, place learning via reinforcement conditioning is unrealistic, since each location does not generally provide a reward or penalty. In this sense, the system shares some problems with existing direct RL schemes (see Section 4.1.2)

Shapiro & Hetherington (1993) have implemented and simulated a variant of Zipser's model using a backpropagation network which takes as input the absolute visual angles subtended by landmarks. Aside from the biological implausibility of backpropagation, the main problem with this network is that learning place fields requires thousands of supervised (i.e. teacher guided) training sessions, in contrast to the quick, unsupervised formation of place fields by place cells in rats.

Sharp's (1994) competitive learning network also resembles Zipser's model. This network has been successfully tested in a simulated in a cylindrical environment similar to the ones used in neurophysiological studies of places cells (see Muller et. al., 1991), specifically by simulating the response of input cells to the angles and distances to landmarks.

Penna & Wu (1993) have proposed a semi-egocentric neural system based on the Tour and Qualnav AI paradigms of Kuipers & Levitt (1988 — see Section 5.2.1). This system learns a place as a "list" of landmarks ordered by azimuth relative to a compass reference point such as "North", and assumes that all landmarks are always visible (no occlusion). To incorporate knowledge of landmark identity, the system assumes that all landmarks are recognizable and distinguishable.

Scholkopf & Mallot (1995) propose a system for navigating in grid mazes. This system defines a place as a collection of distinguishable views of maze junctions with the same successor view (i.e., following an action). Note, however, that such deterministic, grid-based map-making schemes do not extend (generalize) easily to the real world (see Sections 3.2.3). Because locations in the maze are coarsely quantized, view recognition becomes a trivial matching operation (especially without noise). No landmark extraction or invariances need be considered. In effect, the system ignores the signal-to-symbol problem.

Finally, Gaussier & Zrehen (1995) propose a system that conjunctively encodes the identity and absolute angles of visual landmarks. This system was developed independently from an earlier version of the system proposed here, which includes an architecture with similar properties first proposed by Bachelder et. al. (1993).

*Architectures Based on McNaughton's Egocentric Model*

Both allocentric and semi-egocentric models ignore the occurrence of the controversial heading tuned place cells, mainly because heading specificity may be artifactual (heading tuned place fields typically arise in radial arm mazes where the rat runs in mainly two directions). However, McNaughton's fully egocentric model (McNaughton, 1989; McNaughton et. al., 1991) maintains that a place cell responds to a *restricted local sensory view* of an environment. The model explains the persistence of place cell activity following the removal of landmarks by including a mechanism for pattern completion (e.g. Carpenter & Grossberg, 1991). A very simple egocentric "view cell" model has also been proposed by Zipser (1986), where place cells learn whether landmarks reside to the left or right of the rat's midline. McNaughton's model forms the basis for the place learning architecture proposed here, the main difference being the use of body heading, rather than head direction, to define the local view.

### Neurocomputational Architectures for Map Formation

Apart from place cell learning, the existing neural architectures form maps based on either *metrical* or *relational* models of cognitive mapping.

*Architectures Based on O'Keefe's Euclidean Model*

Metrical models conjecture that place cell activity codes for metrical coordinates in an environment. The Euclidean model proposed by O'Keefe & Nadel (1978) supposes that the overlapping place fields provide a mechanism for deducing accurate coordinates within an environment by the pattern of activity across the learned place cells. The model further posits that place cell activity signals the distances to place field centers.

Touretzky et. al. (1994) take a similar approach. Their neurocomputational system associates place units (learned using Zipser's model) with cartesian coordinates measured with respect to some salient reference point, such as the animals nest. A place unit becomes active due to either the sensory input or a radial basis function applied to the current dead-reckoned coordinates. Note, however, that place cell firing rate profiles do not often resemble radially symmetric basis functions (cf Figure 1).

Finally, Burgess et. al. (1995) suggest that the phase of place cell firing with respect to the theta rhythm codes for the direction of the place field center (with respect to the rat) as a consequence of the angular positions of cues. This type of coding requires the existence of place fields that cover the entire environment, which is typically not found.

In general, architectures based on metrical models are prone to dead-reckoning error accumulation over long distances, which may result in invalid learned coordinates or cue directions for place cells. Metrical models also ignore the evidence for relational learning in the hippocampus.

*Architectures Based on Relational Models*

Relational models take the stance that the hippocampus constructs either *topological* or *diktiometric* relations between pairs of place cells. This idea was originally proposed by Tolman himself (1948), who proposed that cognitive maps might take the form of a directed graph called a *means-end field*.

The differences between relational models mainly concern the form of the relations. Topological models neglect spatial information (notice a parallel with Section 5.2.1). Several topological architectures (Penna & Wu, 1993; Schmajuk & Blair, 1993; Schmajuk et. al., 1993) use Kohonen self-organizing feature maps (see Simpson, 1990; see Section 3.2.5), which construct links between cells based on measures of feature similarity at multiple scales (see Section 3.2.5). But these networks learn off-line, and do not truly capture the topological structure of environments (Scholkopf & Mallott, 1995). In particular, two physically adjacent places might have different features (e.g. two opposite sides of a doorway), and two physically distant places might have similar features (e.g. similar looking places in two separate rooms).

Other topological architectures use RNNs to code sequence transitions between patterns of place node activity (Hetherington & Shapiro, 1993; Prepscius & Levy, 1994; see Section 3.3.4 for a review of RNNs). In a sense, these architectures generalize over topological models because they can control the degree of sparseness of network connections using competitive shunting inhibition (Prepscius & Levy, 1994). These architectures are also capable of learning sequences of place activity in a single trial using a Hebbian learning rule with either pre- or post-synaptic gated decay (Minai & Levy, 1993a). However, they have not yet been extended to learning multiple sequences in an efficient manner.

Diktiometric models associate spatial information with relations between places (again, see Section 5.2.1). One way to code relations without relying on featural similarity or learning individual place sequence characteristics is to encode spatial information along with the relations between temporally adjacent places. For example, Muller's cognitive graph model (Muller et. al., 1991)

purports that a group of displace cells learn the distances between place field centers. The model suggests that the hippocampus constructs a directed graph with place cells as the nodes and the lateral meshwork synapses as the edges (see Section 2.2.1 for the relevant anatomy). These synapses learn the distances via Hebbian modification during temporal overlap in the firing of place cell action potentials. Note, however, that distance coding models assume place cell firing rate patterns to be radially symmetric and have uniform extent, which is not typically found. They also implicitly assume that the rat runs with uniform velocity throughout the maze.

Another way to encode spatial information is to associate movements with relations between neighboring places. McNaughton's model (1989) learns associations between place cells conditional on relative egocentric turning and forward body movements. The model assumes that place cells are learned using an egocentric approach. Thus, it gives heading tuned place cells an important role in cognitive mapping. Biologically-plausible mechanisms for how this model might be used to achieve dead-reckoning skills have also been proposed (McNaughton et. al., 1991). Though some researchers have discussed similar ideas (e.g. Gaussier & Zrehen, 1995), McNaughton's model has not yet been realized by any neurocomputational system.

Scholkopf & Mallot (1995) come perhaps the closest to embodying McNaughton's ideas. Their map-making system learns relations in terms of actions using two interacting layers. A map layer constructs a adjacency graph between places, while a movement layer constructs modulatory connections to the adjacency graph from action inputs. (Note: the authors claim that such modulatory connections encode actions more efficiently than the conjunction of movement and place suggested by McNaughton, 1989; but these two schemes are computationally equivalent.) However, the system implicitly assumes the environment to be deterministic, which is obviously not the case even for controlled rat experiments. It learns binary relations and modulatory connections, and only considers highly quantized actions (left, right, back, left turn, right turn). Thus, it does not easily extend to open real-world environments (see Section 3.3.2).

**Architectures for Goal-seeking Behavior**

Not surprisingly, both metrical and relational architectures have advantages and disadvantages for navigation and goal-seeking behavior, assessed here in the context of long-term prediction, route planning, and environment exploration (see Chapter 3). In general, architectures (e.g. Touretzky et. al., 1994) based on metrical models (e.g. O'Keefe & Nadel, 1978; Burgess et. al., 1995) can directly compute the coordinate that follows from executing an action, and can plan direct paths to goals simply by generating vector differences between coordinates and using dead-reckoning (path integration) to follow such vectors (assuming, of course, that such coordinates are correct — see Section 5.2.2). In this respect, they can easily calculate shortcuts and detours. Furthermore, they can explore an environment simply by moving to locations without place fields. However, they lack mechanisms for learning to avoid or favor certain places along the way to the goal, based on reinforcing events previously experienced during exploration.

On the other hand, architectures based on relational models make can make both short- and long-term predictions of place activity by propagating evidence *forwards* in time (recursively diffusing activity along place transitions). Most can exploit expectations in order to incorporate contextual information and improve place recognition (e.g. Scholkopf & Mallot, 1995). Interestingly, RNN

architectures with lateral inhibition (Prepscius & Levy, 1994) exhibit "fixed point" behavior, which supports the prediction of entire sequences of place activity (these fixed points resemble the resonances between top-down and bottom-up activity in ART). However, they currently cannot learn sequences that change at varying rates.

Planning in relational architectures usually entails a "spreading of activation" *backwards* in time from the goal (recursively diffusing activity in the opposite direction along place transitions). Such schemes qualitatively resemble parallel instantiations of the graph-like planning techniques employed by model-based reinforcement learning systems (see Section 4.1.3). Ideally, this spreading is modulated by connections to pain or pleasure centers (learned via reinforcement conditioning) so that the rat can avoid or favor certain places on the way to the goal (Schmajuk et. al., 1993; Schmajuk & Blair, 1993). Architectures without such modulation (e.g. Scholkopf & Mallot, 1995) implicitly assume that all actions incur the same cost (e.g. time, distance, or energy), and require that place fields be uniformly spaced throughout the environment, which is not generally true. The same can be said of architectures that use RNNs to learn temporal sequences (Hetherington & Shapiro, 1993; Prepscius & Levy, 1994).

Because architectures based on topographic relational models (e.g. Schmajuk et. al., 1993; Schmajuk & Blair, 1993; Penna & Wu, 1993; Hetherington & Shapiro, 1993; Prepscius & Levy, 1993) neglect spatial information, the act of planning does not provide appropriate actions for achieving goals; it only provides the sequence of places along the most efficient path to the goal. On the other hand, architectures based on diktiometric models (e.g. Muller et. al., 1991) can compute appropriate actions for achieving goals, and architectures based on action-tagged diktiometric models (McNaughton, 1989; Scholkopf & Mallot, 1995) automatically yield actions. In fact, diktiometric models allow the planning of novel routes (e.g. shortcuts, and detours) — despite arguments to the contrary (see Muller et. al., 1991) — provided that the cognitive map can compute an aggregate trajectory or action by integrating the spatial relations or actions between places along a previously explored route to the goal.

Finally, architectures based on relational models can drive the exploration of environments by keeping track of learned transitions. For example, the system proposed by Scholkopf & Mallot (1995) drives exploration using a binary completion matrix.

## 5.3 Summary

This chapter has discussed issues concerning the construction of task-independent models of environments for the purpose of visual navigation using landmarks. First, it discussed active vision techniques applicable to landmark search, tracking, and visual feature extraction. Ideally, the processing requirements for these tasks become more manageable if a landmark is simply defined as a view-based, spatially and temporally stable set of visual features, rather than the stereotypical self-contained object. This definition allows one to locate and track landmarks using pop-out cues computed with preattentive perceptual grouping operations.

Second, the chapter discussed various AI approaches to visual map-making and navigation for mobile robots. Existing AI approaches include both reconstructive techniques, which produce

metrical models of environments that are somewhat error-prone and time-consuming for large-scale navigation, and qualitative techniques, which learn distinctive places and their adjacency relations. Existing methods for distinctive place definition are either landmark- or image- based, and are largely sensitive to viewing conditions. None provide real-time mechanisms for adapting learned places to dynamic environments, and very few provide real-time methods for performing accurate, robust localization within distinctive places. Existing methods for learning the relations between places tend to rely on the simple symbolic mechanisms discussed in Chapter 3.

Finally, the chapter has discussed existing neurocomputational architectures based on conceptual biological models of spatial cognitive mapping on both computational and plausibility grounds. Only those architectures based on fully egocentric (as opposed to semi-egocentric and allocentric) models of place cell learning exhibit insensitivity to landmark occlusion, dead-reckoning, and compass measurements. Furthermore, only those architectures based on diktiometric, relational (as opposed to metrical and purely topological) models of cognitive map formation can perform route planning and navigation without relying upon accurate dead-reckoning. Together, egocentric place learning and action-based diktiometric, relational cognitive mapping best explain the biological data described in Section 2.2.1, and reveal a possible reason for the existence of heading-tuned place cells in the hippocampus. Both of these approaches are present in McNaughton's (1989) model of spatial cognitive mapping, but architectures based on this model have not yet been implemented in a realistic environment.

Part II of this thesis describes several architectures comprising a neurocomputational system for navigation strategy learning for a mobile robot. These architectures are motivated by the biological data presented in Chapter 2, especially by McNaughton's model of spatial cognitive mapping. They also adhere to the abstract principals of representation and task learning presented in the last two chapters, and to the domain-specific principals of map-making and navigation presented in this chapter.

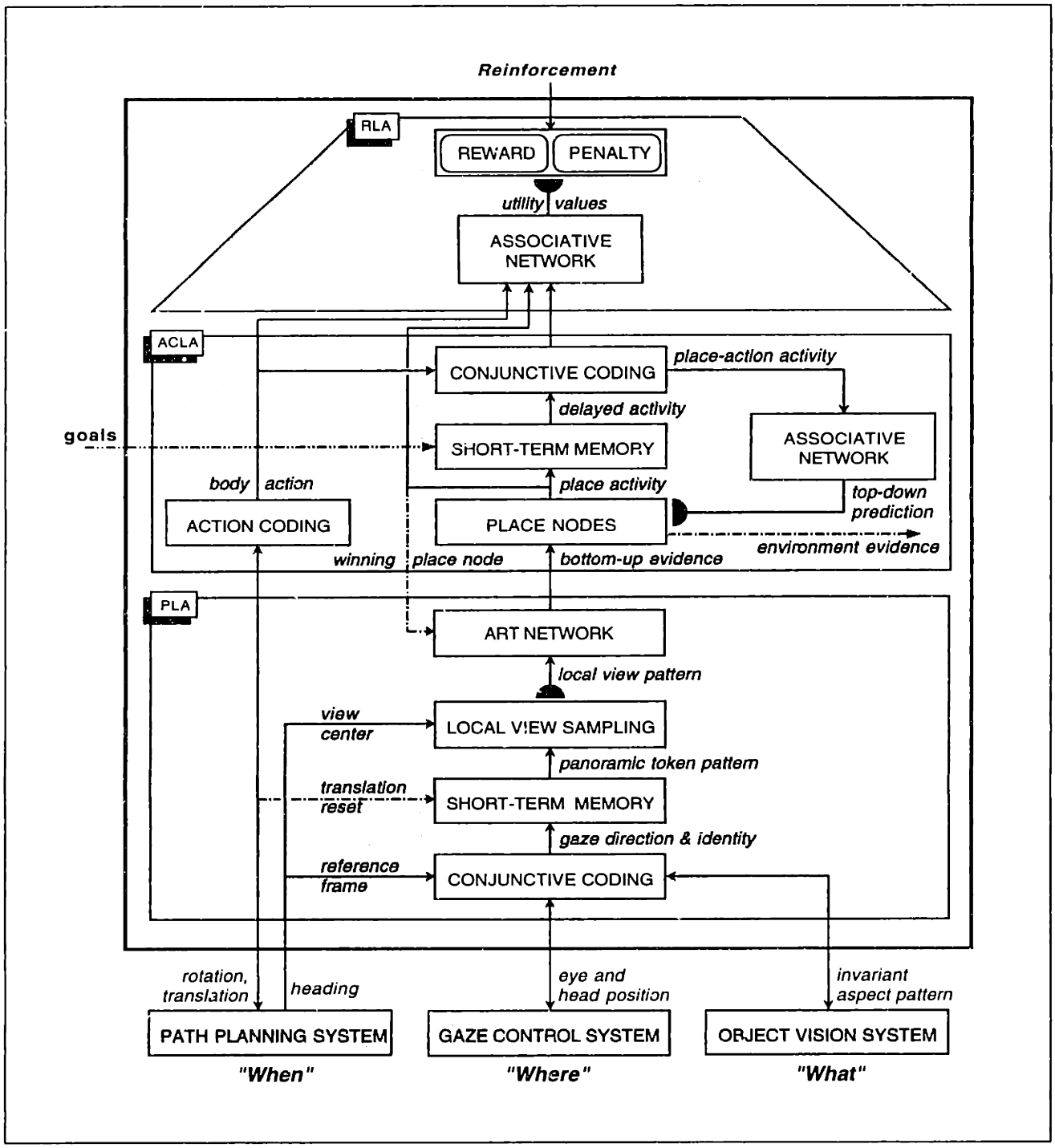


**FIGURE 16: THE NAVIGATION STRATEGY LEARNING SUB-SYSTEM (NSLS).**
*The neural sub-system for learning navigation strategies is conceptually separated into three neurocomputational architectures: a place learning architecture (PLA), an action consequence learning architecture (ACLA), and a reinforcement learning architecture (RLA). The main inputs to the system include landmark identity cues from the object vision system (the "What"), eye and head position from the gaze control system (the "Where"), rotation, translation, and heading from the path planning system (the "When"), and a reinforcement signal and set of goals from external sources.*

# Part II Neurocomputational Architectures: Design and Implementation

Based on the motivations and background presented in Part I of this thesis, the following chapters conceptually describe and mathematically define (using the notation defined in Appendix B) a number of neurocomputational architectures comprising a fully integrated system for map-making, map evaluation, and navigation. At base level, this system includes supporting sub-systems for landmark vision and locomotion (described in Chapter 6), including a *featural grouping sub-system* (FGS), a *gaze control sub-system* (GCS), an *object vision sub-system* (OVS), and a *path planning sub-system* (PPS).

At the heart of the navigation system lies a navigation strategy learning sub-system (NSLS), shown in Figure 16. This sub-system emulates spatial cognitive mapping and behavioral learning by biological organisms, and is theoretically capable of place prediction, environment recognition, route planning, and exploration. It comprises three neurocomputational architectures: a *place learning architecture* (PLA), an *action consequence learning architecture* (ACLA), and a *reinforcement learning architecture* (RLA).

Map-making in the NSLS entails learning a diktiometric relational map of locally distinctive places defined by characteristic views. This learning is collectively performed by the PLA and ACLA (described in Chapters 7 and 8 respectively). The general approach is based on the notion that a useful map qualitatively predicts *what* and *where* landmarks become visible to a creature or robot *when* it executes a locomotive action, requires neither 3D reconstruction nor precise 2D mensuration of physical structures, and adapts to gradual changes in the environment. In accordance with this notion, the NSLS learns a map based entirely on views of visual landmarks (the "What"), imprecise egocentric gaze directions to these landmarks (the "Where"), and rough short-term measurements of locomotive actions (the "When").

Given a learned map, map evaluation then takes the form of an indirect reinforcement learning, performed by the RLA (described in Chapter 9).

With the exception of the FGS and PPS, each of these architectures have been implemented independently on the mobile robot MAVIN (Mobile Adaptive VIsual Navigator), a Cybermotion mobile platform with a miniature pan-tilt unit serving as a neck for a binocular vergence apparatus (see Appendix A). They have also been integrated on MAVIN in the context of teletraining (described in Chapter 10).

CHAPTER 6

# *Landmark Vision, and Locomotion*

The proposed system for navigation strategy learning requires a number of supporting architectures capable of searching for landmarks, tracking them during self motion, and assessing their location and identity. It also requires an architecture for following a desired route by detecting and planning safe paths around obstacles in the environment, and assessing the locomotive actions actually taken while following such paths.

This chapter describes some of the supporting architectures for landmark-based visual navigation on the mobile robot MAVIN. The chapter begins by defining the various coordinate systems in which the inputs to the various architectures are measured. It then discusses a possible architecture for landmark grouping and feature extraction, and the adoption of a simplified version of this architecture on MAVIN for purposes of demonstrating map-making and navigation architectures in real-time. Next, it describes an architecture for gaze control and its role in searching for and tracking visual landmarks, and goes on to detail the object vision system in the context of landmark shape coding for place learning and recognition. Finally, it presents a possible architecture for path planning, and describes the simplifications made to the environment in order to bypass its implementation.

Each of the architectures described in this and subsequent chapters employ geometrical input measurements made with respect to the physical robot. These measurements, most of which are shown in the diagrams in Figures 17 and 18, are described in Tables 1 and 2, respectively. Descriptions of the architectures also employ a number of variables representing the activities of populations of nodes, described separately in each of the following sections (and subsequent chapters). Unless otherwise stated, each of these activities ranges between 0 and 1.

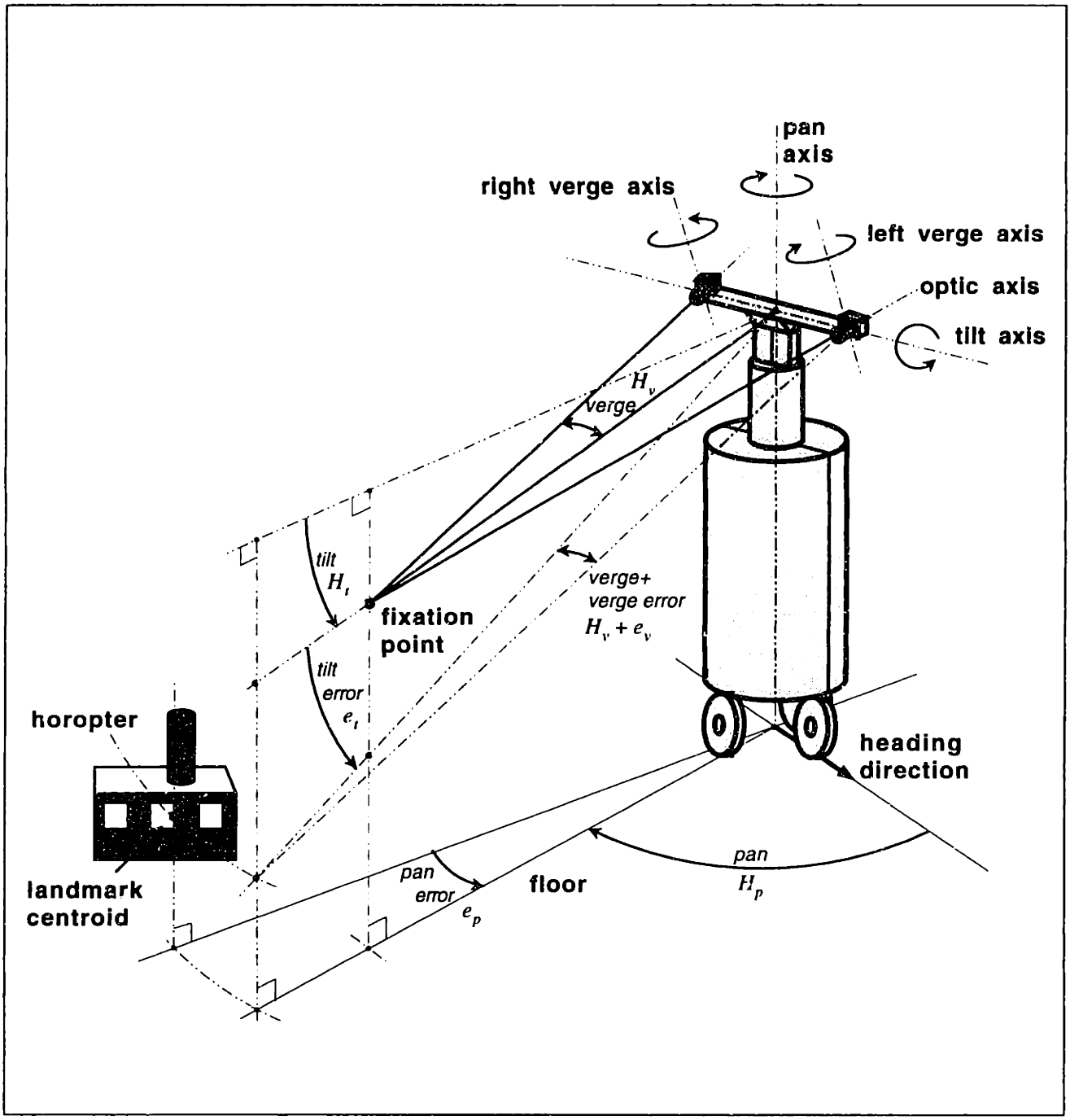


**FIGURE 17: HEAD- AND BODY-CENTERED ROBOT GEOMETRY.**
*Head-centered variables include camera vergence, and pan, tilt, and vergence tracking errors. Body-centered variables include the current head pan and tilt positions.*

**TABLE 1: HEAD- AND BODY-CENTERED PHYSICAL VARIABLE DEFINITIONS.**

| Variable | Definition |
|---|---|
| $\bar{H} = [H_p, H_t, H_v]$ | The robot's measured head position, consisting of a pan position $H_p$ (the angle between the robot's mid line direction and the projection of its line of sight onto the horizontal plane), a tilt position $H_t$ (the angle between the horizontal plane and the line of sight), and a vergence position $H_v$ (the angle between the line of sight and the optic axis of each of the cameras). |
| $H_t^{min}, H_t^{max}$ | The absolute minimum and maximum tilt positions of the robot's head, respectively. These values are given for MAVIN in Appendix A. |
| $\Delta\bar{H}^{in} = [\Delta H_p^{in}, \Delta H_t^{in}, \Delta H_v^{in}]$ | The measured change in the robot's head position over the last time interval $\Delta\tau$. |
| $\Delta\bar{H}^{out} = [\Delta H_p^{out}, \Delta H_t^{out}, \Delta H_v^{out}]$ | The commanded change in the robot's head position. |
| $[x_l, y_l]$<br>$[x_r, y_r]$ | The measured *image* coordinates of the feature centroid of the currently fixated landmark relative to the optic axis in the left and right FOV, respectively. |
| $[x'_l, y'_l]$<br>$[x'_r, y'_r]$ | The computed *angular* coordinates of the feature centroid of the currently fixated landmark relative to the focal point and the optic axis in the left and right FOV, respectively. |
| $\bar{e} = [e_p, e_t, e_v]$ | The computed angular tracking error for the currently fixated landmark, measured with respect to the focal point and the optic axes in terms of the pan error $e_p$, the tilt error $e_t$, and vergence error $e_v$. |

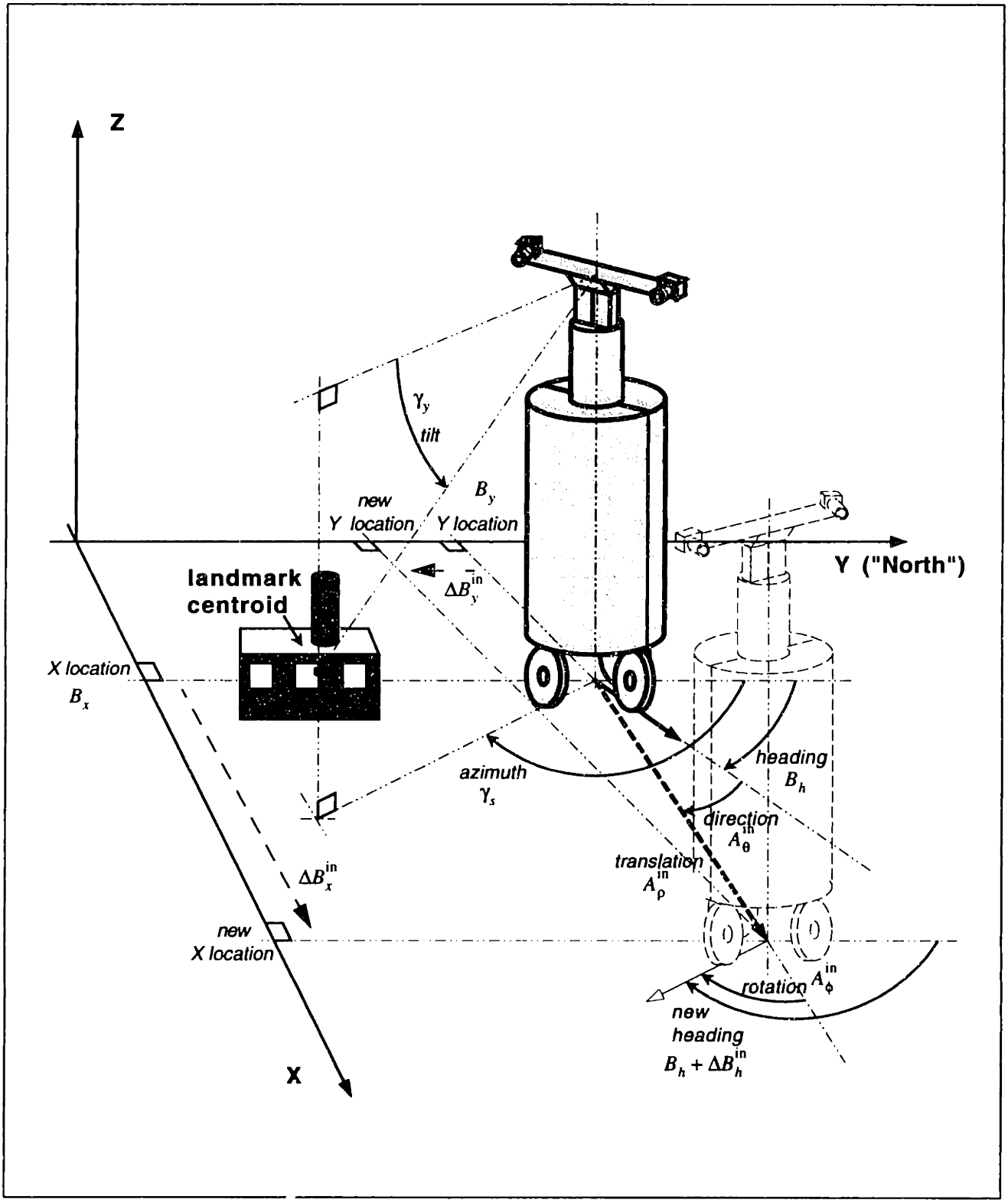


**FIGURE 18: ENVIRONMENT- AND BODY-CENTERED ROBOT GEOMETRY.**
*Environment variables include body position (X, Y and heading), and the azimuth and tilt of the currently fixated landmark. Body-centered variables include Euclidean (positional) and polar (directional, translational, and rotational) representations of body movement.*

**TABLE 2: ENVIRONMENT- AND BODY-CENTERED PHYSICAL VARIABLE DEFINITIONS.**

| Variable | Definition |
|---|---|
| $\bar{B} = [B_x, B_y, B_h]$ | The robot's measured body position within the environment, consisting of a location $(B_x, B_y)$, measured in an arbitrary but fixed (consistent) Euclidean coordinate system, and a heading $B_h$, measured clockwise with respect to the $y$ axis (i.e. "North."). |
| $\Delta\bar{B}^{in} = [\Delta B_x^{in}, \Delta B_y^{in}, \Delta B_h^{in}]$ | The measured change in the robot's body position over the last time interval $\Delta\tau$. |
| $\Delta\bar{B}^{out} = [\Delta B_x^{out}, \Delta B_y^{out}, \Delta B_h^{out}]$ | The commanded change in the robot's body position. |
| $\bar{A}^{in} = [A_\theta^{in}, A_\rho^{in}, A_\phi^{in}]$ | The measured locomotive action executed over the last time interval $\Delta\tau$, expressed in body-centered coordinates as a direction $A_\theta^{in}$ (with respect to the previous heading direction $B_h$), a translation $A_\rho^{in}$ (distance travelled with respect to the previous location $(B_x, B_y)$), and a final rotation $A_\phi^{in}$ (with respect to the direction of movement $A_\theta^{in}$). |
| $\bar{A}^{out} = [A_\theta^{out}, A_\rho^{out}, A_\phi^{out}]$ | The commanded locomotive action, again expressed in body-centered coordinates as a direction $A_\theta^{out}$, a translation $A_\rho^{out}$, and a final rotation $A_\phi^{out}$ (see above). |
| $\bar{\gamma} = [\gamma_s, \gamma_y]$ | The computed direction of the currently fixated landmark as assessed from the robot's current location in terms of azimuth $\gamma_a$ (with respect to the $y$ axis) and tilt $\gamma_t$ (with respect to the horizontal plane). |

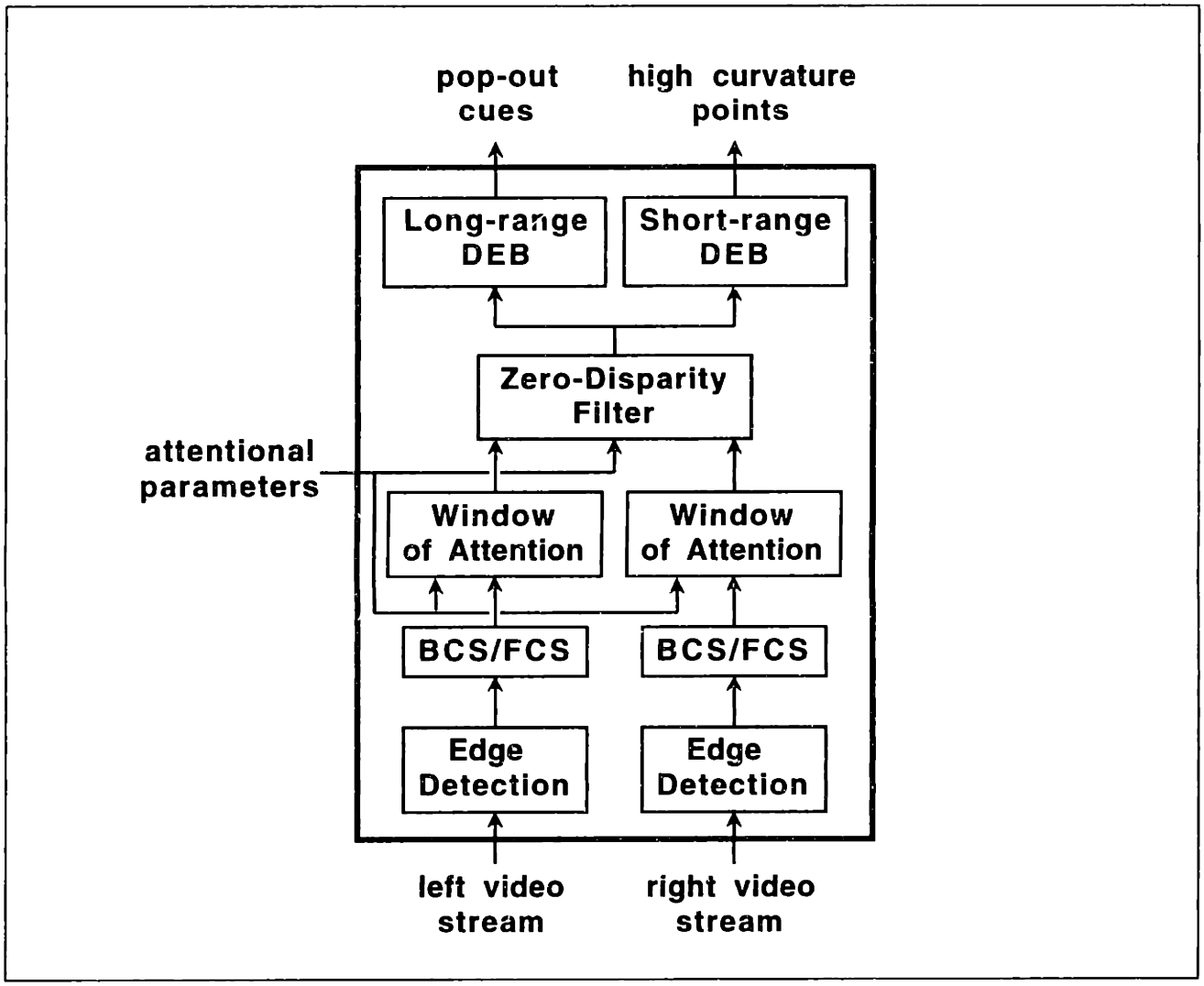


**FIGURE 19: THE FEATURAL GROUPING SUB-SYSTEM (FGS).**
*The featural grouping sub-system extracts stable pop-out cues that indicate the possible positions for visual landmarks, as well as visual features for assessing the identity and pose of visual landmarks.*

# 6.1 Grouping and Feature Extraction

## 6.1.1 Neural Architecture

As described in Section 5.1, landmark-based navigation requires that a mobile robot be capable of searching quickly for landmarks in the visual panorama, tracking possible landmarks during self-motion, and extracting stable features from each landmark. To be practical, all of these processes require some type of grouping operation. The map-making sub-system proposed herein poses no exception to this requirement. It employs both the directions to visual landmarks, determined by a gaze control sub-system (GCS), and the identity and pose of visual landmarks, determined by the object vision sub-system (OVS). Both of these sub-systems receive their input from a featural

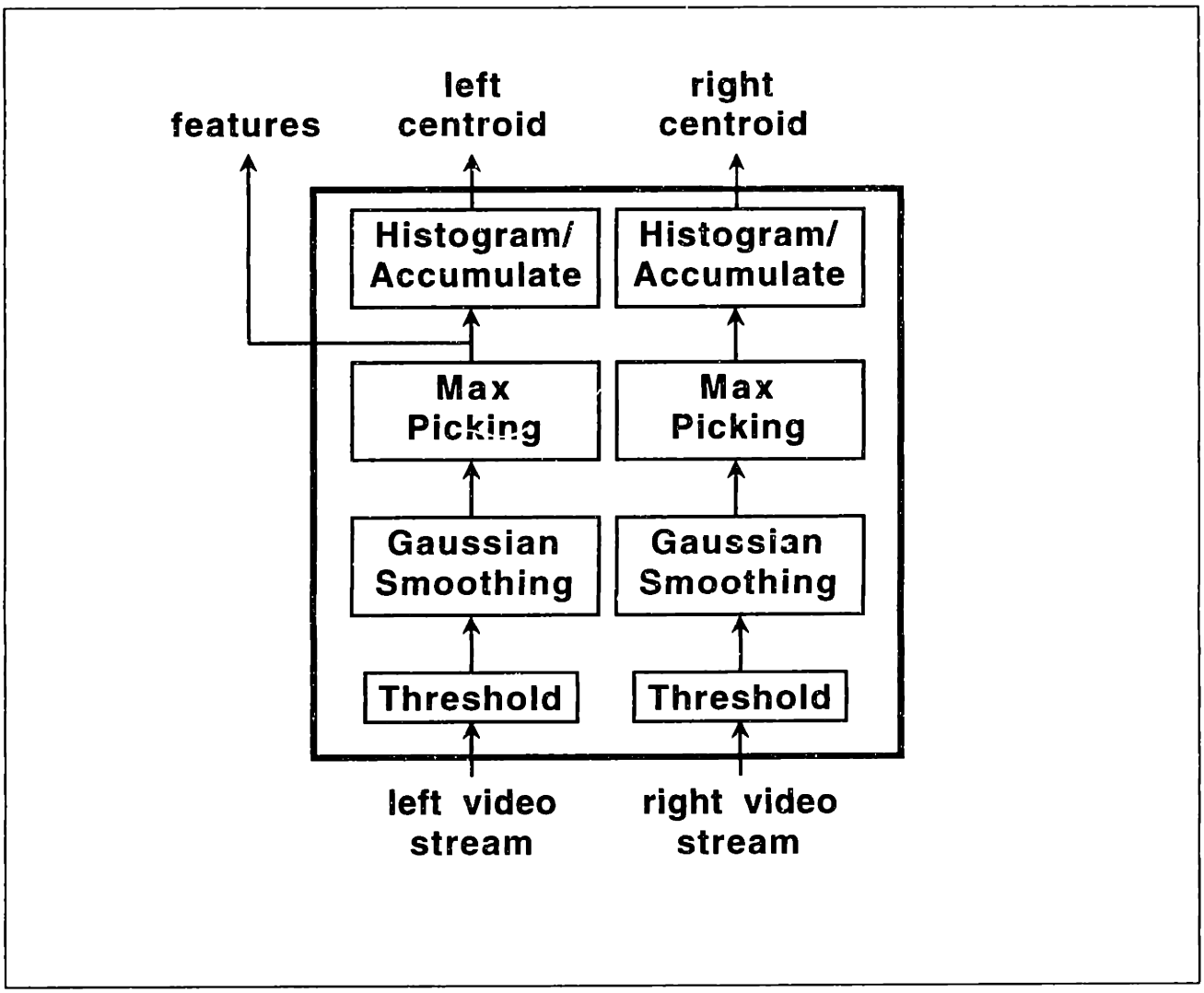


**FIGURE 20: THE SIMPLIFIED FEATURAL GROUPING SUB-SYSTEM (FGS).**
*A simplified version of the FGS extracts bright points independently in each of the binocular images, and computes the centroids of these points.*

grouping sub-system (FGS). The flow of information both within and between these sub-systems is reminiscent of the anatomy of spatial and object processing streams in the primate visual system (cf Figure 10).

The FGS (see Figure 19) takes as input the binocular video imagery provided by the visual sensors and the parameters for attentional processing, and computes spatially and temporally stable pop-out cues and visual features suitable for searching, tracking, and assessing the identity and pose of visual landmarks. It first extracts edge contours in both video streams and applies the BCS/FCS to group these contours over the entire FOV and remove clutter (see Grossberg, 1994; see Section 5.1.1). Next, it performs attentional processing operations in order to "segment" objects from the background. This segmentation takes the form of a 3D window of attention, implemented using a combination of binocular zero-disparity filtering and monocular masking operations (see Section 5.1.3). Finally, a DEB employs short-term diffusion processing to extract high curvature points along the segmented contours, and long-term diffusion to determine stable

"centroids" for perceptual groups of these contours (see Section 5.1.1). The centroids of perceptual groupings that persist over multiple frames define pop-out cues indicating the gaze directions to possible landmarks.

### 6.1.2 Simplifications

As described in the previous section, the FGS employes processes for preattentive and purposive grouping using a variety of both neural and conventional image processing modules. Unfortunately, edge extraction, contour enhancement, boundary segmentation, and perceptual grouping requires a phenomenal amount of computation. Even when the computations of some of the more demanding operations (e.g. those concerning the BCS/FCS) are simplified and approximated, each of these processes requires special purpose, real-time hardware. Together, they greatly exceed the real-time capabilities of most laboratories.

This drawback presents serious difficulties for demonstrating the proposed map-making system in real-time. However, given that segmentation and purposive grouping are largely outside the scope of this research, the implementation concentrates on demonstrating the high level tasks in real-time by dividing the problem of map-making into *low-level* visual processing tasks (encompassing contour extraction, perceptual grouping, feature extraction, and purposive grouping) and *high-level* visual processing tasks (encompassing visual search, visual tracking, shape processing, place recognition, action consequence learning, utility learning, and navigation), and mimicking as best as possible the processes of grouping and feature extraction. The imitation of feature extraction entails defining landmarks by bright lights affixed to cardboard models of 3D buildings (like the ones in Figure 32), and using simple thresholding and max-picking operations to extract point features (see Figure 20). This process effectively mimics the extraction of feature points such as high-curvature points on contours in visible imagery. The imitation of grouping operations entails keeping landmarks well separated in space (e.g. placing them in the four corners of the laboratory), thereby insuring that no two landmarks ever enter the same FOV.

Note, however, that some independent empirical experiments have also been performed with regard to independently demonstrating the viability of some of the low-level visual processes not fully implemented here (Bachelder & Waxman, 1992). In particular, corner points have been extracted in real-time from silhouettes of high contrast visual objects. This extraction, implemented on the PIPE (see Figure 21), entails extracting high-contrast edges, segmenting contours using zero-disparity filtering (see Section 5.1.2), and picking out high curvature points using second derivatives (see Kitchen & Rosenfeld, 1982). The resulting features prove quite insensitive to movements of the object. Simplified forms of some low level processes (e.g. shunted center-surround processing, edges and corner extraction, the zero-disparity filter, and the DEB) have also been implemented in real-time on the PIPE video rate processor (see Appendix A), and others (e.g. the BCS/FCS) have been implemented serially on workstations.

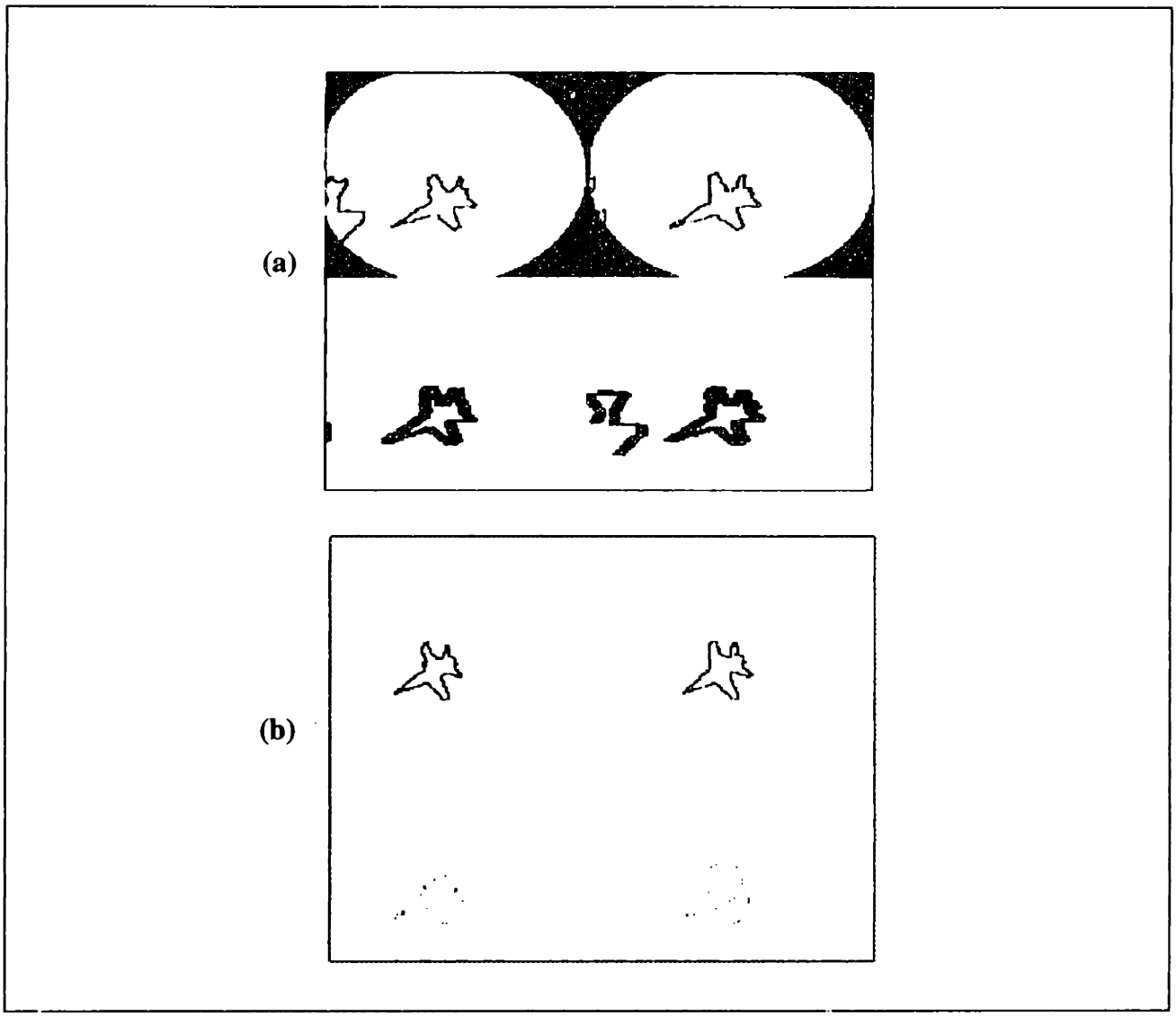


**FIGURE 21: A SIMPLE TEST OF FEATURE SEGMENTATION IN DEPTH.**

*Stable feature points for the silhouette of a model aircraft are segmented in real time from those of another by employing a simplified version of the architecture shown in Figure 19. (a) The upper left and right quadrants show the results of edge extraction within a circular window-of-attention for left and right monocular video streams, respectively. The lower left and right quadrants show the same edges "fattened" in the horizontal dimension according to the current "horopter width" for right and left cameras, respectively. This horopter width determines the window of attention in depth, as implemented by the zero-disparity filter. (b) Upper left and right images show the results of zero-disparity filtering (a short time later), while lower left and right images show the corners (high-curvature points) extracted along these contours. (Adapted from Bachelder & Waxman, 1992)*

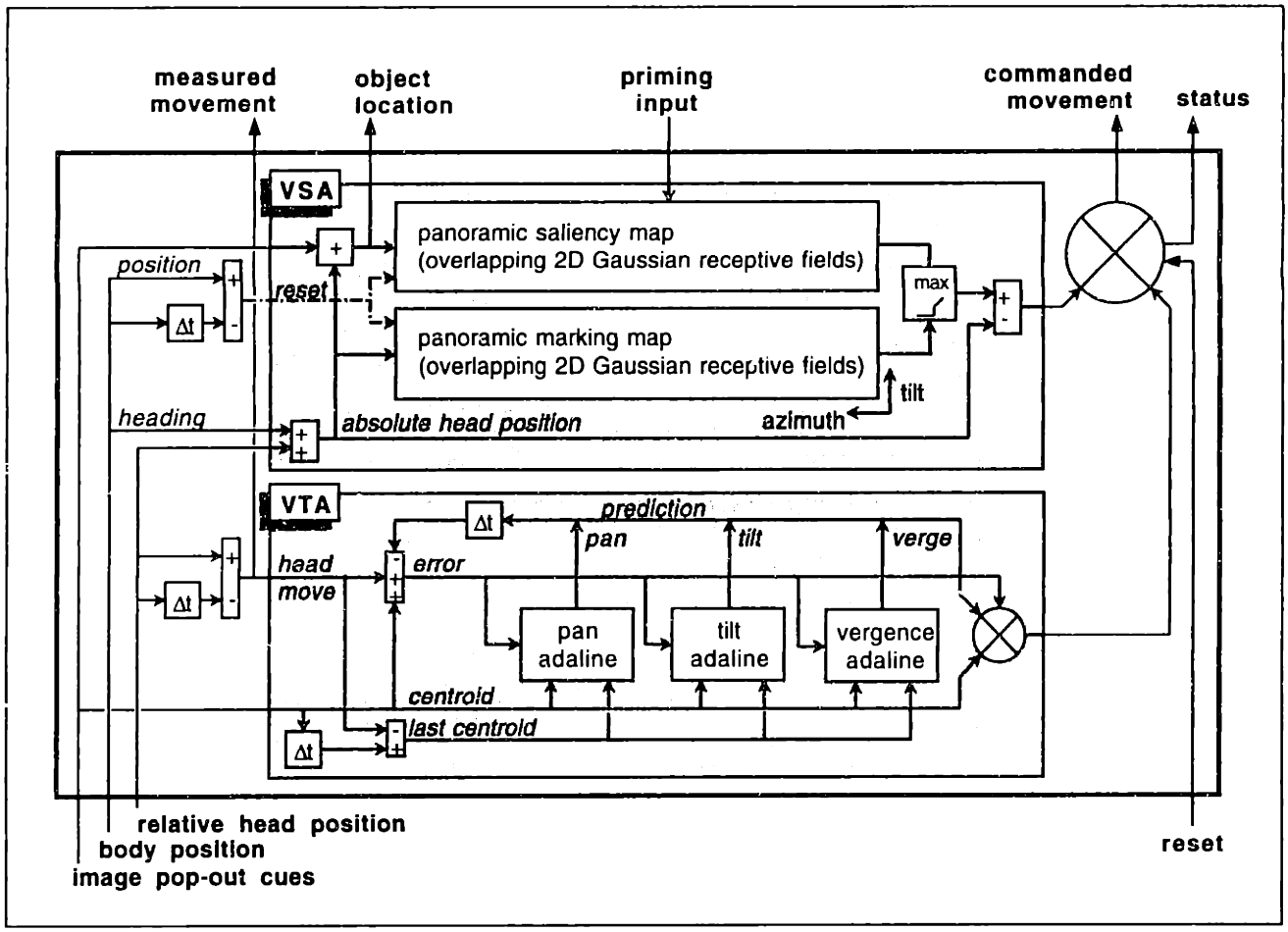


**FIGURE 22: THE GAZE CONTROL SUB-SYSTEM (GCS).**

*The gaze control system (GCS) consists of separate but complementary architectures for visual search (the VSA) and visual tracking (the VTA). These architectures take as input the pop-out cues and features extracted using the featural grouping sub-system (FGS).*

# 6.2 Gaze Control

## 6.2.1 Neural Architecture

The GCS (see Figure 22) takes as input the absolute directions of pop-out cues in the environment (from the FGS), the current head and body positions of the robot, and a signal from the PLA indicating whether the robot should continue tracking the current landmark, or search for a new one. It comprises a *visual search architecture* (VSA) and a *visual tracking architecture* (VTA).

The VSA employs a *saliency map* and a *marking map* to find and serially attend to pop-out cues in the environment panorama (see Section 5.1.1). Both maps coarsely code for the space of possible monocular viewing directions in the visual panorama with respect to "North" using a set of 2D overlapping Gaussian receptive fields. They are updated after each head movement, and gradually reset (decay) over time and when the robot translates through the environment. The marking map

is cumulatively activated by the current gaze direction of the robot head, and thereby records the directions previous searched. The saliency map is cumulatively activated by the locations of pop-out cues of grouped features in the FOV. A top-down priming of the map, either by expectations generated by the PLA or obstacle inquiries made by the path planning sub-system (see Section 6.4), is also possible. For example, the saliency map might be primed proportionally to the differences between the two greatest contenders for place recognition, thereby directing the visual search in order to efficiently resolve ambiguity (similar to the mechanism for object recognition proposed by Seibert and Waxman, 1993). Overt visual attention in the form of a static saccade is directed by that part of the visual panorama with the largest saliency map value and a marking map value below some threshold.

The VTA consists of an attentional controller, and three separate controllers for pan, tilt, and vergence. The attentional processing module (not shown) effects a 3D window of attention in the FGS in order to encompass the tracked object and exclude others. Each of the controllers employs an adaline network (see Simpson, 1990), which learns to predictively track the centroid of the visual target (see also Baloch & Waxman, 1991a,b) within this window. Much like the strategy apparently used by biological systems (see Section 5.1.2), the VTA effects camera movements according to the output of the adalines when the tracking error is small (called a predictive saccade here, since the cameras are not capable of velocity control), and otherwise effects the camera movements according to the error (called a static fixation). Note that the VTA might also perform computations in order to accept or reject the tracked cue as a landmark (e.g. using motion information to determine whether the tracked object is stationary).

Note that both the VSA and VTA issue motion commands to the robot's head, but the head motion actually executed by the robot depends on whether the GCS is currently in tracking mode or searching mode. The GCS automatically enters tracking mode immediately after saccading to the location of a pop-out cue. However, it only enters searching mode when it loses sight of the cue, or when signalled by the PLA. This signal indicates that the OVS has had enough time to assess the appearance of the fixated object and either accept it as a landmark or reject it as a distractor.

### 6.2.2 Dynamics

Both the VSA and the VTA assume that some other visual process extracts and groups visual features for prospective landmarks (see Section 6.1.1). This assumption is currently satisfied by computing the centroids of the extracted landmark features in each FOV by the simplified FGS, which are then converted into angular coordinates for each camera via the perspective projection (pinhole imaging model) equations:

$$(x'_l, y'_l) = \left( \text{atan}\left(\frac{2}{I_x} x_l \tan\left(\frac{\Phi_x}{2}\right)\right), \text{atan}\left(\frac{2}{I_y} y_l \tan\left(\frac{\Phi_y}{2}\right)\right)\right) \quad (1)$$

$$(x'_r, y'_r) = \left( \text{atan}\left(\frac{2}{I_x} x_r \tan\left(\frac{\Phi_x}{2}\right)\right), \text{atan}\left(\frac{2}{I_y} y_r \tan\left(\frac{\Phi_y}{2}\right)\right)\right) \quad (2)$$

where $\Phi_x$ and $\Phi_y$ define the angular FOV of the cameras in the $x$ and $y$ directions, and $I_x$ and $I_y$ are the spatial dimensions (resolution) of the imagery in pixels (the values of these parameters for

TABLE 3: VISUAL SEARCH ARCHITECTURE (VSA) VARIABLE DEFINITIONS.

| Variable | Definition |
|---|---|
| $M$ | The 2D pattern ($n_x \times n_y$ matrix) representing the activities of nodes in the marking map. |
| $S$ | The 2D pattern ($n_x \times n_y$ matrix) representing the activities of nodes in the saliency map. |

MAVIN are given in Appendix A). These angular coordinates are then used to compute the current head displacement $\bar{e}$ with respect to the visual landmark and the two lines of sight via the equations:

$$e_p = \frac{(x'_l + x'_r)}{2} \tag{3}$$

$$e_t = \frac{(y'_l + y'_r)}{2} \tag{4}$$

$$e_v = \frac{(x'_l - x'_r)}{2}. \tag{5}$$

(Note: this conversion is only approximate, since it assumes that moving the head does not affect the nodal point of the camera lenses with respect to the robot's body. The approximation holds for large distances between the landmark and the robot relative to the baseline distance between the two cameras. More accurate error assessment would otherwise require determining the distance to the tracked object.)

For visual tracking, the three components of the head displacement $\bar{e}$ constitute a tracking error, and are respectively sent to each of the three adalines in the VTA, along with the most recent head movement $\Delta H^{in}$, and the error measured prior to this movement. For each dimension, the next head motion (i.e. $\Delta H_p^{out}$, $\Delta H_t^{out}$, or $\Delta H_v^{out}$) is determined by the output of the corresponding adaline if the error is less than some threshold $\Upsilon$, and is otherwise determined by the error itself.

For visual search, the head displacement $\bar{e}$ serves as input to the saliency and marking maps (a mathematical definition of which can be found in Table 3). First, it is converted to an absolute direction $\bar{\gamma}$ (measured with respect the environment) using the equations:

$$\gamma_a = B_h + H_p + e_p \tag{6}$$

$$\gamma_t = H_t + e_t. \tag{7}$$

(Note this landmark direction assessment is also passed on to the PLA in order to define the spatial direction to the current landmark with respect to "North.") These coordinates activate each of

the nodes with 2D Gaussian receptive fields in the $n_x \times n_y$ saliency map (where $n_x$ and $n_y$ are the azimuth and tilt dimensions, respectively) according to the update equation

$$S_{ij} \Leftarrow S'_{ij} + \text{pos}(S_{ij} - S'_{ij}) \cdot e^{\rho\left(c_i^x\right)\Delta\tau} \tag{8}$$

where $i$ and $j$ are the azimuth and tilt indices, respectively, $\Delta\tau$ is the length of time since the last update, and

$$S' = \text{RF}^2\left(\gamma; \begin{pmatrix} n_x \\ n_y \end{pmatrix}, \begin{pmatrix} -\pi \\ H_t^{\min} \end{pmatrix}, \begin{pmatrix} \pi \\ H_t^{\max} \end{pmatrix}, \begin{pmatrix} p \\ f \end{pmatrix}\right) \tag{9}$$

(see Appendix C for an in-depth description of receptive field coding). Note that when $S'_{ij} > S_{ij}$, equation 8 is simply the discrete form of the short-term memory equation

$$\frac{dS_{ij}}{dt} = \rho\left(c_i^x\right)(S'_{ij} - S_{ij}) \,. \tag{10}$$

Otherwise, it simply implements a latch (i.e. $S_{ij} = S'_{ij}$).

In a similar fashion, the marking map is activated by the absolute head direction according to the equation

$$M_{ij} \Leftarrow M'_{ij} + \text{pos}(M_{ij} - M'_{ij}) \cdot e^{\rho\left(c_i^x\right)\Delta\tau} \tag{11}$$

where

$$M' = \text{RF}^2\left(\begin{pmatrix} B_h + H_p \\ H_t \end{pmatrix}; \begin{pmatrix} n_x \\ n_y \end{pmatrix}, \begin{pmatrix} -\pi \\ H_t^{\min} \end{pmatrix}, \begin{pmatrix} \pi \\ H_t^{\max} \end{pmatrix}, \begin{pmatrix} p \\ f \end{pmatrix}\right). \tag{12}$$

In both cases, the vector $[c_i^x, c_j^y]$ represents the azimuth and tilt of the center of receptive field $ij$ (coding azimuth $i$ and tilt $j$), given by

$$[c_i^x, c_j^y] = \text{FC}^2\left(i, j; \begin{pmatrix} n_x \\ n_y \end{pmatrix}, \begin{pmatrix} -\pi \\ H_t^{\min} \end{pmatrix}, \begin{pmatrix} \pi \\ H_t^{\max} \end{pmatrix}, \begin{pmatrix} p \\ f \end{pmatrix}\right) \tag{13}$$

(again, see Appendix C). The modulating term $\rho(c_i^x)$ dictates the rate of decay for each of the receptive field elements as a function of the degree to which translational movements through the environment affect the appearance and viewing direction of objects in the spatial direction corresponding to the center of the receptive field. It is given by

$$\rho(\varphi) = \delta + \kappa A_\rho^{\text{in}} \sin\left(\text{atan}\left(\frac{\Delta B_x^{\text{in}}}{\Delta B_y^{\text{in}}}\right) - \varphi\right) \tag{14}$$

where $\delta$ is the passive decay rate, and $\kappa$ is the travelling decay rate. Map elements coding for space in front of the robot decay at rate $\delta$, while elements perpendicular to the direction of travel decay at rate $\delta + \kappa A_p^{in}$. (This measure, of course, completely ignores the fact that the viewing direction and appearance of distant objects will change much less rapidly than that of nearby objects. It also ignores the effects of changes in the tilt component of the viewing direction, especially for nearby objects that are much lower or higher than the robot. These effects could in principal be incorporated at the expense of simplicity.) After each of these map updates, $\Delta\theta^{out}$ is computed by taking the difference between the current absolute head direction and the direction $[c_i^x, c_j^y]$ closest to the current head direction for which $S_{ij} > \xi$ and $M_{ij} < \zeta$ (where $\xi$ and $\zeta$ are the saliency and marking thresholds, respectively).

The head motion actually executed by the robot is determined by the GCS mode. The GCS begins in search mode. It switches to track mode whenever a centroid that satisfies the saliency and marking tests described above enters the FOV of both cameras (a new target has been found). It switches back to search mode whenever the centroid disappears from view (the target has been lost), or when instructed to do so from an external source (e.g. the PLA).

## 6.3 Object Vision

The object vision system (OVS) is simply a variant of the Seibert-Waxman object vision system described in Section 2.2.2, shown in more detail in Figure 23. The system takes as input a sequence of image frames. From each frame it segments out and extracts stable object features (e.g. edges and high curvature points) using several neural network architectures. These features also drive eye motions, which keep the object within the FOV and reset short-term memory structures. Note that the OVS uses the FGS described in Section 6.1 to extract point features (high curvature points) and their centroids (contour centroids), and the GCS (described in Section 6.2) to control eye motions.

The OVS transforms the extracted features via a shift to centroid, a *log-polar mapping* (akin to the connections between LGN and primary visual cortex), and a final shift to centroid in log-polar space. This transformation achieves invariance to translation, scale, and orientation around the line of sight. Next, it coarsely codes the resulting feature pattern using a small array (e.g. 9x9) of overlapping 2D Gaussian receptive fields (see Appendix C), thereby creating an invariant 2D view pattern largely insensitive to deformations due to rotation in depth. This invariant pattern feeds an ART network, which clusters these views into *aspect categories*. The winner-take-all (binary) output of the ART network feeds a set of *aspect nodes* with short-term memory (passive decay) dynamics. Finally, an associative network called an *Aspect Network* uses a Hebbian rule with both pre- and post-synaptic gated decay to learn the allowable transitions between aspect categories by witnessing the temporal overlap between the activity of each of their respective aspect nodes.

As a consequence of this unsupervised approach to learning, an object is thereby quantized in a self-organizing fashion into *prototypical views*, each associated with a learned *aspect template*. The resulting aspect categories parcel the explored portion of the viewing sphere into aspect regions (see Figure 13). Recognition of any single aspect category provides static evidence for the object for a single image frame, while the learned transitions between these aspect categories

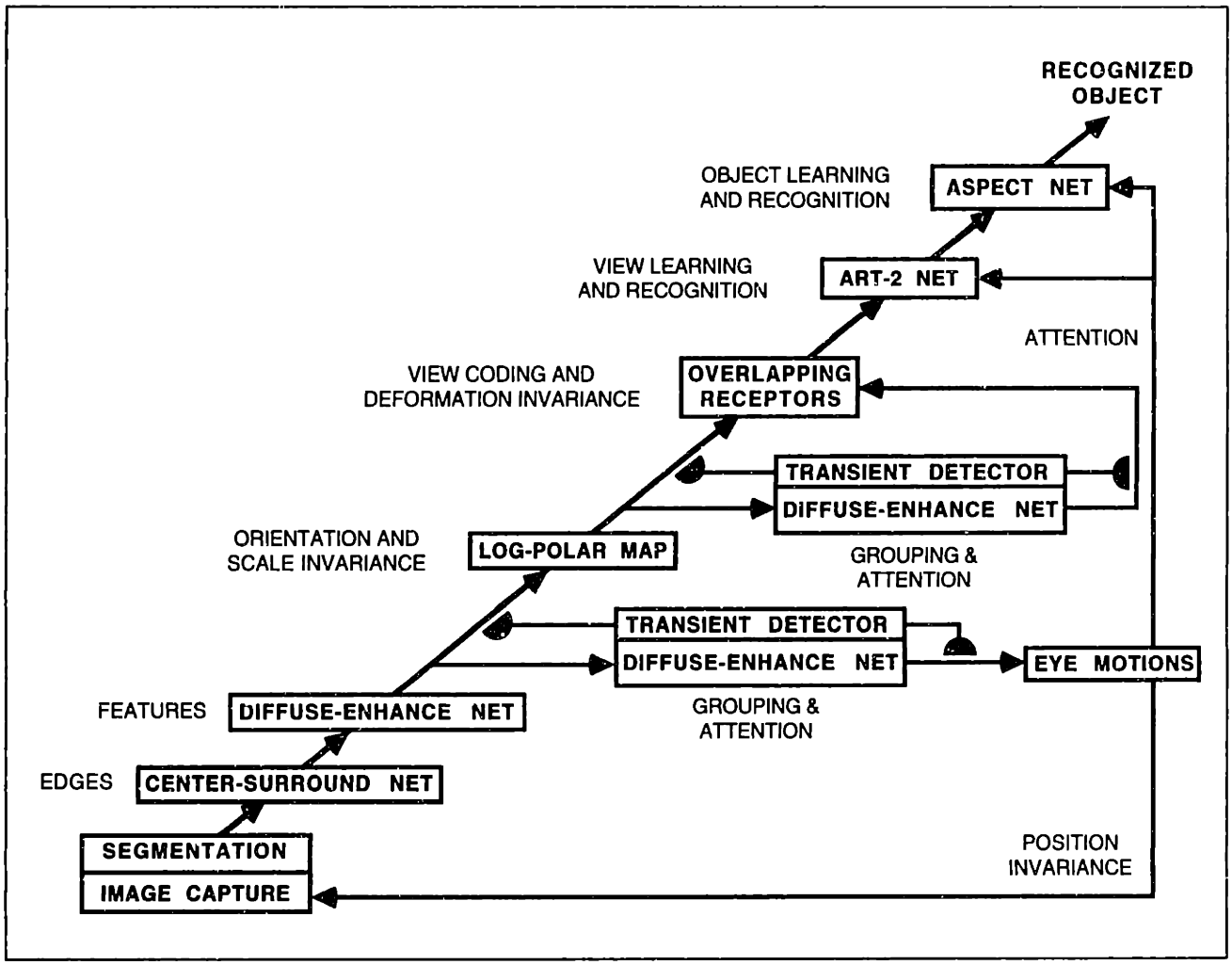


**FIGURE 23: THE OBJECT VISION SUB-SYSTEM (OVS).**
*The neural system for unsupervised visual 3D object learning and recognition (adapted from Seibert & Waxman, 1992).*

resolve ambiguity by accumulating dynamic evidence for the object over multiple image frames. Both static and dynamic evidence are combined, and indicated by the activity of an *object node*.

As alluded to in Section 2.2.3, a recent biologically-inspired modification to the Seibert-Waxman object vision system removes the final shifting operation in the polar dimension of the log-polar space, thereby preserving orientation (though not scale) specificity. As a result of coarse coding by the overlapping receptive fields in the formation of the 2D invariant view pattern, the learned aspects become broadly tuned (as opposed to invariant) to orientation. Interestingly, this modification overcomes several computational instabilities with finding the centroid of nearly symmetric views of objects. This modification has also been incorporated into the OVS.

The OVS provides the place learning architecture described in the next chapter with a code for the invariant appearance of the currently fixated landmark, which essentially determines the landmark's identity and pose. This code might consist of the pattern of aspect evidence, object evidence, or invariant view nodes. A mathematical definition of the variables corresponding to each

TABLE 4: OBJECT VISION SYSTEM (OVS) VARIABLE DEFINITIONS.

| Variable | Definition |
|---|---|
| $\bar{o}$ | The 1D pattern ($n_o$-dimensional vector) representing the activities of the object nodes (current evidence for each of the learned objects). |
| $\bar{v}^i$ | The 1D pattern ($n_i^o$-dimensional vector) representing the activities of the view category nodes (F2 layer in the ART network) for object $i$, assessed before winner-take-all competition (current evidence for each of the learned aspects of each of the learned objects). |
| $F$ | The 2D pattern ($n_s \times n_r$ matrix) representing the activities of the invariant view pattern (the result of applying a set of 2D Gaussian receptive fields to the log-polar transformed spatial features). |

of these possibilities is presented in Table 4. The OVS is also capable of accepting or rejecting objects as landmarks for the PLA based on some qualitative appearance criteria. Criteria for landmark acceptance might be as general as requiring a minimum number or density of visual features, or as specific as requiring that the current view be an instance of an aspect of one of a particular set of objects. Finally, the OVS is capable of answering questions regarding what the robot might expect to see in a particular place or environment. The identity dimension of each place template (see next chapter) constitutes a set of appearance expectations for that place, each of which form an appropriate input to the ART network in the OVS, thereby allowing landmark identification.

## 6.4 Path Planning

### 6.4.1 Proposed Architecture

Obstacle avoidance is largely outside the scope of this research. The idea of qualitative mapping in large-scale space allows a robot to separate the computations associated with route planning between distinctive places from those concerned with path planning within distinctive places (see Section 5.2.1). Note, however, that path planning can be thought of as the lowest level of a hierarchical approach to navigational planning (see Section 3.4), receiving sub-goals from the ACLA. In fact, such a hierarchical approach lessens the demands placed upon the path planner, since only local paths must be considered at any given time.

To illustrate this point, a brief and general description of such a simple path planning sub-system (PPS) is included here as part of a fully integrated navigation system (see Figure 24). This architecture takes as input the current body position of the robot $B$, as well as the desired body-centered locomotive action $\bar{A}^{out}$ issued as a result of the route planning processing in the ACLA (see

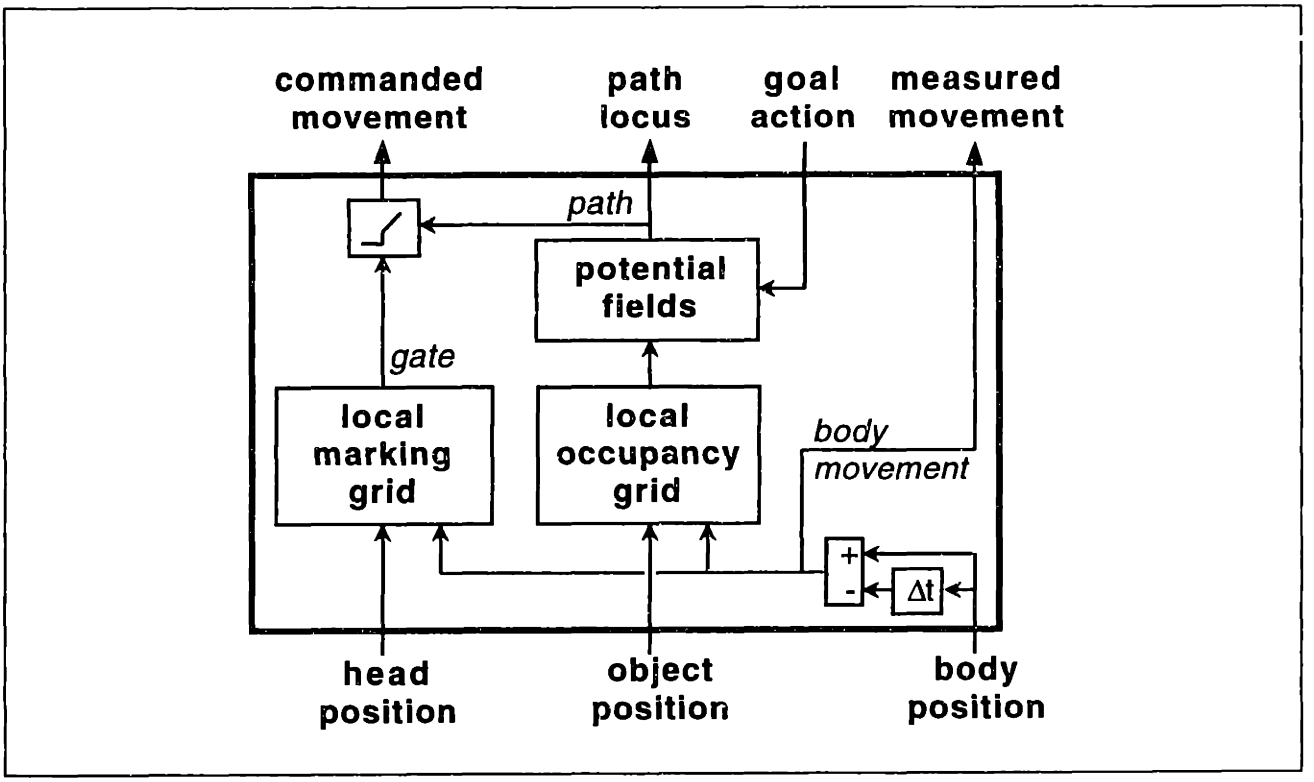


**FIGURE 24: A PATH PLANNING SUB-SYSTEM (PPS).**

*A path planner might simply record the positions of landmarks within a local body-centered coordinate system using a local occupancy grid, and use a potential fields approach to determine the best path within the grid. Movement commands might also be gated by a local marking map that records the positions in the environment that have been previously searched by the gaze control system.*

Chapter 8). It computes the most recent change in body position $\Delta B^{in}$ (over some specified action time interval $\tau_A$) and converts it to the body-centered action representation $\bar{A}^{in}$ via

$$A_\theta^{in} = \left( \text{atan}\left(\frac{\Delta B_x^{in}}{\Delta B_y^{in}}\right) - B_h \right)_{-\pi}^{\pi} \tag{15}$$

$$A_\rho^{in} = \sqrt{\left(\Delta B_x^{in}\right)^2 + \left(\Delta B_y^{in}\right)^2} \tag{16}$$

$$A_\phi^{in} = \left(\Delta B_h^{in} - A_\theta^{in}\right)_{-\pi}^{\pi}. \tag{17}$$

This locomotive action is then passed on to the ACLA, while the absolute body heading $B_h$ is passed on to the PLA.

The desired movement $\bar{A}^{out}$ then becomes a goal vector (location) in a local, body-centered metric coordinate system. To move the robot to this goal location safely, the PPS might employ coarse reconstructive techniques, such as depth from binocular disparity, to build up a local map of obstacles (landmarks and non-landmarks) in the robot's immediate vicinity. To deal with uncer-

tainty, the local map might be constructed using a set of 2D overlapping Gaussian receptive fields cumulatively activated by the location of the currently fixated object, thereby forming a structure similar to an *occupancy grid* (see Elfes, 1987). The size of these receptive fields might be a function of distance from the origin, reflecting a decrease in accuracy of assessed object positions with distance from the robot. The robot could also employ a marking grid similar to the marking map used in the GCS in order to record the areas in the environment that have already been visually searched for obstacles (cumulatively activated by the current fixation point distance and direction). Again, both of these maps would decay over time. However, evidence in the two maps might simply be shifted as the robot makes translational movements.

Using the occupancy map, the robot might employ parallel path planning techniques, such as *potential fields*, to continuously update a possible collision-free path to the current local goal location (see Hwang & Ahuja, 1992). The locus of locations along the current path could then prime the corresponding visual directions in the saliency map of the GCS, thereby "requesting" that the GCS direct the robot's attention to unexplored portions of the path so that obstacles might be detected. When all the points along the current path have been marked on the marking grid, the PPS could then issue the motor command $\Delta B^{out}$ corresponding to the first location in the computed path.

Under this scenario, place recognition and path planning (and perhaps even other gaze-intensive systems with little to do with visual navigation) utilize the GCS concurrently, essentially using the saliency map as a "blackboard".

## 6.4.2 Simplifications

In implementing the proposed navigation system, the PPS is, for now, simply bypassed. The environment is assumed to contain no obstacles to avoid. In place of the PPS, the navigation system simply measures the current body position $B$, computes the recent body movement $\Delta B^{in}$ and converts it to the body-centered action representation $\bar{A}^{in}$ via equations 15-17, and issues $B_h$ to the PLA and $\bar{A}^{in}$ to the ACLA. Likewise, it takes the desired body-centered locomotive action $\bar{A}^{out}$, converts it to the environment-centered representation $\Delta B^{out}$ via

$$\Delta B_x^{out} = A_\rho^{out} \cos\left( B_h + \frac{\pi}{2} - A_\theta^{out} \right) \tag{18}$$

$$\Delta B_y^{out} = A_\rho^{out} \sin\left( B_h + \frac{\pi}{2} - A_\theta^{out} \right) \tag{19}$$

$$\Delta B_h^{out} = \left( -A_\phi^{out} - A_\theta^{out} \right)_{-\pi}^{\pi} \tag{20}$$

and issues it directly to the actuators.

## 6.6 Summary

This chapter has described the role of some of the supporting sub-systems for featural grouping (FGS), gaze control (GCS), object vision (OVS), and path planning (PPS). The FGS computes pop-out cues and high curvature points from binocular visual input indicating the possible positions and visual features for landmarks in the FOV, respectively. These cues and features are fed to the GCS and the OVS, similar to the splitting of object and spatial vision streams in the primate visual system. The GCS consists of two complementary architectures for visual search (VSA) and visual tracking (VTA). The VSA employs topographic saliency and marking maps to serially search for visual landmarks, while the VTA uses neuro-controllers to keep the currently fixated object in the FOV. Concurrently, the OVS processes the appearance of the currently fixated object, and either rejects or accepts it as a visual landmark. Finally, the PPS executes high level locomotion commands while keeping the robot from colliding with obstacles. A possible architecture has only briefly been described here for completeness, and to clarify the types of structures and interactions the sub-system might have with the rest of the navigation system.

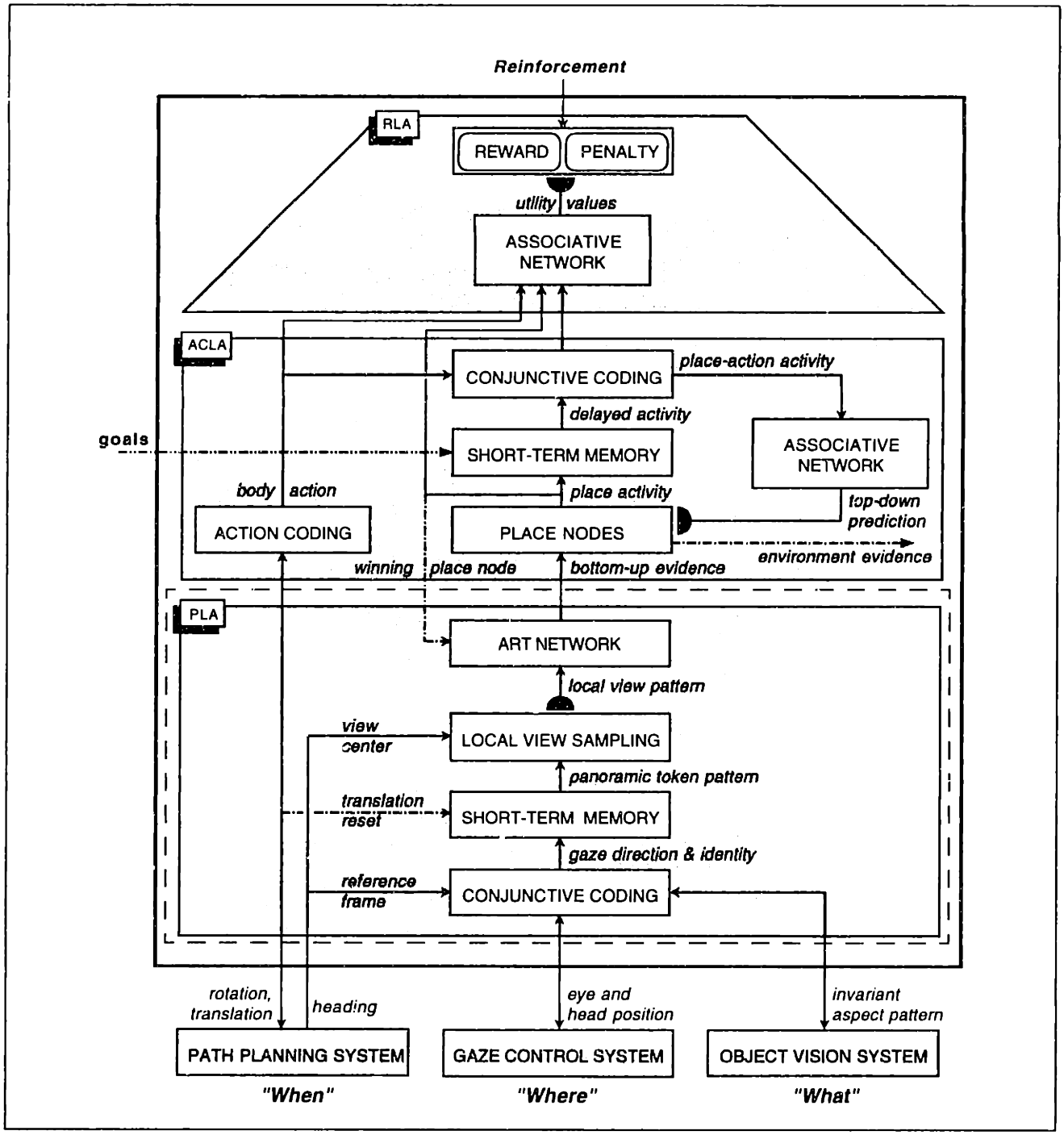


**FIGURE 25: THE PLACE LEARNING ARCHITECTURE (PLA).**
*The place learning architecture, highlighted here as part of the navigation strategy learning sub-system, takes as primary input the appearance (the "What") and spatial direction (the "Where") of the currently fixated landmark.*

CHAPTER 7

# *Place Learning and Recognition*

The place learning architecture (PLA), highlighted in Figure 25, combines two possible view-based methods for defining places using landmarks: by the *spatial directions* (the "Where") of visible landmarks, and by the *visual appearance* (the "What") of visible landmarks (Bachelder & Waxman, 1994; see Figure 26 and Figure 27, respectively). It self-organizes (i.e., clusters) an environment into place categories, each defined by an adaptable template pattern representing a coarse conjunctive coding of 2D landmark shape and spatial direction within a restricted local view of the environment. In the sense that this template represents a prototypical "iconic snapshot," the PLA at some level implements McNaughton's (1989) local sensory view model of hippocampal place cells (see Section 5.2.2).

This chapter first describes the PLA at the architectural level. It then discusses various ways to code landmark identity within the PLA, goes on to define the dynamics of panoramic local view coding, and reports results of implementing a semi-egocentric version of the architecture on the mobile robot MAVIN. After a brief analysis of these preliminary results, the chapter describes the dynamics of restricted local view coding, and reports the results of implementing a fully egocentric version on MAVIN and using it to learn the visual environment. Finally, it discusses and presents the results for a way in which to learn places at multiple spatial scales.

## 7.1 Neurocomputational Architecture

To coarsely code landmark shape (the "What"), the PLA employs the OVS to form invariant landmark views. Possible candidates for these views include the pattern of object evidence across learned landmark representations, the pattern of aspect evidence across such representations, or the unclassified 2D invariant view (see Table 4 in Chapter 6). Each of these candidates has its effect on performance (see Section 7.2). To coarsely code landmark direction (the "Where"), the PLA uses internal measurements of body position and head direction, and external measurements of the fixated 2D landmark centroid within the visual FOV. This computation involves two independent, fixed populations of nodes with overlapping Gaussian receptive fields. The first popula-

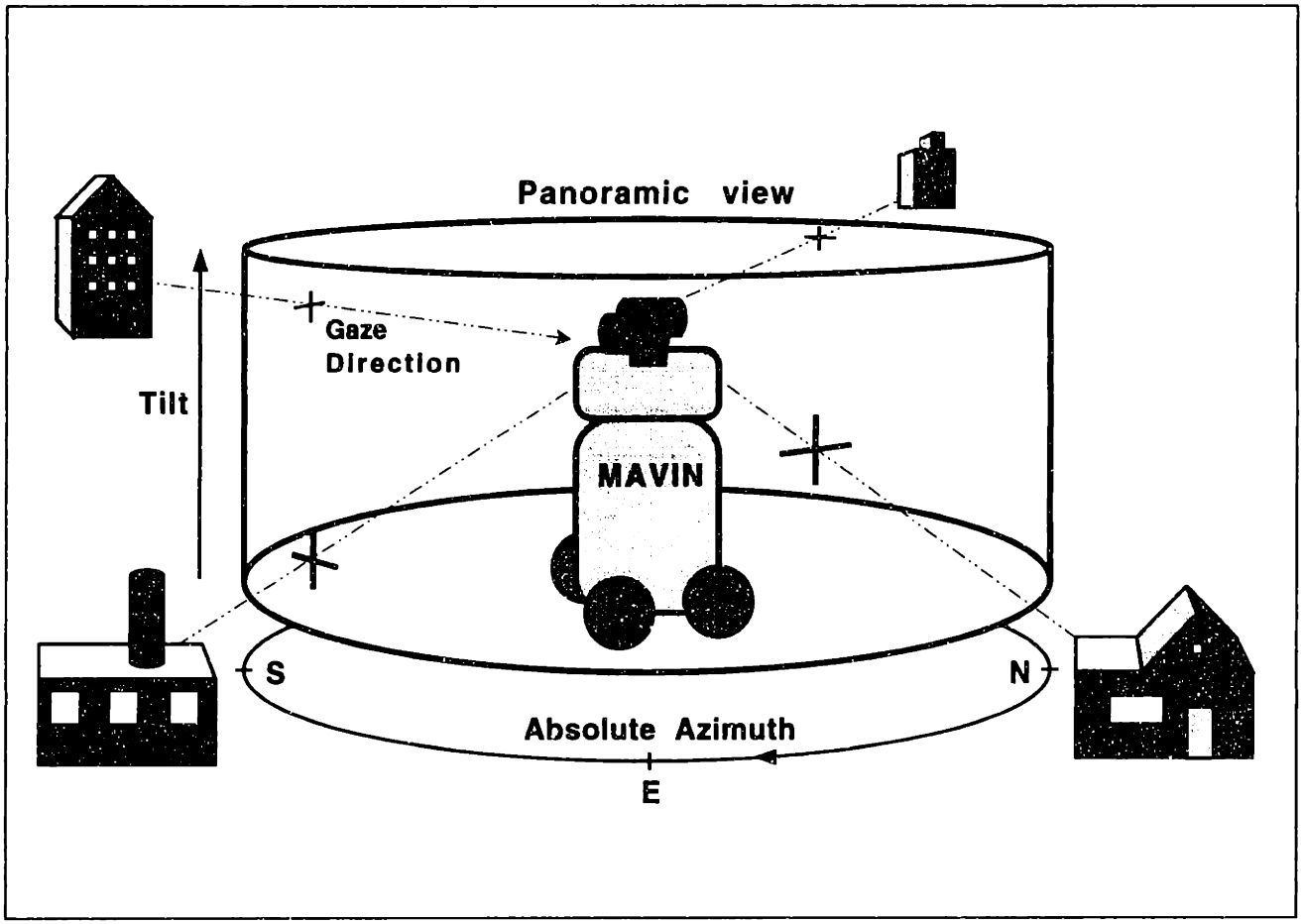


**FIGURE 26: A LANDMARK POSITION-BASED DEFINITION OF PLACES.**
*Defining places by prototypical locations within the landmark constellation. The location of the robot is indicated by the distribution of gaze directions (absolute azimuth and tilt) to the feature centroids of landmarks, plotted here in a panoramic view of the environment.*

tion coarsely codes for landmark azimuth, computed using the horizontal coordinates of the landmark centroid, the pan position of the head, and the heading of the robot (with respect to an arbitrary but locally consistent reference direction, such as "North"). The second population coarsely codes for landmark elevation, computed using the vertical coordinates of the landmark centroid, and the tilt position of the head.

To perform place recognition, the robot must continually search the visual panorama for landmarks (see Figure 28). During this process, a short-term topographic memory gradually builds up a panoramic pattern from the current conjunction of spatial and identity coding nodes. This memory, reminiscent of the fixed mapping scheme proposed by Hinton (1990) for forming whole-part hierarchies, resets (decays) gradually over time, and when the robot makes translational movements. Concurrently, a local view sampling mechanism takes a restricted "snapshot" of the panoramic memory in the direction of the robot's heading. Once the robot has sufficiently visually searched the portion of the panoramic scene corresponding to the local view, this snapshot pattern is fed to an ART network, and place learning and recognition proceeds. Finally, the unprocessed output of the ART network feeds a set of sensory place nodes.

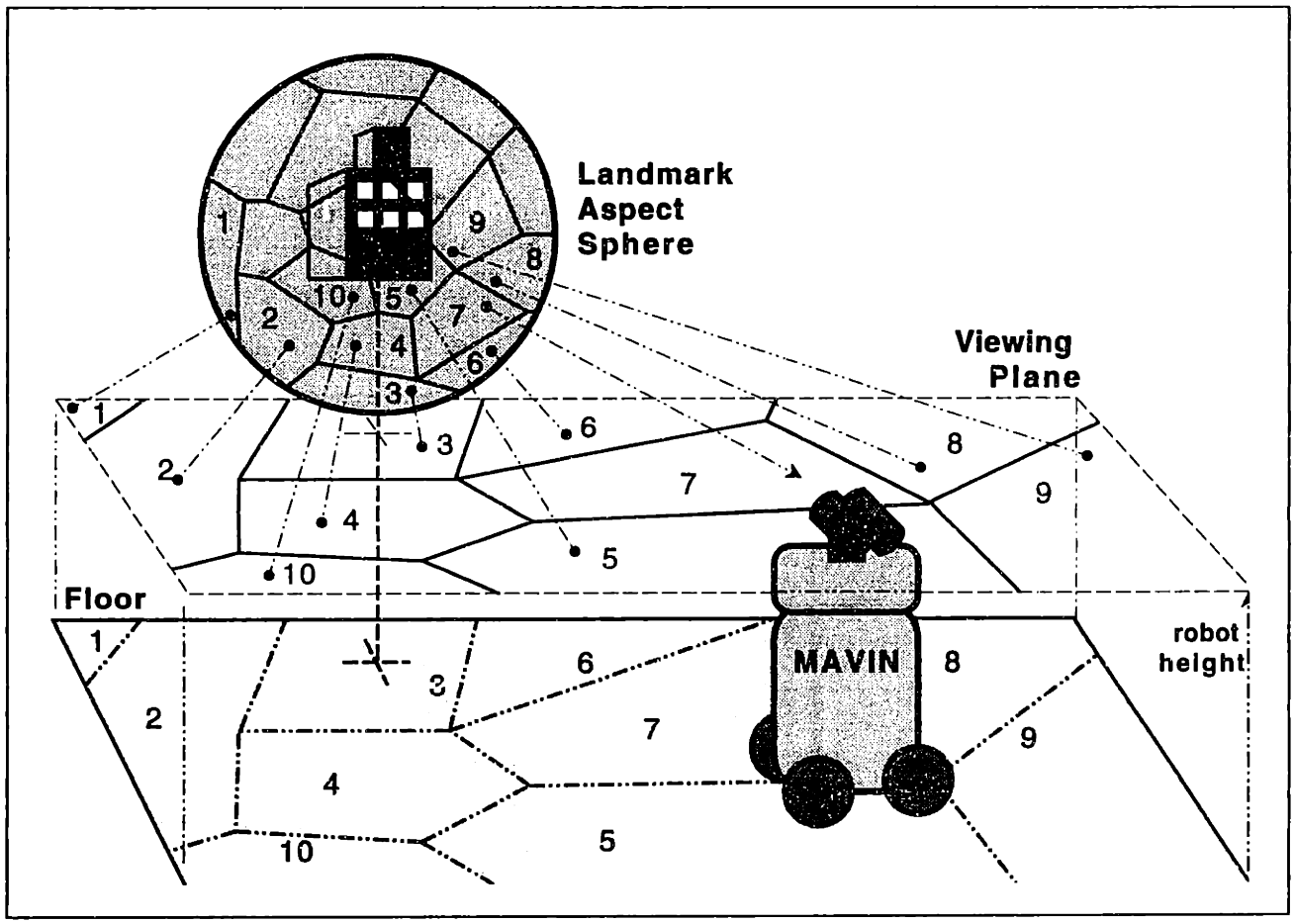


**FIGURE 27: A LANDMARK APPEARANCE-BASED DEFINITION OF PLACES.**
*Defining places by the identity and pose of a single visual landmark. The environment is parcelled into numbered place regions, shown conceptually here on the floor, that correspond to the numbered landmark aspect categories visible from the viewing plane. (Note that the shapes of the aspect regions are simplified here for purpose of illustration.)*

This approach draws from the convergence of spatial and object vision streams in the hippocampus of primates (see Section 2.2.1), and therefore possesses many of the features of other landmark- and image-based approaches to place recognition and topological map-making (see Section 5.2.1). Its major difference lies in the fusion of object form and direction, and in the computation of rotationally insensitive views via non-uniform spatial processing, both of which have a number of computational advantages. Concisely stated, these differences serve to decrease sensitivity to imaging conditions, speed up the place recognition search process, and free the robot from problems associated with dead-reckoning. They also pave the way for learning relational cognitive maps in terms of relative forward and turning motions of the robot.

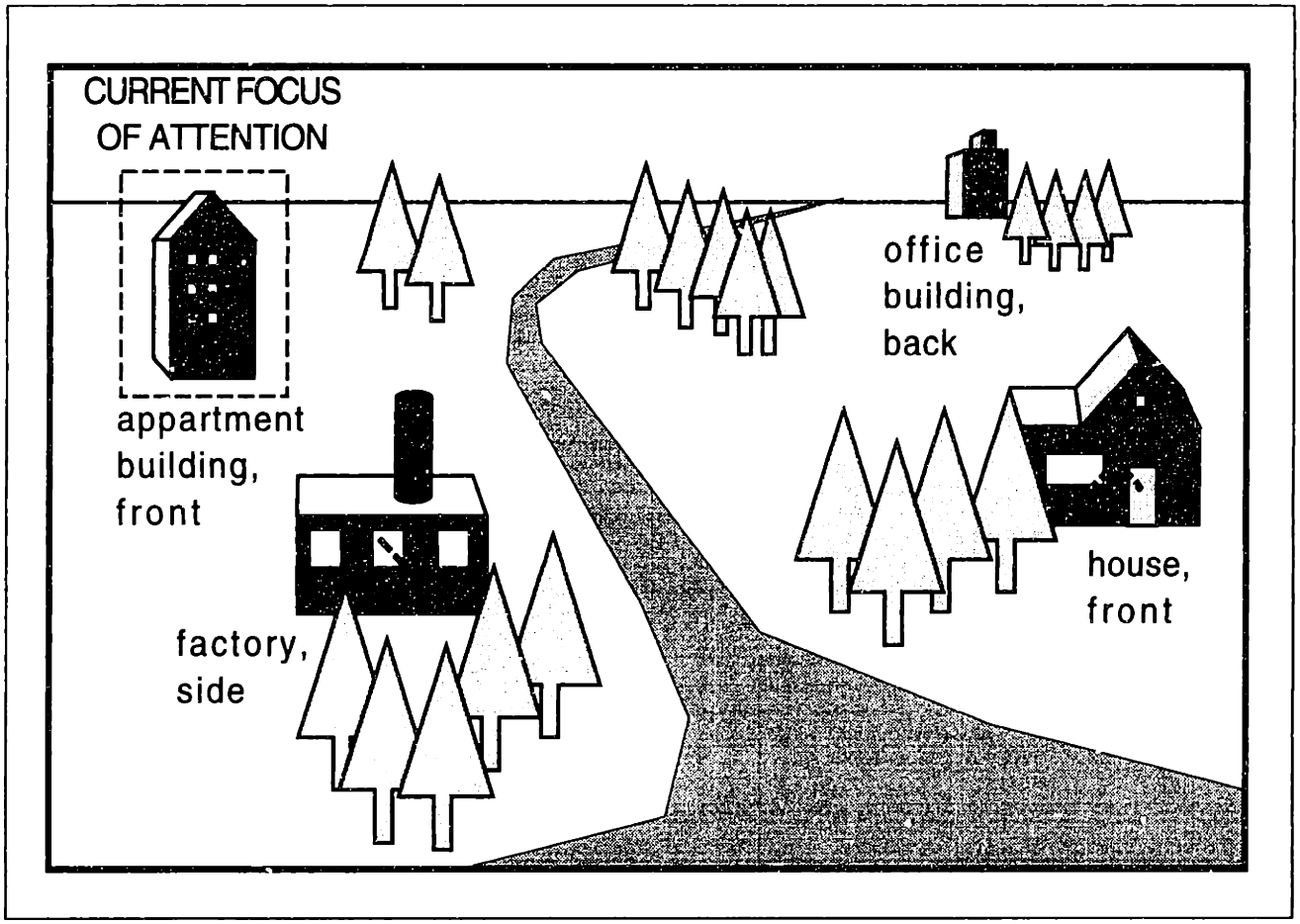


**FIGURE 28: A LOCAL VIEW DEFINITION OF PLACES.**

*Defining a place by a combination of identity, pose, and spatial direction of visible landmarks within a local view of the environment. To assess this local view, the robot must continually search the visual panorama by sequentially attending to individual landmarks. This particular view can be likened to what a paperboy might see from a particular place along his route, with "X" marks indicating visual directions of landmarks to which he has previously attended.*

## 7.2 Shape Processing in the Object Vision System

As mentioned earlier (see Section 7.1), the PLA might employ several different representations of landmark shape using various intermediate representations in the OVS (see Section 2.2.2). Early implementations of the PLA hierarchically represented landmark shape using the pattern of object or aspect evidence across the resident (previously learned) models for landmarks in the OVS (see Bachelder & Waxman, 1994, for a discussion). Using the object evidence $\bar{o}$, the PLA can only map environments with 3 or more unoccluded, previously learned landmarks, primarily because it ignores landmark pose information. Using the aspect evidence $\text{cat}(\bar{v}^1, \ldots, \bar{v}^{n_{n_o}})$, which is more biologically plausible, the PLA can map environments consisting of at least one unoccluded, previously learned landmark, but place templates become quite large because there are an order of magnitude more aspects $(n_1^o \cdot n_2^o \cdot \ldots \cdot n_{n_o}^o)$ than objects $(n_o)$. Note that neither hierarchical coding scheme requires that correct landmark recognition be performed in order to perform place learn-

ing or recognition; it only requires that the shape of landmarks remain roughly the same between recognition episodes, thereby creating sufficiently similar token node patterns. Furthermore, the performance of place recognition with hierarchical shape coding degrades gracefully with the degree of landmark occlusion, since the gradual occlusion of a single landmark will gradually change only a single portion of the evidence pattern.

Unfortunately, such hierarchical coding implies that learning and adapting landmark representations lower in the hierarchy (i.e. in the OVS) causes instability in the place representations at the top of the hierarchy (in the PLA), and also causes the place templates to grow. Practically speaking, then, hierarchical shape coding requires that landmark learning strictly precede place learning. To meet this requirement, MAVIN has used the OVS to learn and subsequently recognize visual landmarks during a separate landmark exploration phase, mainly by tracking and viewing each landmark from novel vantage points while traversing around it, and by witnessing the unfolding of aspect categories and category transition events (see Bachelder & Waxman, 1992, 1994). Perhaps an even more disadvantageous consequence of this approach, however, is that it places even more stringent demands upon the process of landmark segmentation and feature extraction. That is, it requires that features be extracted exclusively from each landmark.

A more recent method for coding invariant landmark shape uses the fixed-size, unclassified (orientation specific) 2D invariant view pattern $\text{cat}(\boldsymbol{F})$ from the OVS (Bachelder et. al., 1994; Bachelder & Waxman, 1995). This method retains the ability to learn an environment using only a single landmark, as well as the ability to perform place recognition without performing landmark recognition. Additionally, it allows landmark learning and place learning to be performed concurrently without interference. It can also deal with the partial or complete occlusion of landmarks during both learning and recognition, provided that such occlusions remain consistent from all positions (i.e. landmarks and obstacles remain stationary, at least in the short term). Hence, this method truly captures the invariant appearance of landmarks within a snapshot of the environment! In fact, such landmarks need not correspond to separable physical objects (see Section 5.1.1). One disadvantage of this method is that it makes the integration of new information into the panoramic view more difficult (from multiple gazes), basically because it forgoes the pattern normalization operations that are performed by the ART network in the OVS. Methods for sensor fusion might alleviate this difficulty (Luo & Kay, 1989). Another disadvantage is that the identity of landmarks in a particular place is not ready obtained by looking at the place representation. Instead, one must apply the landmark recognition system to each portion of place template (like a visual search through memory).

## 7.3 Panoramic Coding Dynamics

The dynamics of place learning and recognition in the PLA (see Section 7.1) are presented here using the variables described in Table 5. After each head movement, which takes time $\tau_{H}$ to effect, the following steps are taken to update these variables:

1. The spatial input $\gamma$ for the azimuth and tilt position, respectively, of the currently fixated landmark (gathered from the search architecture in the GCS — see equations (6) and (7) in Chapter 6) is coded by three populations of nodes with overlapping 1D Gaussian receptive fields (see Appendix C). That is,

TABLE 5: PLACE LEARNING ARCHITECTURE (PLA) VARIABLE DEFINITIONS.

| Variable | Definition |
|---|---|
| $\bar{r}^a, \bar{r}^t, \bar{r}^I$ | The 1D patterns (vectors with dimensions $n_a$, $n_t$, and $n_I$ respectively) representing the activities of the azimuthal, tilt, and object identity coding nodes, respectively. |
| $T$ | The 3D pattern ($n_a \times n_t \times n_I$ array) representing the activities of the token nodes. $T_{ijk}$ is the activity of the token node corresponding to the panoramic spatial direction $ij$ (azimuth $i$ and tilt $j$) and object coding node $k$. |
| $V^P$ | The 3D pattern ($n_a \times n_t \times n_I$ array) representing the activities of the short-term token memory nodes. This pattern codes for the current *panoramic* local view of the environment. |
| $V^R$ | The 3D pattern ($n_{Ra} \times n_t \times n_I$ array) representing the activities of the restricted local view nodes. $V^R_{ijk}$ is the activity of the token node corresponding to the local view spatial direction $ij$ (local azimuth $i$, and tilt $j$) and object coding node $k$. This pattern codes for the current *restricted* local view of the environment. |
| $\bar{s}$ | The 1D pattern ($n_p$-dimensional vector) representing the activities of the *sensory* place nodes. |

$$\bar{r}^a = \mathrm{RF}_I(\gamma_a; n_a, -\pi, \pi) \tag{21}$$

$$\bar{r}^t = \mathrm{RF}_I(\gamma_t; n_t, H_t^{\min}, H_t^{\max}) \tag{22}$$

where $n_a$ and $n_t$ are the number of azimuth and tilt coding nodes respectively.

2. The landmark features are processed in the OVS to produce a set of $n_I$ identity coding nodes, $\bar{r}^I$ (see previous section), which are normalized according to

$$\bar{r}^I \Leftarrow \frac{\bar{r}^I}{|\bar{r}^I|} \tag{23}$$

(see previous section for how these object identity coding nodes are determined).

3. The activity of the identity coding nodes $\bar{r}^I$ is combined with the azimuth and tilt coding nodes $\bar{r}^a$ and $\bar{r}^t$ to produce a set of token nodes via the equation

$$T = \mathrm{conj}(\bar{r}^a, \bar{r}^t, \bar{r}^I). \tag{24}$$

This 3D pattern indicates the direction and identity of the currently fixated landmark.

4. The instantaneous activities of the token nodes $T$ are fed to a short-term memory $V^P$, which encodes all the landmarks in the visual panorama, and decays over time during translational movements and in the absence of input, according to the short-term memory equation

$$V^P_{ijk} = T_{ijk} + \text{pos}(V^P_{ijk} - T_{ijk}) \cdot e^{-\rho(c^a_i)\tau_H} \tag{25}$$

where $\rho(\varphi)$ is as defined in equation (14) (see Chapter 6), and

$$c^a_i = \text{FC}_p(i; n_a, -\pi, \pi) \tag{26}$$

is the receptive field center of the azimuth coding node $i$. Equation 25 implements a passive decay when the incoming token node activity is smaller than the activity of the corresponding panoramic view memory node. Otherwise, it implements a latch.

## 7.4 Results for a Semi-Egocentric Version

In early versions of the PLA (see Figure 29), the local view sampling simply passed the entire panoramic view pattern, given by the vector $\text{cat}(V^P)$, to the ART network (with vigilance $\omega$ and learning rate $\alpha$). This occurred once the visual panorama had been sufficiently searched, as indicated by the VSA marking map $M$ in the GCS ($M_{ij} > \zeta$ for all elements $ij$). The sensory place nodes $s$ were then determined by extracting the activity of the F2 (output) category nodes in the ART network before winner-take-all competition. Using this panoramic operation with the parameters shown in Table 6, the PLA learns panoramic patterns of landmark positions with respect to a global orientational frame of reference (e.g. "North").

Early version of the PLA also employed a hierarchical approach to represent landmark shape using the object evidence $o$ from the OVS (see Section 2.2.2). Thus, MAVIN first had to learn each of the prospective landmarks individually by traversing around them on a pre-programmed path (see Figures 30, 31, and 32). Subsequently, MAVIN learned places from panoramic local views of landmark objects witnessed from various locations spaced at roughly one foot intervals along a pre-programmed path through the laboratory (Bachelder et. al., 1993; Bachelder & Waxman, 1994; see Figure 33).

A number of general qualities have been observed while using this version in the laboratory. Due to its separate coding of shape and direction, it can recognize places invariant to both the robot's heading and the landmark imaging conditions (e.g. lighting and image scaling). Due to coarse coding mechanisms, which insure that panoramic patterns change gradually with robot movement, this recognition also exhibits insensitivity to small changes in the robot's location or in the positions and poses of landmarks. As a result, each of the learned place categories corresponds to a place region, an area from which the visual appearance, location, and poses of the landmarks remain sufficiently similar to its learned place template (see Figure 34). These mutually exclusive regions effectively parcel the explored 2D area of the environment (see Figure 35). Furthermore, the activity profile of a particular place category, which describes its sensory evidence as a function of location, falls off roughly exponentially from a peak as the robot translates away from the center of its corresponding place region and into a neighboring place region (see Figure 36).

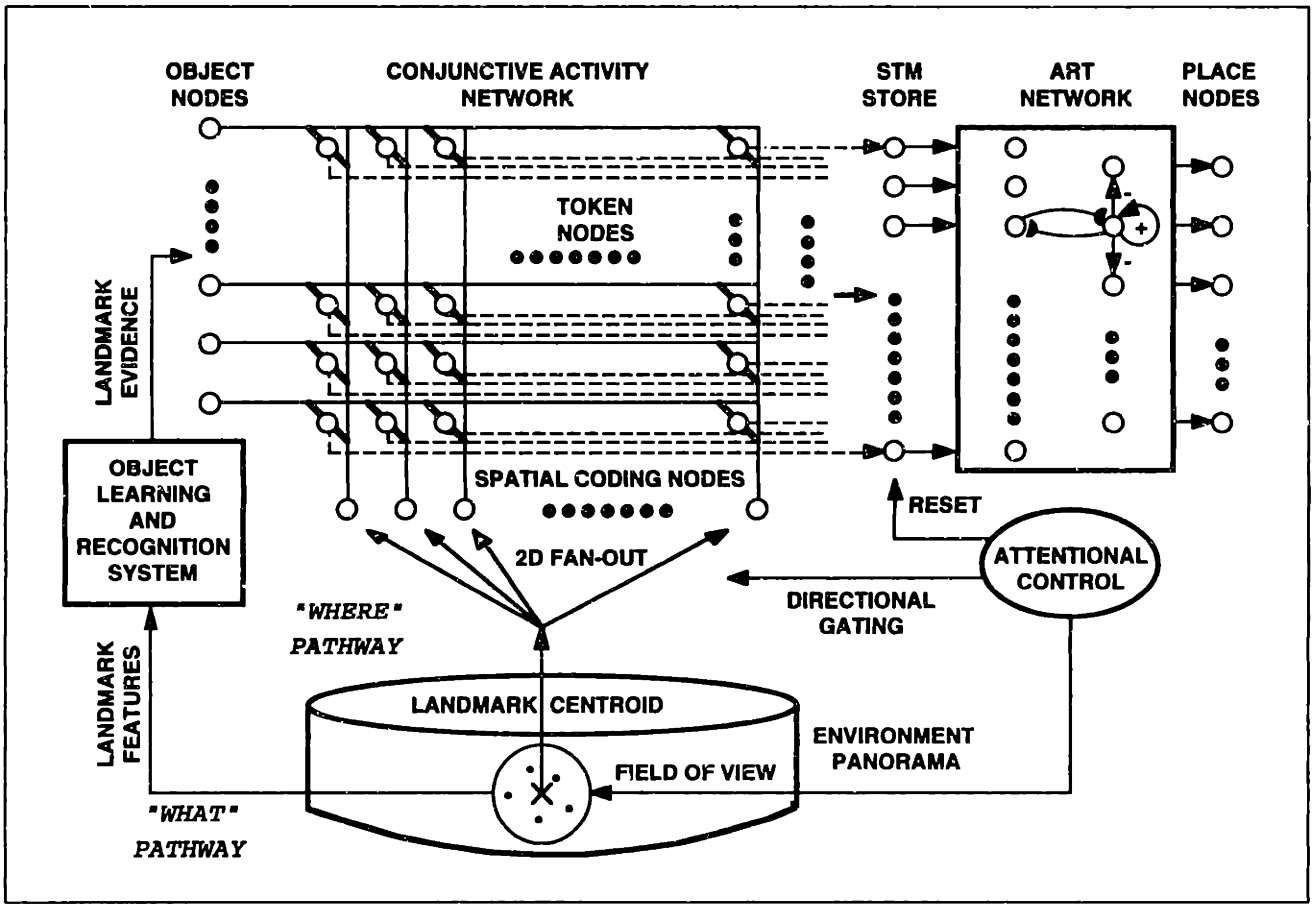


**FIGURE 29: NETWORK ARCHITECTURE FOR A SEMI-EGOCENTRIC VERSION OF THE PLA.**

*The semi-egocentric place learning architecture (Bachelder et. al., 1993; Bachelder & Waxman, 1994) parcels a 3D environment into 2D place regions over which the identities, poses (i.e., aspects), and relative spatial positions of visual landmarks are qualitatively similar.*

**TABLE 6: SEMI-EGOCENTRIC PLA PARAMETER SETTINGS.**

| **Parameter** | $n_s, n_r$ | $n_x, n_y$ | $n_a, n_t$ | $\delta, \kappa$ | $\omega, \alpha$ | $\zeta, \xi$ | $\bar{r}^I$ |
|---|---|---|---|---|---|---|---|
| **Value** | 9,9 | 11,5 | 11,5 | 0 $\text{sec}^{-1}$, 1 $\text{ft}^{-1}\text{sec}^{-1}$ | 0.9,0.1 | 0.95,0.5 | $\bar{o}$ |

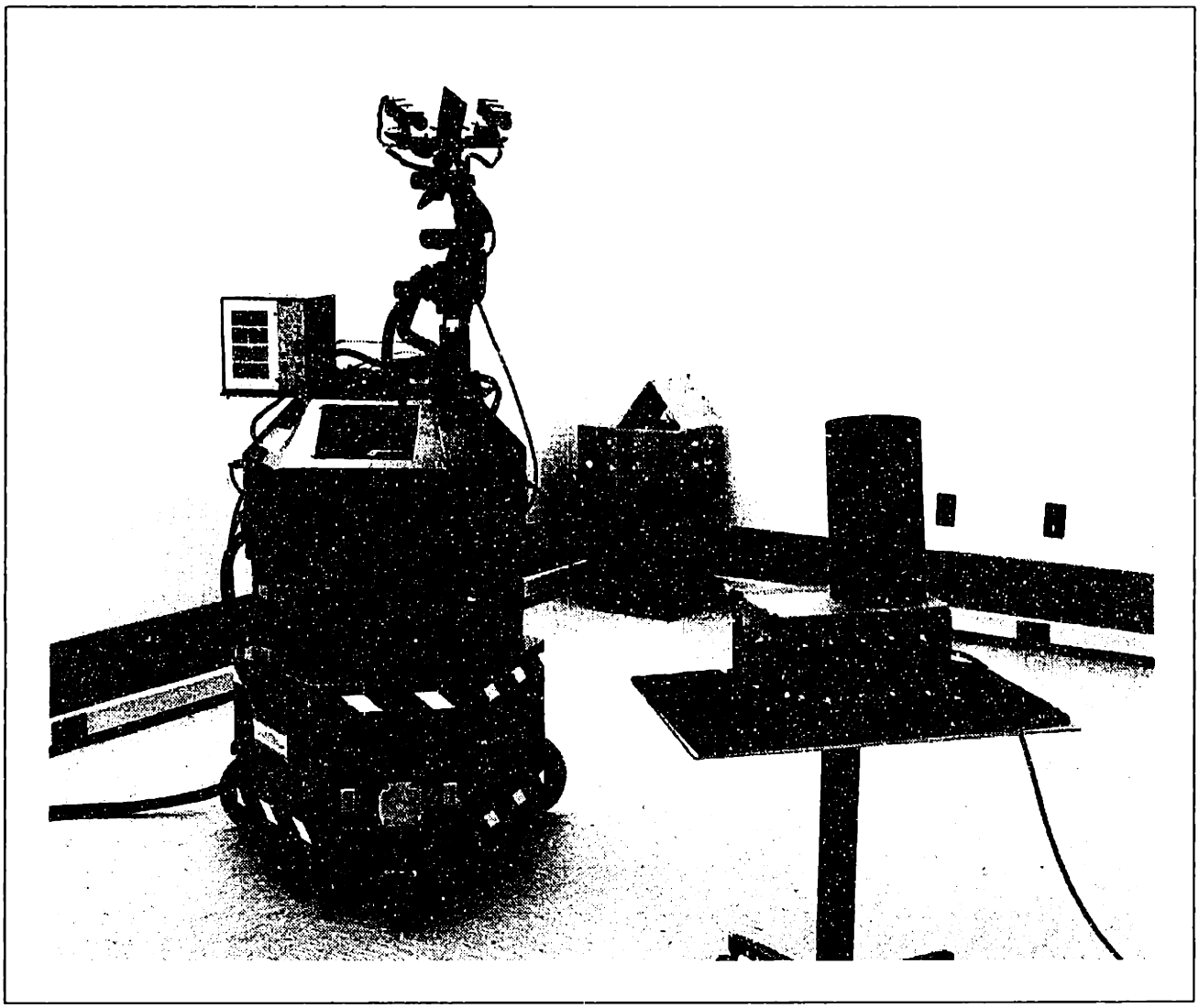

**FIGURE 30: MAVIN EXPLORING A VISUAL LANDMARK.**
*MAVIN (the Mobile Adaptive VIsual Navigator) explores a simple landmark composed of bright lights affixed to a model of a "factory" building.*

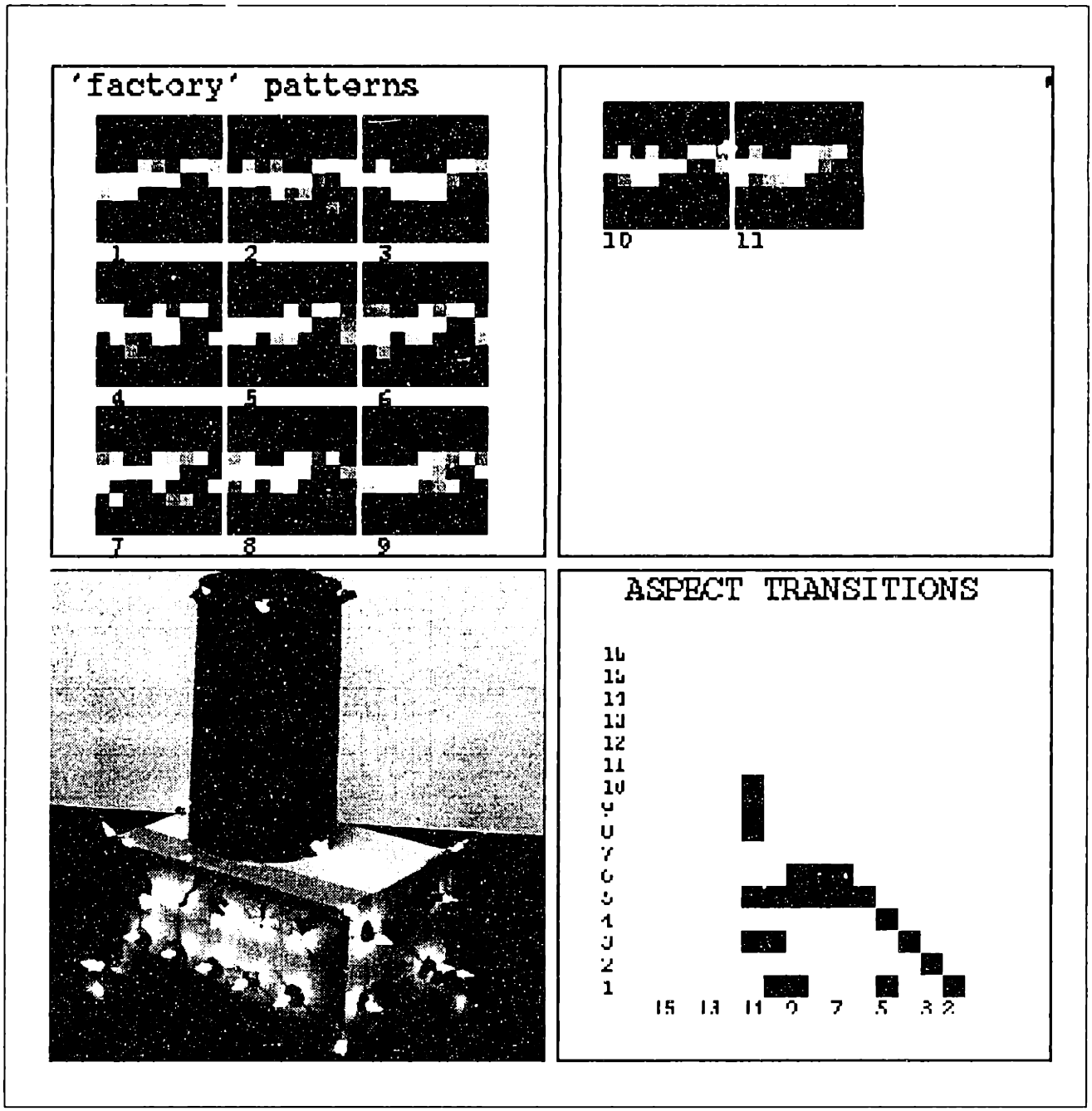


**FIGURE 31: LEARNED ASPECT TEMPLATES AND TRANSITIONS FOR FACTORY LANDMARK.**

*The top two quadrants of this computer screen printout show the 9x9 templates for the factory landmark, learned by MAVIN during the course of visual exploration. The lower right quadrant shows the learned transitions between each of the aspect categories.*

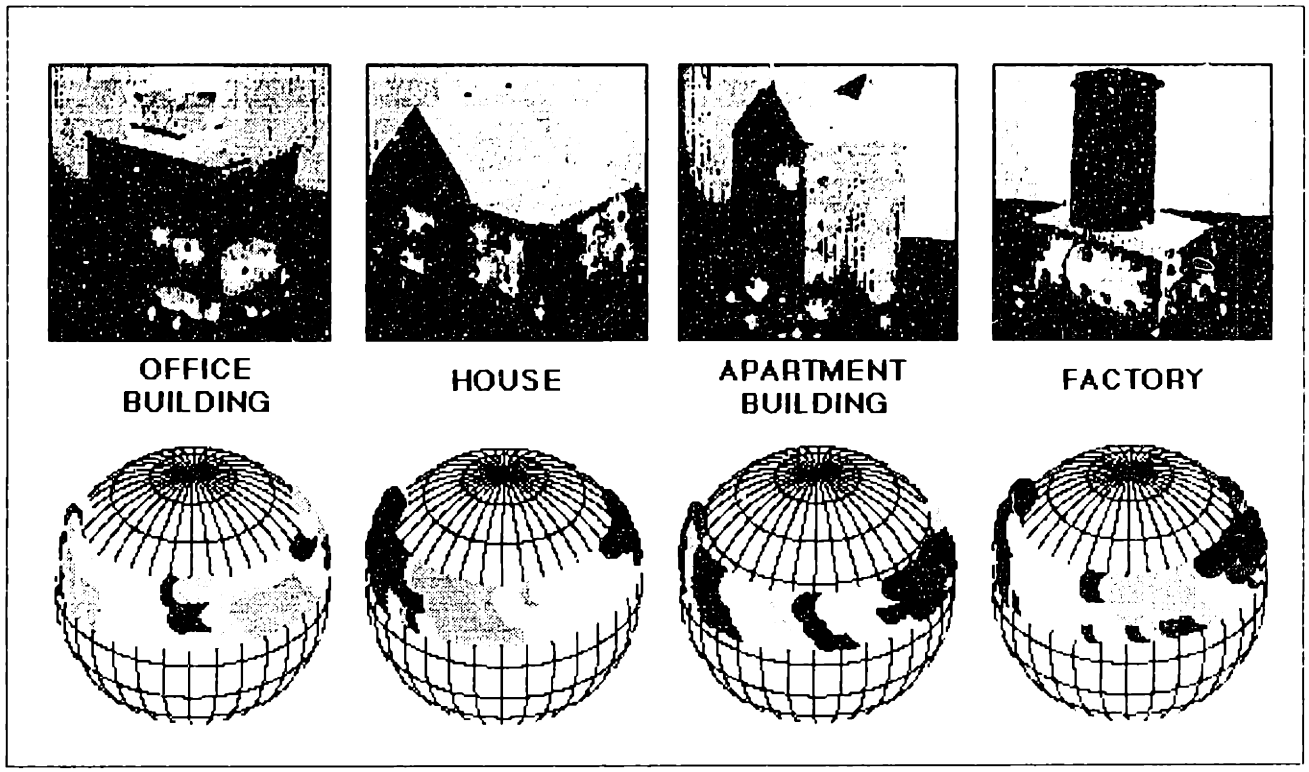


**FIGURE 32: LEARNED LANDMARK ASPECT CATEGORIES.**

*The learning of visual landmarks by the mobile robot MAVIN, using the object vision subsystem (Seibert & Waxman, 1989, 1992). Exploring each of the landmarks ("factory", "house", "apartment building", and "office building") from 216 different viewing directions generates on the order of 10 to 20 aspect categories per landmark. As a consequence, the spherical space of possible viewing directions is parcelled, for each landmark, into aspect regions, as illustrated by the aspect spheres below each of the landmarks.*

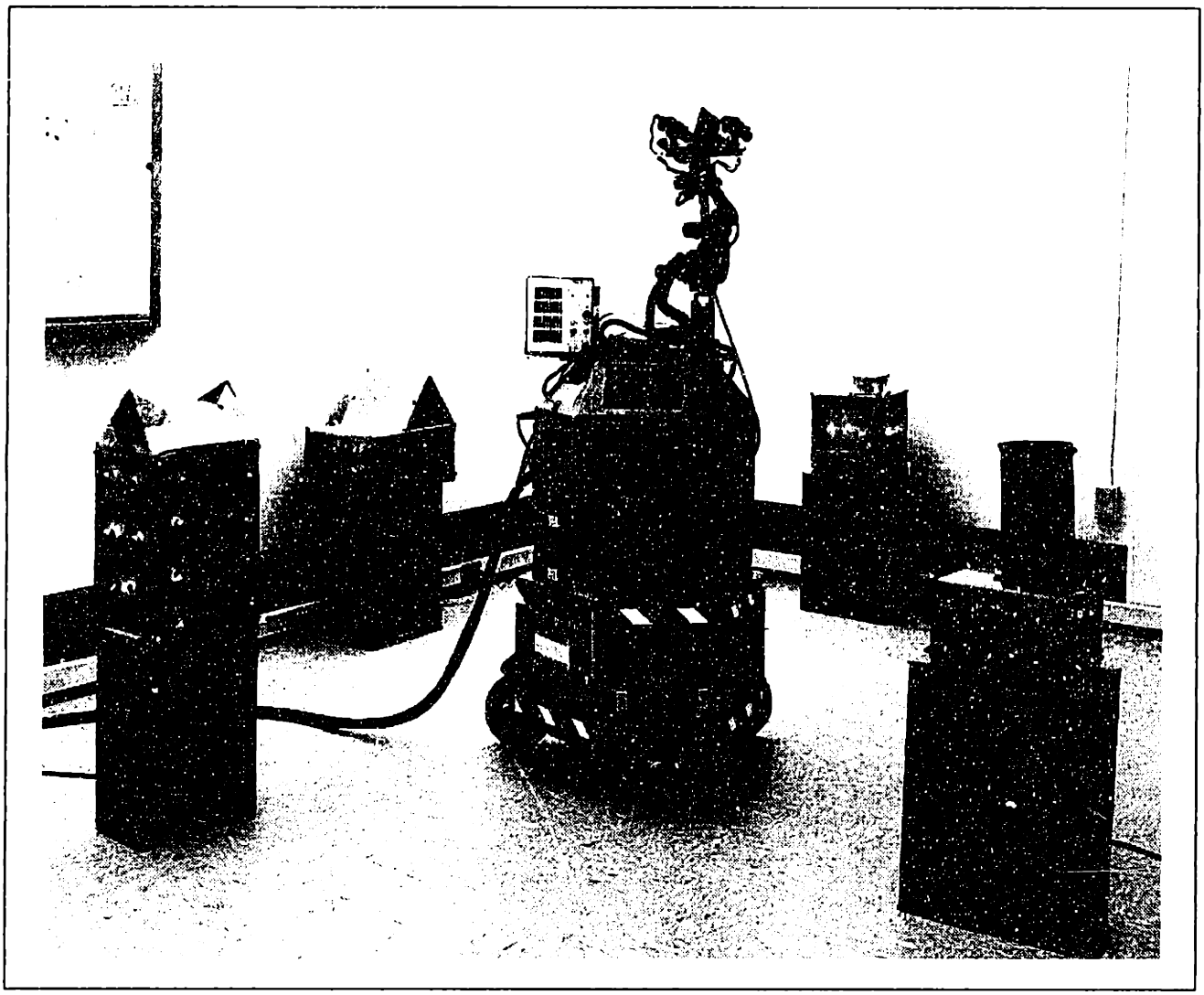

**FIGURE 33: MAVIN EXPLORING A VISUAL ENVIRONMENT.**
*MAVIN among the 4 landmark objects used to define an environment (from left to right: "apartment building", "house", "office building", and "factory").*

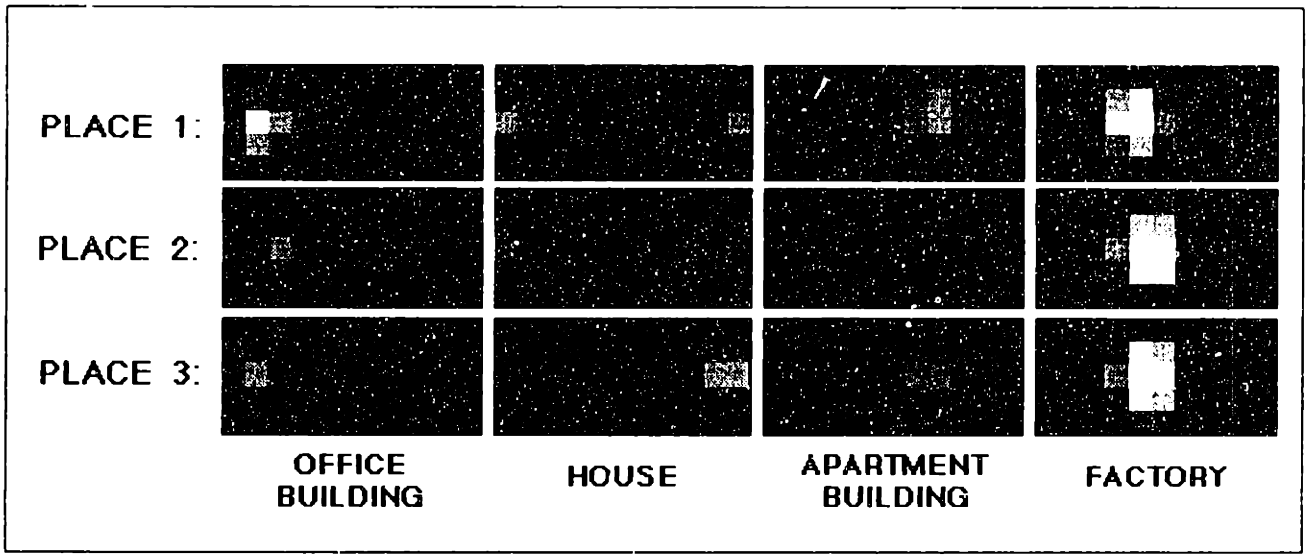


**FIGURE 34: LEARNED HEADING INVARIANT PLACE TEMPLATES.**
*The three patterns shown correspond to each of the place templates for the 3 learned heading invariant place categories. Each of the four blocks of each pattern display the pattern of evidence for the corresponding landmark (office building, house, apartment building, and factory) over the spatial panorama (azimuth and tilt).*

These activity profiles overlap, and intersect at the boundaries between the place regions (decision boundaries).

Not surprisingly, the activity profiles for the learned place nodes qualitatively resemble the spatial firing patterns observed for place cells in rat experiments (cf Figure 6). Furthermore, due to the invariant view-based coding of landmarks and their positions, in addition to the fact that place node activity varies smoothly with changes in landmark position and spatial location, place regions and profiles translate and, for the most part, scale with the environment (at least in theory). Unfortunately, unless "North" is defined with respect to the landmark constellation, place regions and activity profiles do not rotate with the environment. This property arises only as a result of a fully egocentric implementation (see Section 7.6). Note also that the scaling property breaks down in the vicinity of landmarks that reside at a much different elevation than the robot's head, simply because both the appearance and tilt direction to the landmark change with scaling. With this observation in mind, however, it would be interesting to see how the firing rates of rodent place cells change in response to scaling an environment (i.e. the distances between landmarks) in which the proximal landmarks are fixed significantly above the maze apparatus.

On computational grounds, these preliminarily results attest to several advantages of using an HRV learning approach for qualitative map-making, including real-time operation, robustness to various noise sources, and efficient storage of places and environments. In addition, the overlapping nature of the activity profiles illustrate the potential to perform place localization at multiple spatial scales (see Bachelder & Waxman, 1994, for a more thorough discussion).

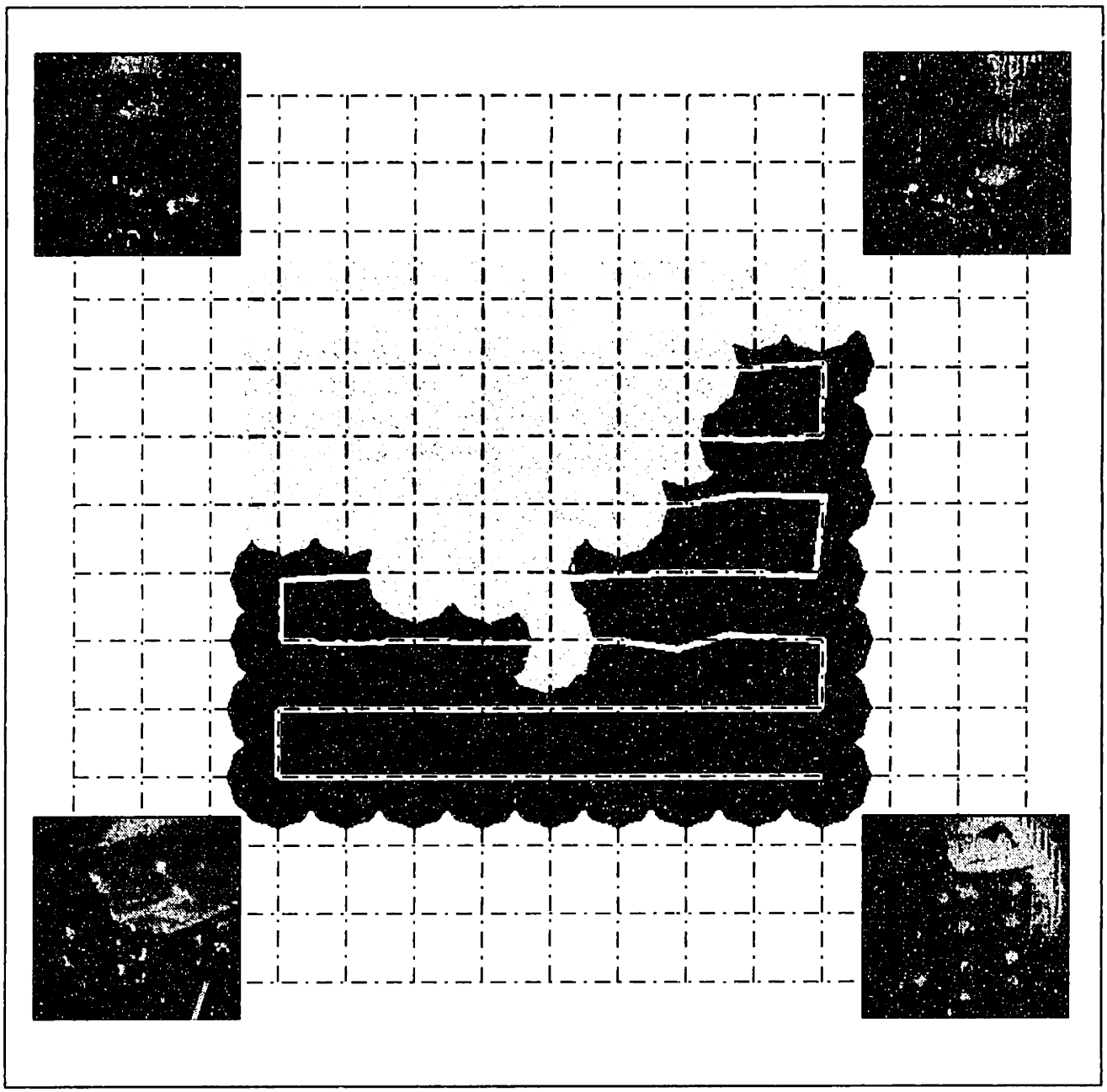

**FIGURE 35: LEARNED HEADING INVARIANT PLACE REGIONS.**

*Shaded place regions for each of the learned place categories parcel the explored area into a qualitative map. This areal view of the laboratory environment iconically shows the approximate locations of each of the landmarks, and traces the pre-programmed exploration path taken by the robot in white. (cf Figure 13)*

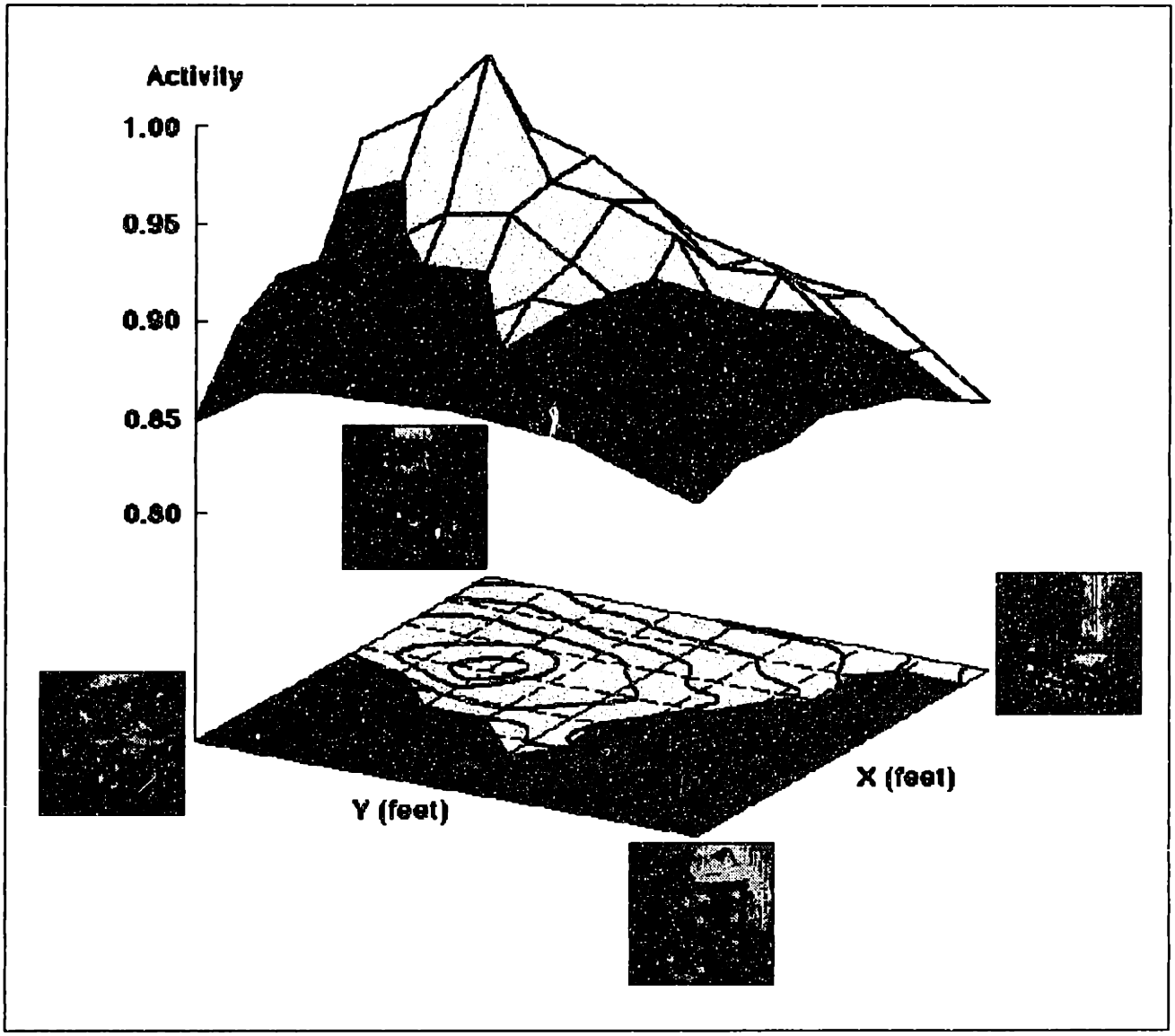


**FIGURE 36: ACTIVITY PROFILE OF A SINGLE HEADING INVARIANT PLACE NODE.**
*The activity profile for place node #3 over the explored 7' x 8' portion of the environment (iso-activity contours drawn at 0.02 intervals, between 0.81 and 1.00). The place regions for each of the 3 learned place categories are indicated on both the profile and the contour plane by different shades (place regions 1, 2, and 3 are black, dark grey, and light grey respectively). The decision boundaries between these regions indicate a spatial overlap with the activity profiles of the other two place nodes. (Note that some of the "bumps" in this and subsequent profiles may be due to inaccuracies in the dead-reckoned coordinates used to measure and plot the activity.)*

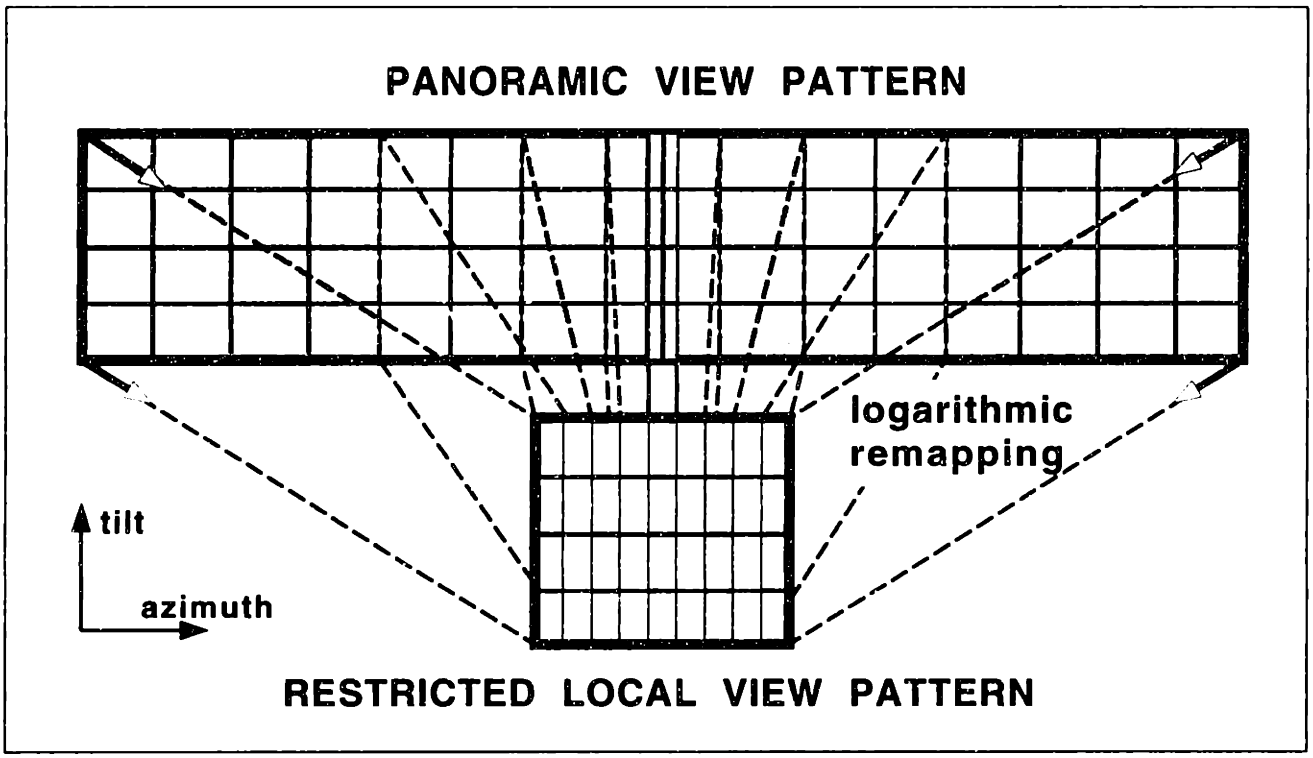


**FIGURE 37: THE CONCEPT OF LOCAL VIEW PROCESSING.**
*Restricted processing of the local view by a logarithmic transformation magnifies information in front of the robot and diminishes information in the periphery.*

## 7.5 Restricted Local View Processing Dynamics

In the sense that semi-egocentric place learning relies entirely on 2D views of landmarks and environments, it does at some level implement the local sensory view model of hippocampal place cells described by McNaughton (1989), even though the local view is panoramic. However, the use of globally oriented, panoramic representations poses two problems for place recognition. The first problem is one of efficiency. Before place recognition can occur with any degree of confidence, the robot must stop at each location and turn completely around in order to search the entire panorama for visible landmarks — a time consuming, cumbersome operation for a robot with relatively slow head, eye, and body movements. The second problem is one of robustness. The PLA must align incoming patterns with stored ART templates by using an accurate compass to indicate a global directional sense (e.g. "North" — see discussion in Section 5.2.2). While the accuracy of place recognition is largely insensitive to variations in this directional sense, the ability to localize the robot within place regions (using the activity of learned place nodes) is not.

In recent implementations, the local view sampling performs a non-uniform processing about the heading of the robot (illustrated in Figure 37). It centers (shifts) the panoramic pattern memory with respect to the robot's current heading, logarithmically transforms the shifted memory in azimuth, and applies an array of nodes with overlapping Gaussian receptive fields, effectively computing a restricted local view pattern. More specifically, it shifts and logarithmically transforms the centers $\tilde{c}^a$ of the azimuth coding nodes in azimuth according to the equation

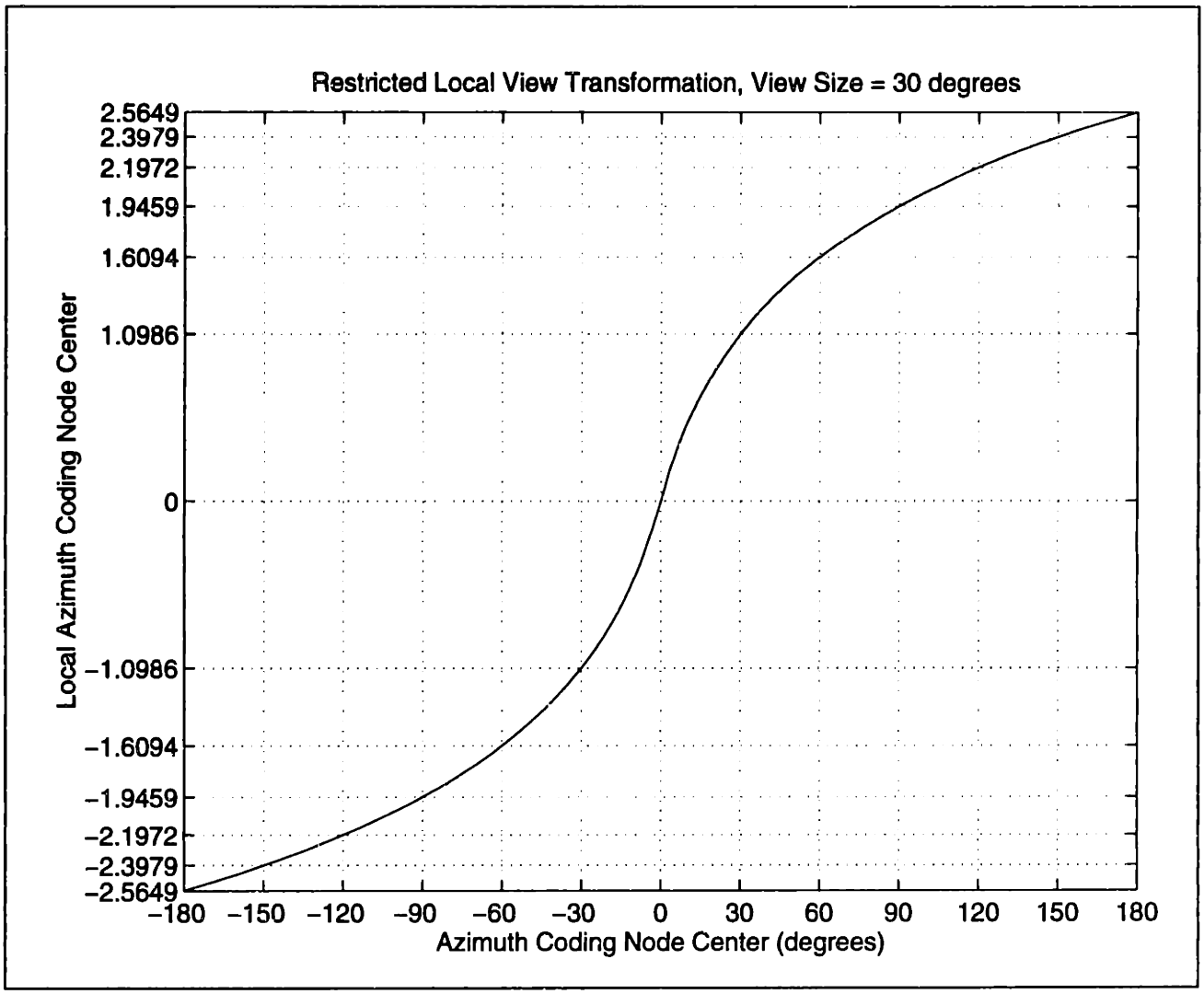


**FIGURE 38: THE LOGARITHMIC RESTRICTED LOCAL VIEW TRANSFORMATION.**
*Azimuth coding node centers are logarithmically transformed into local azimuth centers.*

$$c_i^{Ra} = \mathrm{RV}(c_i^u) \tag{27}$$

where

$$\mathrm{RV}(\varphi) = \log\left(1 + \frac{2}{\sigma_R}\left|(\varphi - B_h)_{-\pi}^{\pi}\right|\right)\mathrm{sgn}\left((\varphi - B_h)_{-\pi}^{\pi}\right) \tag{28}$$

is the restricted local view transform, and $\sigma_R$ is the effective width of the local view. This spatial remapping yields a new set of "local azimuth" centers in a periodic space with minimum and maximum values given by

$$c^R_{\max} = \mathrm{RV}(B_h + \pi) = \log\left(1 + \frac{2\pi}{\sigma_R}\right) \tag{29}$$

$$c^R_{\min} = \mathrm{RV}(B_h - \pi) = -\log\left(1 + \frac{2\pi}{\sigma_R}\right) = -c^R_{\max} \tag{30}$$

(see Figure 38). A set of local azimuth receptive fields are then applied to each of the new centers, yielding a vector

$$\bar{r}^{Ra}_i = \mathrm{RF}_p(c^{Ra}_i; n_{Ra}, c^R_{\min}, c^R_{\max}) \tag{31}$$

for each azimuth coding node $i$, where $n_{Ra}$ is the number of local azimuth receptive fields. Finally, the activities for each of the nodes in the restricted local view pattern $V^R$ with the same tilt index $j$ and identity index $k$ are determined by weighting each of the local receptive fields vectors $l$ by the activity of its corresponding panoramic token memory node $V^P_{ljk}$, i.e.

$$V^R_{ijk} = \left[\sum_{l=1}^{n_a} V^P_{ljk} \cdot \bar{r}^{Ra}_l\right]_i. \tag{32}$$

(An alternate way of determining the local view pattern would be to apply a set of receptive fields to the *activity* as well as the center of each of the azimuth coding nodes, take their conjunction, and then determine the local view nodes by taking the max over all azimuthal indices. This method would keep the activity coding separate from the spatial location coding, but would also produce a local view pattern of size $n^{\Delta} \times n^{Ra} \times n^t \times n^l$ rather than $n^{Ra} \times n^t \times n^l$, where $n^{\Delta}$ is the number of "activity coding nodes.")

This overall transformation resembles the log-polar transformation in the OVS (see Section 2.2.2). Though it does in fact represent information from the entire panorama, it has the effect of magnifying information directly in front of the robot, and diminishing information in the periphery (much like a fovea magnifies, on the cortex, the information along the line of sight). The scale of the logarithmic transformation, defined by $\sigma_R$, governs the extent of this magnification, and hence the "size" of the local view. Thus, the robot learns spatially "restricted" local views with respect to an egocentric frame of reference (i.e. the heading direction $B_h$), rather than learning globally oriented panoramic local views.

Now, the local view pattern, given by $\mathrm{cat}(V^R)$, is presented as input to the ART network once the *restricted* local view has been sufficiently searched. As before, the activities of the sensory place nodes $s$ are determined by extracting the activity of the F2 (output) category nodes in the ART network before winner-take-all competition ensues. To determine when the search is sufficient, the local view transformation defined above is also applied to the marking map $M$ in order to produce a local view marking map $M^R$. That is,

$$M^R_{ij} = \left[\sum_{l=1}^{n_x} M_{lj} \cdot \mathrm{RF}_p(\mathrm{RV}(c^x_l); n_{Rx}, c^R_{\min}, c^R_{\max})\right]_i \tag{33}$$

where $n_{Rx}$ is the number of local marking azimuth nodes. The search is considered sufficient when $M^R_{ij} > \zeta$ for all $ij$.

TABLE 7: EGOCENTRIC PLA PARAMETER SETTINGS.

| Parameter | $n_s, n_r$ | $n_x, n_y$ | $n_a, n_t$ | $\delta, \kappa$ | $\omega, \alpha$ | $\zeta, \xi$ | $\sigma_R$ | $n_{Ra}, n_{Rx}$ | $\bar{r}^I$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 9,9 | 18,6 | 18,6 | 0 $sec^{-1}$, 1 $ft^{-1}sec^{-1}$ | 0.9,0.1 | 0.95,0.5 | $\pi/6$ | 9,6 | cat($F$) |

# 7.6 Results for a Fully Egocentric Version

Using this restricted view version of the PLA with the parameters shown in Table 7, MAVIN learned places as before, by traversing a pre-programmed exploration path in a (slightly larger) 7 x 9 ft. area of the laboratory, and stopping at one-foot intervals to build up a panoramic token pattern (see Bachelder et.al., 1994; Bachelder & Waxman, 1995). At each stopping point, however, MAVIN also rotated 360° at 10° intervals in order to capture restricted local views. (In practice, MAVIN would learn restricted local views as they naturally arose during the course of exploration along an arbitrarily random path through the environment.) As before, this version of the PLA has learned places invariant to imaging conditions, and the recognition of these places is insensitive to small movements of the robot and landmarks. Now, however, places have become broadly tuned (as opposed to invariant) to the robot's heading. Thus, the learned heading tuned place categories effectively parcel the space of explored environment positions (both headings and locations) into place regions (see Figures 39 and 40). Their local view patterns exhibit insensitivity to turning motions of the robot, primarily due to the overlapping receptive fields, but also due to the fuzziness of the view boundaries (see Figure 41). Their activity profiles, now a function of heading as well as location, fall off roughly exponentially as the robot translates or rotates, and overlap in heading as well as location (see Figures 42, 43, 44, and 45). In short, the non-uniform restricted processing insures that view patterns change gradually as the robot turns (landmarks fade as they move into the periphery), which would not occur if a sharp window were used (the landmarks would abruptly disappear).

Restricted processing results in an order of magnitude more place categories. However, these categories have much smaller templates than the previous heading invariant categories. Restricted processing also allows the robot to keep moving through the environment, rather than having to step and look around, because most of the relevant information in the local view lies directly in front of the robot. Therefore, restricted processing actually accelerates the search for landmarks by allowing the robot to neglect peripheral directions. Note that the modified PLA still requires a directional sense to code the azimuthal direction of landmarks (see 6.2.1). However, compass readings must remain consistent only during the visual search of a single restricted local view, so that the relative spatial relations between landmarks remain accurate during turning movements.

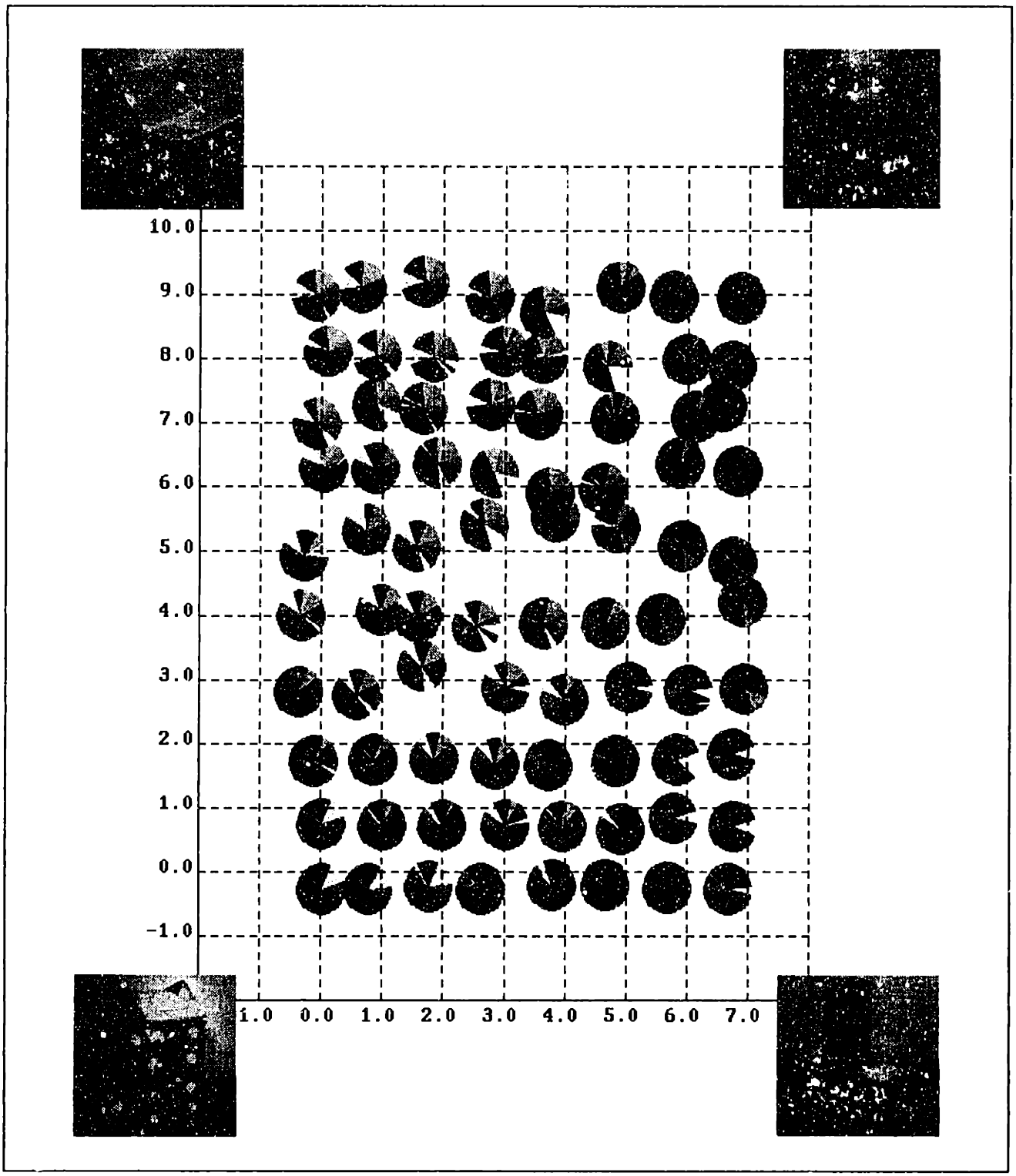


**FIGURE 39: LEARNED HEADING TUNED PLACE REGIONS.**

*A parcelling of a 7'x9' area of the lab (by multi-shaded discs at sampled locations), as well as the range of possible heading directions for each of the sampled locations (by oriented wedges within the discs), into 49 place regions. This parcelling illustrates both the location and heading specificity of place categories.*

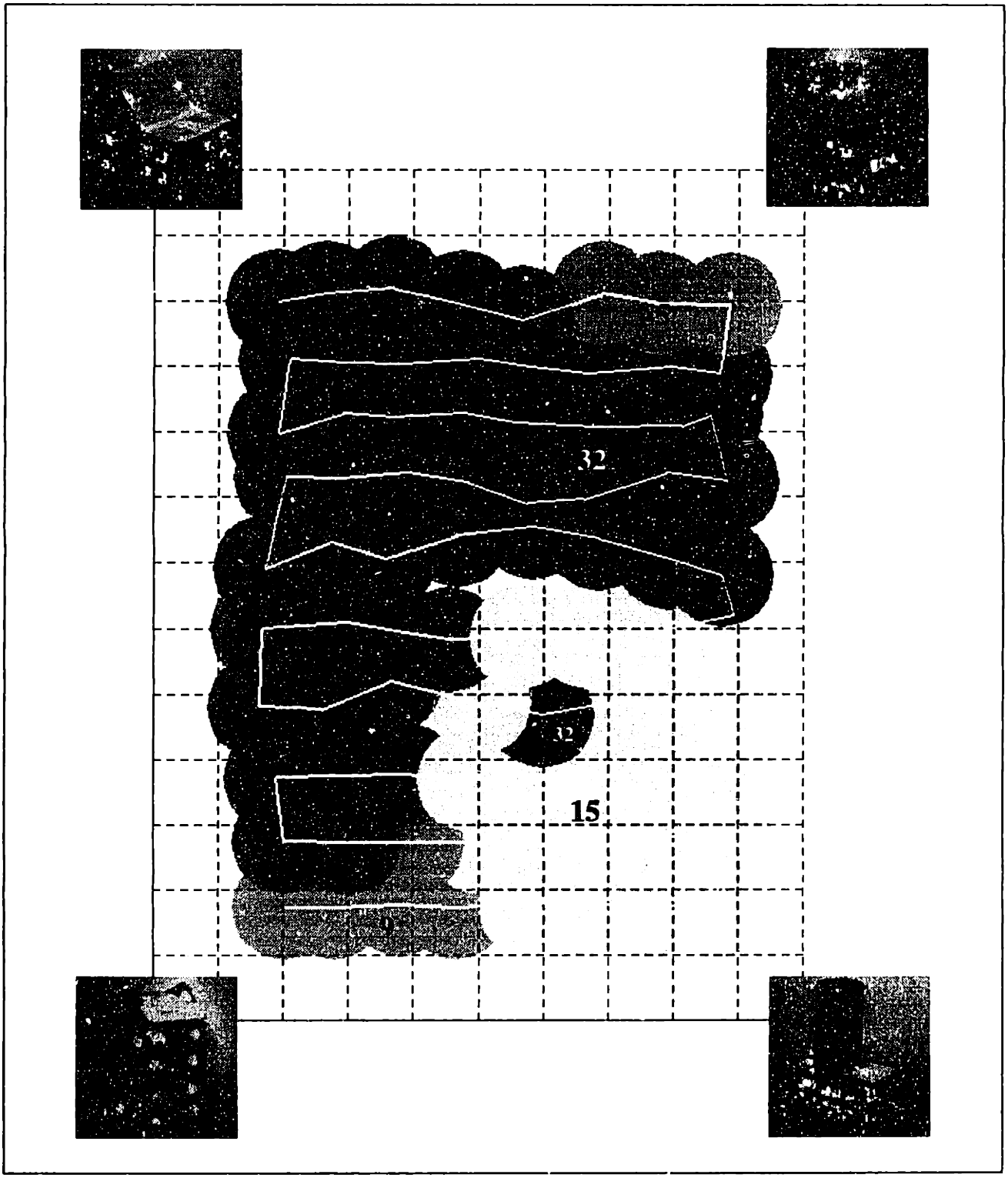


**FIGURE 40: LEARNED HEADING TUNED PLACE REGIONS AS A FUNCTION OF LOCATION.**
*Place regions for a 240° heading (the heading specificity of place node 32) across the explored area, with the exploration path indicated in white (cf Figure 14).*

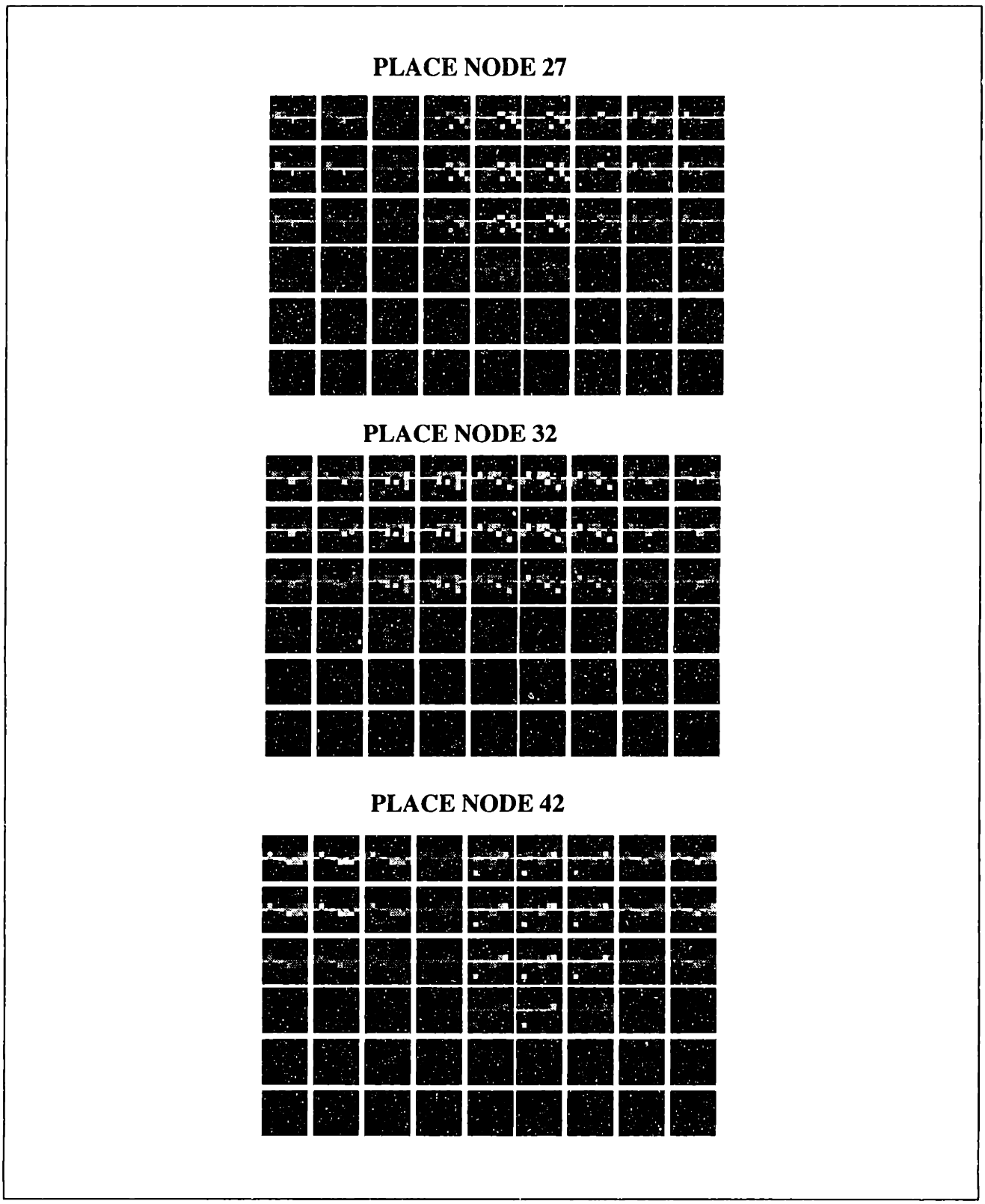


**FIGURE 41: LEARNED HEADING TUNED PLACE TEMPLATES.**
*Heading tuned place templates corresponding to 3 of the 50 learned place nodes. The azimuth and tilt dimensions of each pattern are indexed by the large blocks, while the shape dimension is indexed by the spatial pattern within each block.*

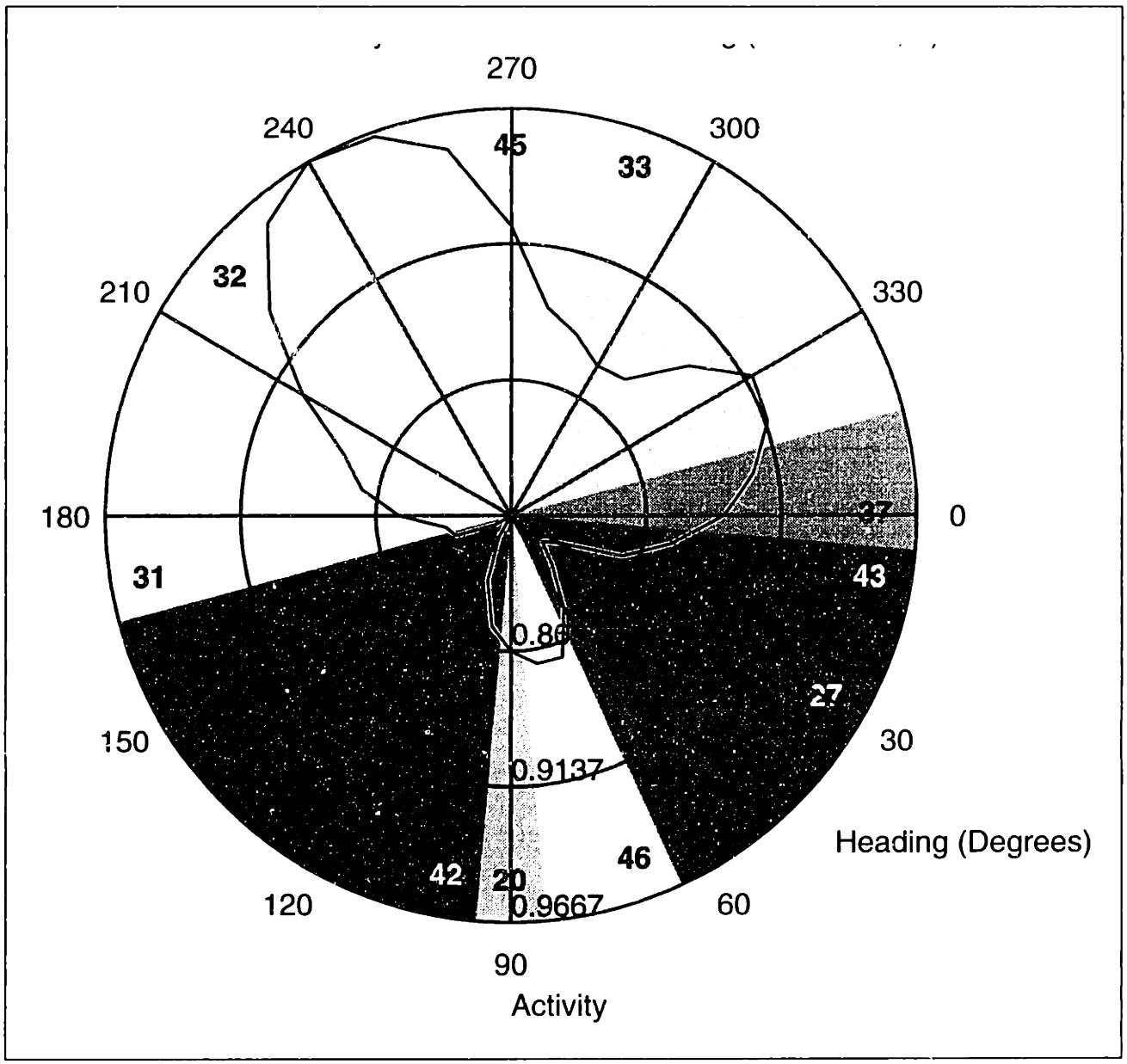


**FIGURE 42: ACTIVITY OF A SINGLE HEADING TUNED PLACE NODE AS A FUNCTION OF HEADING.**

*The polar activity profile for place node 32, superimposed upon labelled decision regions, as a function of heading for location (5',8').*

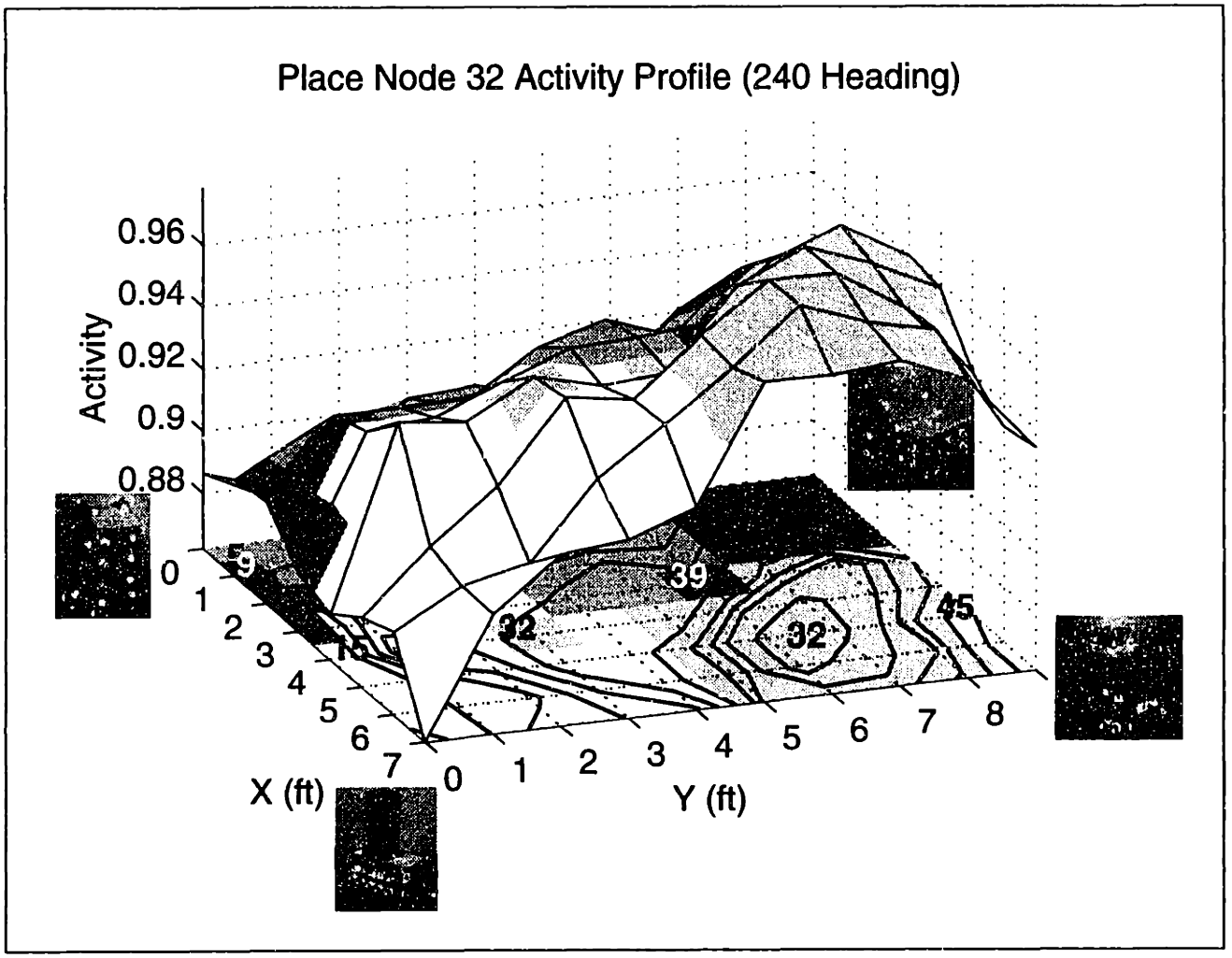


**FIGURE 43: ACTIVITY OF A SINGLE HEADING TUNED PLACE NODE AS A FUNCTION OF LOCATION.**

*The activity profile of place node 32, along with labelled decision regions (shown on the floor and superimposed on the profile), as a function of location for a 240° heading (cf. Figure 36).*

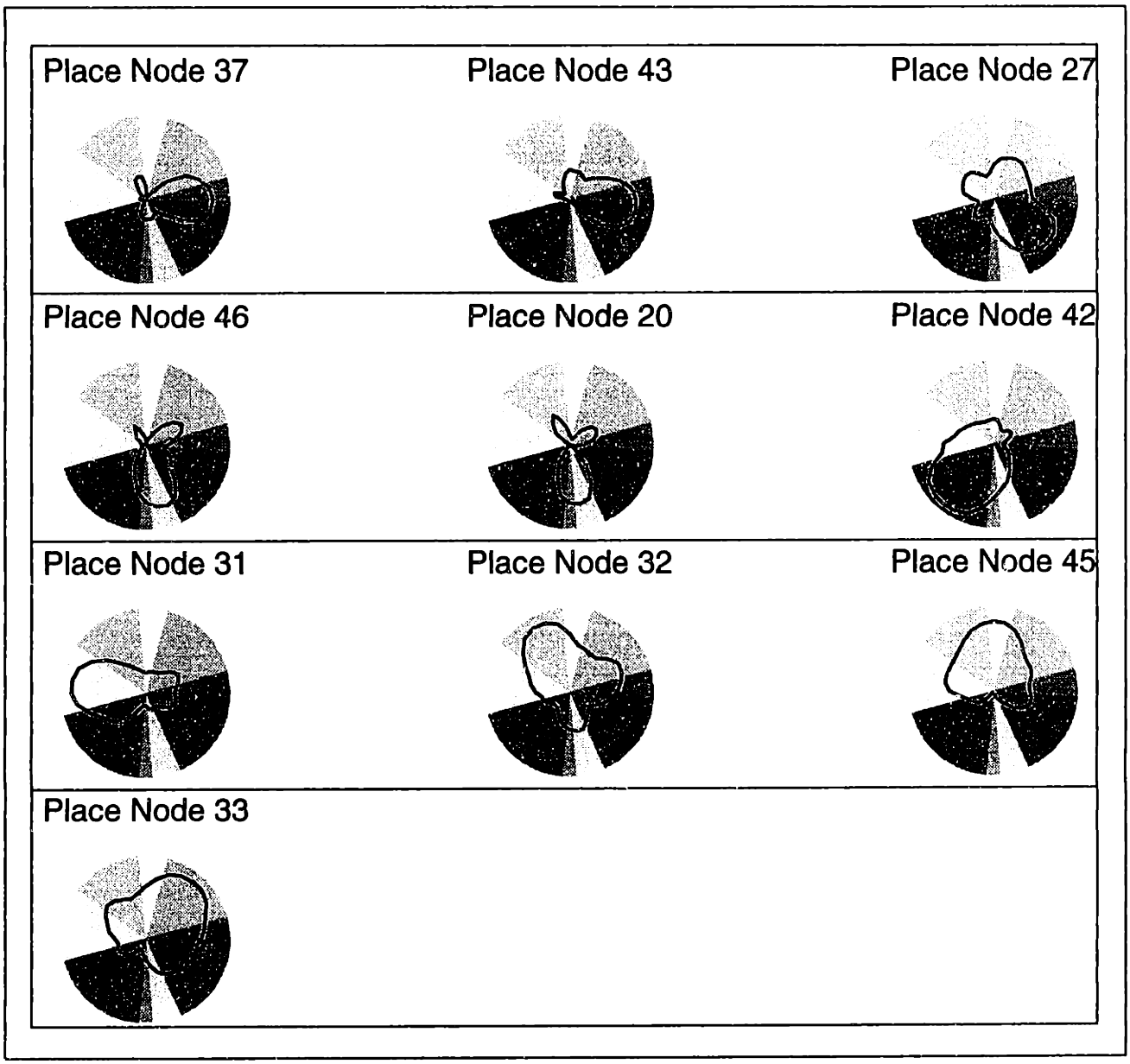


**FIGURE 44: HEADING TUNED PLACE NODE ACTIVITIES AS A FUNCTION OF HEADING.**
*The activity profile for each of the most active place nodes as function of heading for location (5',8'). Note that each node is broadly tuned to a particular heading. All activities are plotted on the same scale. The heading direction scale for each disk is the same as that of Figure 42.*

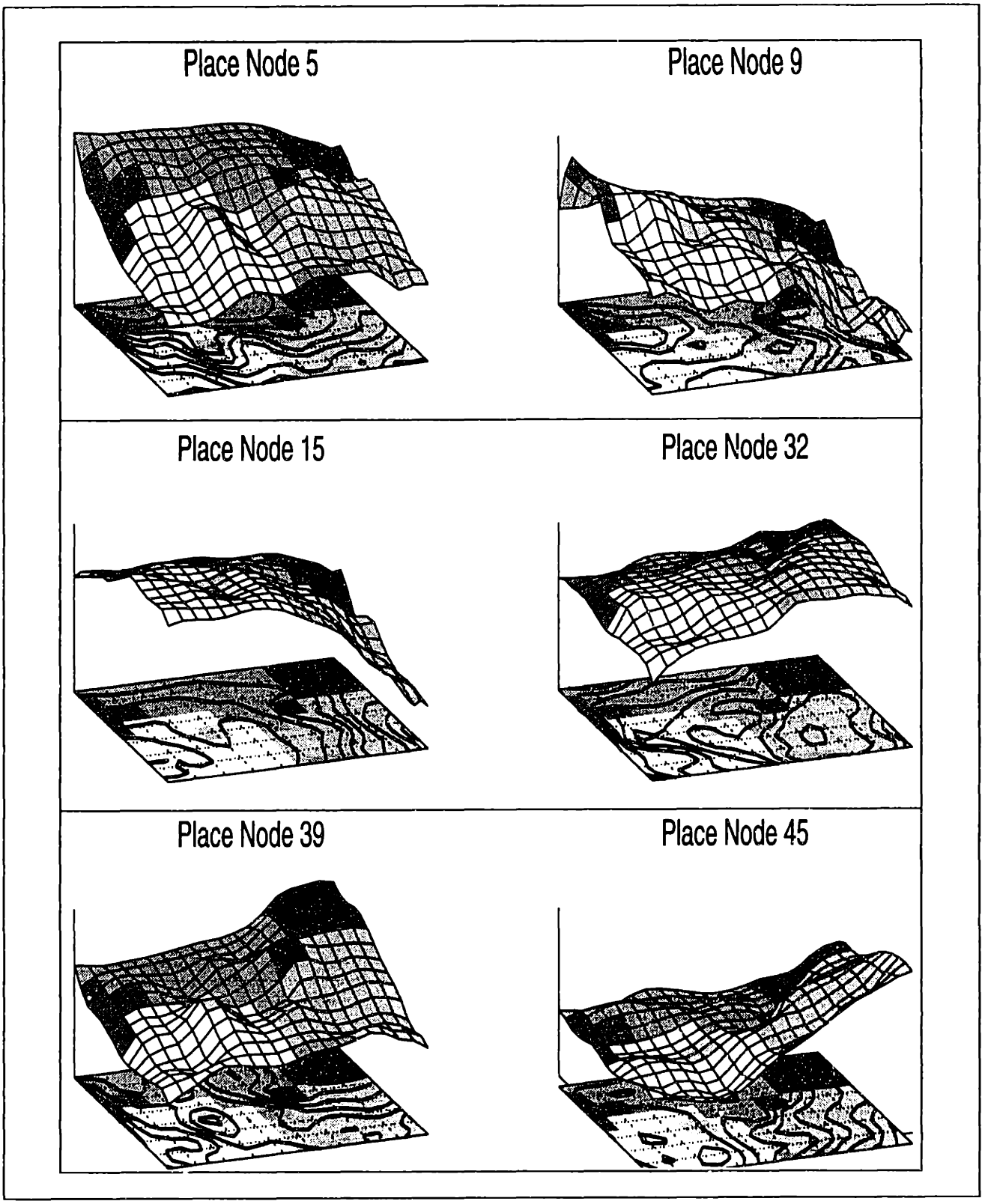


**FIGURE 45: HEADING TUNED PLACE NODE ACTIVITIES AS A FUNCTION OF LOCATION.**
*The activity profiles for each of the most active place nodes as a function of location for a 240° heading. Note that each node is broadly tuned to a particular location. All activities are plotted on the same scale. The location scale for each plot is the same as that of Figure 43.*

## 7.7 Learning Places at Multiple Spatial Scales

As previously mentioned, the vigilance parameter in the ART network regulates the degree to which the environment is parcelled into place categories. The larger the vigilance, the more finely parcelled the environment becomes. This feature suggests one possible way in which environments might be learned at multiple spatial scales, which in turn allows the robot to exhibit more precise control over its location. In effect, the vigilance setting directly trades off generalization for discrimination. Note, however, that vigilance does not affect the typical shape and size of each of the activity profiles, only the degree of overlap. This property is reminiscent of the large degree of overlap between an apparently large number of place cells in the rat hippocampus (see Section 2.2.1).

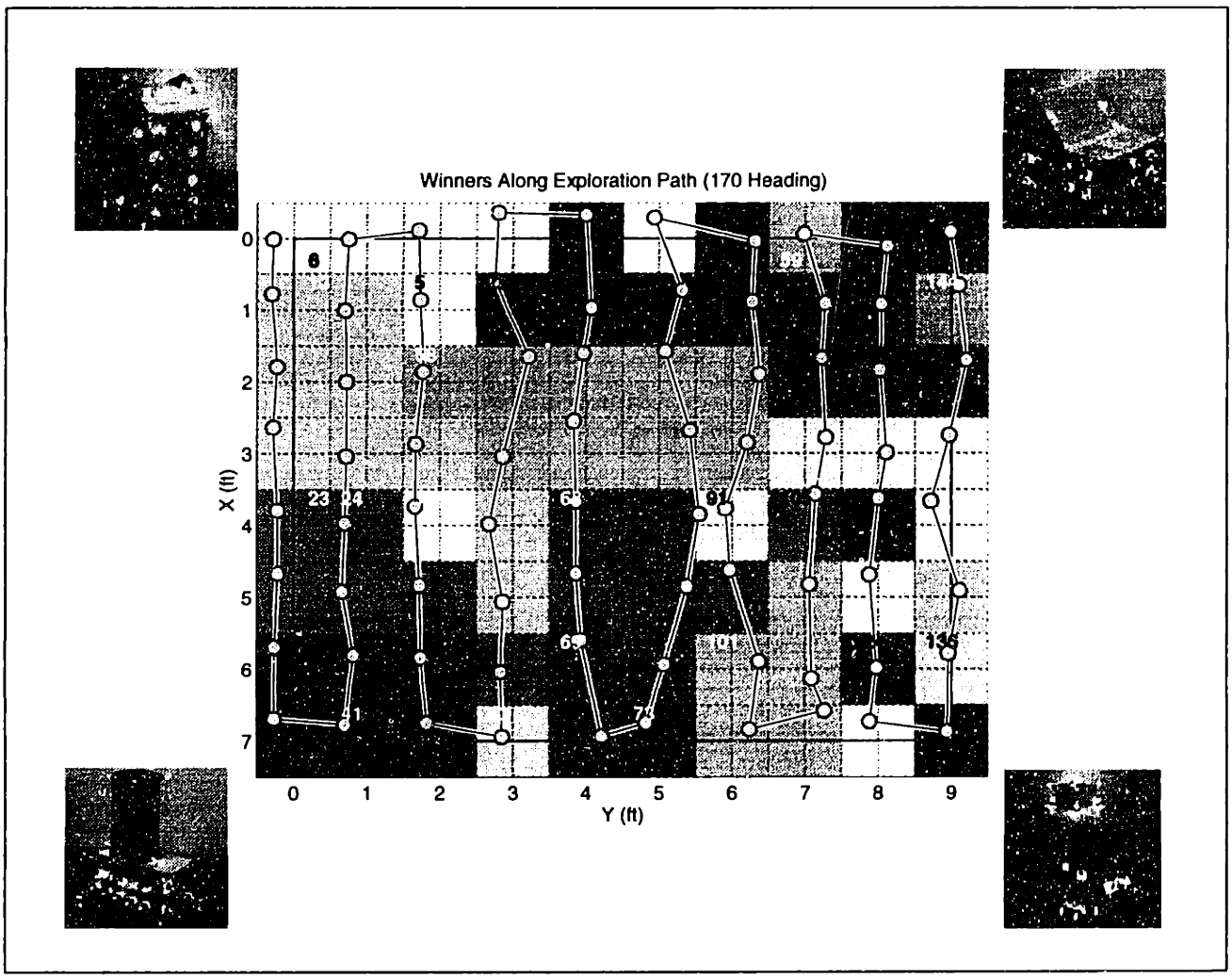


**FIGURE 46: LEARNED HEADING TUNED PLACE REGIONS AS A FUNCTION OF LOCATION (SMALL SCALE).**

*Place regions for a 170° heading (cf Figure 40). Also shown is the exploration path taken by the robot and the stopping points along this path (marked by circles).*

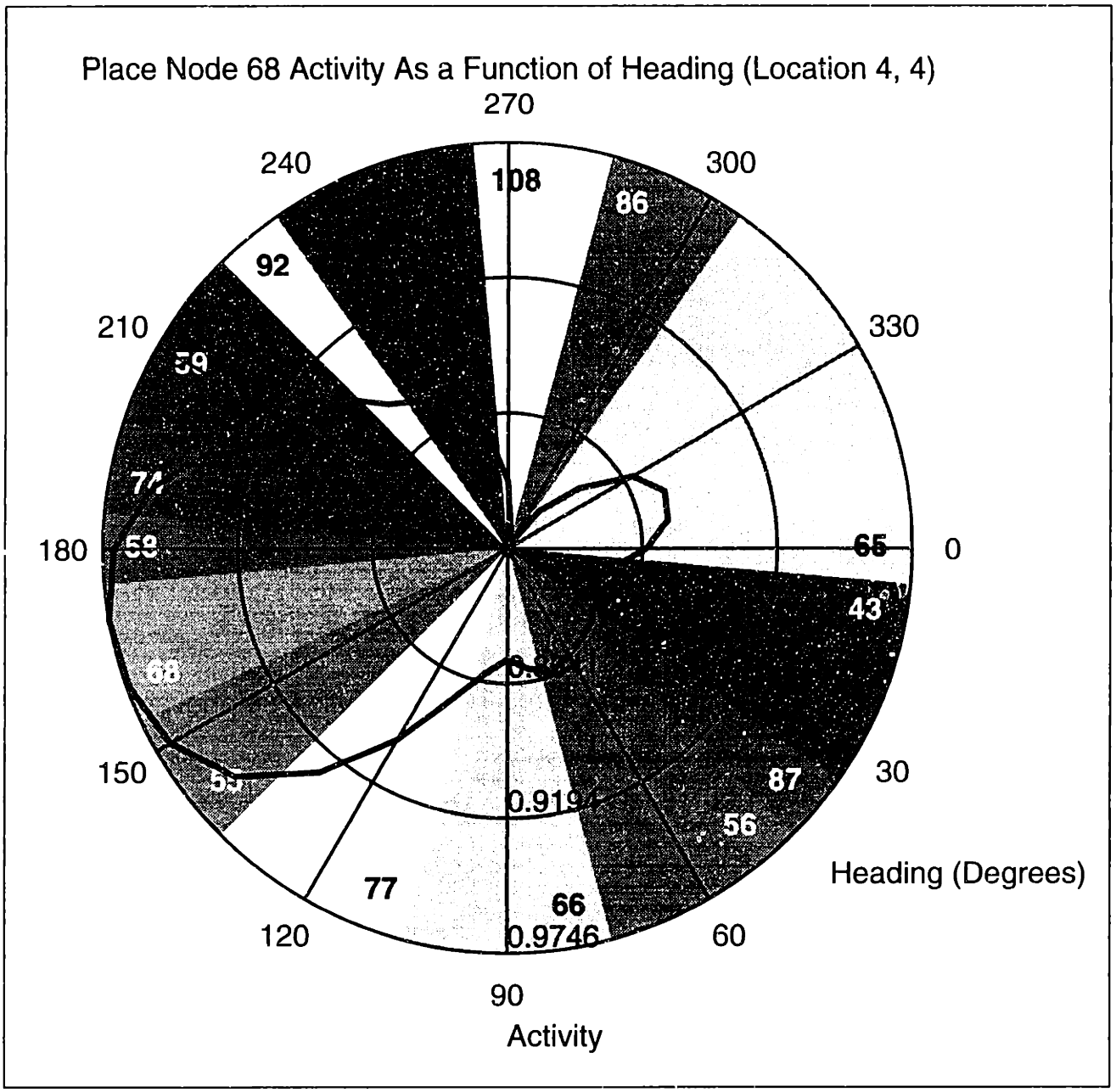


**FIGURE 47: ACTIVITY OF A SINGLE HEADING TUNED PLACE NODE AS A FUNCTION OF HEADING (SMALL SCALE).**

*The polar activity profile for small-scale place node 68, superimposed upon labelled decision regions, as a function of heading for location (4',4') (cf. Figure 42).*

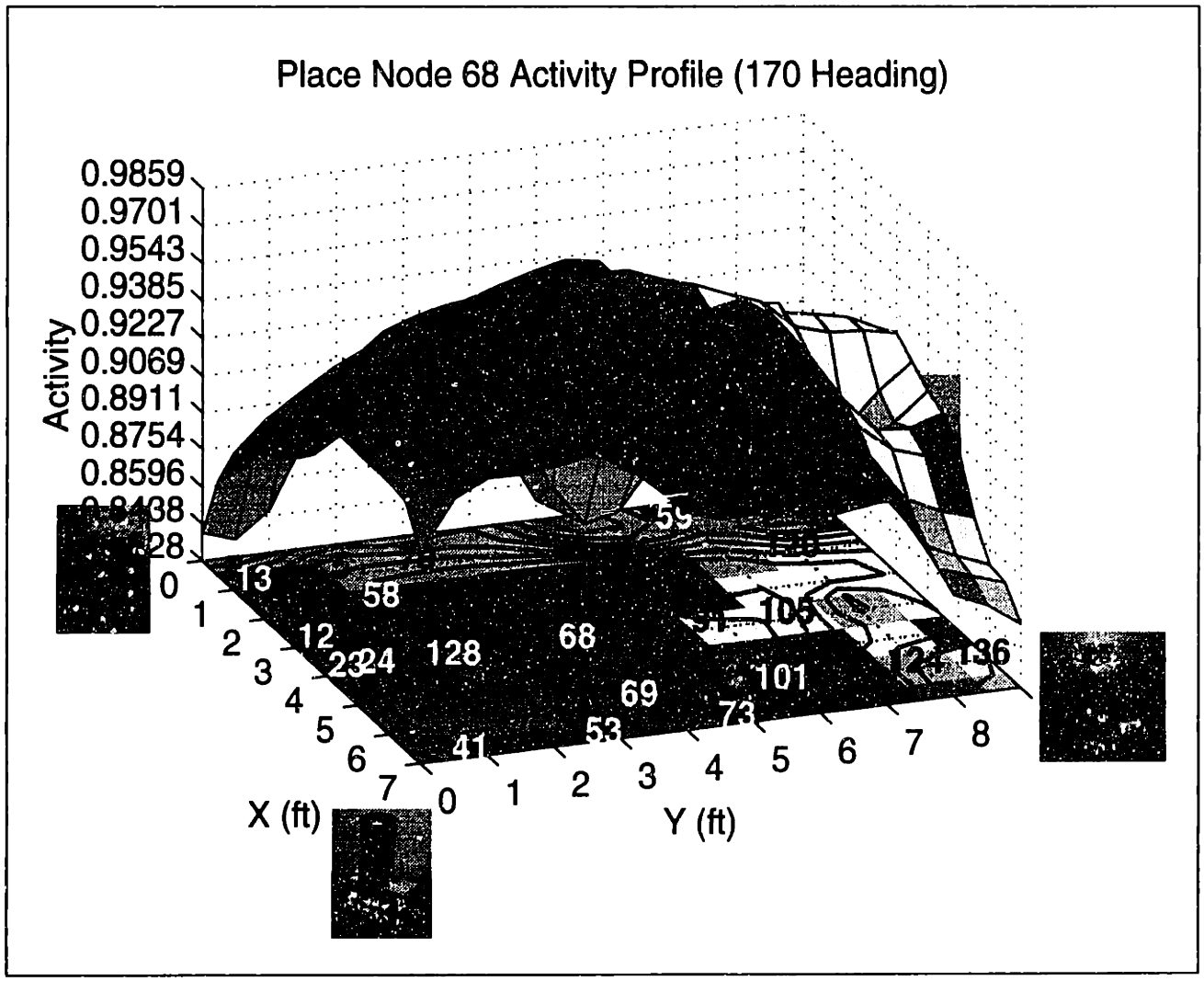


**FIGURE 48: ACTIVITY OF A SINGLE HEADING TUNED PLACE NODE AS A FUNCTION OF LOCATION (SMALL SCALE).**

*The activity profile of small-scale place node 68, along with labelled decision regions (shown on the floor and superimposed on the profile), as a function of location for a 170° heading (cf. Figure 43).*

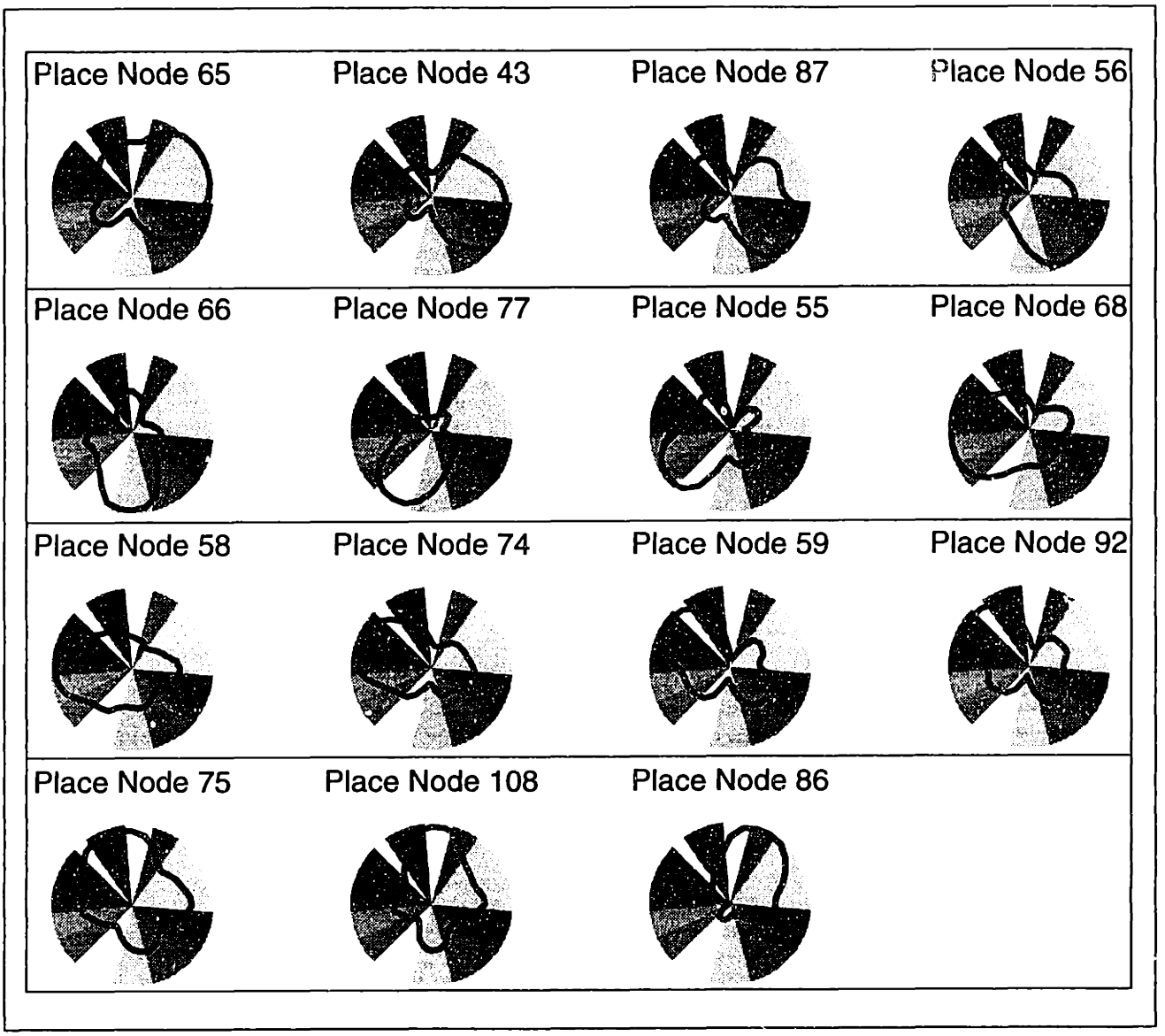


**FIGURE 49: HEADING TUNED PLACE NODE ACTIVITY AS A FUNCTION OF HEADING (SMALL SCALE).**

*The activity profile for each of the most active small-scale place nodes as function of heading for location (4',4') (cf. Figure 44).*

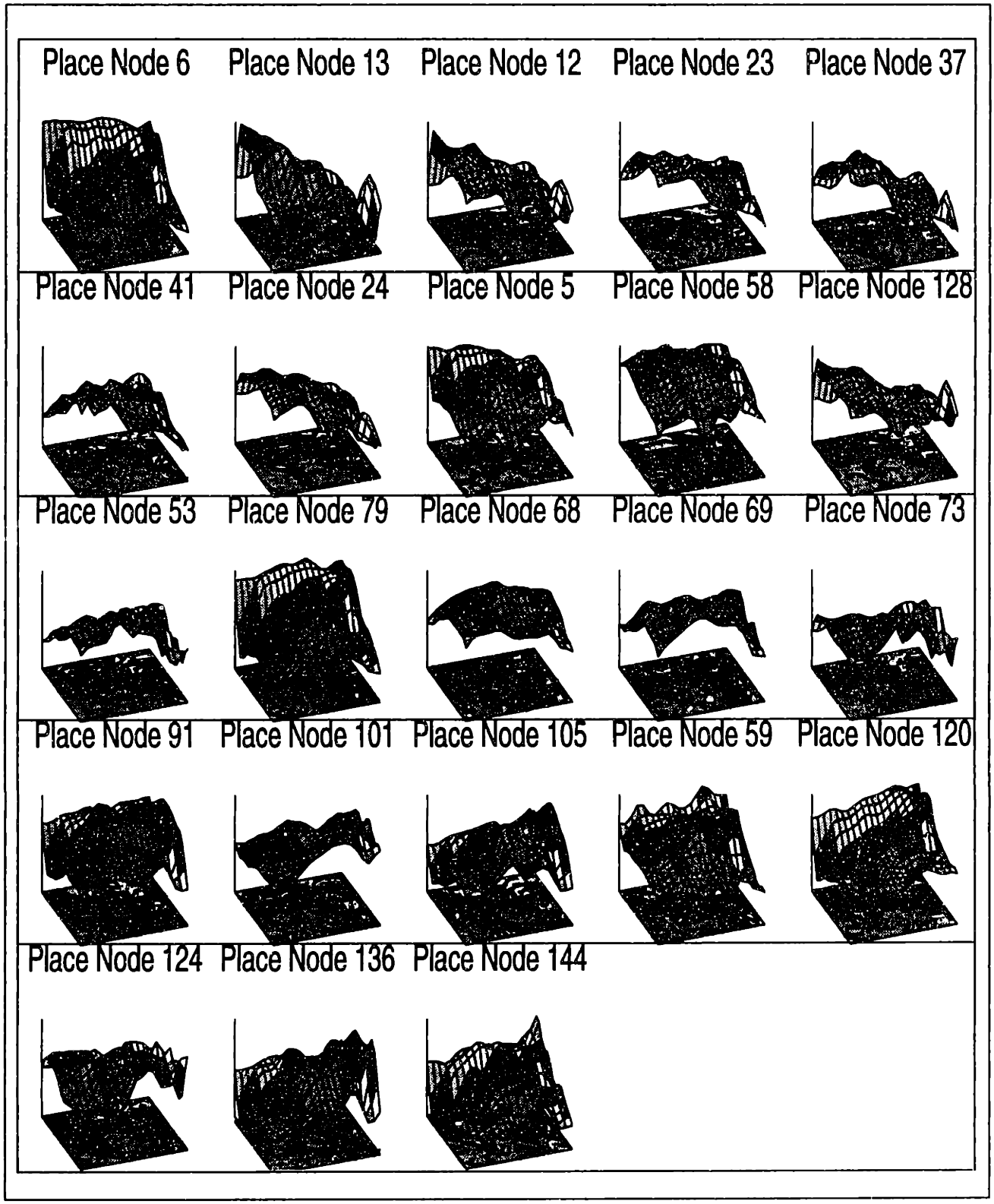


**FIGURE 50: HEADING TUNED PLACE NODE ACTIVITY AS A FUNCTION OF LOCATION (SMALL SCALE).**

*The activity profiles for each of the most active small-scale place nodes as a function of location for a 170° heading (cf. Figure 45).*

To illustrate this mechanism, the PLA was again retrained within the same laboratory environment, with a vigilance of 0.95 instead of 0.9. This time, the PLA learned 145 place categories instead of 49. Figure 46, which shows the regions as a function of location for a 170° heading, illustrates the finer parcelling that results (the figure shows 19 regions as compared to the 6 shown in Figure 40). This effect is also illustrated for the heading dimension by Figure 47, which plots the activity of a single place node as a function of heading for a single location (the figure shows a parcelling into 14 wedges as compared to the 10 shown in Figure 42). Note that the degree of tuning for the selected place node does not qualitatively differ from that of the node selected in Figure 42, just the degree of overlap with nearby place regions. The same can be said with respect to location, as witnessed by the profile in Figure 48. As before, one can get a good feel for how the environment is coded for this smaller spatial scale by examining Figures 49 and 50, which show the activity profiles for the most active place nodes as a function of heading for a single location, and as a function of location for a single heading, respectively.

This learning at multiple scales does not, of course, say anything about how the PLA might integrate several different spatial scales. Future work with regard to this problem is discussed in the conclusion (Section 11.4.5).

## 7.8 Summary

This chapter has conceptually and mathematically described the place learning architecture (PLA). The PLA learns view-based representations of places by combining the visual appearance (the "What") and visual directions (the "Where") of landmarks. This combination resembles the convergence of spatial and object vision streams in the hippocampus of primates, can be thought of as a fusion of landmark- and image- based methods for defining distinctive places, and effectively implements McNaughton's (1989) local view model of place learning. It also resembles the view-based learning of aspect categories in the OVS.

An early semi-egocentric version of the PLA learns heading invariant places defined by panoramic local views. Implementing this version on the mobile robot MAVIN has demonstrated a number of properties of the view-based approach, including real-time operation, robustness to various noise sources, an efficient storage of places and environments, and the potential to perform sub-place localization at multiple spatial scales. Note, however, that the robot must search the entire visual panorama for landmarks at each location. It must also use an accurate compass.

A fully egocentric version of the PLA, employing a logarithmic restricted local view transformation, addresses these problems. Implementing this version on MAVIN has verified that the general properties of the semi-egocentric version are retained, but has also revealed that the architecture learns heading tuned places — that is, place nodes tuned not only to the location of the rat, but also to the absolute direction the robot faces with respect to the environment. Simple experiments with this version have demonstrated the ability to learn such heading tuned places at multiple spatial scales simply by varying the vigilance parameter of the embedded ART network.

The heading specificity of learned place nodes paves the way for learning relational maps in terms of relative forward and turning motions of the robot. Without heading specificity, the robot must

measure the actions or spatial relations relating adjacent places relative to a global directional frame of reference. The next chapter describes an architecture that takes advantage of this heading tuned place learning in order to learn and use a diktiometric map of an environment.

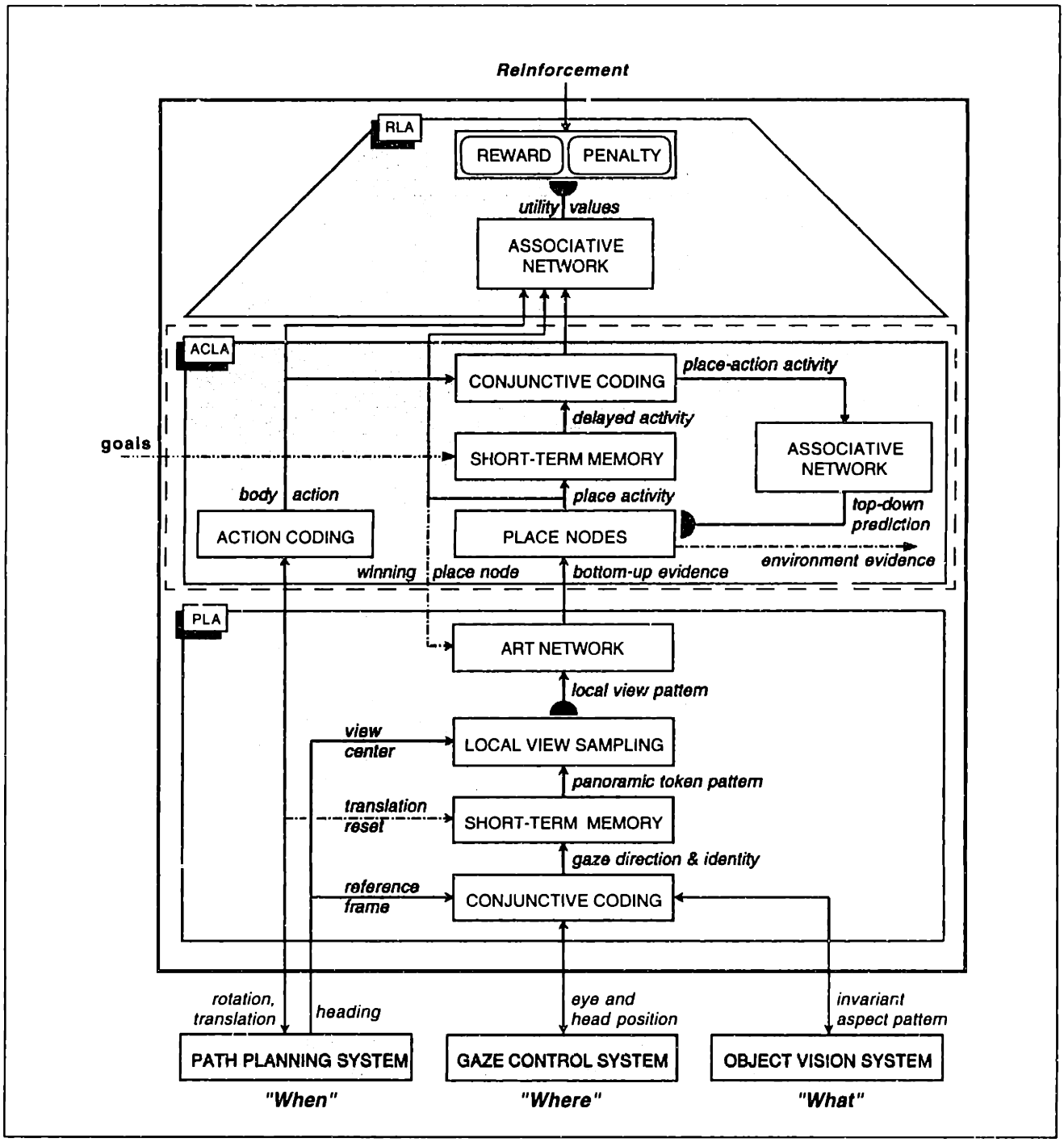


**FIGURE 51: THE ACTION CONSEQUENCE LEARNING ARCHITECTURE (ACLA).**
*The action consequence learning architecture, highlighted here as part of the navigation strategy learning sub-system, takes as primary input the bottom-up sensory place evidence from the PLA (coding for the "What" and "Where"), and the recent locomotive action executed by the robot (the "When").*

CHAPTER 8

# *Action Consequence Learning and Prediction*

The action consequence learning architecture (ACLA), highlighted in Figure 51, emulates cognitive mapping in the rat hippocampus by building a relational model of an environment reminiscent of an aspect graph, thereby completing the analogy with object learning and recognition (see Section 2.2.3). It learns the allowable transitions between neighboring places in terms of the turning and forward body motions that the robot must perform to physically move between their corresponding place regions.

This chapter first describes the ACLA at the architectural level, and qualitatively describes the concept of using the architecture to perform place prediction and environment recognition. It then defines the dynamics of the architecture, and gives a mathematical account of a possible probabilistic interpretation based on a POMDP. Finally, it describes the results of simulating the architecture using data gathered by the mobile robot MAVIN.

## 8.1 Neurocomputational Architecture

The ACLA (detailed in Figure 52) takes as input the *sensory* (bottom-up) evidence for places gathered using the PLA ("What" and "Where"), as well as the relative turning and forward body motions executed by the robot ("When") within a fixed time interval. It performs contrast-enhancement operations on the sensory place evidence, thereby producing a sparse pattern of evidence across place categories. It also coarsely and independently codes the directional, translational, and orientational movements of the robot using three separate, fixed populations with overlapping Gaussian receptive fields. It then forms a pattern of place-action nodes from the conjunction of a short term memory (representing the *contextual place evidence* at the last instant in time) and the outputs of the three receptive field populations. This pattern feeds a localist heteroassociative network that learns to predict consequences in terms of the sensory place evidence at the next instant in time. Finally, predictions are recurrently integrated with the incoming sensory place evidence in order to assess the current contextual place evidence based on both visual input and history. In this manner, the architecture concurrently maintains the evidence for multi-

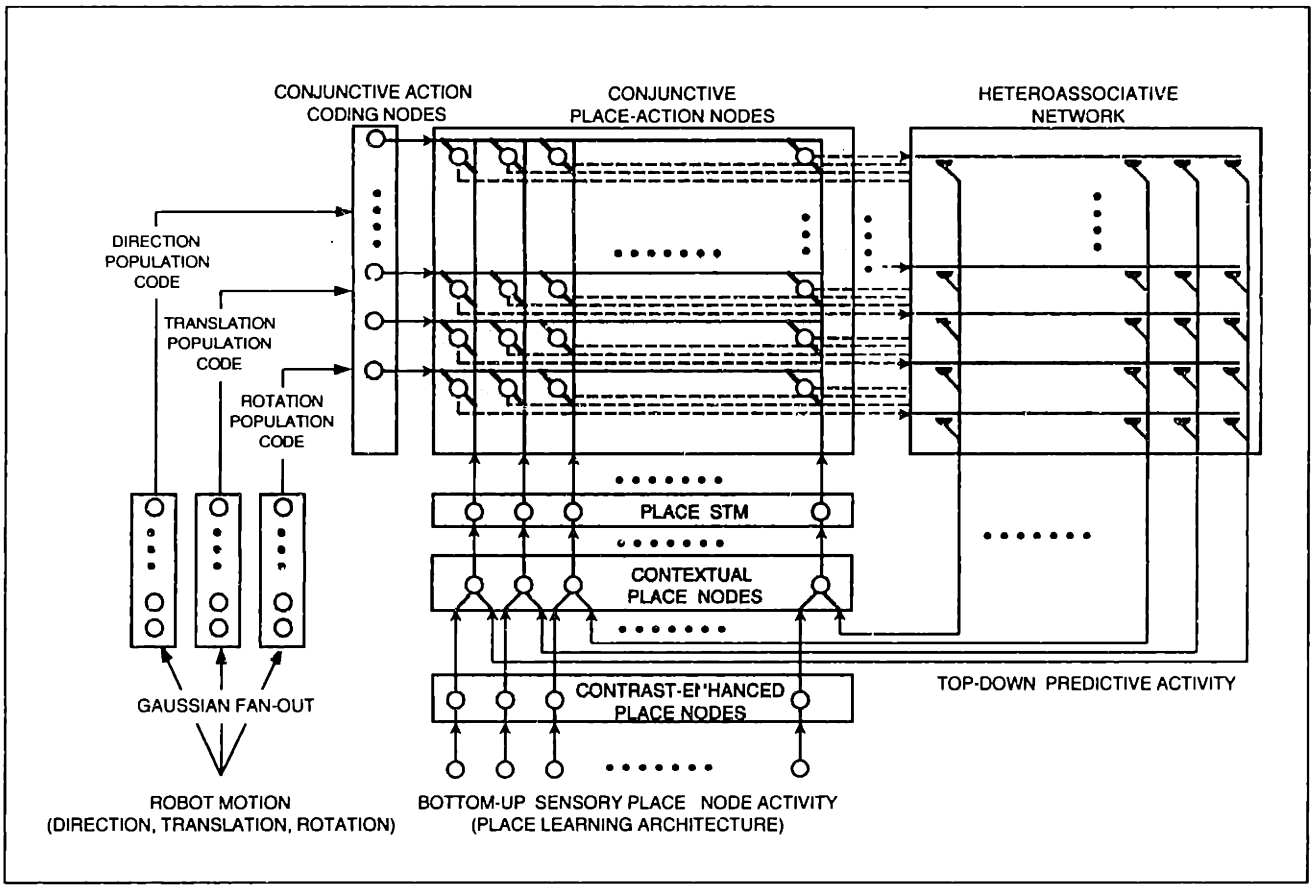


**FIGURE 52: NETWORK ARCHITECTURE FOR THE ACLA.**

*The action consequence learning architecture performs unsupervised transition learning within an associative network between a set of contextual place nodes, which contain the current evidence for places, and a set of place-action nodes, which conjunctively code the action taken in the previous place. The contextual place evidence at the next instant in time arises from a combination of top-down predictive evidence form the associative network, and bottom-up evidence from the place learning architecture.*

ple place hypotheses, even though it makes a choice in order to effect competitive learning for both places and action consequences.

The ACLA can be viewed as a localist RNN with a static "hidden layer" forming a conjunction of state and action units, and an input layer forming a conjunction of sensory input and predictive feedback (see Section 3.3.4). Its main advantages over most existing RNNs lie in its learning of action consequences in a single trial, in its refinement of these consequences in a few additional trials, and in its ability to learn multiple sequences independently of time. The ACLA also resembles a number of diktiometric map-making techniques (see Section 5.2.1), but avoids the problems associated with purely symbolic representations of such maps.

## 8.2 Place Prediction and Environment Recognition

The ACLA has been designed to robustly support *long-term prediction*, *environment recognition*, *route planning*, and *exploration* (see Section 5.2). Long-term prediction and environment recognition, discussed here, require only the learned map, embodied by the PLA and the ACLA described above. Route planning and exploration, discussed later in Section 9.2, additionally require some learned utility over places and actions (i.e. the entire navigation strategy).

Long-term prediction refers to assessing the expected outcome of an arbitrary length sequence of forward and turning body movements. The ACLA performs predictive tasks simply by initializing its short-term memory to indicate the starting place, and by sequentially simulating ("imagining") the sequence of actions while forcing uniform sensory place evidence (indicating that the PLA has no visual information regarding the current place), a processes equivalent to running the robot around "blindfolded".

Environment recognition involves gathering evidence for possible environments by assessing, for each learned map, the likelihood of experiencing the sequence of observed places as a consequence of executing a sequence of movements. The degree of match between the top-down prediction and the bottom-up sensory evidence provides an estimate of this likelihood, and is analogous to the accumulation of static (single view) and dynamic (temporal sequences of views) evidence for object recognition (see Section 2.2.2).

## 8.3 Dynamics

The discrete time dynamics of the ACLA are presented here using the variables described in Table 8 and the architecture shown in Figure 52. The architecture is first initialized as follows.

### 8.3.1 Initialization

1. The action integration counter $n_A$ is reset, i.e.

$$n_A = 0. \tag{34}$$

2. The short-term memory nodes $\bar{m}$ are normalized to reflect uniform distribution of activity over places, i.e.

$$\bar{m} = 1/n_p. \tag{35}$$

3. The activity of the environment node $z_o$, indicating the current evidence for the environment currently in working memory, is normalized to reflect a uniform distribution of activity across all environments, i.e.

$$z_o = 1/n_e \tag{36}$$

where $n_e$ is the number of known environments.

**TABLE 8: ACTION CONSEQUENCE LEARNING ARCHITECTURE (ACLA) VARIABLE DEFINITIONS.**

| Variable | Definition |
|---|---|
| $\bar{z}$ | The 1D pattern ($n_e$-dimensional vector) representing the activities of the environment nodes (across all ACLA architectures). $z_0$ represents the scalar evidence for the environment whose map currently resides in working memory. |
| $\bar{r}^{\theta}, \bar{r}^{\rho}, \bar{r}^{\phi}$ | The 1D patterns (vectors with dimensions $n_\theta$, $n_\rho$, and $n_\phi$ respectively) representing the activities of the directional, translational, and rotational coding nodes, respectively. |
| $\bar{a}$ | The 1D pattern ( $(n_\theta \cdot n_\rho \cdot n_\phi)$ -dimensional vector) representing the activities of the action coding nodes. |
| $\bar{b}$ | The 1D pattern ($n_p$-dimensional vector) representing the activities of the *bottom-up* place nodes. |
| $\bar{p}$ | The 1D pattern ($n_p$-dimensional vector) representing the activities of the *contextual* place nodes. |
| $\bar{m}$ | The 1D pattern ($n_p$-dimensional vector) representing the activities of the *short-term memory* place nodes. |
| $D$ | The 2D pattern ($n_p \times (n_\theta \cdot n_\rho \cdot n_\phi)$ matrix) representing the activities of the place-action nodes. $D_{ij}$ is the activity of the place-action node corresponding to place $i$ and action $j$. |
| $\bar{t}$ | The 1D pattern ($n_p$-dimensional vector) representing the predictive output of the associative network, and indicating the top -down predictive evidence for places. $t_i$ is the predictive evidence for place $i$. |

4. Action equivalence classes (i.e. actions that are known *a priori* to produce the same consequences) are determined, such as groups of action coding nodes that represent actions for which $A_\rho^{in} = 0$ and $A_\phi^{in} + A_\theta^{in}$ is constant (a turn in place).
5. Certain weights in the associative network are set to reflect prior knowledge of action consequences. For example, the transition template from any given place-action node that codes an action for which $A_\rho^{in} = 0$ and $A_\phi^{in} + A_\theta^{in} = 0$ (a "null" action) is simply given by a vector pattern for which all the elements are zero except that element corresponding to the current place. That is, taking no action at all will always result in remaining in the same place.

### 8.3.2 Algorithmic Implementation

After initialization, the following procedure is executed after each time interval $\Delta\tau$:

1. The recently executed locomotive action $\bar{A}^{in}$, measured over the last action time interval $\tau_A = n_A \cdot \Delta\tau$ by the PPS (see equations (15)-(17) in Section 6.4.1), is coded by three populations of nodes with overlapping Gaussian receptive fields (see Appendix C). Their outputs are respectively given by

$$\bar{r}^{\theta} = \mathrm{RF}_p(A_{\theta}^{in}; n_{\theta}, -\pi, \pi) \tag{37}$$

$$\bar{r}^{\rho} = \mathrm{RF}_f(A_{\rho}^{in}; n_{\rho}, 0, A_{\rho}^{max}) \tag{38}$$

$$\bar{r}^{\phi} = \mathrm{RF}_p(A_{\phi}^{in}; n_{\phi}, -\pi, \pi). \tag{39}$$

   where $n_{\theta}$, $n_{\rho}$, and $n_{\phi}$ are the respective numbers of direction, translation, and rotation coding nodes, and $A_{\rho}^{max}$ is an upper bound on the maximum distance travelled between any two place regions.

2. These separate populations are combined to form the action coding nodes $\bar{a}$ via the equation

$$\bar{a} = \mathrm{cat}(\mathrm{conj}(\bar{r}^{\theta}, \bar{r}^{\rho}, \bar{r}^{\phi})). \tag{40}$$

   At this point, any obvious action equivalence classes may be accounted for by finding the maximum activity for all of the action nodes in each equivalence class, and setting only one of these action nodes (i.e. the one with the smallest index) to this maximum value, while setting all others to zero.

3. The incoming raw sensory place activities $\bar{s}$, containing the raw evidence for each of the $n_p$ learned place nodes, is contrast-enhanced using the sigmoid function in order to form the bottom-up place activities. That is,

$$b_i = S(s_i; \nu, \mu) \tag{41}$$

   where $\mu$ and $\nu$ are the slope and position of the sigmoid inflection point respectively.

4. The bottom-up evidence $\bar{b}$ is combined with the top-down evidence $\bar{t}$ (defined shortly) to form the contextual place nodes $\bar{p}$ via the equation

$$p_i = (b_i t_i) / \hat{z}_o. \tag{42}$$

   where $\hat{z}_o$ is a normalization factor given by

$$\hat{z}_o = \bar{b} \cdot \bar{t}. \tag{43}$$

   This normalization term indicates the instantaneous bottom-up evidence for the environment whose map currently resides in local memory, determined by the degree of match between bottom-up sensory evidence and top-down expectations for places in the map.

Note that, when the PLA and ACLA are integrated, the activities of the contextual place nodes $\bar{p}$, rather than the sensory place nodes $\bar{s}$, are used to choose a place node "winner" $i_{max}$ for adapting the ART network weights in the PLA (see Section 7.1).

5. Given the instantaneous environment evidence $\hat{z}_o$, the activity of the environment node is updated according to the equation

$$z_o \Leftarrow z_o \cdot \hat{z}_o . \tag{44}$$

In essence, this computation combines static (bottom-up) evidence with dynamic (top-down history) evidence, similar to the accumulation of object evidence by object nodes in the Aspect Network (see Section 2.2.2). A normalization across all environment nodes then occurs, i.e.

$$\bar{z} \Leftarrow \frac{\bar{z}}{|\bar{z}|} . \tag{45}$$

To perform environment recognition, one simply chooses the environment $i$ with the maximum evidence $z_i$. (This entire process requires either that each of the maps be loaded into working memory serially, or that the mapping processes for all environments run in parallel).

6. The new winning place node $i_{max}$ is compared to the previous winning place node $i'_{max}$. If these nodes differ, the action integration counter $n_A$ is reset, and the short-term memory nodes latch the current contextual place evidence, i.e.

$$n_A = 0 \tag{46}$$

$$\bar{m} = \bar{p} . \tag{47}$$

Otherwise, the action integration counter gets incremented, i.e.

$$n_A \Leftarrow n_A + 1 . \tag{48}$$

(Note that in a continuous time version of the ACLA, the short-term memory nodes would be activated according to the equation

$$\frac{d}{dt}\bar{m} = \lambda(\bar{p} - \bar{m}) \tag{49}$$

where $\lambda$ controls the time scale for the architecture.)

7. The short term memory place nodes are conjunctively combined with the action coding node activity to form the action-place nodes, via the equation

$$\boldsymbol{D} = \text{conj}(\bar{m}, \bar{a}) . \tag{50}$$

8. The vector cat($\boldsymbol{D}$) forms the input, along with the desired output $\bar{b}$, to a localist associative network, the dynamics of which are presented in Appendix D. The predictive output of this network computes the top-down evidence $\hat{\imath}$. Each of the weight templates in this network represent the expected distribution of bottom-up place evidence that follows from executing a prototypical action $j$ from the center of a particular place region $i$. Since actions are limited in

duration, these distributions represent "fuzzy" topological relations between nearby places. Note that the use of this network requires that the cat($\boldsymbol{D}$) vectors that result in different place evidence $\boldsymbol{b}$ after a particular action be linearly independent, which is satisfied by the fact the both place activity profiles and action coding node receptive fields overlap. It also requires that the input cat($\boldsymbol{D}$) be a localist representation, which is also satisfied due to the fact that each of the place and action coding nodes are, for the most part, respectively tuned to unique place regions and locomotive actions.

During the course of this procedure, each of the structures in the ACLA (the sets of nodes, the connections, and the associative network) grow with the number of place nodes $n_p$. Thus, implementations on serial computers must either initially allocate at least as many nodes and connections as the architecture will ever possibly need, or dynamically allocate new nodes and connections as the number of places increases.

## 8.4 A Probabilistic Interpretation

The architecture dynamics described in the previous section are reminiscent of a variety of probabilistic techniques. Action consequence (top-down) place prediction resembles state prediction by an MDP, and dynamic (contextual) place recognition resembles an incremental (step-wise) form of the forward-backward procedure for HMMs (see Shen, 1994), as well as the Bayesian state estimation techniques used for POMDPs (White, 1993; see Appendix E). In fact, the network equations actually approximate those of a POMDP, with probabilistic actions as well as states, contingent on a few assumptions.

### 8.4.1 Intuition

One intuitive way of understanding the correspondence between the ACLA and a POMDP is to realize that the "states" and "actions" have been carefully chosen so that the current place and action are all that is needed in order to determine the next place. The next place (i.e. next location and heading) depends only on the current place (i.e. current location and heading) plus the last action (i.e. direction with respect to the current heading, a translation, and a final rotation). Furthermore, both the places (positions) and the actions (motions) are only partially observable in terms of their bottom-up, graded evidence. Graded evidence for place recognition is obtained from static (bottom-up) activity determined by the PLA, the process of which resembles an ML approach to concept learning. Graded evidence for actions is obtained from analog receptive fields, which perform a sort of course probabilistic coding.

Note that this Markovian interpretation would not be possible without resorting to a global frame of reference if the places coded only for location, invariant to heading. In this case, the next place (i.e. next location) would depend not only on the current place (i.e. current location) and action, but also on the direction from which one entered the current place (absolute heading) — a non-Markovian "momentum." One could, of course, make the absolute heading a part of the action representation rather than that of the places, but this coding would require an accurate compass. A similar dilemma would also arise if one defined actions in other ways, such as using average velocity instead of distance travelled.

TABLE 9: ACLA PROBABILISTIC VARIABLE DEFINITIONS.

| Variable | Definition |
|---|---|
| $Z_i$ | The statement: "Situated in environment $i$." Note that $Z_o$ stands for the statement: "Situated in the environment whose representation is currently stored in working memory." |
| $P_i$ | The statement: "Currently situated in state $i$." |
| $P_i'$ | The statement: "Was situated in state $i$ $\tau_A$ seconds ago." |
| $\Theta_i$ | The statement: "The body movement just executed was in direction $i$." |
| $\Sigma_i$ | The statement: "The body movement just executed included a translation $i$." |
| $\Phi_i$ | The statement: "The body movement just executed included a rotation $i$." |
| $A_j$ | The statement: "Just finished executing action $j$." |

With this intuition in mind, a mathematical correspondence between the ACLA and a POMDP can be made, based on the following assumptions (see Table 9 for notation).

## 8.4.2 Assumptions

1. The contrast-enhanced bottom-up place evidence $\bar{b}$ provides the likelihood of each of the places — of "seeing" the current restricted local view $V^R$ in each of the places — given that the robot is situated in the environment currently stored in "working memory." That is, for every $i$,

$$b_i \equiv \Pr\left( V^R | Z_o, P_i \right). \tag{51}$$

   Hence, if we were to make a choice for place recognition by picking the winner of these input values (which, incidentally, is what ART normally does), then we would be performing a maximum likelihood (ML) classification (see Section 3.2.4). This interpretation is reasonable, since $b_i$ does in fact indicate how close the sensory input resembles each of the category prototypes. Furthermore, the use of a sigmoid to contrast-enhance the raw sensory place evidence $\bar{s}$ produces Gaussian-like profiles for each place category as a function of position.

2. Each action coding node indicates the posterior probability of having executed its corresponding motion category. That is,

$$r_i^{\theta} \equiv \Pr(\Theta_i) \tag{52}$$

$$r_i^{\rho} \equiv \Pr(T_i) \tag{53}$$

$$r_i^{\phi} \equiv \Pr(\Phi_i) . \tag{54}$$

This too is a reasonable assumption, since each category has a Gaussian spatial distribution, and since the probability distribution over categories is normalized.

3. The environment is Markovian (see Appendix E):

   (a) The probability of seeing the current restricted local view $V^R$ is only dependent on the current environment and the current place. That is,

$$\Pr\left( V^R | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, P_i \right) = \Pr(V^R | Z_o, P_i) \tag{55}$$

   for all $i$.

   (b) The probability of having taking any action $j$ is only dependent on independent directional, translational, and rotational body motions. That is,

$$\Pr\left( A_{j(j_\theta, j_\rho, j_\phi)} | Z_o, \{\bar{A}^{\text{in}}, V^R\}, P_i' \right) = \Pr(\Theta_{j_\theta}) \cdot \Pr(\Sigma_{j_\rho}) \cdot \Pr(\Phi_{j_\phi}) \tag{56}$$

   for all $i$ and $j(j_\theta, j_\rho, j_\phi)$. (Note: there is a unique value of $j$ for every combination of values for $j_\theta$, $j_\rho$, and $j_\phi$.)

   (c) The current place is dependent only on the last place and last action. That is,

$$\Pr\left( P_i | Z_o, \{A^{\text{in}}, V^R\}, P_l', A_j \right) = \Pr(P_i | Z_o, P_l', A_j) \tag{57}$$

   for all $i$, $j$, and $l$.

4. The probability of having executed any locomotive action $\bar{A}^{\text{in}}$ is independent of the current environment $Z_o$.

### 8.4.3 Interpretations

The probabilistic assumptions outlined in the previous section allow one to interpret, and subsequently explain, the ACLA architecture using probability theory as follows:

1. The activity of each contextual place node indicates the posterior probability of its corresponding place category, given the sequence of actions and local restricted views up to and including the current time. That is,

$$p_i \equiv \Pr\left( P_i | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, V^R \right). \tag{58}$$

   Note, then, that choosing the current place based on the maximum contextual place node corresponds to using the maximum *a posteriori* (MAP) criterion (see Section 3.2.4).

2. The activity of the environment node $y_i$ indicates the likelihood of environment $i$, given the history of actions and restricted local views up to the current time. That is,

$$y_i = \frac{\Pr\left( \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, V^R \middle| Z_i \right)}{\sum_{j=1}^{n_e} \Pr\left( \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, V^R \middle| Z_j \right)}. \tag{59}$$

The choice of environment for recognition therefore uses the ML criterion. Furthermore, the computation performed by equation (44) resembles an incremental form of the forward-backward procedure for HMMs (See Rabiner and Juang, 1986).

3. The transitional weights in the associative network represent Markov transition probabilities between states. That is,

$$w_l^{k(i,j)} \equiv \Pr(P_l | Z_o, P_i', A_j) . \tag{60}$$

(Note: there is a unique index $k$ for every combination of values for $i$ and $j$. Action-place node $k(i,j)$ corresponds to place $i$ and action $j$.)

4. The top-down evidence indicates the probability distribution over places, conditional upon the sequence of restricted local views experienced up to but not including the current time, and the sequence of actions performed just prior to the current time. That is,

$$t_i \equiv \Pr\left( P_i | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}} \right). \tag{61}$$

### 8.4.4 Verification

To verify these interpretations, let us first examine the combination of bottom-up and top-down evidence (see equations (42) and (43)). It turns out that, under these interpretations, this operation performs a recursive Bayesian update (see Pearl, 1988). The Bayesian update formula is, in our case, given by

$$\Pr\left( P_i | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, V^R \right) = \Pr\left( P_i | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}} \right) \cdot \frac{\Pr\left( V^R | Z_o, \{\bar{A}^{\text{in}}, V^R\} \bar{A}^{\text{in}}, P_i \right)}{\Pr\left( V^R | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}} \right)}. \tag{62}$$

This formula essentially provides a way to incorporate the new restricted local view evidence $V^R$ into the current probability estimate $\Pr(P_i | Z_o, \{\bar{A}^{\text{in}}, V^R\} \bar{A}^{\text{in}})$. After simplifying with (55), substituting equations (51), (58), and (61) yields

$$p_i = \frac{(t_i b_i)}{\Pr\left( V^R | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}} \right)}. \tag{63}$$

Due to the mutually exclusive nature of places (their probabilities sum to 1) and the constancy of $\Pr(V^R|Z_o, \{\bar{A}^{in}, V^R\}, \bar{A}^{in})$ over all places, this expression is equivalent to the place node equation (42). It is also clear from equations (42) and (63) that

$$\hat{z}_o = \Pr\left(V^R|Z_o, \{\bar{A}^{in}, V^R\}, \bar{A}^{in}\right) \tag{64}$$

which, by the definition of conditional probability, equation (59), and assumption 4, is equal to

$$\begin{aligned}\hat{z}_o &= \frac{\Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}, V^R\middle|Z_o\right)}{\Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}\middle|Z_o\right)} = \frac{z_o \cdot \sum_{j=1}^{n_c} \Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}, V^R\middle|Z_j\right)}{\Pr\left(\bar{A}^{in}|Z_o, \{\bar{A}^{in}, V^R\}\right) \cdot \Pr\left(\{\bar{A}^{in}, V^R\}\middle|Z_o\right)} \\ &= \frac{z_o \cdot \sum_{j=1}^{n_c} \Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}, V^R\middle|Z_j\right)}{\Pr\left(\bar{A}^{in}|Z_o, \{\bar{A}^{in}, V^R\}\right) \cdot \left(z_o' \cdot \sum_{j=1}^{n_c} \Pr\left(\{\bar{A}^{in}, V^R\}\middle|Z_j\right)\right)} = \Omega\frac{z_o}{z_o'}\end{aligned} \tag{65}$$

where $z_o'$ is the evidence for the environment one action time step $\tau_A$ in the past, and

$$\Omega = \frac{\sum_{j=1}^{n_c} \Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}, V^R\middle|Z_j\right)}{\Pr\left(\bar{A}^{in}|Z_o, \{\bar{A}^{in}, V^R\}\right) \cdot \sum_{j=1}^{n_c} \Pr\left(\{\bar{A}^{in}, V^R\}, \bar{A}^{in}, V^R\middle|Z_j\right)} \tag{66}$$

is a constant for the current time. Given that $\bar{z}$ is normalized, equation (65) is equivalent to equations (44) and (45) when $\tau_A = \Delta\tau$, where $\Omega$ is the normalization factor $|\bar{z}|$.

Next, we examine the establishment of action-place nodes $\bar{a}$ in equation (40) and the computation of top-down evidence $t$ using equations (49) and (50) and the associative network. First, note that, by equations (56), (40) and (52)-(54), each action coding node $j$ can be described by

$$a_{j(j_\theta, j_\rho, j_\phi)} = \Pr\left(A_{j(j_\theta, j_\rho, j_\phi)}|Z_o, \{A^{in}, V^R\}, P_i', \bar{A}^{in}\right) \tag{67}$$

for any value of $i$. Now, by equations (46)-(48) and (58), each short-term memory node $i$ represents the posterior probability for place $i$ one action time step $\tau_A$ in the past, or

$$m_i = \Pr\left(P_i'|Z_o, \{\bar{A}^{in}, V^R\}\right) = \Pr\left(P_i'|Z_o, \{\bar{A}^{in}, V^R\}, \bar{A}^{in}\right). \tag{68}$$

(Due to causality, this probability is independent of the current action $\bar{A}^{in}$ in the absence of any information about its consequences.) Now, substituting this equation and equation (67) into equation (50) yields

$$D_{ij} = \Pr\left(A_j, P_i' | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \tag{69}$$

(by definition of conditional probability). Therefore, an action-place node signals the probability of having just executed direction-translation-rotation action $j(j_\theta, j_\rho, j_\phi)$ in place $i$, given the sequence of actions and sensations up to but not including the current time. Substituting equations (60) and (69) into the feedforward prediction equation (11i) for the heteroassociative network (in Appendix D) then yields

$$t_l = \sum_{i=1}^{n_p} \sum_{j=1}^{n_\theta \cdot n_\rho \cdot n_\phi} \Pr(P_l | Z_o) \cdot \Pr\left(P_i', A_j | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \tag{70}$$

which, by equation (57), can be rewritten as

$$t_l = \sum_{i=1}^{n_p} \sum_{j=1}^{n_\theta \cdot n_\rho \cdot n_\phi} \Pr\left(P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, P_i', A_j\right) \cdot \Pr\left(P_i', A_j | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right). \tag{71}$$

By the definition of conditional probability, this equation becomes

$$t_l = \sum_{i=1}^{n_p} \sum_{j=1}^{n_\theta \cdot n_\rho \cdot n_\phi} \Pr\left(P_i', A_j, P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right)$$

$$= \sum_{i=1}^{n_p} \sum_{j=1}^{n_\theta \cdot n_\rho \cdot n_\phi} \Pr\left(A_j | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, P_i', P_l\right) \cdot \Pr\left(P_i', P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right)$$

$$= \sum_{i=1}^{n_p} \left( \Pr\left(P_i', P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \sum_{j=1}^{n_\theta \cdot n_\rho \cdot n_\phi} \Pr\left(A_j | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, P_i', P_l\right) \right). \tag{72}$$

Since actions are mutually exclusive (their probabilities sum to 1), the second (inner) summation of this equation equals 1, and we are left with

$$t_l = \sum_{i=1}^{n_p} \Pr\left(P_i', P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \tag{73}$$

which, by definition of conditional probability, yields

$$t_l = \sum_{i=1}^{n} \Pr\left(P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \cdot \Pr\left(P_i' | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}, P_l\right) \tag{74}$$

$$= \Pr\left(P_l | Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right) \cdot \sum_{i=1}^{n_{pp}} \Pr\left(P_i' | P_l, Z_o, \{\bar{A}^{\text{in}}, V^R\}, \bar{A}^{\text{in}}\right). \tag{75}$$

TABLE 10: ACLA PARAMETER SETTINGS.

| Parameter | $A_\rho^{max}$ | $n_\theta, n_\rho, n_\phi$ | $\lambda$ | $\beta$ | $\mu, \nu$ |
|---|---|---|---|---|---|
| Value | 5 ft. | 23,6,23 | 1 | 0.1 | 10,1.0 |

Finally, due to mutually exclusive places, the second term of this equation equals 1, and we are left with the top-down evidence interpretation (61).

# 8.5 Simulations Using Mobile Robot Data

The ACLA has been implemented on a Sparc 10 workstation with the parameters set according to Table 10. The details of this implementation are described in the following sections.

## 8.5.1 Training Methods

Initial tests of this version (Bachelder & Waxman, 1995) have simulated the exploration of the laboratory environment off-line, specifically by feeding the ACLA a set of triples, each consisting of the current contextual place activities $\bar{p}$, a locomotive action $\bar{A}^{in}$, and the resulting sensory place activities $\bar{s}$. All of the sensory place evidence patterns consist of actual measurements of the learned place nodes presented earlier (see Sections 7.6 and 7.7). These measurements were made by MAVIN at each of the 2,880 sampled positions (80 locations x 36 headings) that were originally used to learn these places. A total of 8.2944 x $10^6$ triples were created by synthesizing actions between ordered pairwise sets of these sampled positions. This data set was then widdled down to 82,000 triples by considering only those triples for which $A_\rho^{in} < A_\rho^{max}$.

Following this learning phase, the predictive ability of the ACLA was tested by giving it starting activities $\bar{p}$ and sequences of actions $\bar{A}_1^{in}, \bar{A}_2^{in}, \ldots, \bar{A}_n^{in}$, and comparing the resulting predicted place sequence $\bar{p}_1, \ldots, \bar{p}_n$ to the sequence that would have been indicated by the PLA without any action information (i.e. $(\bar{s}_1/|\bar{s}_1|), \ldots, (\bar{s}_n/|\bar{s}_n|)$ ). So that results could be compared to the rotational parcelling in Figure 42, the first sequence consists of rotations of 10° intervals at a sampled location. To test the ACLA's generalization ability, the second sequence consists of more-or-less random actions that differ from those used to train the ACLA.

## 8.5.2 Training Results

Not surprisingly, the learning results depend upon the spatial scale with which the PLA learned new place nodes, determined primarily by the vigilance parameter of the ART' network (see Section 7.7). For a vigilance of 0.9 (large scale), the ACLA learned and adapted approximately 1500 transition templates. Training required approximately 15 hours on a Sparc 10 for 3 passes through the data (approximately 0.2 seconds for each triple). For a vigilance of 0.95 (small scale), the

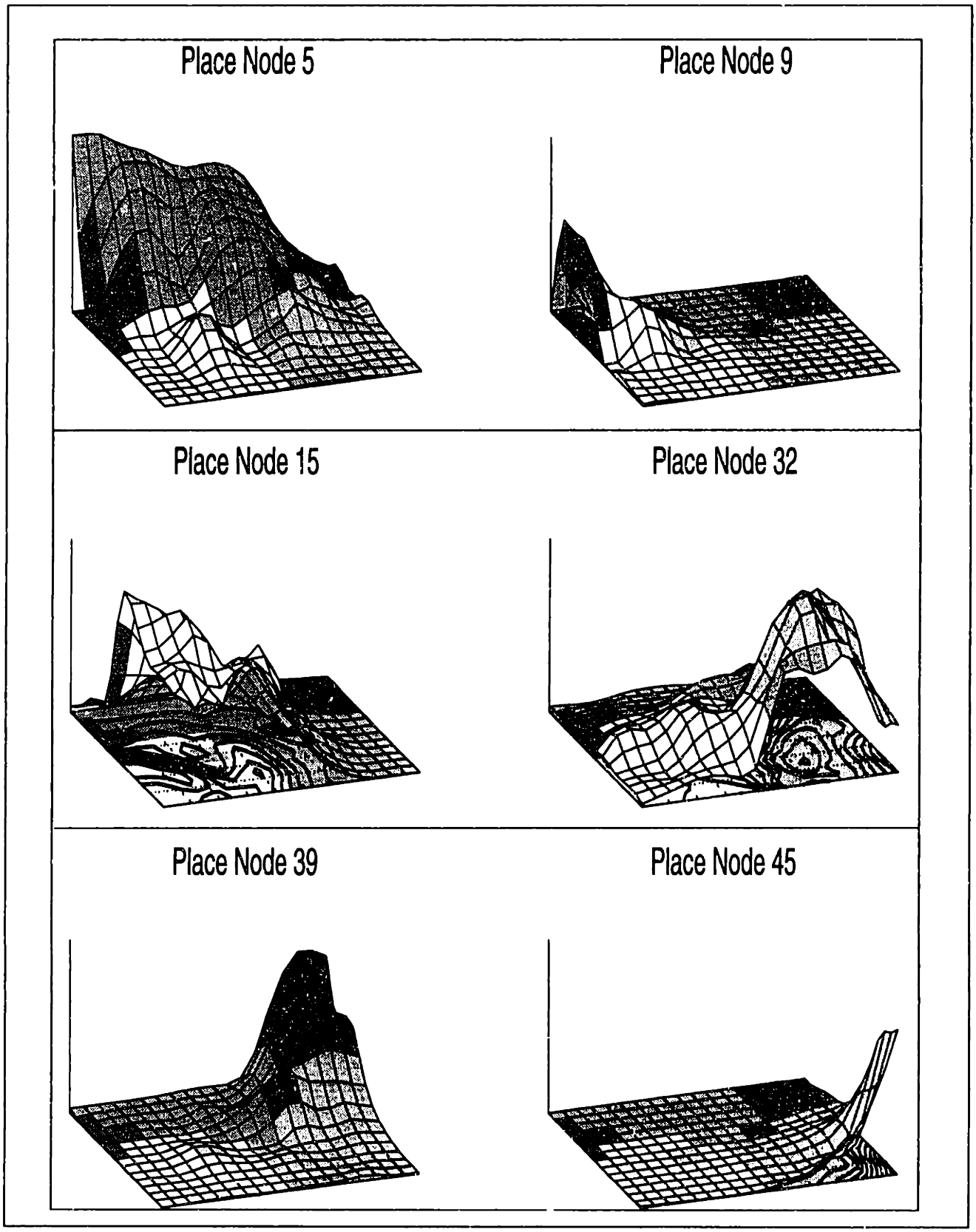


**FIGURE 53: CONTRAST-ENHANCED PLACE NODE ACTIVITY AS A FUNCTION OF LOCATION (LARGE SCALE).**

*The contrast-enhanced activity profiles for each of the most active large scale contextual place nodes as a function of location for a 240° heading (sensory input only).*

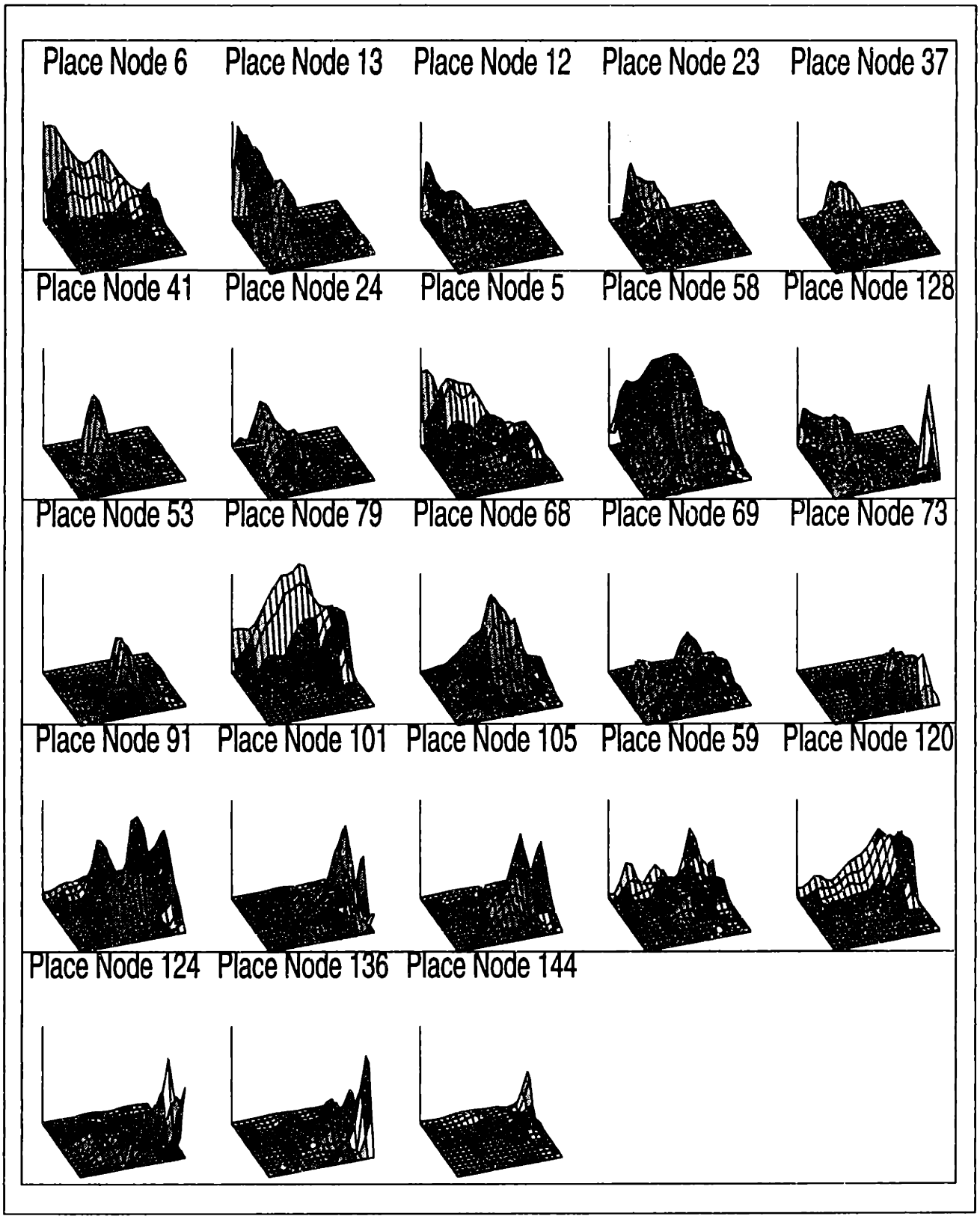


**FIGURE 54: CONTRAST-ENHANCED PLACE NODE ACTIVITY AS A FUNCTION OF LOCATION (SMALL SCALE).**

*The contrast-enhanced activity profiles for each of the most active small scale contextual place nodes as a function of location for a 210° heading (sensory input only).*

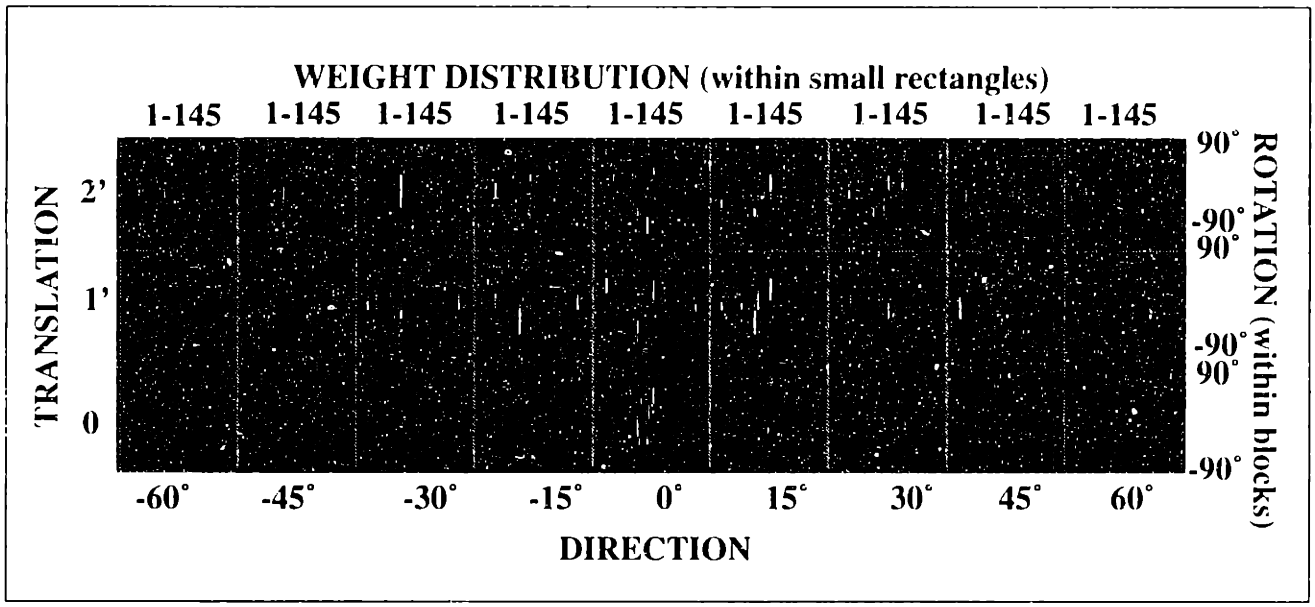


**FIGURE 55: WEIGHT TRANSITION TEMPLATES FOR A SINGLE PLACE.**

*This computer screen frame shows all the place-action transition templates for small scale place #32 (vigilance = 0.95). Each large block corresponds to a given direction (horizontal axis) and translation (vertical axis), while each horizontal rectangle within a given large block corresponds to a given rotation (vertical axis). Within each rectangle is shown the weight template as consecutive bars (from left to right) for each of the numbered places (horizontal axis). The brightness of each bar is proportional to the corresponding weight.*

architecture learned approximately 10,000 transition templates, and required 45 hours for a single pass through the data. (Note that this is, or course, a fairly exhaustive set of data. Presumably, the architecture should be capable of generalizing from a smaller amount of data.)

For each of the vigilance settings, the appropriate parameters for the contrast-enhancement of incoming sensory place evidence were determined with the generalization vs. discrimination trade-off in mind (see Section 3.2.1). On the one hand, a sufficient amount of overlap between each of the activity profiles is necessary for generalization, while on the other a decent amount of activity fall-off with distance from the center is necessary for discrimination. The contrast-enhanced contextual place node activity profiles as a function of position for each of these spatial scales are shown in Figures 53 and 54, respectively. The activity of these nodes as a function of heading for each of these scales is shown in Figures 56 and 57, respectively. It is important to note that contrast-enhancement does not affect the decision boundaries for each of the place regions (relative ordering of evidence is retained).

The training results are exemplified by Figure 55, which shows the learned weight templates for a single place category (over all actions). Note that these templates are sparsely distributed. In fact, there are $O(n_p)$ transitions rather than $O(n_p^2)$ transitions. For this reason, the implementation was able to utilize a sparse matrix representation for the associative network.

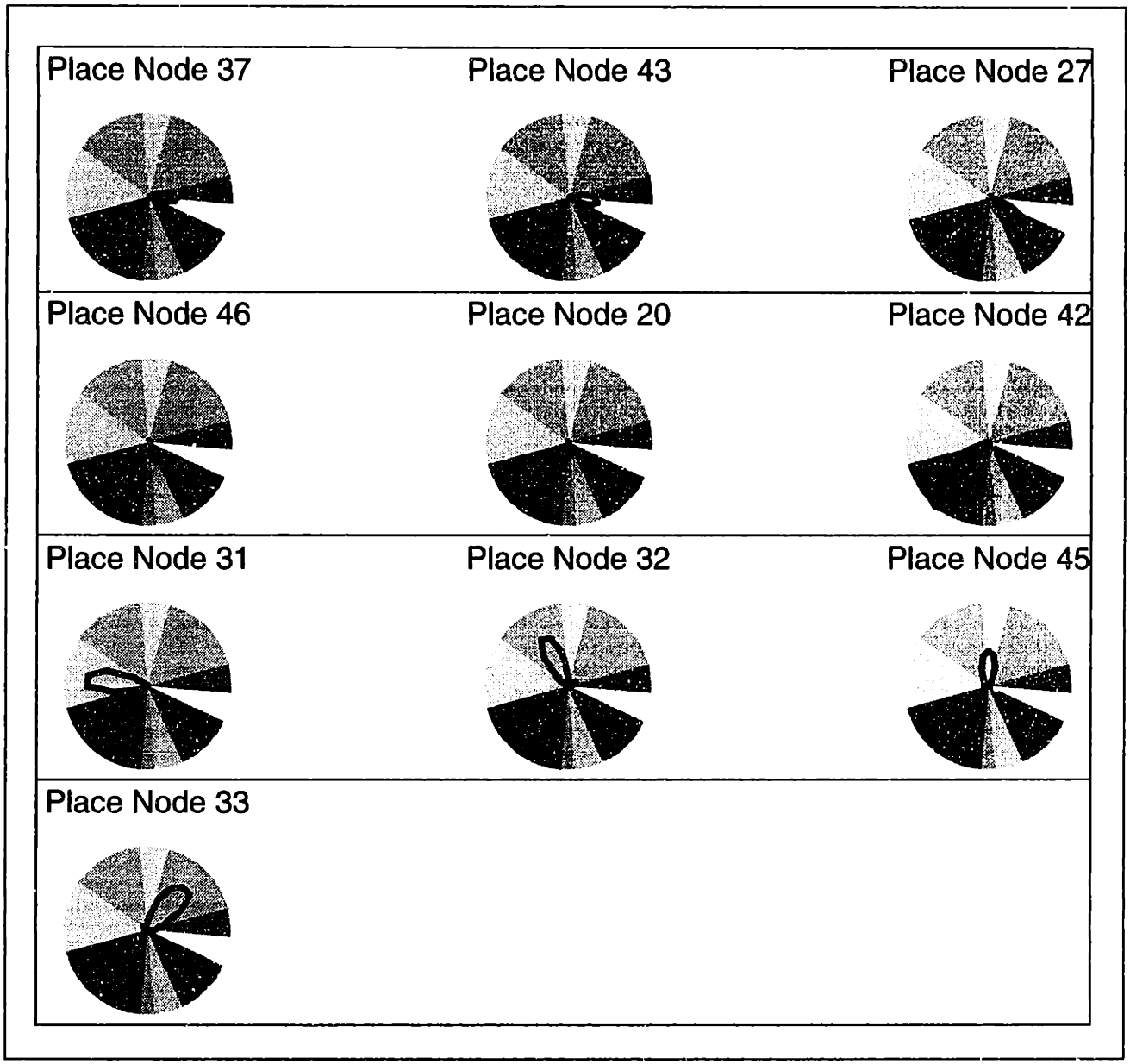


**FIGURE 56: CONTRAST-ENHANCED PLACE NODE ACTIVITY AS A FUNCTION OF HEADING (LARGE SCALE).**

*The contrast-enhanced activity profiles for each of the most active large scale contextual place nodes as a function of heading for location (5',8') (sensory input only).*

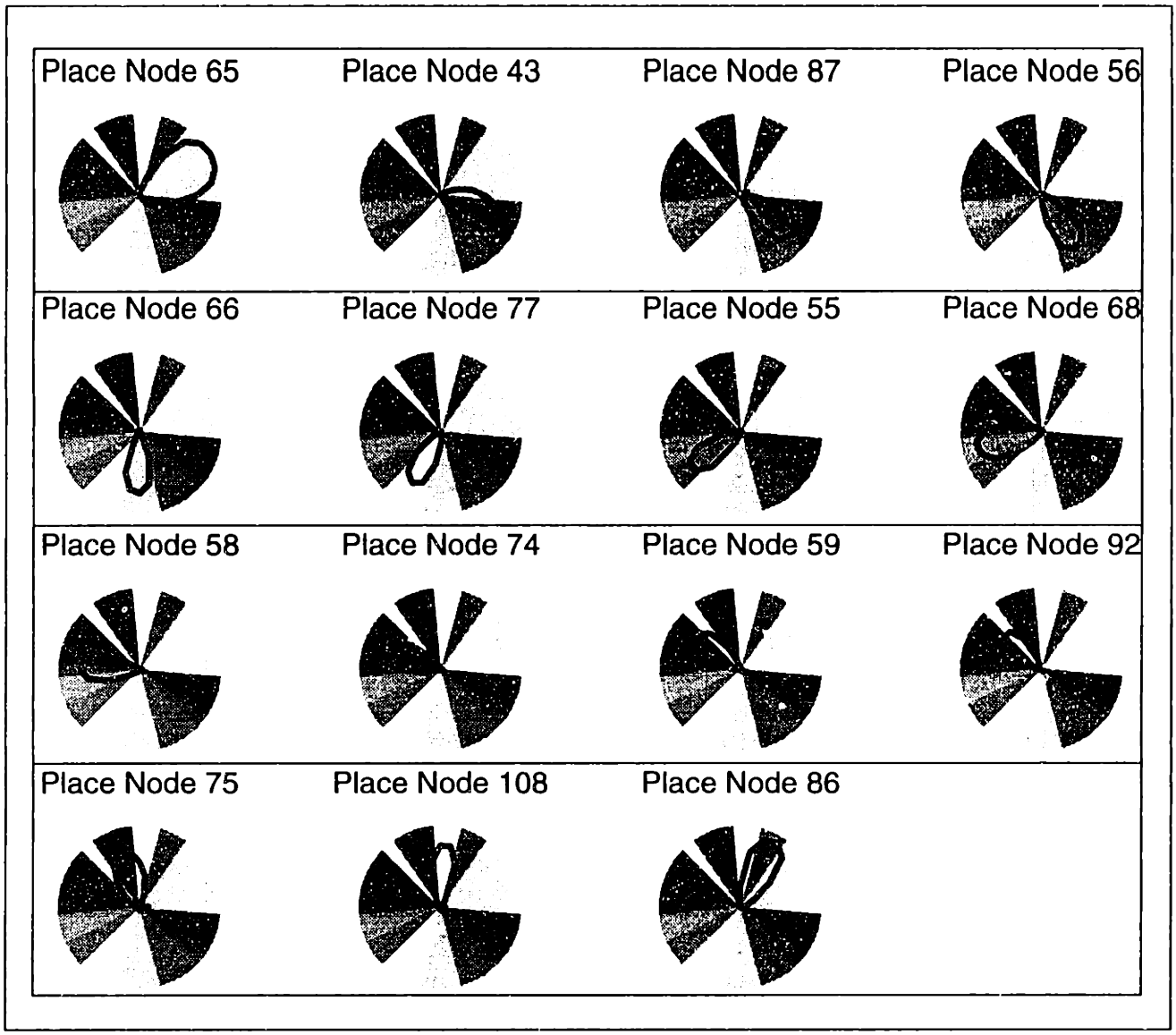


**FIGURE 57: CONTRAST-ENHANCED PLACE NODE ACTIVITY AS A FUNCTION OF HEADING (SMALL SCALE).**

*The contrast-enhanced activity profiles for each of the most active small scale contextual place nodes as a function of heading for location (4',4') (sensory input only).*

### 8.5.3 Prediction Results

Prediction results from two action sequences for each of the two vigilance values are presented in Figures 58-61. Results for the smaller vigilance value (larger scale map) indicate that the trained architecture is capable of roughly predicting the next place region for moderately long action sequences (around 5). It has difficulty with longer sequences, mainly because the fuzziness of the conjunctive pattern (which helps for generalization) diffuses activity through the architecture with time, causing the winning place to become less pronounced. The resulting decrease in confidence can be viewed as a slow increase in uncertainty over time with the absence of visual input.

For two reasons, the results for the larger vigilance value (smaller scale map) are much better in terms of decision-making ability. First of all, the smaller scale is more in accordance with the scale of each of the actions, and is therefore more appropriate for predicting the consequences of a wide variety of actions in an area of the laboratory that is only 7' x 9' in size. Another way of saying this is that the distinguishability of places should be commensurate with the distinguishability of actions. Second, a dynamic thresholding operation was incorporated into the contrast-enhancing operations for the contextual place evidence. That is, the operation

$$p_i = \max(p_i - p_{k(n_w)}, 0) \tag{76}$$

was appended to step 4 in the algorithmic implementation (after equation 42 in Section 8.3.2), where $n_w$ is the minimum number of place nodes needed to define a particular location, and $k(n_w)$ is the index of the contextual place node with the $n_w$th largest activity. (In this particular case, $n_w$ was set to 5.) This addition yields a new *competitive* contrast-enhancement step that, together with the contrast-enhancement operations on the sensory input, approximates the results of a cooperative-competitive shunting network with sigmoidal feedback (see Grossberg, 1988). The overall effect is better discrimination between places during the predictive task.

Though not shown in these diagrams, the prediction does eventually break down beyond a certain number of actions. One reason for this breakdown might simply be the fact that, due to the qualitative nature of the learned map, one cannot expect that predicted place activities precisely match the sensory activity. Place regions come in all shapes and sizes, and cannot be expected to yield the perfect results that one would expect from, say, a radial basis function. Furthermore, a few place nodes have split place fields (as witnessed by some of the activity profiles in Figure 54), which means that the learned transition templates for such nodes can only be accurate for a portion of their place regions. Additional machinery is therefore required to split such nodes when detected so that the architecture can disambiguate between them using history. This splitting operation is outside the scope of this thesis, but is discussed in more detail in Section 11.4.4. In any case, these occurrences, as in rat experiments (see Section 2.2.1), are quite rare.

Despite the apparent breakdown of place node activity prediction beyond a critical number of actions, it is important to note that the failure to predict activities does not necessarily reflect a failure in predicting effective position. Activity results are rather deceiving in this respect, because place cells are not arranged topographically. The robot might actually make predictions that are effectively only a few feet off from its actual course, yet the place activity might look quite different. Even in the event that place activity does reflect large errors in location after sev-

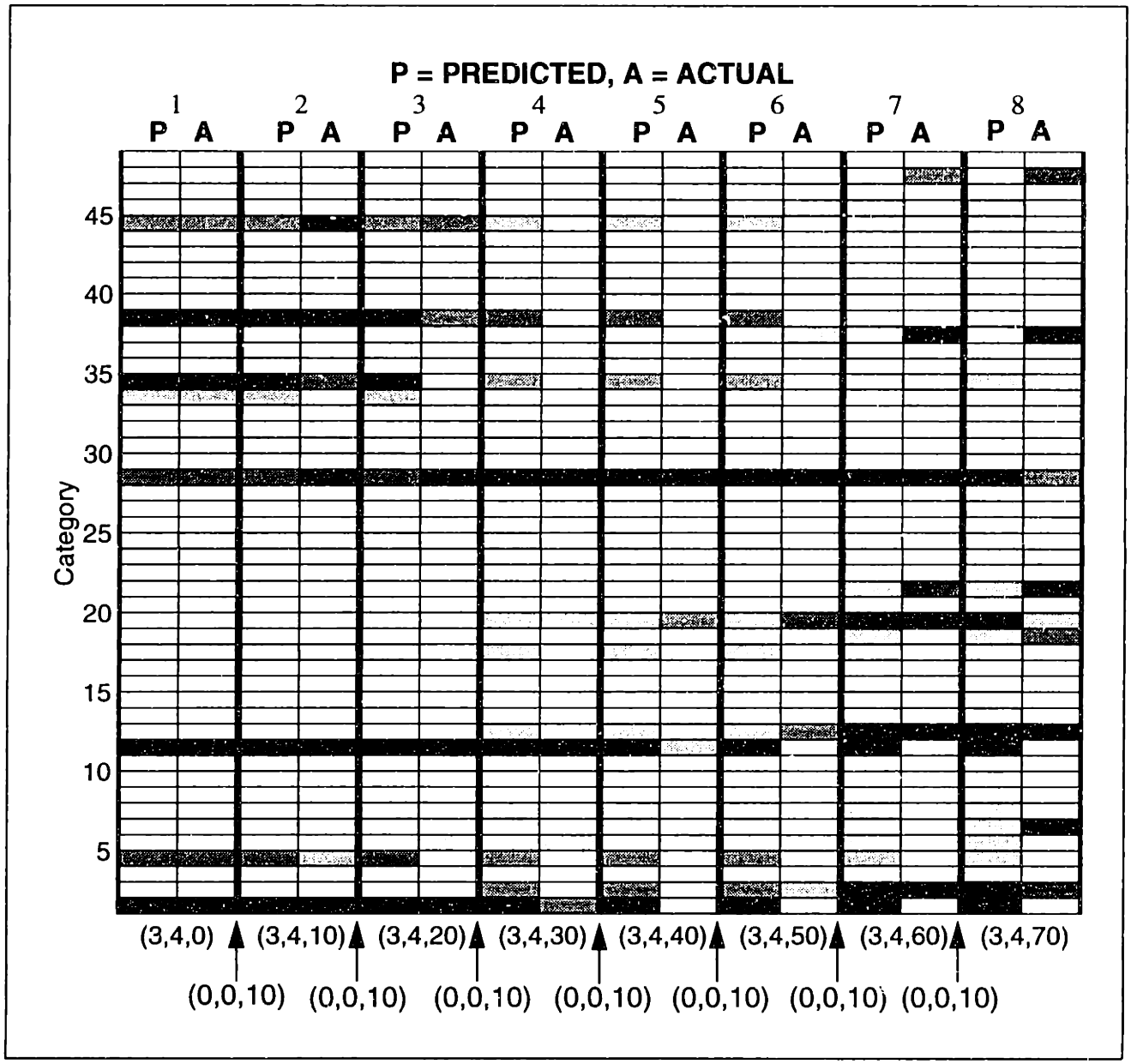


**FIGURE 58: SIMULATION 1 OF LONG-TERM PLACE PREDICTION (LARGE SPATIAL SCALE).**

*Long-term prediction results of simulating a large scale version of the ACLA with real data gathered by MAVIN. The action sequence, in this case effecting a clockwise turn in place in 10° increments, produces a sequence of 8 place predictions comparable to those recognized by the place learning architecture for the corresponding positions. Each double column of this chart corresponds to a position along the resulting path (labelled directly below the column in cartesian coordinates as $B_x$ in feet, $B_y$ in feet, and $B_h$ in degrees), and vertically displays the pattern of contextual place node evidence (P) and the pattern of bottom-up evidence from the PLA (A) side by side, thereby allowing a direct comparison of the predicted activities with those that result from observing landmarks. The actions taken between each of these locations (measured in polar coordinates as $A_\theta^{in}$ in degrees, $A_\rho^{in}$ in feet, and $A_\phi^{in}$ in degrees) are indicated by arrows.*

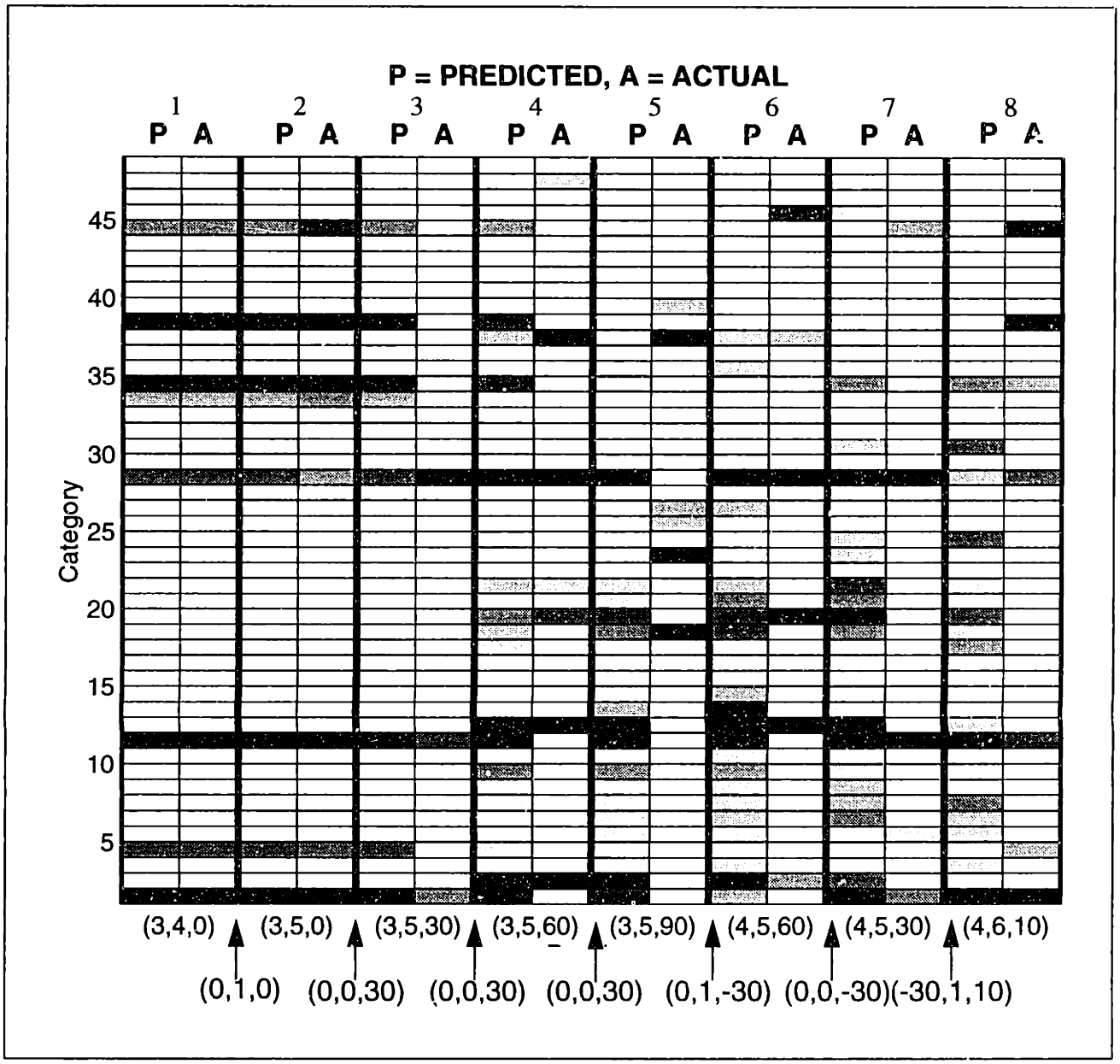


**FIGURE 59: SIMULATION 2 OF LONG-TERM PLACE PREDICTION (LARGE SPATIAL SCALE).**
*Long-term prediction results of simulating a large scale version of the ACLA with real data gathered by MAVIN. The action sequence, in this case a more-or-less random sequence of actions, produces a sequence of 8 place predictions comparable to those recognized by the place learning architecture for the corresponding positions. (See the caption of Figure 58 for a more in depth description of this chart.)*

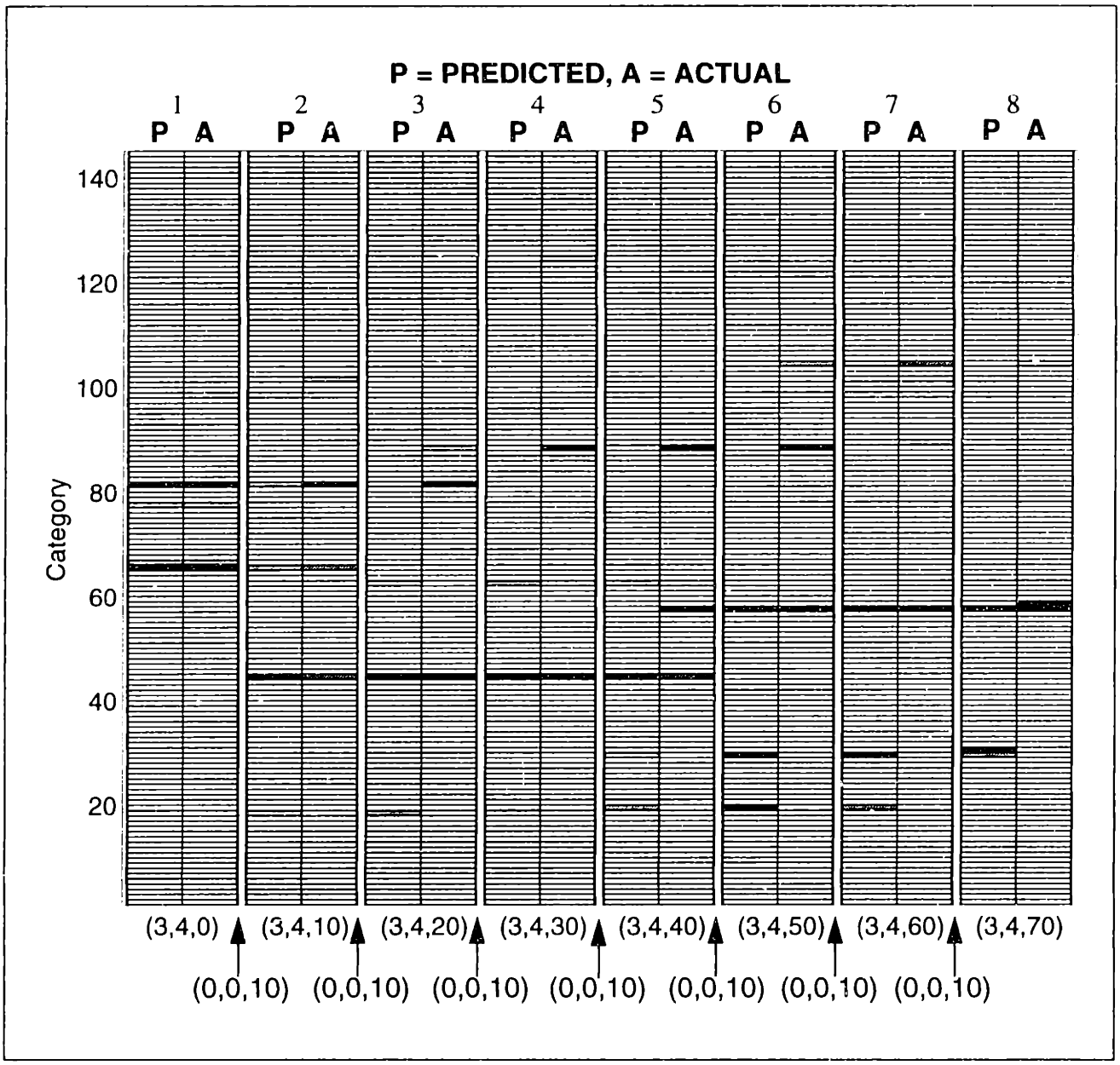


**FIGURE 60: SIMULATION 1 OF LONG-TERM PLACE PREDICTION (SMALL SPATIAL SCALE).**

*Long-term prediction results of simulating a small scale version of the ACLA with real data gathered by MAVIN. The action sequence, in this case clockwise turn in place in 10° increments, produces a sequence of 8 place predictions comparable to those recognized by the place learning architecture for the corresponding positions (cf. Figure 58). (See the caption of Figure 58 for a more in depth description of this chart.)*

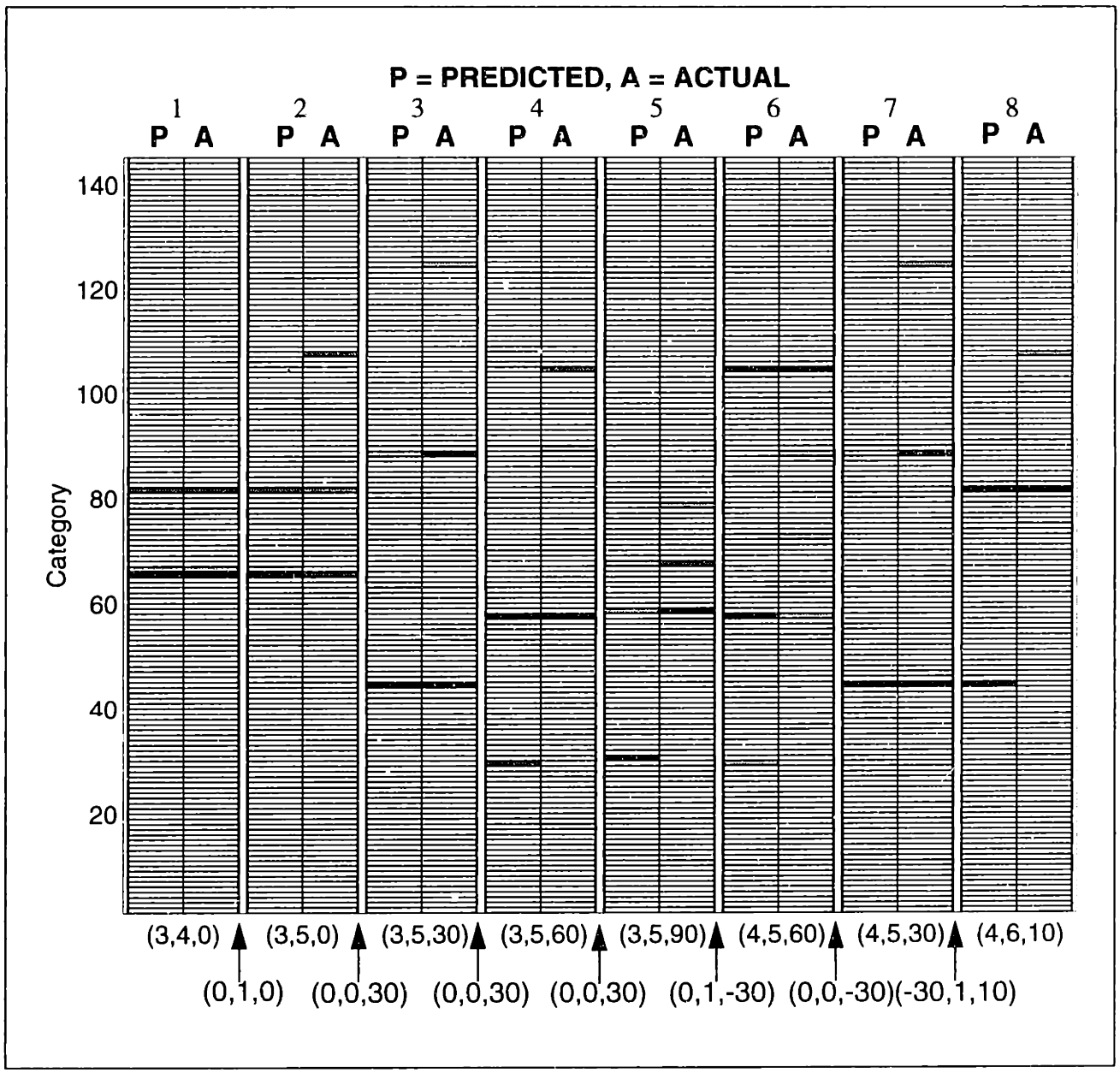


**FIGURE 61: SIMULATION 2 OF LONG-TERM PLACE PREDICTION (SMALL SPATIAL SCALE).**

*Long-term prediction results of simulating a small scale version of the ACLA with real data gathered by MAVIN. The action sequence, in this case a more-or-less random sequence of actions, produces a sequence of 8 place predictions comparable to those recognized by the place learning architecture for the corresponding positions (cf. Figure 59). (See the caption of Figure 58 for a more in depth description of this chart.)*

eral actions, the ability to predict place node activity over even a small number of actions frees the robot from having to search for landmarks after every action. The sensory evidence must be obtained in order to keep the prediction on track only after every few actions, which greatly speeds up the entire navigation process (recall that the time needed for head motions defines the critical time path for place recognition).

## 8.6 Summary

This chapter has conceptually and mathematically described the action consequence learning architecture (ACLA). The ACLA learns the relations between places conditional on coarsely coded locomotive actions. It resembles a localist RNN, but can perform fast learning and learn maps rather than individual sequences. It basically learns a diktiometric map of an environment without resorting to symbolic techniques. In fact, its neural dynamics can be interpreted probabilistically in the framework of a POMDP. Simulation with data gathered using the mobile robot MAVIN has attested to the ability to use the resulting learned map for predicting the consequences of locomotive actions in the absence of sensory input. Note, however, that the performance of the ACLA on such predictive tasks depends greatly on the spatial scale employed. This dependence places even greater importance upon the learning and integration of maps at multiple spatial scales. Furthermore, the problems caused by place nodes with split place regions illuminate a need for extra machinery that splits such nodes when detected. Despite these drawbacks, however, the predictive ability of the architecture effectively speeds up navigation by decreasing the demands placed on the time-consuming landmark search operations required for place recognition.

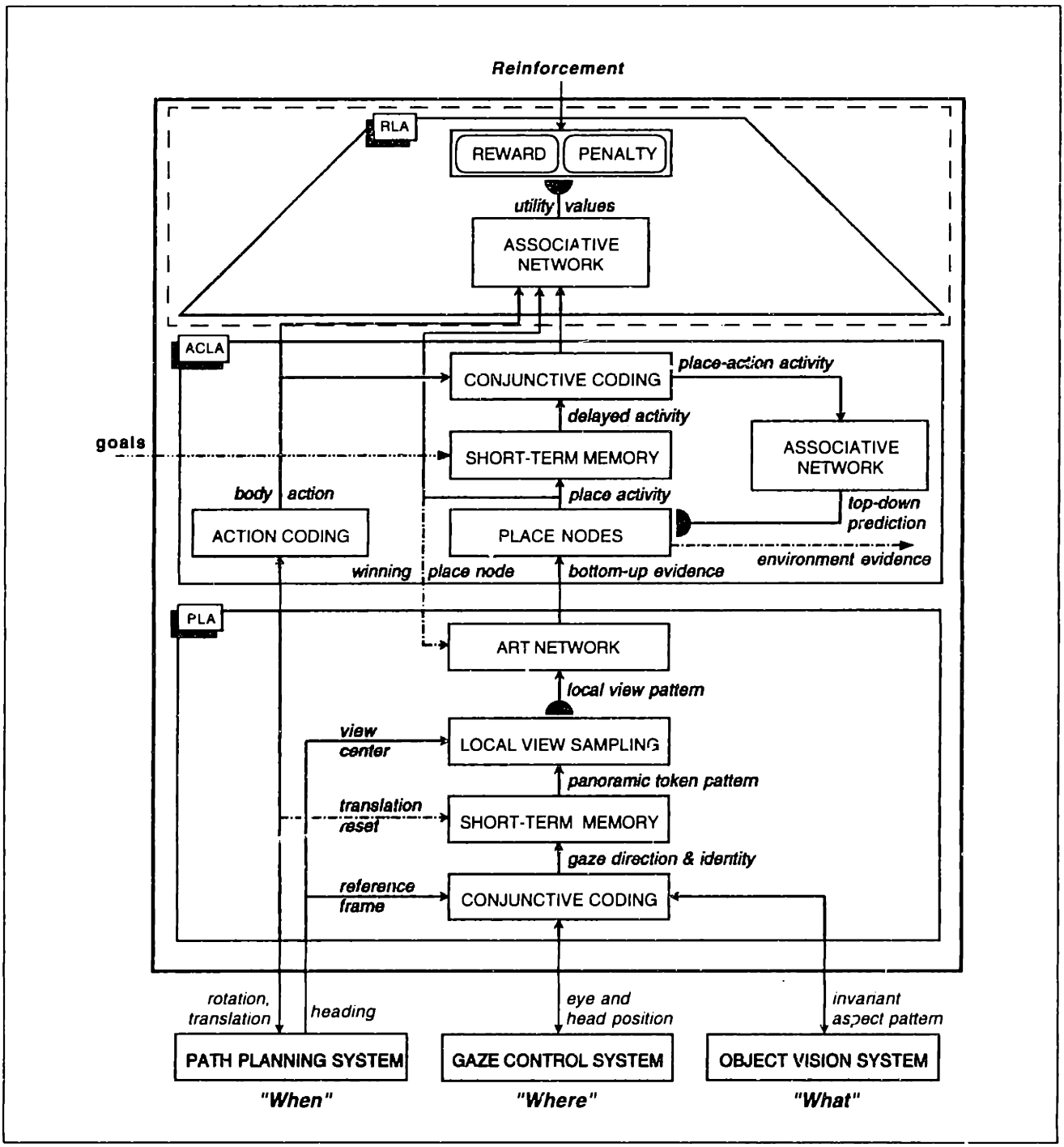


**FIGURE 62: THE REINFORCEMENT LEARNING ARCHITECTURE (RLA).**
*The reinforcement learning architecture, highlighted here as part of the navigation strategy learning sub-system, takes as primary input a reinforcement signal, a set of goals, and the contextual place evidence (coding the "What" and "Where"), action coding node activities (coding for the "When"), and the place action node activities (coding for the conjunction of "What", "Where", and "When") from the ACLA.*

CHAPTER 9

# *Indirect Reinforcement Learning and Route Planning*

The reinforcement learning architecture (RLA), highlighted in Figure 62, emulates a cognitivist account of learning in the rat hippocampus (see Section 2.1.1) by directly learning the relative utility of each of the places and actions learned by the PLA and ACLA architectures described in the previous two chapters.

This chapter first describes the RLA at the architectural level, and qualitatively describes the concept of using the architecture in conjunction with the ACLA to perform route planning and exploration. It then goes on to define the short-term memory and learning dynamics within the RLA, and the planning and exploration dynamics within both the RLA and ACLA. Finally, it describes the results of simulating the architectures together in these capacities using data gathered by the mobile robot MAVIN.

## 9.1 Neurocomputational Architecture

Due to the localist, indirect approach to task learning embodied by constructing a map using the PLA and ACLA (see Sections 4.1.3, 3.2.5, and 3.3.4), the proposed architecture for reinforcement learning is a simple one. Since the PLA and ACLA self-organize diffuse inputs from the environment into meaningful, mutually exclusive categories (e.g. places and actions) represented by single units, the situations that lead to the delivery of reward or punishment can be assessed simply by examining the relative evidence for each of these categories. The main problem is to incrementally learn the correlation between each category and the delivery of rewards and penalties, thereby determining the relative *utility* of each category. This approach allows such utility values to be incrementally adapted as the environment and performance criteria change.

The RLA exploits this simple idea by associatively learning the expected utility (e.g. reward/punishment) of places, actions, and place-action pairings with respect to some navigational performance criteria. The architecture employs yet another localist heteroassociative network, again with a combination of fast and slow learning, to construct direct associations between special

externally controlled "pain" and "pleasure" categories and the internally updated place and action categories (and possibly their conjunctions).

## 9.2 Route Planning and Exploration

As previously mentioned, the ACLA is useful for both route planning and environment exploration. Route planning refers to determining the optimal sequence of motions that brings the robot from its current location to a desired place region, in the sense that it incurs the least penalty and the most reward along the way. In conjunction with the RLA, the ACLA provides a substrate for optimal route planning via a recursive diffusion of activity backwards through its heteroassociative network from the goal place, or even a set of possible goal places (see Section 5.2.2). This diffusion, modulated by the learned utility values propagated backwards through the associative network in the RLA, iteratively computes an evaluation of each of the actions in each of the places with respect to the goal. Planning then becomes a matter of executing the best action at each place, essentially determined by following the gradient.

Exploration entails traversing unvisited parts of the environment with the purpose of expanding the current map, ideally while maximizing utility. Interestingly, environment exploration can be thought of as a special case of route planning. Because the ACLA implicitly records possible avenues for exploration by the place-action category pairs without learned expectation templates, optimal exploration entails planning and following routes to the places with absent templates, then executing the actions corresponding to those missing templates. This technique resembles the use of update graphs by Mozer & Bachrach (1991 — see Section 3.3.4).

## 9.3 Dynamics

### 9.3.1 Memory and Learning

The dynamics for reinforcement learning in the RLA, presented here using the variables described in Table 11, are relatively straightforward. First, the weight templates in the RLA are initialized as follows:

1. The weight templates for all place and place-action nodes are initialized to $\left(\frac{1}{2}, \frac{1}{2}\right)$, corresponding to neither reward nor punishment.
2. The weight templates for action coding nodes are initialized according to the amount of time $\tau_j$ required for the robot to execute the corresponding action $j$ at maximum velocity. That is,

$$w^{j(j_\theta, j_\rho, j_\phi)} = \left[1 - \frac{\tau_{j(j_\theta, j_\rho, j_\phi)}}{\tau_{max}}, \frac{\tau_{j(j_\theta, j_\rho, j_\phi)}}{\tau_{max}}\right] \tag{77}$$

where

$$\tau_{j(j_\theta, j_\rho, j_\phi)} = \max\left(\frac{\left|c^{\rho}_{j_\rho}\right|}{\Gamma} + \frac{\left|c^{\theta}_{j_\theta}\right| + \left|c^{\phi}_{j_\phi}\right|}{\omega}, \Delta\tau\right), \tag{78}$$

TABLE 11: REINFORCEMENT LEARNING ARCHITECTURE (RLA) VARIABLE DEFINITIONS.

| Variable | Definition |
|---|---|
| $R$ | The reinforcement signal ("reward", "penalty", or "none"). |
| $\bar{g}$ | A list of goal places. |
| $\bar{r}^r = \left(r_r^r, r_p^r\right)$ | The 1D pattern (2-dimensional vector) representing the activities of the reinforcement coding nodes (reward and penalty nodes). |

$$\tau_{max} = \frac{A_\rho^{max}}{\Gamma} + \frac{2\pi}{\omega}, \tag{79}$$

$\Gamma$ and $\omega$ are the maximum driving and turning velocities of the robot (Appendix A contains the values of these parameters for MAVIN), and $\Delta\tau$ is the action time interval for the ACLA architecture. This initialization results in the initial weight distribution shown in Figure 63.

Following this initialization, the RLA receives inputs from the place nodes $\bar{p}$, action coding nodes $\bar{a}$, and place-action nodes $\boldsymbol{D}$ in the ACLA. It also receives the reinforcement signal $R$, and the goal list $\bar{g}$. It maintains a short-term memory of the recent reinforcement (reward) via the equation

$$\frac{dr_r^r}{dt} = \eta \cdot \begin{cases} 1 - r_r^r \text{ if } R = \text{reward} \\ 0 - r_r^r \text{ if } R = \text{penalty} \\ \frac{1}{2} - r_r^r \text{ otherwise.} \end{cases} \tag{80}$$

It then forms the penalty dimension of the reinforcement coding nodes via complement coding, i.e.

$$r_p^r = 1 - r_r^r \tag{81}$$

where $\eta$ is the time constant for the memory. Finally, it presents the vector $\text{cat}(\bar{p}, \bar{a}, \text{cat}(\boldsymbol{D}))$ as input to the associative network, along with the vector $\bar{r}^r$ as the desired output (see Appendix D).

## 9.3.2 Route Planning and Exploration

Assuming that the utility values for each of the places and actions in the map have been learned sufficiently for the desired task according to the dynamics presented above, they can then be used to influence route planning computations in the ACLA. To perform route planning with respect to the goal $\bar{g}$, the following initialization steps are first taken:

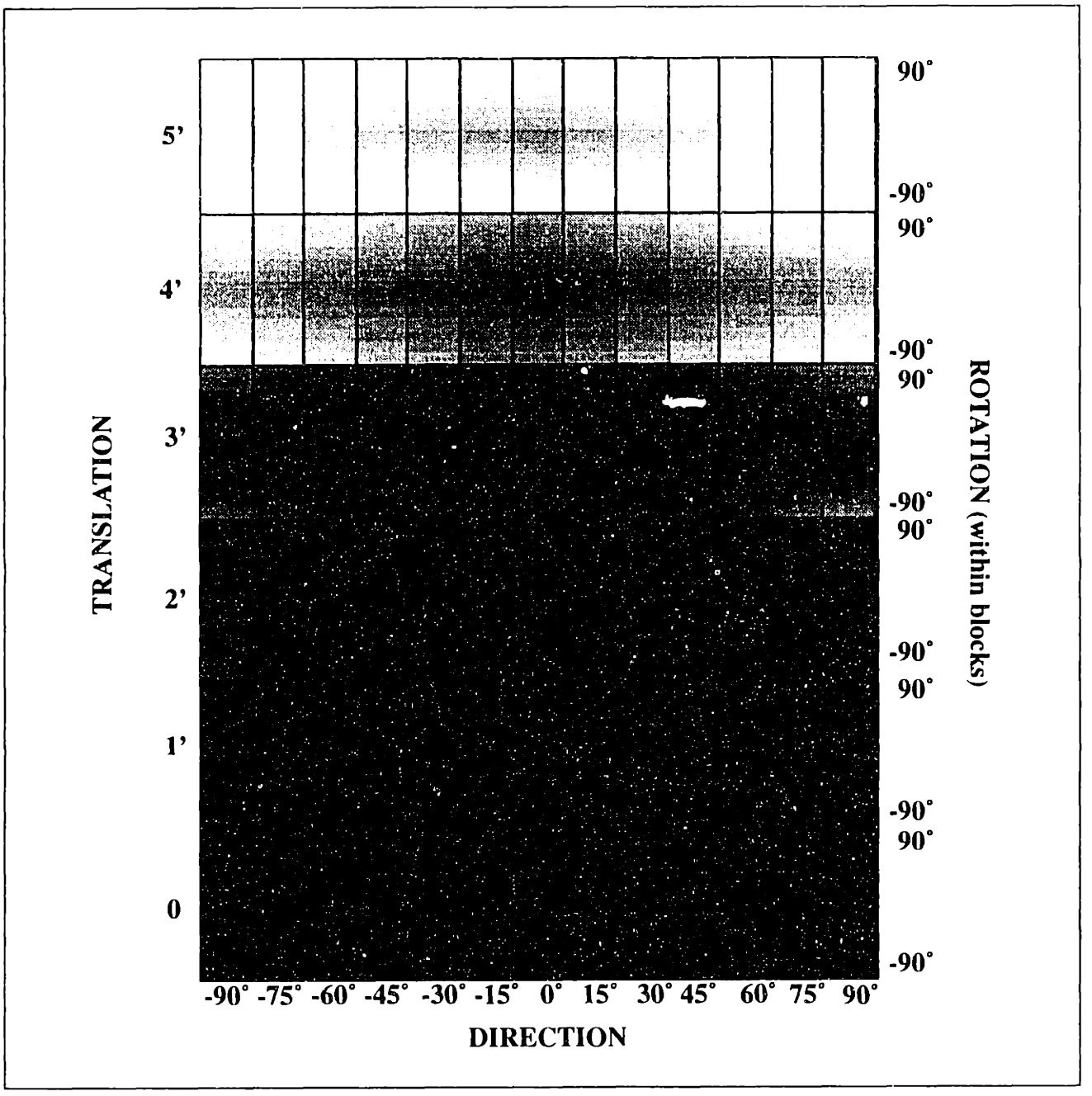


**FIGURE 63: INITIAL UTILITY WEIGHT DISTRIBUTION FOR ACTION CODING NODES.**
*Each block corresponds to a given direction (horizontal axis) and translation (vertical axis). Within each box is plotted the "cost" (penalty) portion of the weight templates for each rotation (vertical axis). The brightness of each bar is proportional to the corresponding weight. (Note: the "reward" portions of each of the templates is simply the inverse of the "cost" portion.)*

1. The reward and punishment nodes in the RLA are set to 0 and 1 respectively (i.e. $\bar{r}^r = (0, 1)$ ).
2. The contextual place nodes $\bar{p}$ in the ACLA are initialized to zero.
3. For planning, each of the unlearned templates in the RLA are temporarily initialized to $(0, 1)$, reflecting minimum utility for unexplored portions of the environment. For exploration, they are temporarily initialized to $(1, 0)$.
4. For exploration, each of the elements of the completion vector $h$ in the ACLA associative network (see Appendix D) is temporarily set to 1.

The contextual place nodes then become the source for a "diffusion" of activity backwards through the ACLA, modulated by learned utility values from the RLA. During this diffusion, the transition synapses and contextual place nodes become conductive and capacitive mediums respectively. One iteration of the diffusion process progresses as follows:

1. The contextual activity $\bar{p}$ is propagated backwards through the associative network in the ACLA (see Appendix D), yielding an input distribution $\partial$. Simultaneously, activity is propagated backwards through the associative network in the RLA, yielding a vector $\text{cat}(\bar{u}^p, \bar{u}^a, \bar{u}^d)$ that effectively defines the utility values for places, actions, and actions performed in places, given by $\bar{u}^p$, $\bar{u}^a$, and $\bar{u}^d$, respectively (more precisely, they correspond to the inverse utility, or cost).
2. $\partial$, $\bar{u}^d$, and $\bar{u}^a$ are combined in order to back-activate the place-action nodes $D$ via the equations

$$\boldsymbol{Q} = \begin{bmatrix} [\bar{q}^1]^T \\ [\bar{q}^2]^T \\ \cdots \\ [\bar{q}^{n_p}]^T \end{bmatrix} = \text{invcat}\,(\bar{u}^d + \partial, n_p, n_\theta \cdot n_\rho \cdot n_\phi) \tag{82}$$

$$\bar{q}^i \Leftarrow \bar{q}^i + \bar{u}^a \tag{83}$$

$$D_{ij\,(j_\theta, j_\rho, j_\phi)} = \bar{a}\,(j_\theta, j_\rho, j_\phi) \cdot \bar{q}^i . \tag{84}$$

Here, $\bar{a}\,(j_\theta, j_\rho, j_\phi)$ represents a diffusion of activity through the action nodes, determined using equations (37) - (40) with the action $\bar{A}^{\text{in}} = \bar{A}\,(j_\theta, j_\rho, j_\phi)$, where

$$\bar{A}\,(j_\theta, j_\rho, j_\phi) = \left( c^\theta_{j_\theta}, c^\rho_{j_\rho}, c^\phi_{j_\phi} \right) \tag{85}$$

and

$$c^\theta_i = \text{FC}_p(i; n_\theta, -\pi, \pi) \tag{86}$$

$$c^\rho_i = \text{FC}_f(i; n_\rho, 0, A^{\max}_\rho) \tag{87}$$

$$c^\phi_i = \text{FC}_p(i; n_\phi, -\pi, \pi) . \tag{88}$$

Note that $\bar{a}(j_\theta, j_\rho, j_\phi)$ can be computed off-line for all values of $j_\theta$, $j_\rho$, and $j_\phi$.

3. The activity of the place-action nodes $\boldsymbol{D}$ back-activates the short-term memory nodes $\bar{m}$ via the equation

$$m_i = \min_{\substack{j = 1 \ldots (n_\theta \cdot n_\rho \cdot n_\phi) \\ h_{k(i,j)} = 1}} (D_{ij}) \tag{89}$$

That is, the place-action nodes with learned templates compete across actions.

4. The short-term memory is combined with $\bar{u}^p$ in order to determine the place node activity, given by

$$\bar{p} = \bar{u}^p + \bar{m}. \tag{90}$$

5. The activities of the contextual place nodes corresponding to the goal places $\bar{g}$ are reset to zero, i.e.

$$p_{g_1} = p_{g_2} = \ldots = p_{g_{n_g}} = 0. \tag{91}$$

This diffusion process terminates when the percentage change in each of the elements of $\bar{p}$ is less than some threshold $\varepsilon$. Note that it might actually be interleaved with the forward (predictive) operation of the ACLA and RLA, provided that the short-term memory $\bar{m}$ can be stored and restored between diffusion and prediction episodes.

Once the diffusion has terminated, the appropriate action for any given distribution of place activity $\bar{p}$ is determined via hill climbing within the network. That is, the next action from any given place is determined by selecting the action $j$ for which the place activation achieves the minimum value via the equation

$$j\left(j_\theta^{\min}, j_\rho^{\min}, j_\phi^{\min}\right) = \operatorname{argmin}_{\substack{j_\theta = 1 \ldots n_\theta \\ j_\rho = 1 \ldots n_\rho \\ j_\phi = 1 \ldots n_\phi}} a_{j(j_\theta, j_\rho, j_\phi)} \tag{92}$$

where

$$a_j = \frac{\sum_{i=1}^{n_p} h_{k(i,j)} p_i D_{ij}}{\sum_{i=1}^{n_p} h_{k(i,j)} p_i}. \tag{93}$$

The next action is then determined by $\bar{A}^{\text{out}} = \bar{A}(j_\theta^{\min}, j_\rho^{\min}, j_\phi^{\min})$.

## 9.4 Probabilistic Interpretation

Evaluation of each of the places and actions allows one to resume the comparison initiated in the previous chapter (see Section 8.4), namely that a learned strategy (consisting of places, locomotive actions, a normalized expectation template for each place-action pair, and reward/penalty

associations with places and actions) resembles a POMDP (states, actions, a conditional probability distribution for each state-action pair, and a payoff function over states and actions). With the addition of reinforcement learning, planning by diffusion resembles a "soft" form of optimal policy determination for a POMDP (Singh, 1994). The contextual place node activities code for the maximum expected future reward received after entering each of the places, and diffusion takes the place of iteratively solving for the value function. The main difference is that equation (84) modifies the "Q" values in order to take the uncertain nature of actions into account. That is, $a_k(j_\theta, j_\rho, j_\phi)$ might be interpreted as the probability of "accidentally" executing action $k$ while trying to execute action $j(j_\theta, j_\rho, j_\phi)$. Under this interpretation, the weighted summation over $\bar{a}(j_\theta, j_\rho, j_\phi)$ computes the future expected reward for *trying* to execute action $j(j_\theta, j_\rho, j_\phi)$.

Of course, this resemblance illuminates a possible difficulty, namely that policy determination requires exponential time with respect to the number of states (places). However, as pointed out in Section 4.1.3, policies need not be completely optimal in practice, and useful satisficing policies generally require very few iterations, particularly for topological structures (which have sparse transition probabilities — $O(n_p)$ rather than $O(n_p^2)$, where $n_p$ is the number of places), and especially when the optimal achievement of goal states requires very few transitions. Furthermore, hierarchical planning principals should apply to navigational planning over long distances in real-time, despite the sizes of learned maps (Davis, 1986; Yeap, 1988; Levitt & Lawton, 1990; see Section 3.4). At the very bottom of the navigation hierarchy are the computations associated with local path (motion) planning for obstacle avoidance (for a review, see Hwang & Ahuja, 1992; McDermott, 1992), performed using the path planning sub-system (PPS — see Section 6.4). The next scale might consist of heading tuned places and their action transitions, while the larger scales might be constructed either by learning environments at multiple spatial and temporal scales (by varying the vigilance and action integration times, respectively), or by pooling the place categories hierarchically and learning simple relations between them. For instance, heading invariant categories might be created by pooling all heading tuned place categories with purely rotational transitions between them (more on this idea in Section 11.4.5).

**TABLE 12: RLA PARAMETER SETTINGS.**

| Parameter | η | ε | β |
|---|---|---|---|
| Value | 1 | 1% | 0.1 |

## 9.5 Route Planning on MAVIN

Initial experiments with the RLA did not verify the learning algorithm, since it requires integrating the entire system. Learning experiments are therefore presented in the next chapter. Instead, planning was performed by MAVIN using the initialized weights in the RLA in conjunction with the ACLA trained in Section 8.5. The parameters for these planning operations are shown in Table 12. Note that in practice one would not normally specify a set of places as goals to the robot, since such places are actually only internally meaningful. Instead, one would perhaps sup-

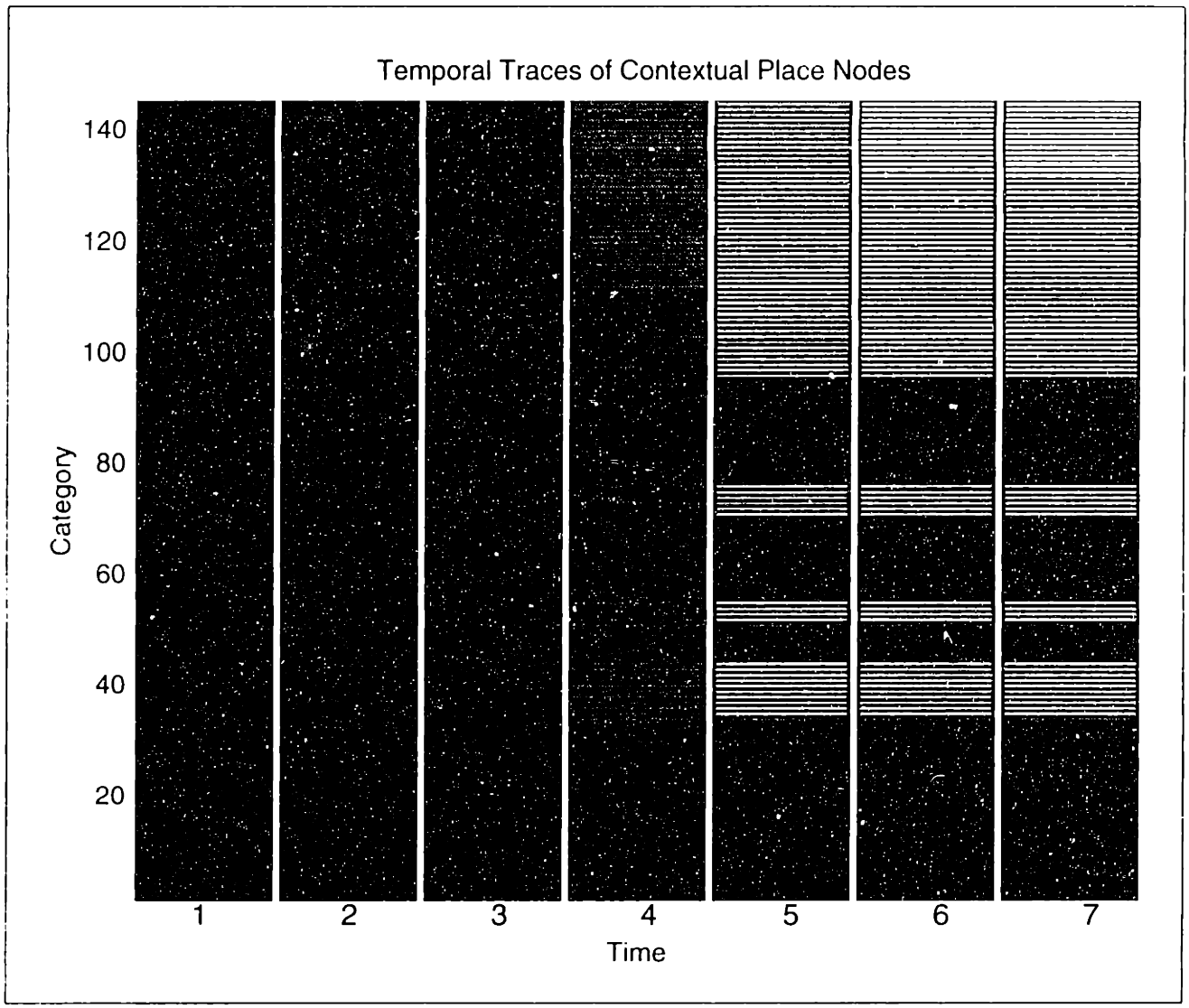


**FIGURE 64: DIFFUSION OF ACTIVITY OVER PLACE NODES (NO TRAINING, GOAL = PLACE #1).**
*The activity of each of the place nodes as a function of time during the diffusion process. The brightness of each element is proportional to the corresponding activity level.*

ply the name of an object in the environment, which through learning would be associated internally with a set of places that could then serve as goals. However, such secondary association is beyond the scope of this thesis.

Two sets of experiments, corresponding to two possible goals, were conducted on the mobile robot MAVIN. In the first set of experiments, the ACLA was told by the operator that the goal was to reach place node 1, whose place activity profile peaks at position (0',0',0°). For this goal, the diffusion process terminated after 7 iterations, each iteration taking approximately 10 seconds. (Note: the time required to perform the diffusion is, unfortunately, impeded on a serial machine by the sparse representation used to store weights in the ACLA, which is necessary in order to keep memory requirements reasonable.) The activities of each of the place nodes as a function of time for this process are shown in Figure 64. The final activity of each of the place nodes is also

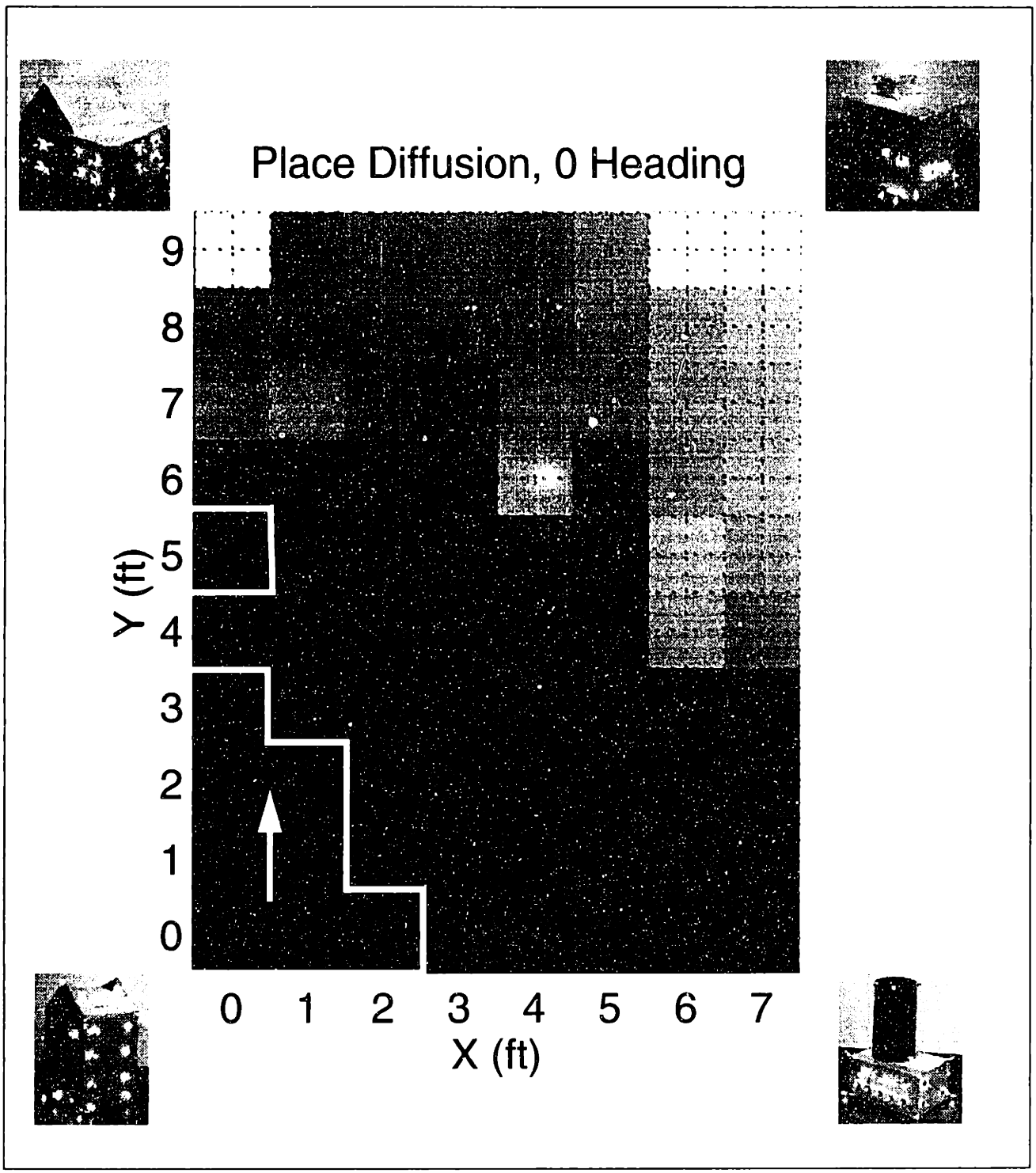


**FIGURE 65: FINAL PLACE ACTIVITY AFTER DIFFUSION AS A FUNCTION OF LOCATION (NO TRAINING, GOAL = PLACE #1).**

*Each location in the environment is colored according to the diffusion activity of the PLA place node with the most sensory evidence for that location (brightness is proportional to activity level). The goal region and preferred heading is indicated in white. (Note: a more accurate measure of utility for any given location would be a weighted average of the diffusion activity according to the contrast-enhanced evidence for each of the places.)*

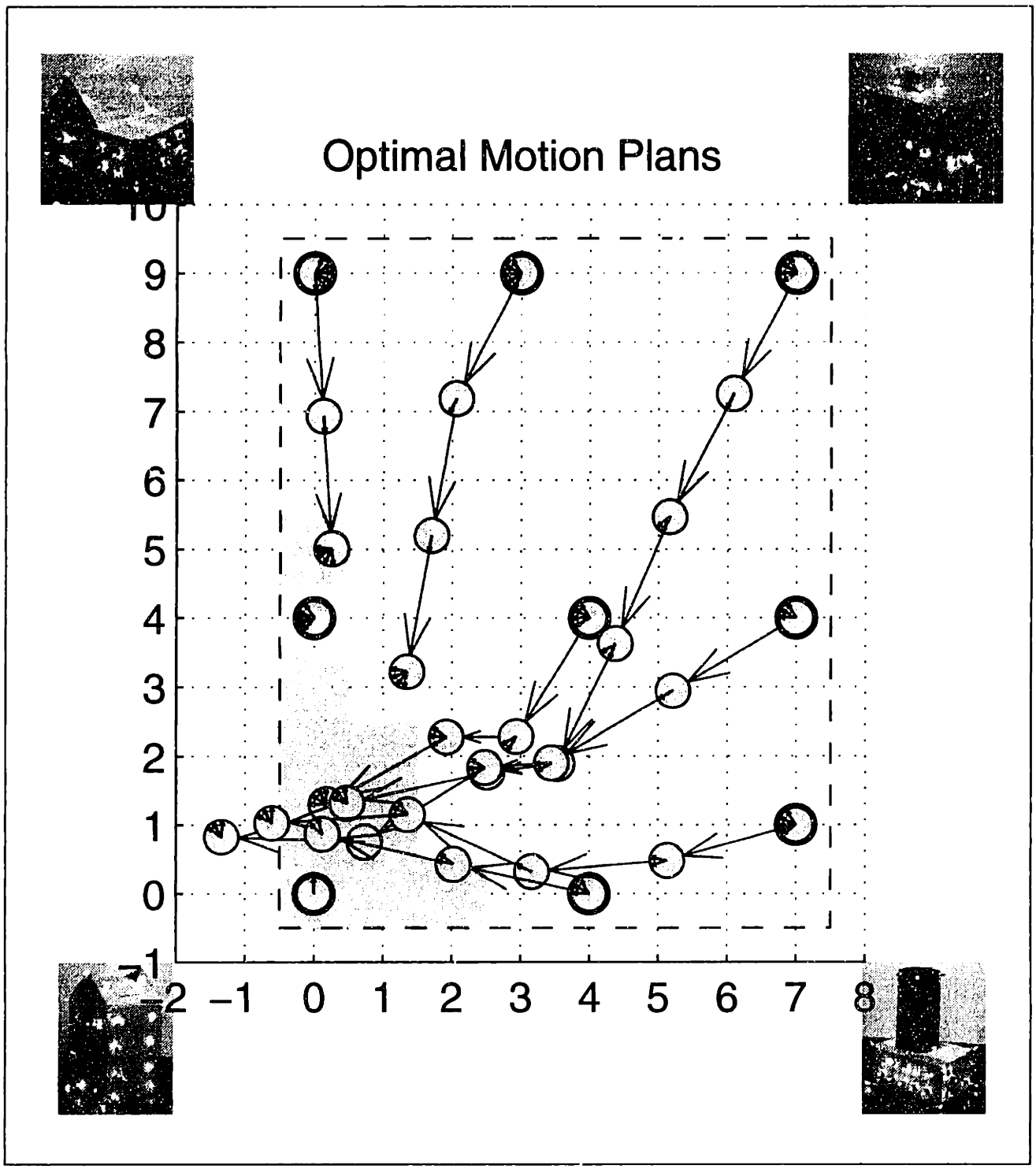


**FIGURE 66: ROUTE PLANS EXECUTED BY MAVIN (NO TRAINING, GOAL = PLACE #1).**
*Routes navigated by MAVIN from various starting points in the laboratory. The goal region is indicated by the gray area in the lower left-hand corner of the environment. Circles with radial lines indicate the positions and headings of the robot at points along each route. Black rings indicate the starting positions.*

plotted as a function of location in Figure 65. Note that the activity rises as a function of distance from the goal.

Following termination of the diffusion process, MAVIN was instructed to navigate to the goal from various starting locations in the laboratory. The routes actually followed by MAVIN from these locations are shown in Figure 65. At every 3rd location along each route, MAVIN scanned the visual panorama for landmarks in order to provide sensory evidence for places via the PLA. At the intermediate locations, place evidence was supplied by long-term prediction operations of the ACLA.

A number of observations can be made with respect to this first set of results. First, note that the paths taken to the goal are not always the most direct routes, mainly because the relations between places are qualitative rather than precise (place activity profiles come in all shapes and sizes), but also due to the fact that split place regions exist (see discussion in Section 8.5.3). Both split and elongated regions cause the learned transitions of corresponding place nodes to be inaccurate for a substantial portion of the environment. For the most part, this inaccuracy is not a severe problem, since planning is determined by weighing the diffusion activity by the pattern of place evidence. Since place evidence profiles overlap in space, this weighting sufficiently disambiguates different locations within split place regions. However, the contribution of inaccurate transitions to this weighting often causes the plan to deviate significantly from what would otherwise be a more direct route.

A related problem is that the robot often travels almost all the way to the goal region, but then starts rotating back and forth ad infinitum. This behavior occurs because the planned action differs greatly for locations near the goal, depending upon which side of a place decision boundary the robot sits. Sometimes, the planned action for the left side of a heading boundary is a turn to the right, while the planned action for the right side is a turn to the left, resulting in oscillatory behavior. It may be possible to alleviate this problem by executing a random action when such an oscillation is detected (e.g. a forward movement). But random actions might cause the robot to move into unexplored or dangerous territory. A better solution is simply to plan routes at smaller spatial scales when the goal is close at hand. Intuitively, more precise actions are necessary for approaching the goal, as opposed to simply heading generally towards it from afar. This idea also fits in nicely with hierarchical planning (see Section 3.4). Therefore, the need for multiple spatial scales arises once again. In this particular case, it would probably be best to hand the planning over to the path planning system when the robot moves within a small distance from the goal (in this case, only a few feet!), since planning then becomes a matter of moving to a visible location.

Finally, notice that the robot actually moves outside the explored portion of the environment a few times on its way to the goal, which is itself in the corner of the explored territory. In this case, such behavior poses little problem, since the local views just outside the explored area are similar enough to those experienced within the area. However, the further the robot strays from this area, the more probable it will become "lost" and have to find its way to the goal by retracing its steps back into the explored area (by following the learned transitions in the backwards direction).

In the second set of experiments, the goal was changed to place node 68 in order to test the planning process for a place more-or-less in the center of the environment (see Figure 36 for the activ-

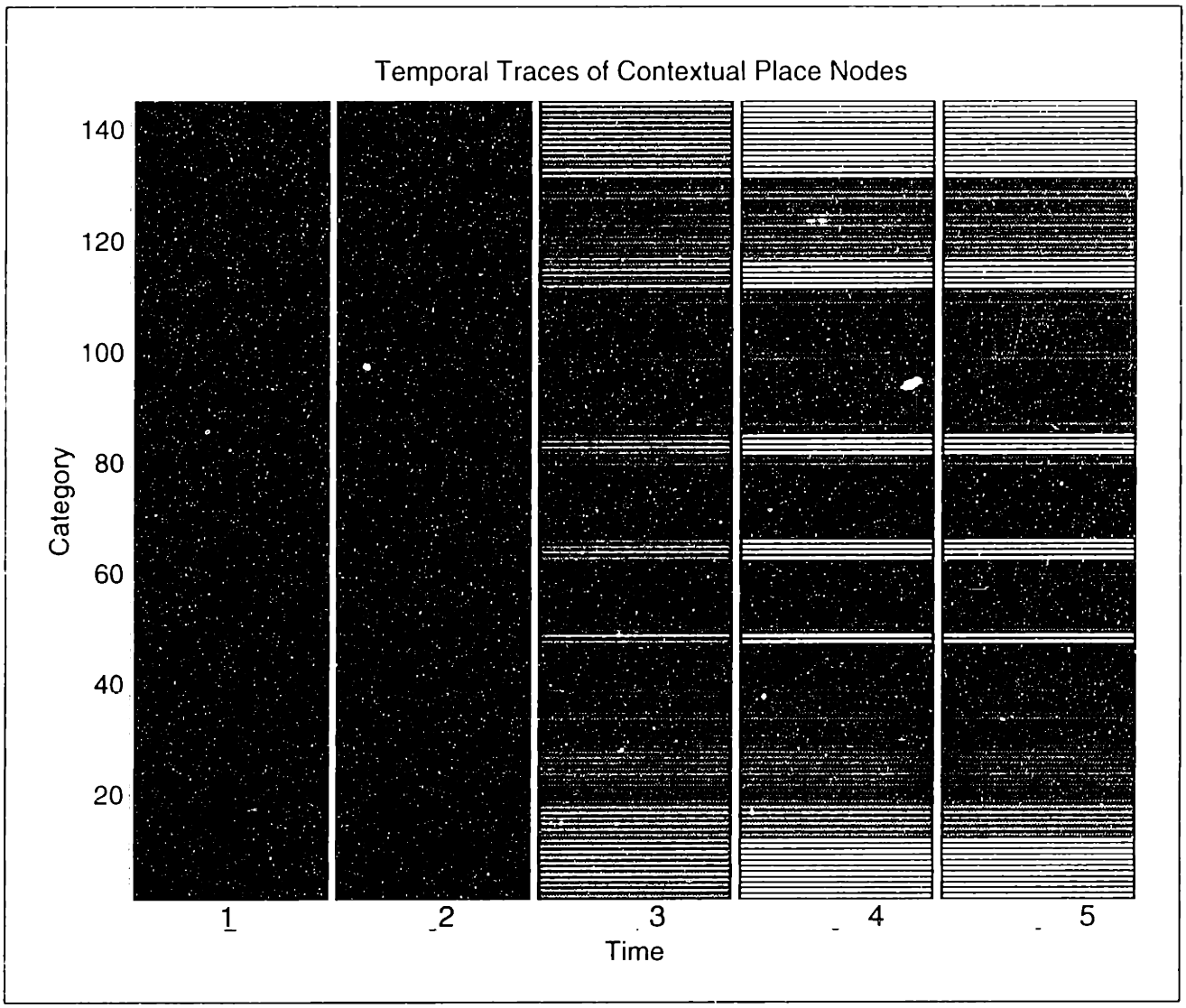


**FIGURE 67: DIFFUSION OF ACTIVITY OVER PLACE NODES (NO TRAINING, GOAL = PLACE #68).**
*The activity of each of the place nodes as a function of time during the diffusion process. The brightness of each element is proportional to the corresponding activity level.*

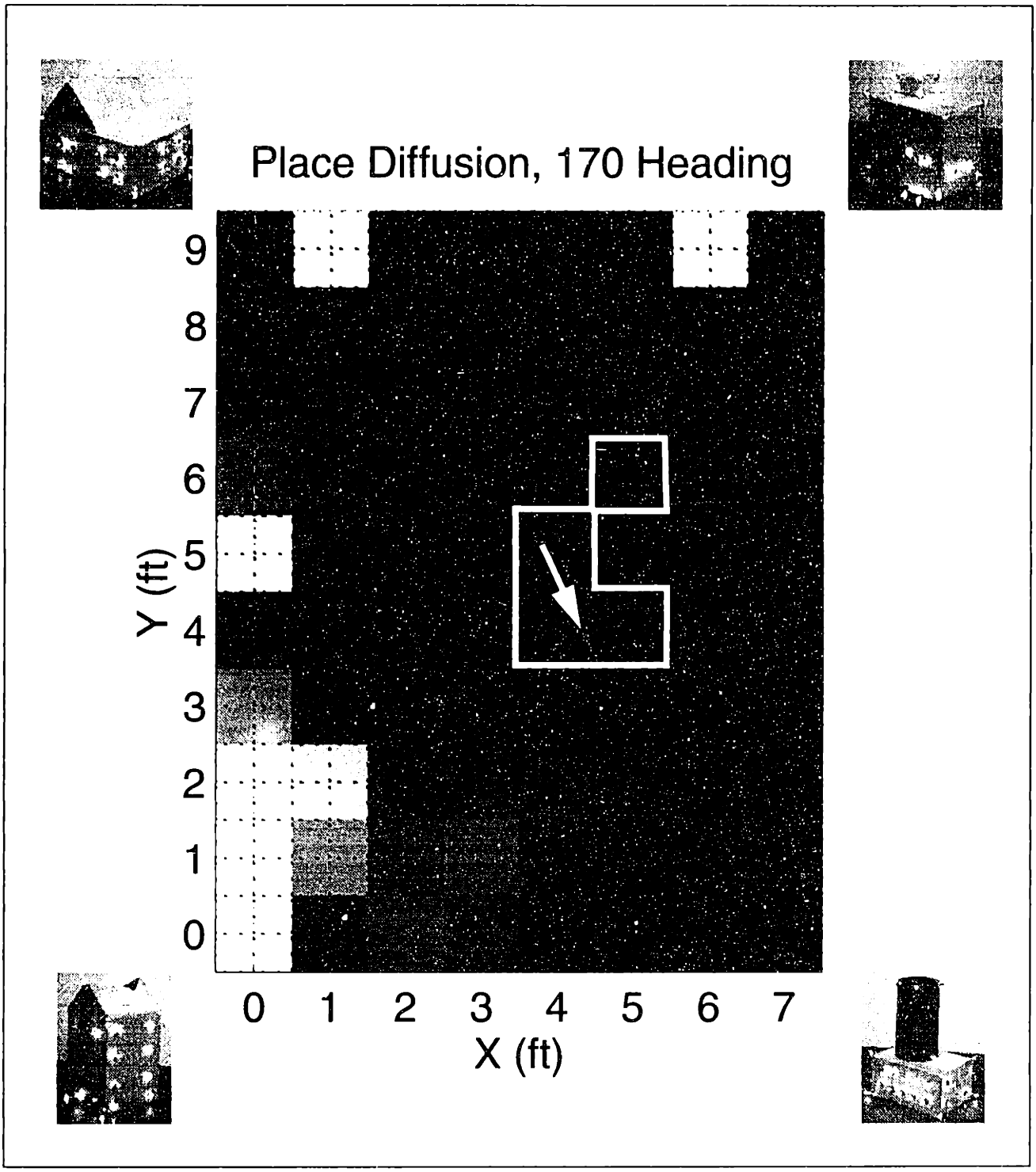


**FIGURE 68: FINAL PLACE ACTIVITY AFTER DIFFUSION AS A FUNCTION OF LOCATION (NO TRAINING, GOAL = PLACE #68).**

*Each location in the environment is colored according to the diffusion activity of the PLA place node with the most sensory evidence for that location (brightness is proportional to activity level). The goal region and preferred heading is indicated in white.*

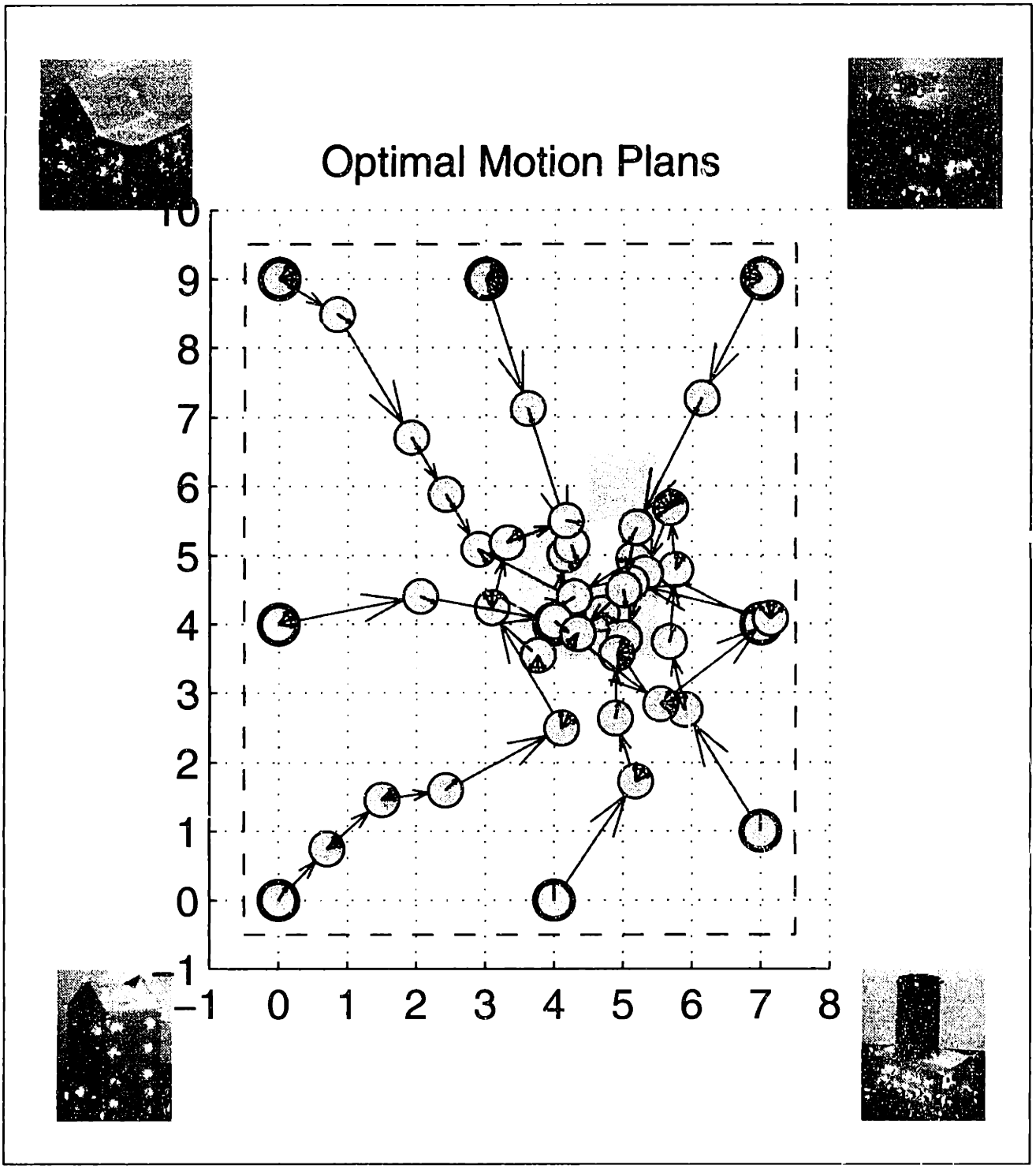


**FIGURE 69: ROUTE PLANS EXECUTED BY MAVIN (NO TRAINING, GOAL = PLACE #68).**
*Routes navigated by MAVIN from various starting points in the laboratory. The goal region is indicated by the gray area in the center of the environment. Circles with radial lines indicate the positions and headings of the robot at points along each route. Black rings indicate the starting positions.*

ity profile of place node 68). For this goal, the diffusion process terminated after 5 iterations. Once again, the activities of each of the place nodes as a function of time are shown in Figure 67. Note the difference between this diffusion and that shown in Figure 64. Also note the difference between the final activities as a function of location for this goal, shown in Figure 68, and those for the previous goal in Figure 65. Once again, the routes actually followed by MAVIN from each of the starting positions to the goals are shown in Figure 69.

The same sort of observations that were made with respect to the first set of planning results can be made with respect to the second. In addition, it is not difficult to see that the routes taken by the robot actually circle the goal once the robot comes within a certain range. This strange behavior is again most likely a consequence of trying to use a map at a spatial scale that is much too large with respect to the final action needed to obtain the goal region. The robot constantly overshoots the region, and has to turn around and try again. As already mentioned, path planning operations are much more appropriate for the level of precision required at this stage of the planning process.

## 9.7 Summary

This chapter has conceptually and mathematically described the reinforcement learning architecture (RLA). Due to the simplicity of indirect methods for reinforcement learning, the architecture consists entirely of a single associative network, and a short-term memory for reward and punishment. The addition of the RLA, as the map evaluation component, to the map-making components embodied by the PLA and ACLA completes the navigation strategy learning sub-system (NSLS), which is capable of place prediction, environment recognition, route planning, and exploration, and resembles a POMDP for a visual environment. This resemblance sheds some light on the computational advantages of implementing the sub-system at multiple spatial scales, and employing hierarchical planning techniques.

The chapter also reported some results of testing the route planning operations of the RLA and ACLA architectures on the mobile robot MAVIN. The diffusion process required to determine the optimal action policy for each of two different goals terminated after a handful of iterations, each iteration taking on the order of 10 seconds. The actions that MAVIN actually executed on its way to each of the goals deviated slightly from the optimal paths (in a Euclidean sense) primarily due to the qualitative nature of the maps, but also due to the presence of a few split place regions. In areas near the goal, the robot's actions often became erratic, sometimes oscillating in place or circling the goal. These results further attest to the need for learning and using maps at multiple spatial scales, and the employment of hierarchical planning techniques.

The next chapter presents a framework in which each of the sub-systems described in this and previous chapters are integrated in the context of teletraining.

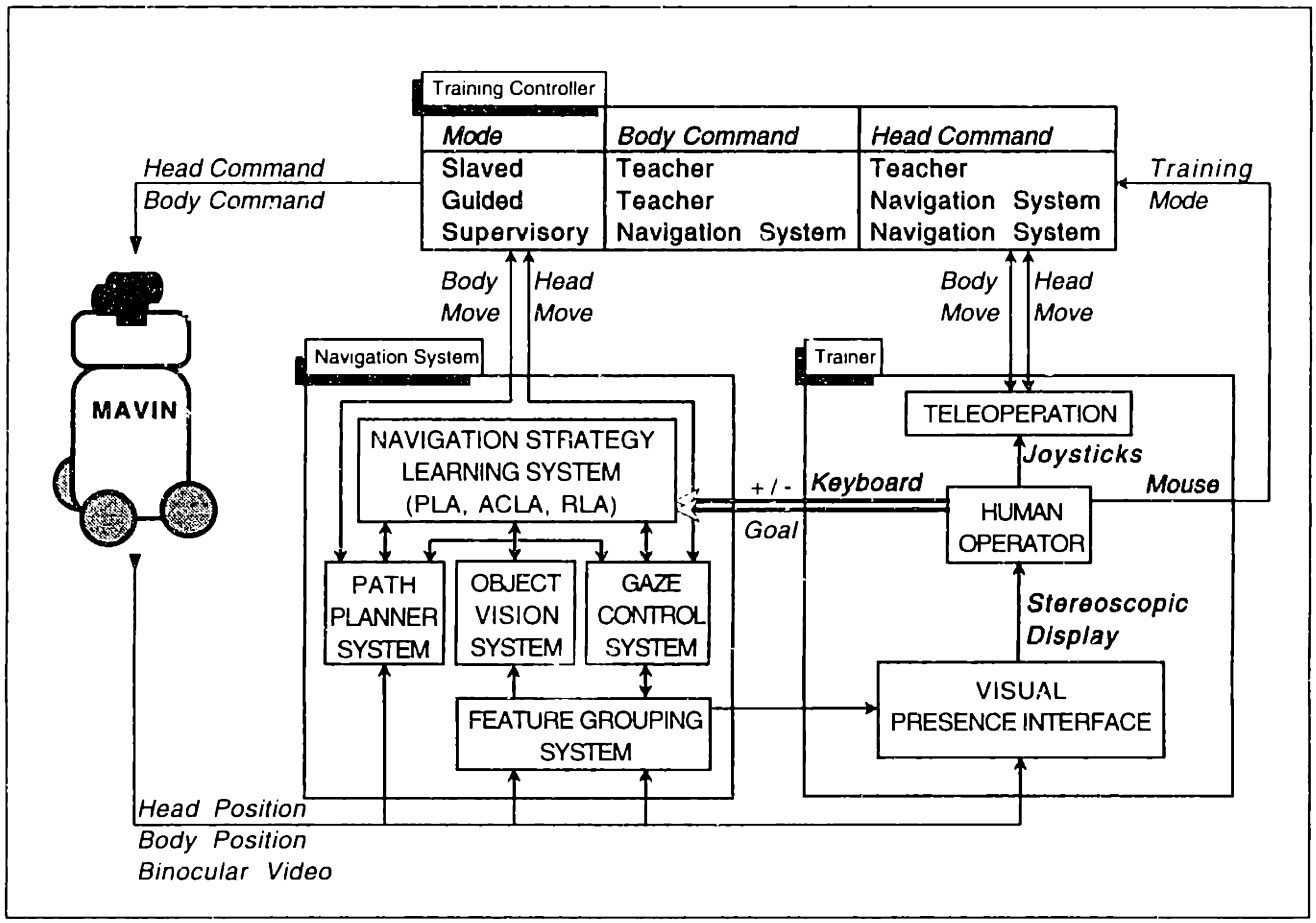


**FIGURE 70: A FRAMEWORK FOR TEACHING NAVIGATION VIA TELETRAINING.**
*The fully integrated system for learning and using navigation strategies can be embedded in a framework for teaching via teletraining.*

CHAPTER 10

# *Navigation Strategy Learning via Teletraining*

The previous four chapters have conceptually and mathematically described the sub-systems that comprise a real-time, fully-integrated, HRV neurocomputational system for learning and using navigation strategies on a mobile robot. Chapter 6 described some of the supporting architectures for landmark vision and locomotion. Chapters 7 and 8 then described two architectures that collectively learn a map of a visual environment: the place learning architecture (PLA), and the action consequence learning architecture (ACLA). Together, the PLA and ACLA can be used to perform long-term prediction and environment recognition. Next, Chapter 9 detailed the addition of an architecture for indirect reinforcement learning (RLA), which completes the navigation strategy learning sub-system (NSLS). The sub-system is capable of route planning and exploration, and resembles a POMDP. Along with peripheral sub-systems for gaze control (the GCS), object vision (the OVS), and path planning (the PPS), it forms the heart of the proposed navigation system.

This chapter describes the integration of each of these sub-systems within the framework of *teletraining*, a from of teleprogramming for reinforcement learning that employs a large degree of telepresence technology.

## 10.1 The Teletraining Framework

As described in Section 2.3, directing the exploratory behavior of a human trainee using restrictive physical guidance can be quite effective in teaching behavior. By the same token, the ability to physically direct the exploratory behavior of a robot, by guiding the robot along possible solutions to a desired task, and pointing out particularly important situations and alternatives along the way, is very attractive.

With this idea in mind, the proposed navigation system can be viewed as a "trainee" for autonomous navigation. It can be placed within a larger framework for training navigation tasks, as shown in Figure 70, consisting of a physical robot, the navigation system, the trainer, and a train-

ing controller. In this teletraining framework, both the trainer and the navigation system experience the sensory output of the physical robot, in this case consisting of measurements of head and body position, as well as binocular video imagery of the external environment. Both the trainer and the navigation system can also issue head and body movement commands to the robot, though the training controller decides which commands actually get executed. In any case, the trainer is able to view the commands issued by the navigation system, and vice versa.

When the trainer is human, the form and fidelity requirements for teleprogramming described in Section 4.3 can be satisfied using a combination of analog and symbolic teaching and control. (Note, however, that the trainer may actually be another previously trained navigation system.) The trainer is capable of symbolically issuing rewards and penalties to the navigation system, simply by pressing corresponding keys on a computer console. These keys directly (through internal channels) activate the pain and punishment categories in the RLA. Analogic teaching entails driving the robot around like a car, accomplished using teleoperation coupled with a limited degree of presence, augmented by computer graphics (see Section 4.3.4). The presence interface allows the human trainer to perceive depth by displaying the binocular video stereoscopically.

## 10.2 Training Modes

Within the general teletraining framework, training takes place using three distinct modes: *slaved*, *guided*, and *supervisory*. The current mode is determined by the human operator based on how far the task learning has progressed, measured by the similarity between the navigation (body) commands of the trainer and navigation system. The modes affect operation in the following ways:

1. *Slaved mode*: The trainer completely controls the physical robot (both head and body movements), but can monitor and take hints for the body commands generated by the navigation system. The navigation system receives sensory information and learns passively, but cannot actively direct the sensors in order to search for landmarks, and will therefore rarely be able to perform place learning or recognition. This mode allows the operator to steer the physical robot into remote but important environment locations quickly without collisions with obstacles.
2. *Guided mode*: The trainer again controls the body movements, and can monitor the body movements issued by the navigation system, but the navigation system controls the head movements, and can therefore "look" (actively search, track, and extract features from visual landmarks) in order to perform place learning and recognition, passively learn action consequences, and learn reinforcement events. This mode allows the trainer to direct the robot's attention to important areas of the environment so that it can build up corresponding portions of its internal map, and learn the utility of these areas.
3. *Supervisory mode*: The navigation system controls both head and body motions, but can take suggestions from the commands issued by the human operator. The navigation system can "look" (actively learn and recognize places), "act" (actively learn action consequences), and learn reinforcing events. This mode allows the trainer to test and fine-tune the navigation strategy.

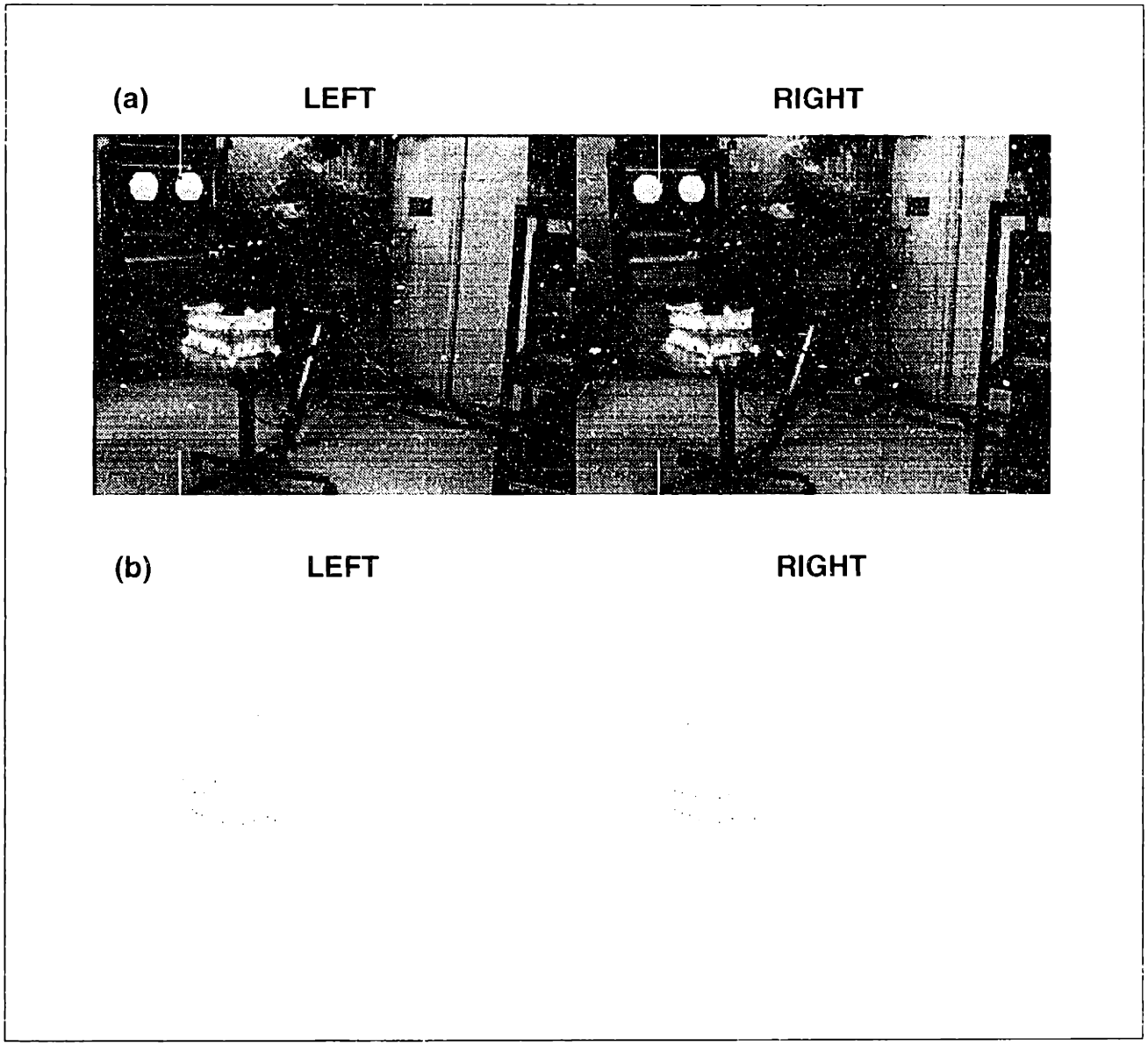


**FIGURE 71: ROBOT VIDEO IMAGERY DISPLAYED TO OPERATOR.**
*(a) Single frames of the binocular video streams displayed stereoscopically to the human operator. The white vertical line indicates the robot's current body direction with respect to the head. (b) Superimposed on each video stream in real-time are the respective point features extracted by the simplified featural grouping subsystem (FGS) within the circular window of attention.*

## 10.3 Implementation on MAVIN

Early implementations of teletraining on MAVIN employed a computer monitor to create a subjective sense of presence. The two binocular video streams from MAVIN's cameras were displayed upon this monitor on alternating frames, allowing the trainer to perceive depth using shutter glasses (see Figure 71). So that the trainer did not become disoriented or sick, computer graphics were superimposed on this display to indicate the pan and tilt direction of the robot's

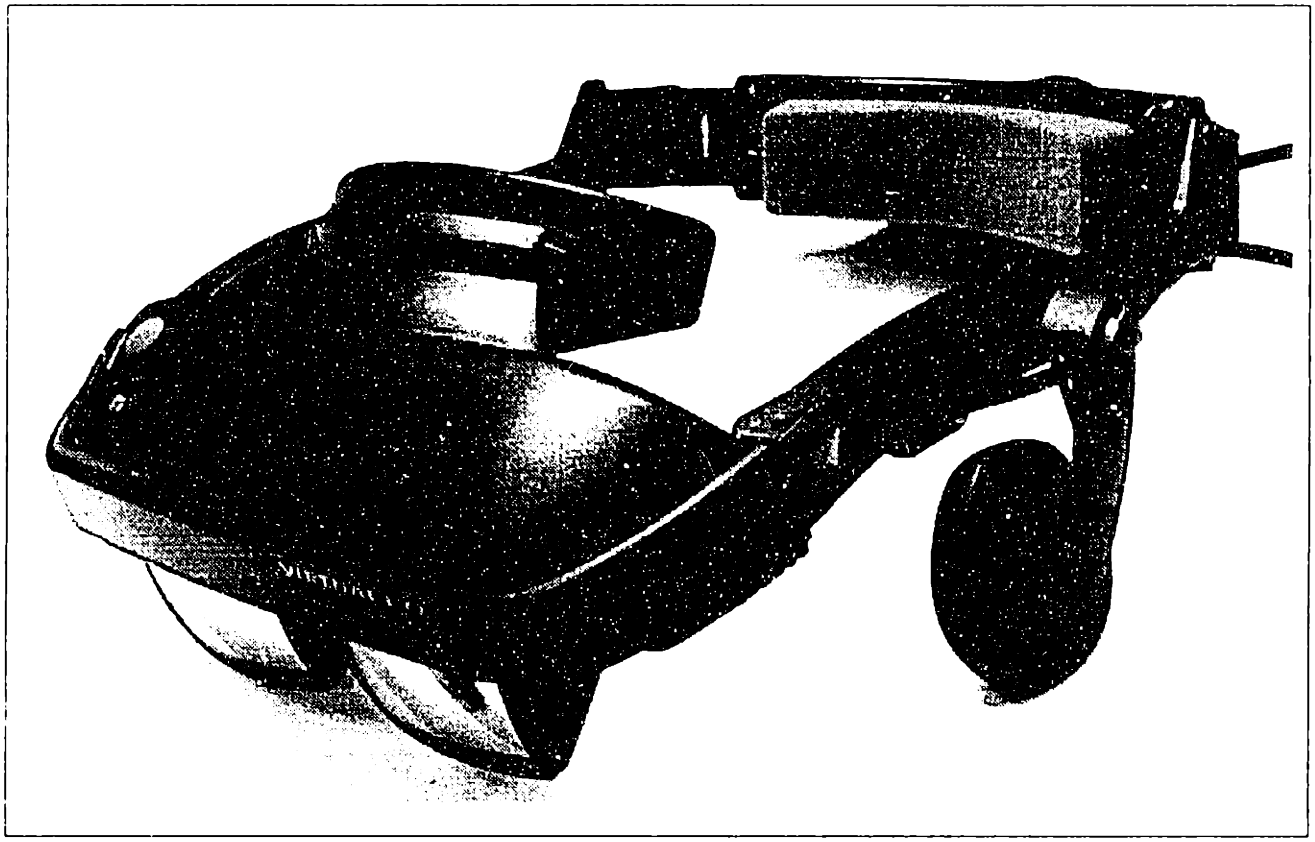

**FIGURE 72: HEAD-MOUNTED DISPLAY WITH HEAD TRACKER.**
*The Virtual i.O i-glasses include two LCD stereoscopic displays (below the visor), a 3-DOF (pitch, yaw, and roll) head tracker (behind the foam padding), and a pair of ear-phones. Each of the LCD displays has a resolution of 256x256, and subtends a 30° field-of-view.*

head with respect to its body midline. So that the trainer and robot perceive the environment in a similar fashion, extracted visual point features were also superimposed. Finally, teleoperative commands to the robot's head and body were issued by the trainer using several joysticks.

Initial subjective experience using this setup indicated that the difference in baseline between the robot's cameras and the operator's eyes does not interfere with the ability to use the resulting depth perception to navigate, despite the fact that the world looks rather stretched. However, the operator's ability to fuse the two binocular images is limited to the small disparities around the horopter. Thus, the ability to control vergence with a joystick (or, if available, an eye tracker) is very important.

More recent implementations have replaced the video monitor with a binocular head-mounted display (see Figures 72 and 73). The orienting graphics remain, but play a much smaller part in the system since the HMD includes a head tracker. Arrow keys on the computer keyboard (instead of joysticks) are still required to move the body and verge the cameras, but head motions are slaved entirely by movements of the operator's head. This new setup can be likened to driving a car.

**FIGURE 73: THE TRAINER SETUP.**
*The operator can direct head motions simply by moving his head while controlling vergence and body motions with arrow keys on a computer monitor. Graphically augmented video from the robot's cameras is presented stereoscopically.*

## 10.4 Training Results

Using the setup described in the previous sections, MAVIN was slaved, guided, and supervised through the laboratory environment by an operator. Though the teletraining framework does not require the robot to have previously learned a map of the environment, in this case the NSLS was loaded with the PLA and ACLA data learned during previous training sessions (see Sections 7.7 and 8.5.1). Under slaved mode, the operator drove MAVIN to three different random locations, marked by pylons on the floor. At each of these locations, the operator placed the system into guided mode, and allowed the robot to scan for landmarks and perform place recognition. At two of these locations, (5',1') and (2,'6'), the operator issued rewards. At the remaining location (3',3'), the operator issued a penalty. In all three cases, the operator continued to rotate the robot clockwise at small (approximately 10°) intervals each time the robot had finished searching the panorama and had recognized the current place, until a complete rotation had been executed. This

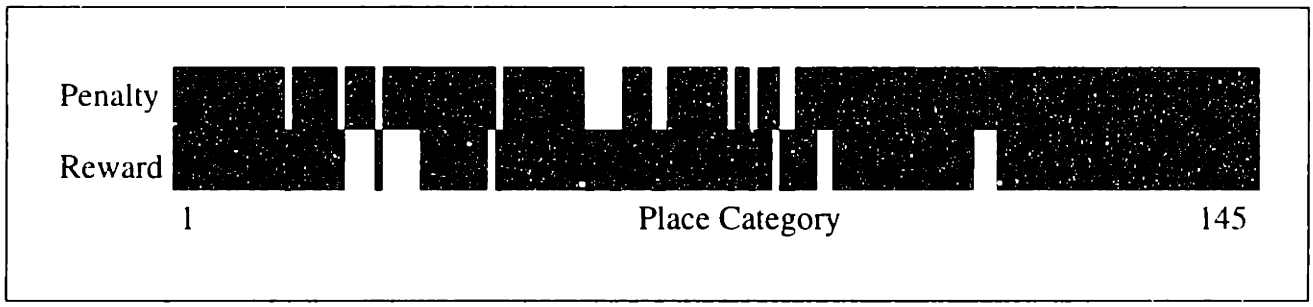


**FIGURE 74: RLA WEIGHT TEMPLATES FOR TRAINED TASK.**
*The two-element weight templates are shown here for each of the 145 learned place categories. The upper portion of the templates correspond to penalty, while the bottom portion corresponds to reward. The brightness of each element is proportional to the corresponding weight.*

rotation was necessary in order to learn the reinforcement values for each of the heading tuned place nodes most active for all the possible headings at a single location. (Note: such rotations would not be necessary if a representation of the environment were utilized wherein heading tuned place nodes were pooled in hierarchy into heading invariant place nodes — see Section 7.7).

This procedure resulted in the learned RLA weights shown in Figure 74. It is important to note that the process of learning reinforcement in this fashion results in templates that reflect the utility of various place regions, not the individual positions in the environment where rewards or punishments were administered. That is, reinforcement at a particular position in the environment results in an evaluation of the place node corresponding to the entire place region which contains that position.

## 10.5 Route Planning Results

Once each of these locations had been visited in the manner described above, the robot was driven to various starting locations, placed into supervisory mode, and given one of two goals. The results can be directly compared to those presented in Section 9.5. The diffusion of activity, final diffusion activity, and route plans executed by MAVIN when the goal was place region 1 are respectively shown in Figures 75, 76, and 77. The diffusion of activity, final diffusion activity, and route plans executed by MAVIN when the goal was place region 68 are respectively shown in Figures 78, 79, and 80.

In addition to exhibiting the same general qualities as the results in Section 9.5, these two sets of results for the trained task generally demonstrate an avoidance of penalty positions and an attraction to reward positions. In some cases, the robot even makes great excursions in order to avoid the penalty location, or goes out of its way to extract a reward. There are, however, some apparent limitations to this behavior. Notice, for example, that some of the routes actually pass quite near reward locations without actually being diverted to them. This sort of behavior has three major causes. First, the qualitative characteristic of the learned map do not allow routes to be planned more accurately than, in this particular case, a few feet (see discussion in Section 9.5).

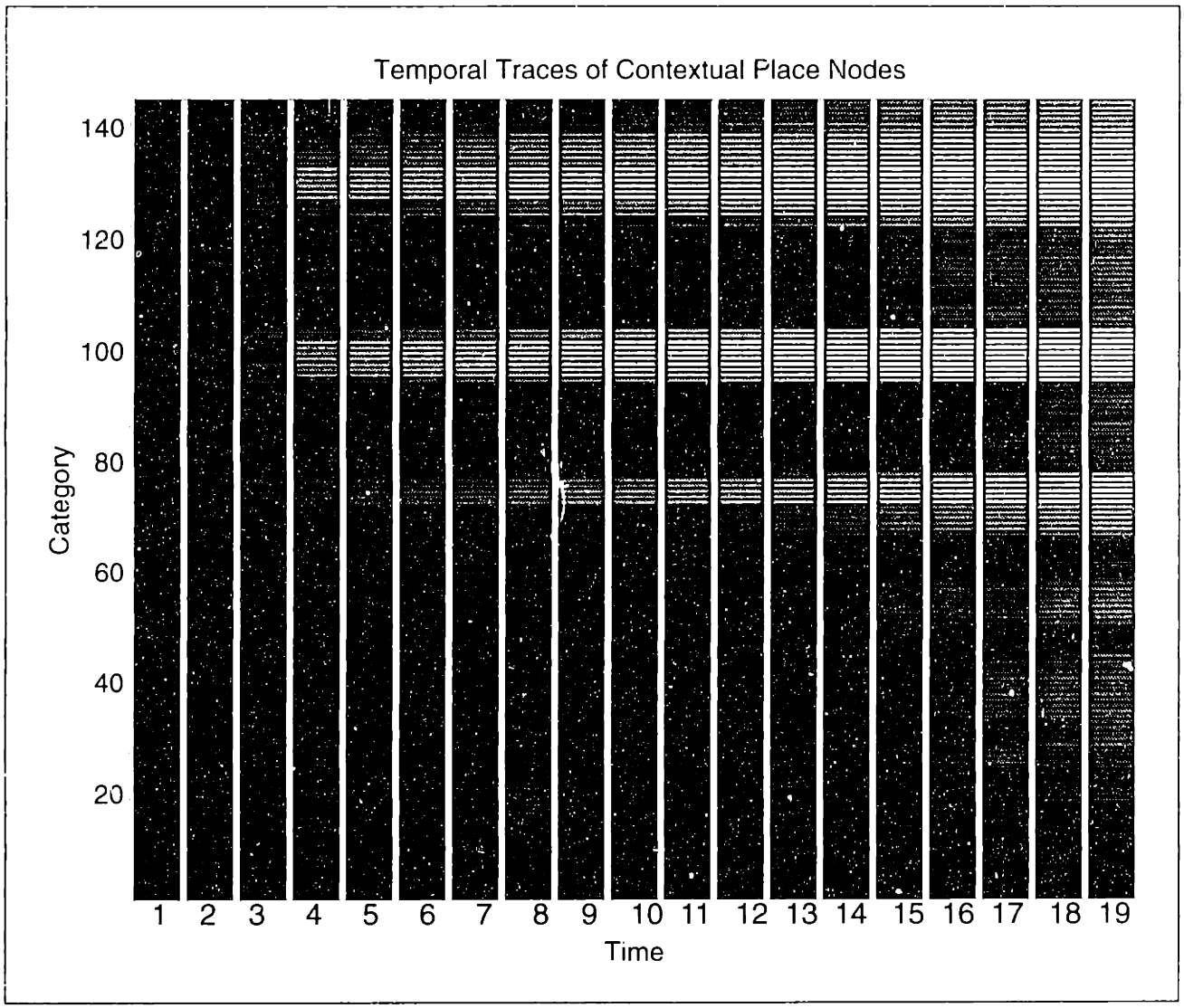


**FIGURE 75: DIFFUSION OF ACTIVITY OVER PLACE NODES (TRAINED TASK, GOAL = PLACE #1).** *The activity of each of the place nodes as a function of time during the diffusion process. The brightness of each element is proportional to the corresponding activity level.*

Second, the learned utility of places relative to the utility of actions is such that travelling to reward locations on the way to the goal does not offset the cost incurred by executing the extra actions necessary to re-route. This is apparently not the case for penalties. To address this problem, one would be tempted to scale the reinforcement values in order to weigh them more heavily than action costs. However, this scaling often results in the rewards for a given location actually exceeding the penalty one would get by remaining at or near that location, which means that the robot will never attempt to reach the goal! Therefore, if one really wishes the robot to visit reward locations, then they should be made part of the goal list.

Finally, recall that the robot learns the utility of entire regions in the environment, not the individual positions were reinforcement has been administered. Thus, the robot merely avoids the local area within which a penalty has been given, and is attracted to the local areas surrounding the positions where the rewards have been given.

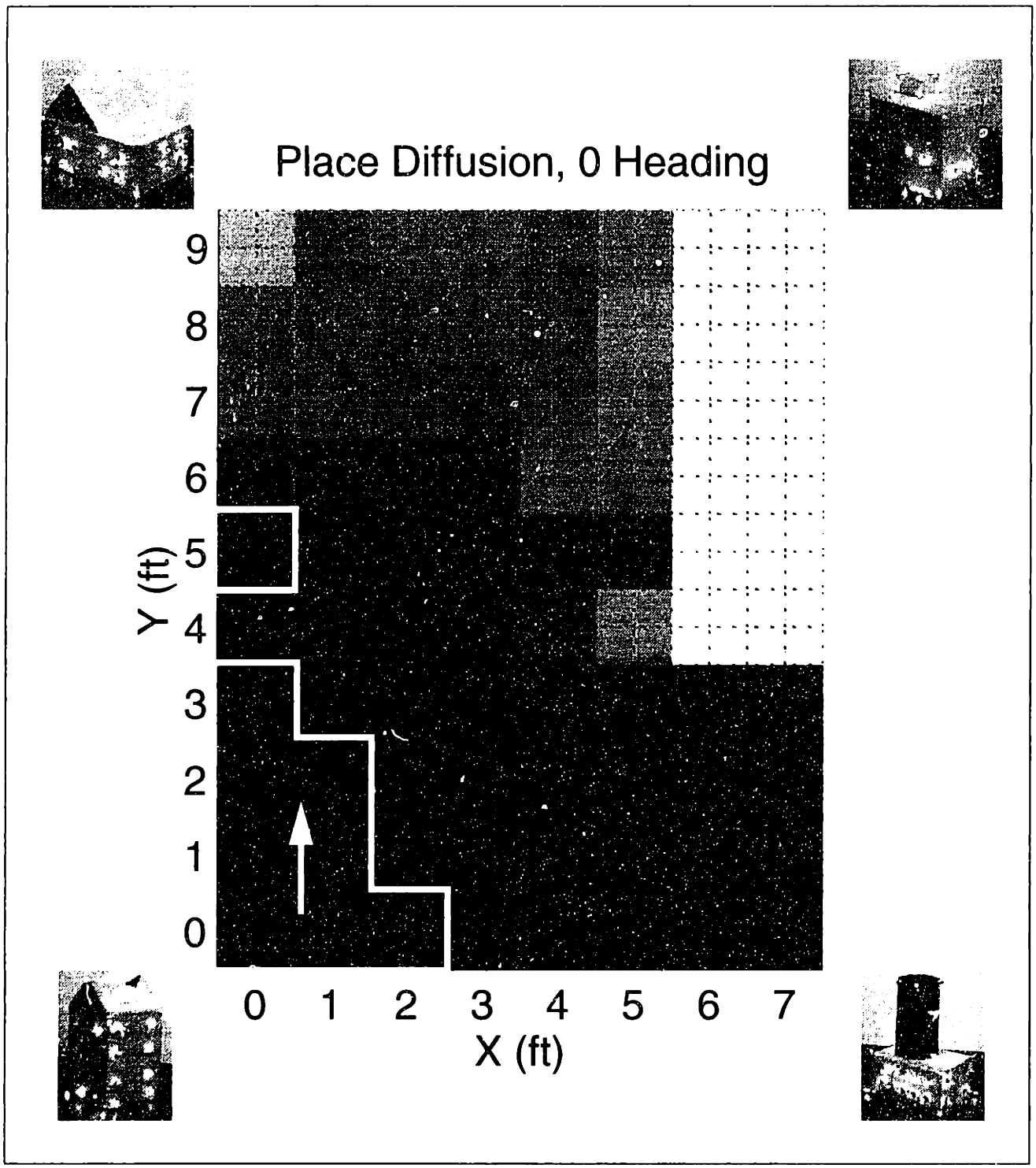


**FIGURE 76: FINAL PLACE ACTIVITY AFTER DIFFUSION AS A FUNCTION OF LOCATION (TRAINED TASK, GOAL = PLACE #1).**

*Each location in the environment is colored according to the diffusion activity of the PLA place node with the most sensory evidence for that location (brightness is proportional to activity level). The goal region and preferred heading is indicated in white.*

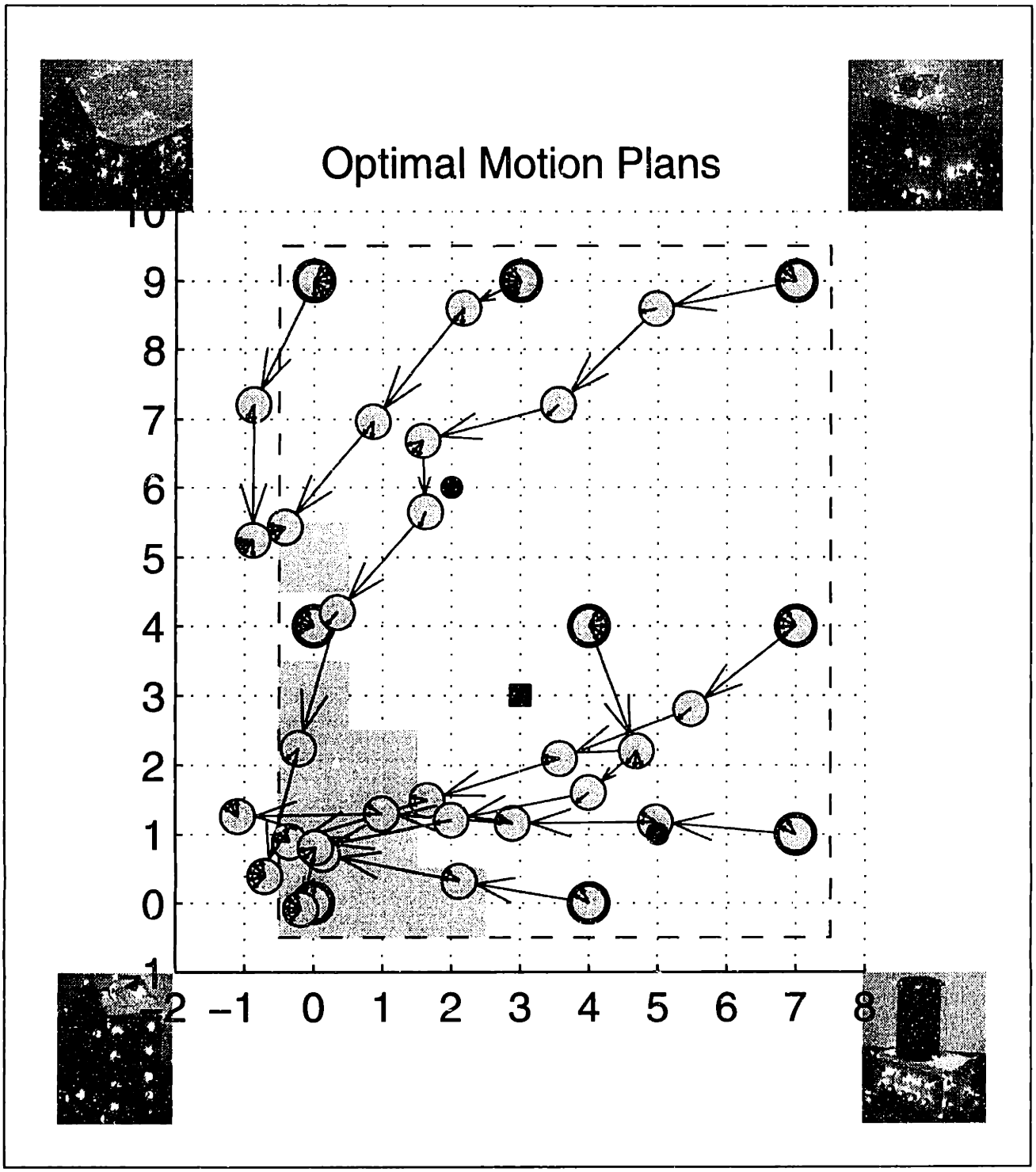


**FIGURE 77: ROUTE PLANS EXECUTED BY MAVIN (TRAINED TASK, GOAL = PLACE #1).** *Routes navigated by MAVIN from various starting points in the laboratory. The goal region is indicated by the gray area in the lower left-hand corner of the environment, while the locations of rewards and penalties issued during training are indicated by colored circles and squares, respectively. Circles with radial lines indicate the positions and headings of the robot at points along each route. Black rings indicate the starting positions.*

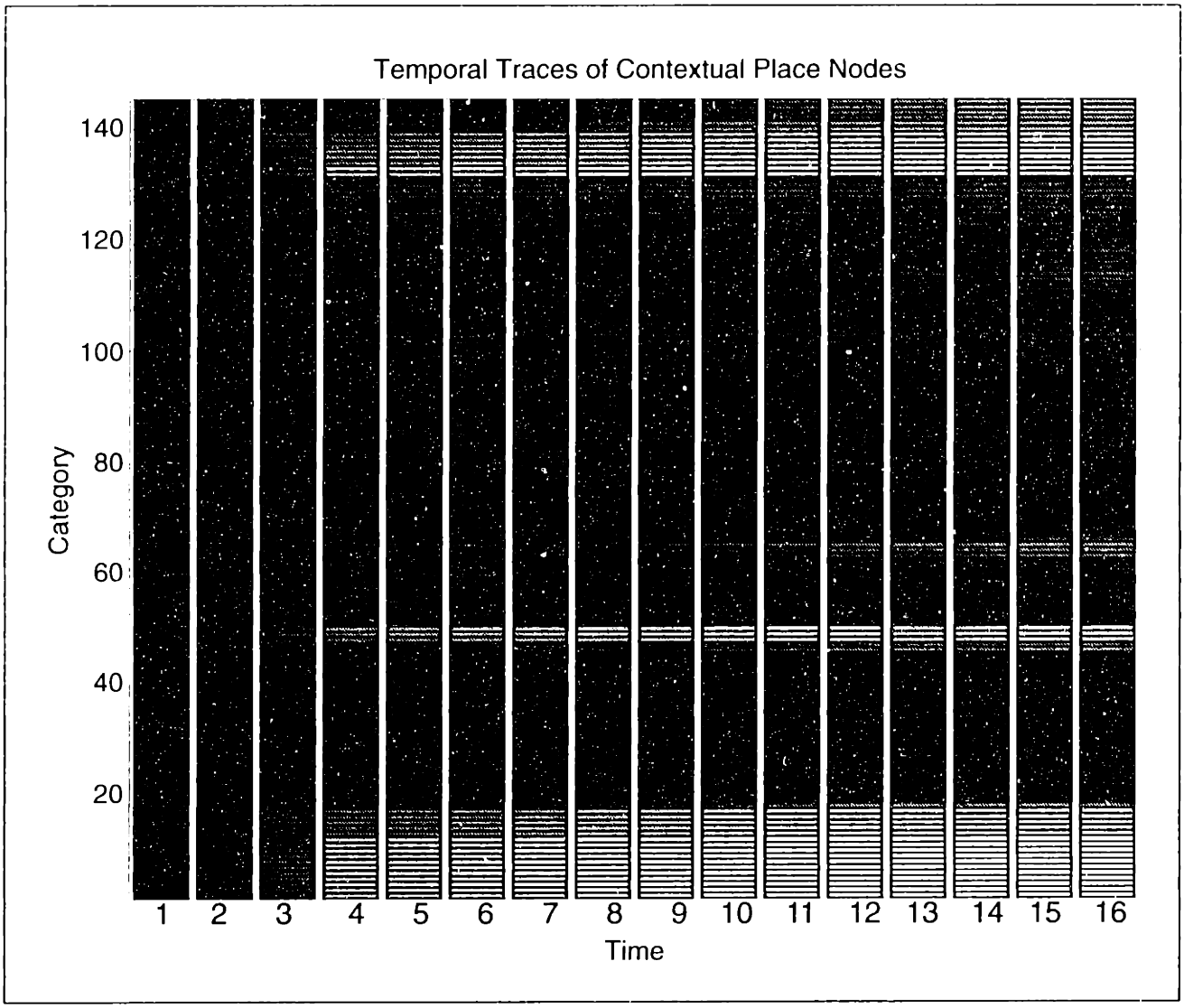


**FIGURE 78: DIFFUSION OF ACTIVITY OVER PLACE NODES (TRAINED TASK, GOAL = PLACE #68).**

*The activity of each of the place nodes as a function of time during the diffusion process. The brightness of each element is proportional to the corresponding activity level.*

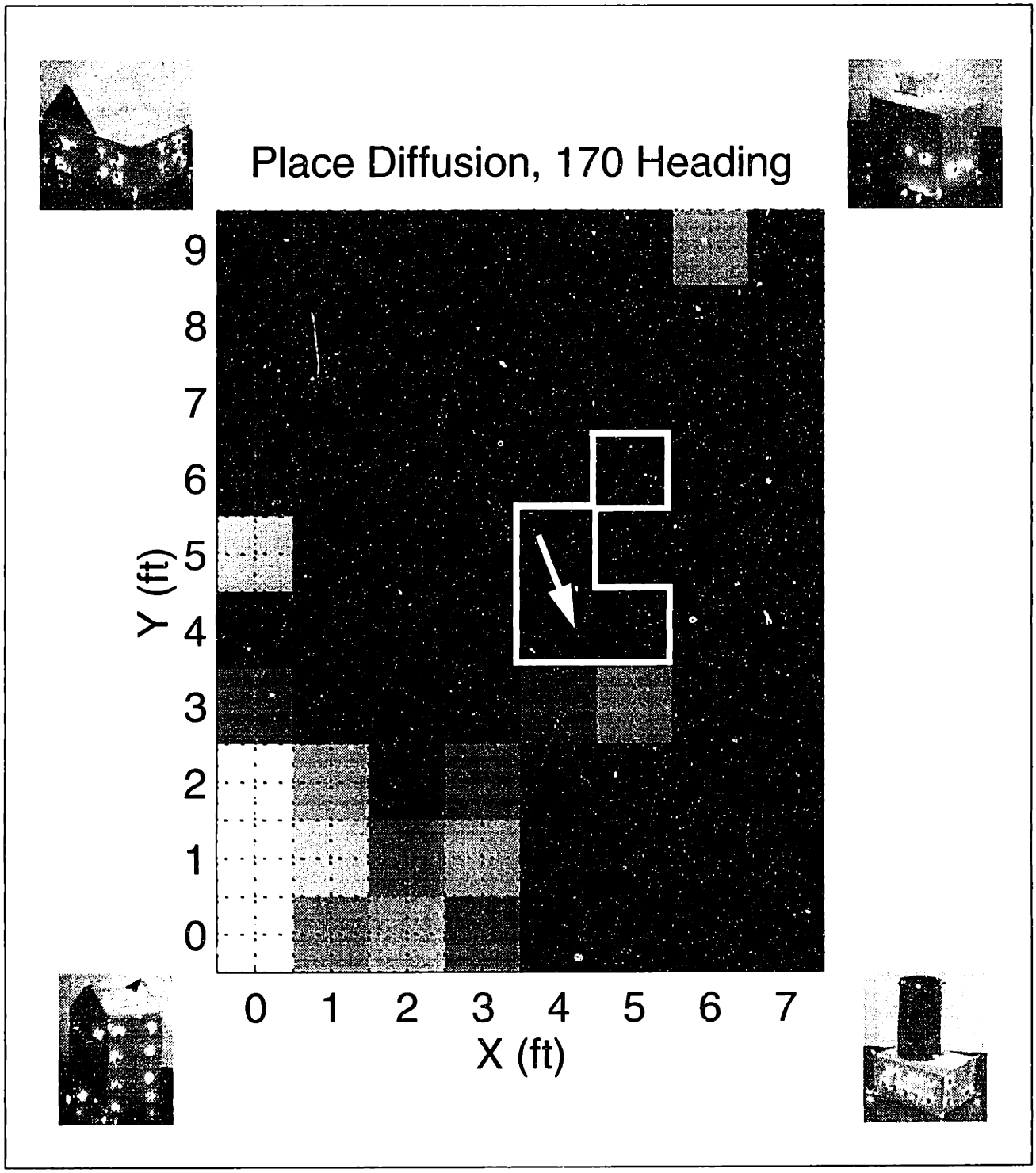


**FIGURE 79: FINAL PLACE ACTIVITY AFTER DIFFUSION AS A FUNCTION OF LOCATION (TRAINED TASK, GOAL = PLACE #68).**

*Each location in the environment is colored according to the diffusion activity of the PLA place node with the most sensory evidence for that location (brightness is proportionai to activity level). The goal region and preferred heading is indicated in white.*

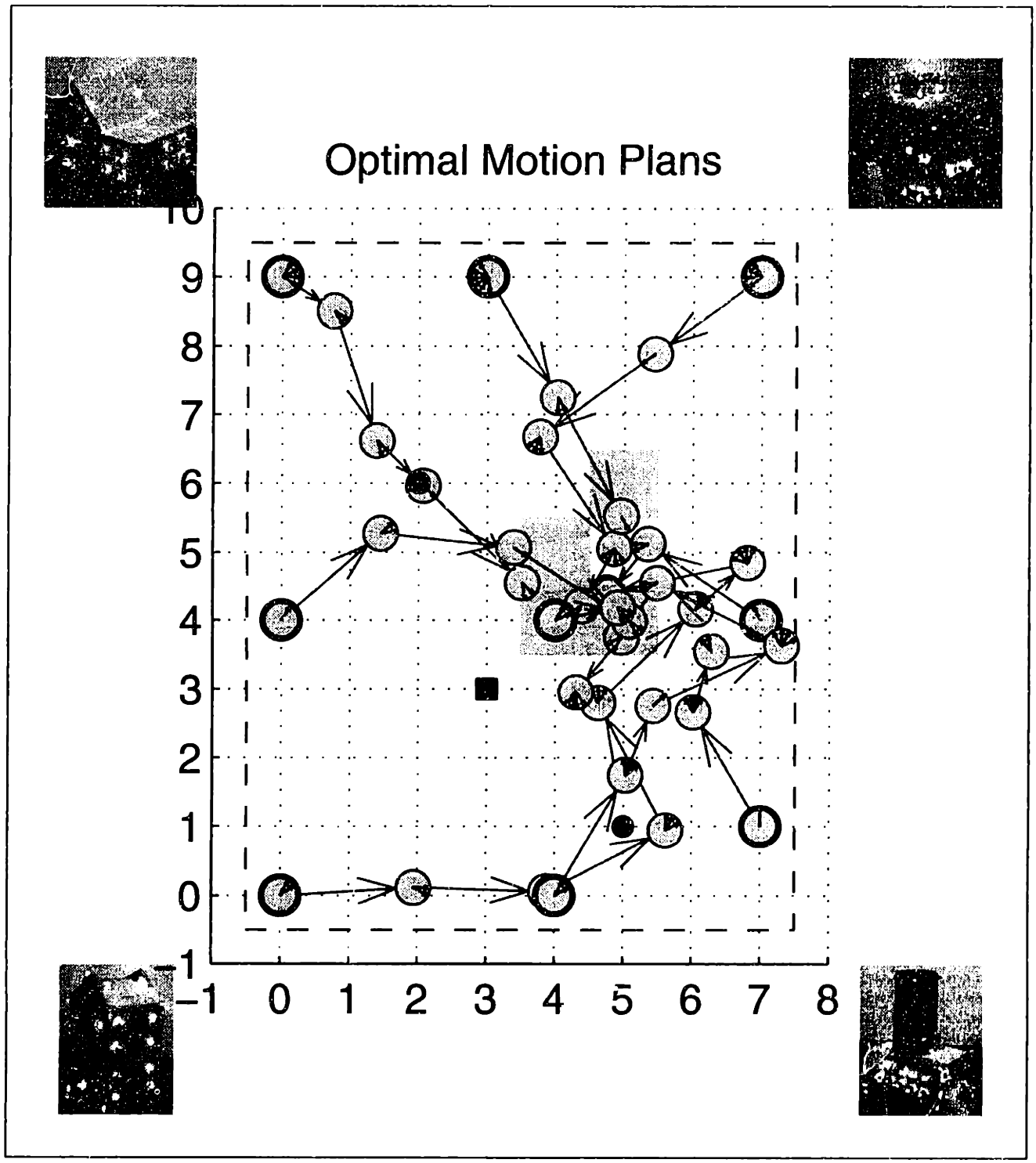


**FIGURE 80: ROUTE PLANS EXECUTED BY MAVIN (TRAINED TASK, GOAL = PLACE #68).** *Routes navigated by MAVIN from various starting points in the laboratory. The goal region is indicated by the gray area in the center of the environment, while the locations of rewards and penalties issued during training are indicated by colored circles and squares, respectively. Circles with radial lines indicate the positions and headings of the robot at points along each route. Black rings indicate the starting positions.*

Because of this general "attraction" or "deflection" by areas in the environment, the robot's routing behavior on these trained tasks resembles the sort of behavior one expects from potential fields approaches to path planning (see Section 6.4). However, it is important to keep in mind that potential fields typically work in a Euclidean coordinate system, unlike the network diffusion performed here. Furthermore, potential fields approaches have the tendency to produce local minima within which the robot might get stuck, which is again not the case with the global diffusion process employed here.

## 10.6 Summary

This chapter has described a framework for training the fully integrated navigation system via teletraining, a technique for teaching new tasks through reinforcement learning using telepresence technology. This training involves a gradual transition from slaved mode (where the human operator controls head and body movements via teleoperation), to guided mode (where the operator controls body movements only), and finally to supervisory mode (where the human merely instructs the robot). The teletraining framework is implemented in the mobile robot MAVIN with the aid of a stereoscopic head-mounted display and head tracker.

Training the robot by driving it under slaved mode to three separate locations, and guiding the robot through a 360° turn at each location while issuing reinforcement, resulted in learned templates in the RLA for many of the previously learned place categories. Using these templates, MAVIN was then able to plan and navigate routes to two goals from a variety of starting positions in the laboratory while avoiding place regions containing the positions where penalties were previously issued, and routing through the place regions containing the positions where rewards were previously issued. The routing behavior of the robot on these trained tasks qualitatively resembles that of potential fields approaches to path planning.

CHAPTER 11

# *Summary, Conclusions, and Future Work*

This thesis has described the development of a system that learns and applies strategies for visual navigation, and its implementation on the mobile robot MAVIN. The system learns, in an unsupervised fashion, a map of a large-scale environment defined by a spatial arrangement of visual landmarks. It also learns, under the guidance of a trainer, how to perform simple navigation tasks in such an environment. The development of this system has taken a cognitivist approach to emulating, at the neurocomputational level, the apparent hierarchical, relational, view-based spatial cognitive mapping of space in the rat hippocampus. Evaluation (testing) of the system was driven by arguments for real-time demonstration on a mobile robot.

## 11.1 Thesis Summary

In Part I, the thesis reviewed the technical background pertaining to training navigation tasks. Biological motivations included the existence of place cells and relational structures in the rat hippocampus, as well as analogous view-based relational structures in the object vision pathway of the macaque monkey. It also included psychological evidence for the effectiveness of using restrictive physical guidance to teach humans new tasks. Computational principals and techniques for learning representations of environments were then presented, which revealed a number of advantages and disadvantages of existing symbolic, statistical, and neural approaches to learning environment models consisting of states and state transitions. Next, the subject of learning new tasks was explored, which not only illuminated a whole set of problems with direct approaches to reinforcement learning, but also exposed a number of gaps in the capabilities of indirect approaches. An analysis of the analogous problems for control learning led the discussion to techniques for teaching skills via teleprogramming, which in turn spawned an analysis of the advantages of employing techniques for achieving telepresence. Finally, the more specific issues related to visual navigation were addressed, including the use of active vision techniques for landmark search, tracking, and feature extraction, and the pros and cons of various AI and biological modelling approaches to map-making and route planning. The later discussions shed some light on a number of problems associated with reconstructive and metrical approaches to map-making, and

explored various alternative methods for image- and landmark-based definitions of distinctive places, and learning and planning with diktiometric maps.

The lessons learned by studying these various principals and techniques were employed in Part II during the development of the proposed system. First, a number of supporting sub-systems for landmark vision and robot locomotion were presented in detail, including a gaze control sub-system (GCS), an object vision sub-system (OVS), a path planning sub-system (PPS), and a featural grouping sub-system (FGS). The former two sub-systems were implemented in their entirety on MAVIN, while the latter two were simplified substantially by limiting the types of environments in which the robot navigates. A view-based place learning architecture (PLA) was then defined, and both semi-egocentric and fully egocentric versions were implemented successfully on MAVIN. Next, a relational action consequence learning architecture (ACLA) was presented and analyzed in the context of a partially observable Markov decision process (POMDP). The architecture was trained using data gathered by MAVIN, and was demonstrated to perform rough predictions of place based on moderately long locomotive action sequences. Finally, the navigation strategy learning sub-system (NSLS) was completed by introducing a reinforcement learning architecture that learns the relative utility of various places and actions in the learned map (embodied by the PLA and ACLA) based on rewards and punishments received through time. The NSLS was first tested on MAVIN without any reinforcement training, which verified that the robot could indeed use the learned places and transitions to plan routes to several goals from various starting locations in the environment. All of the sub-systems were then integrated in the context of teletraining, wherein an operator could train the system by guiding the robot through motions using telepresence technology. Following such a training session, MAVIN demonstrated the ability to plan routes to goals while avoiding areas containing penalty locations and passing though areas containing reward locations.

Overall, the process of implementing the proposed system for visual navigation in (near) real-time on a mobile robot has effectively demonstrated its potential for intra-place localization, environment recognition, route planning, and exploration.

## 11.2 Demonstration, Simplification, and Simulation

The results reported in this thesis are, of course, preliminary due to the simplifications made to the environment in order to ease grouping, feature extraction, and path planning requirements. Note, however, that such simplifications differ in important ways from computer simulations:

1. Features arise from physical structures in the environment, and require real-time extraction and grouping algorithms, which might not be perfect.
2. The robot makes continuous movements, which include errors. Even the emulation of such errors introduces yet another set of assumptions to the simulation process.
3. The designer remains explicitly aware of the specific assumptions underlying any simplifications to environment, unlike computer simulations, where it is often very difficult to know what and in how much detail to simulate. In general, it is very difficult to know what elements of the world are likely to significantly affect performance without finding out first hand. For

example, simulations often neglect the signal-to-symbol problem (see Section 3.2), and therefore have no difficulty using symbolic approaches in conjunction with discrete sensations and actions.

Therefore, real-time implementation on a real mobile robot, even in this simple environment, allows one to make conclusions about the potential for more realistic environments much more soundly than even the most complicated computer simulations. As a case in point, most simulations of map-making in the robotics literature, as well as cognitive mapping in the hippocampal literature, completely ignore the physical act of searching for landmarks or structures in the visual panorama, most likely because the designers do not realize its direct relevance to the task. But this physical act can actually affect the efficiency of map-making and localization, and can therefore affect the design of map-making systems. (See Section 7.5 for a synopsis for how the efficiency of visual search prompted a redesign of the proposed map-making system.)

At the very least, real-time implementation demonstrates the potential usefulness of the proposed system in certain environments where landmarks are easily picked out. The system in its present form might, for example, prove useful in for navigating in environments with beacons, such as the sonar beacons frequently used undersea. It may also prove useful for visual environments where landmarks are sparsely distributed and uncluttered, such as space (and perhaps undersea). Finally, it may prove useful for environments with targets that might easily be segmented from the background using active sensing technologies, such as military environments imaged from airborne vehicles using SAR. In fact, the bright lights used in these experiments greatly resemble the bright returns in SAR imagery.

In any case, prospects for extending the proposed system to more realistic environments mainly depend on the appropriateness of dividing the problem into low and high level visual processing, the main assumption underlying the simplification of environments using bright light patterns. In particular, the existence of a sub-system (i.e. the FGS) for low level visual processing must at least be theoretically possible. One can argue for this division most easily on biological grounds. As discussed in Sections 2.2.2 and 5.1.1, biological organisms use hierarchical processing principals to preattentively group low level visual features, and use these perceptual groups to drive visual search. Though top-down expectations do play a role, perceptual grouping occurs even in situations with little or no expectations (e.g. random-dot stereograms). Biological organisms also purposively group visual features in order to track moving targets and perform object recognition. Based on these living and working examples, it is certainly possible to construct an artificial subsystem that carries out similar low-level visual processing tasks.

## 11.3 Scientific Contributions

*The main contribution of this thesis is an empirical demonstration that viable artificial systems for teaching and utilizing navigating strategies large-scale visual environments can be achieved by emulating, at the neural network and architecture levels, cognitivist biological principals for learning task-independent cognitive maps.* This demonstration has entailed developing a real-time, neurocomputational system for learning strategies for visual navigation via teletraining, and

verifying (testing) the navigation capabilities of this system on the mobile robot MAVIN. The design of this system has incorporated new hierarchical, relational, view-based (HRV) methods for learning and evaluating maps.

The process of integrating both new and existing computational techniques into a working system for visual navigation also makes significant primary and secondary scientific contributions to biological modelling, visual learning and representation, visual navigation, and teleoperation and supervisory control.

### 11.3.1 Biological Modelling

The proposed system demonstrates how reinforcement learning might be accomplished in animals using a cognitivist theory, by faithfully emulating spatial cognitive mapping in the rat hippocampus at the neurocomputational level. As reviewed in Section 2.2.1, current architectures based on models of cognitive mapping for the most part either fail to explain important biological observations, fail to provide the computational mechanisms for navigation and goal-seeking behavior, or lack a practical means for implementation. The proposed system is the first to successfully implement McNaughton's (1989) conceptual local view model of hippocampal map-making in a continuous environment (see Section 5.2.2). It is also the first to demonstrate a biologically plausible neurocomputational model of cognitive mapping on a real mobile robot. At the very least, the results of these implementations might suggest some possible experiments to the biological community.

### 11.3.2 Visual Learning and Recognition

The proposed system also demonstrates the general potential of the HRV learning methodology. It has effectively extended, both hierarchically and by analogy, the principals originally introduced by Seibert & Waxman (1989, 1992) for learning and recognizing visual objects, to the domain of visual map-making and navigation. It recognizes an environment by landmark parts by first learning the spatial layout of landmark aspects rather than just a spatial feature pattern. This approach might also be applicable to learning objects in terms of their constituent parts (see Section 2.2.2).

The system also incorporates actions into learned relations, thereby making way for prediction and control (i.e. exploration, planning, and policy generation). This approach might be applicable to learning motion-induced transitions between object aspects (Cunningham & Waxman, 1994a; see Section 2.2.2), which aid the process of recognizing, inspecting, and manipulating objects.

### 11.3.3 Mobile Robotics

As a secondary contribution, the proposed system introduces new architectures with which a mobile robot can learn, adapt, and apply diktiometric maps of visual environments. The place learning architecture exploits the invariances afforded by landmark-based place definition, as well as the computational matching ease associated with image-based place definition, resulting in an architecture that does not require landmarks to be distinguishable. Due to the distributed nature of its invariant shape processing, it requires neither the prior learning of visual landmarks, nor the

subsequent correct identification of the landmarks, in order to perform place learning or recognition. Furthermore, place learning in this fashion results in a coarse coding of space that allows the robot to deduce both position and heading from the pattern of activity across learned place nodes.

The action consequence learning architecture embodies an empirically efficient method for learning Markov-like maps of visual environments with continuous actions. Very few incremental methods for such rapid task learning currently exist, especially for relatively complex environments with continuous actions and sensations (see Chapter 3). Due to the localist nature of the environment representation, this architecture allows reinforcement learning to proceed simply as a form of correlation learning. Furthermore, the computations it performs resemble standard, efficient algorithms for solving POMDPs. Unlike POMDPs, however, the system is capable of adapting both the places and the transitions to gradual changes in the environment through the use of standard localist neural principals.

The system also illustrates a how action policies for behavior-based robots might be learned for visual navigation using a deliberative approach. Of Brooks (1991) four types of learning for behavior-based systems (see introduction to Chapter 4), the proposed system has addressed both representation learning, via the construction of task-independent maps, and the formation of new behaviors, indirectly through the evaluation of the learned map and a subsequent computation of an optimal action policy. Policy determination takes the form of a parallel, distributed computation involving only local interactions between the elements of the task-independent model (reminiscent of *internalized plans* — see Payton et. al., 1990).

### 11.3.4 Teleoperation and Supervisory Control

Finally, the proposed system illustrates a new paradigm for teleoperation and supervisory control called teletraining. This paradigm addresses the teaching aspect of Sheridan's (1992) five duties for the human supervisor within a typical supervisory control system (see Section 4.3). It allows an operator to perform an off-site mission training via teleprogramming, wherein the semi-autonomous robot learns tasks by observation based on the sensations experienced during teleoperation through a combination of analogic and symbolic mechanisms. (In principal, the operator could also train the system via simulated guidance in *virtual environments*, an artificial world created by a computer — see Rheingold, 1991). However, this training takes the form of a restrictive physical guidance that can be thought of as a directing of exploratory behavior for reinforcement learning, rather than a forced physical guidance for learning control. The use of telepresence (i.e. the subjective feeling of presence) eliminates any need for special command languages beyond the symbolic issuing of rewards and penalties. The system therefore establishes the role of telepresence in training robots to solve complex tasks rather than simply remotely controlling robots precisely from a distance.

## 11.4 Directions for Further Research

As a proof of concept, this thesis has certainly spawned more questions than it has answered. There are many aspects of the proposed system that simply have not been addressed. Parameters have been tweaked only enough to make the system work sufficiently well on simple navigation

tasks. No systematic methods were employed in order to optimize test the system to its limits. Furthermore, no attempt has been made to incorporate and verify many of the features that might make the system more useful. This section describes the incredibly large amount of work that remains.

### 11.4.1 Landmark Occlusion and Incomplete Search

The place learning architecture employs an ART network to learn and recognize local view patterns. This ART network exhibits a certain degree of robustness to noise, and can therefore handle the partial or complete occlusion of visual landmarks. Experiments should be carried out in order to systematically assess just how robust (how much occlusion can be tolerated). Note that permanent occlusions do not concern us here, since they will simply be adapted to over time (see next section). In effect, permanent occlusions simply change the appearance of the landmark from particular viewing positions. But one would hope the system would be fairly robust to the temporary partial or even complete (e.g. removal) of landmarks.

A related topic concerns a possible modification to the system in order to allow the robot to accumulate evidence for places as the search for landmarks progresses, rather than waiting for a certain portion of the local view to be searched. Head movements are still the most time consuming operation of the system, and therefore represent the critical time path for place recognition. To make this search process more efficient, the ART network itself would have to be modified so that it could differentiate between portions of the incoming local view pattern that contain no landmarks and those that simply have not yet been searched. The bottom-up matching process would then occur exclusively in the searched portion of the local view, and the landmark search process could then terminate once some measure of "confidence" (e.g. the difference in evidence between the most active and second-most active place nodes) reaches a given threshold. Note that this modification would greatly increase the importance of priming search locations in the gaze control architecture saliency map in a top-down fashion by locations of the major differences in the patterns of contending places with the most evidence.

### 11.4.2 Dynamic Environments

Each of the architectures proposed in this thesis are theoretically capable of adapting their learned representations to gradual changes in the environment. The PLA constantly updates its local view templates to match the incoming sensory pattern, the ACLA adjusts its transition templates based on prediction errors, and the RLA changes its evaluation of places and actions based on reinforcement events. However, none of these capabilities have been rigorously verified. Future work should attempt to quantify just how robust the architectures are to environment changes by systematically varying various attributes of the visual environment, such as the positions, poses, and even the permanent removal of individual landmarks. This assessment may entail a systematic evaluation of the effects of each of the system parameters to determine just how quickly the system can adapt to changes while still remaining robust to temporary changes like the partial occlusion of landmarks. This trade-off is often referred to as the "plasticity versus stability dilemma."

### 11.4.3 Natural Environments

As discussed in Section 11.2, the results presented in this thesis must be regarded as preliminary due to the simplifications made to the test environment. A major thrust of future work therefore must concentrate on developing more sophisticated methods for dealing with landmarks and obstacles in real-time, thereby extending the system to more realistic environments.

For the case of more natural landmarks, one possibility would be to implement the FGS in its entirety (see Section 6.1.1). But this requires access to powerful real-time, parallel computers. Such computers are expensive and difficult to come by. Undoubtedly, such implementation would entail quite a bit of parameter tweaking, and might even require some ad hoc methods to quickly disambiguate distractors from landmarks. It may also entail specializing the search for landmarks, depending upon the type of environment within which the robot is expected to navigate. Ideally, such landmark disambiguation should employ simple low-level visual processing algorithms (e.g. using texture, color, etc.).

It may also be possible to consider *all* stable visual point features extracted using some low-level processing algorithm as "centroids" of visual landmarks. For example, one might simply apply the DEB to edges directly and use the resulting peaks to define centroids. There are drawbacks to this method, of course. Such features are likely to be less permanent structures (e.g. writing on whiteboards), and would be numerous enough to exert great demands on the speed of head movements. But for some environments, these disadvantages may not be all that serious.

Another possible way to pick out landmarks would be to implement the zero-disparity filter, but center it over a number of different disparity ranges. The output would then generate all the possible locations of matches for all possible distances from the robot. The robot could then average this information over time to remove some false matches, and sequentially attend to the remaining matches in order to verify them more closely.

The case for obstacle avoidance and path planning is quite different. Unlike map-making for large-scale navigation, path planing does benefit form knowing at least the approximate or relative 3D positions of obstacles with respect to the robot, and storing these positions in short-term local maps. Furthermore, it requires assessing the locations of every obstacle in the environment — even those obstacles that may be inappropriate as landmarks. Clearly, then, greatly different representations and computations should be used for performing these different tasks.

### 11.4.4 Error Correction

One of the biggest problems of learning diktiometric maps occurs when two places actually look the same with respect to the features used to define them (in this case, invariant landmark views and their visual directions). In the context of representation learning, this problem can be thought of as a case of perceptual aliasing or hidden state (see Section 3.2.1). Even rats might have this problem, as indicated by the rare but present split place fields (see Section 2.2.1). In a localist network with competitive learning dynamics, such an occurrence can wreak havoc with both learning and recognition. The network might constantly adapt templates for a place node that

represents two entirely different areas in the environment. More often than not, the transition templates for such a place come to represent the transitions for neither of the locations, but rather some "average" of the two, resulting in confusion when either place is exited.

A typical way to handle these situations is to split such nodes when detected (see Section 5.2.1). Though the two resulting place nodes still have similar local view templates, they can be disambiguated using the learned transitions. The challenging part is, of course, actually detecting the problem in the first place. One way to do detect a node with a split place region would be to associate a location with each node indicating the approximate center of the region, and splitting a node if it activates sufficiently far away from this location. This tactic unfortunately gets back to a Euclidean way of thinking about maps, which is not only biologically unappealing, but also computationally dangerous due to the problems associated with dead-reckoning. A more appealing way is to split a place node if the expectations generated while taking an action while in its place region are not sufficiently met. Such a mismatch basically says that the robot was not where it thought it was, and that there must be two places that look alike. Such a tactic resembles the reset that accompanies a mismatch between bottom-up and top-down activity in an ART network.

### 11.4.5 Multiple Scales and Environments

Another important part of the HRV approach that has not yet been completely explored here is the use of hierarchy. Currently, the only use of hierarchy in the system concerns the representation of each environment as a map and its evaluation at multiple spatial scales, and the simultaneous consideration of multiple environments for recognition by assessing their evidence over time. The latter capability should be verified in the future, as this thesis has only reported the results of learning in a single environment. But even verifying this capability does not address the problem of deciding where one environment ends and another begins, and how one might plan routes between environments. This problem is where the analogy between objects and environments breaks down. Objects, unlike environments, are finite and self contained, and they do not blend into each other.

Related to this problem is the issue of implementing a hierarchical control structure for planning. This entails learning hierarchical relations (i.e. equivalence classes) between places and environments at various spatial scales. There are several ways of achieving such a hierarchical structure, some of which have already been described and implemented in Section 7.7. Spatial methods for defining places at multiple scales, such as lowering the vigilance in the ART network, are unfortunately not universally applicable for very large scales (i.e. at the environment level), simply because the larger scales can no longer assume that the local views adequately describe the regions that they represent (see discussion in Section 7.7). A more attractive way to learn a large scale representation would be to exclude landmarks within a certain radius from the robot. This might work well for open environments up to a certain scale, but certainly not for indoor or maze-like environments.

Another more abstract way of learning representations at larger scales is using graph morphology. For example, a heading invariant place node might be formed by pooling the set of heading tuned place nodes that might be reached using only pure rotations. At the environment level, new environments might be formed by pooling sets of place nodes with a large number of interconnections

(transitions). Possible mechanisms for performing this pooling include using simply connected components to define places at one scale by entire groups of places and their transitions at the next scale down, perhaps by specifying a minimum number of interconnections for such groups.

The problem of learning transitions between such "meta" places is also difficult. There immediately arises a question of what time scale is appropriate. One possible solution would be to learn maps at multiple temporal as well as spatial scales. But how could one then integrate information from all of these scales, and how would one then store the incredibly large structure that would result? Another possible solution is to find some way of using continuous, yet efficient, representation for actions.

Most likely, the problem of actually using the hierarchical structure to perform planning operations will spawn another whole set of technical issues.

### 11.4.6 Other Work

Finally, a number of operational issues remain to be verified on a mobile robot. First, it should be verified that the robot can reach one of multiple goals (several different place regions). Second, it should be verified that the robot can deduce the appropriate place region goals from indirect associations with other namable objects in the environment. Third, the robot should be allowed to explore around a completely unexplored environment and build up the places, action consequences, and evaluations from scratch in an interleaved, on-line fashion (which is currently rather impractical, given the relatively slow speed of head motions). Fourth, the navigation system should be implemented in continuous time, and the robot should then be capable of searching and tracking landmarks, learning and predicting places, and executing actions simultaneously. Finally, it should be possible to run an interleaved prediction/planning operation in which the robot can plan the entire route to a goal without executing any actions, and use the resulting route to compute an aggregate action or "detour" route to the goal.

Following the pursuit of each of these improvements, it should then be possible to compare the proposed navigation system, with respect to robustness and accuracy, to state-of-the-art approaches to robot navigation.

APPENDIX A

# *The Mobile Adaptive VIsual Navigator (MAVIN)*

This appendix describes the mobile robot MAVIN (Mobile Adaptive VIsual Navigator) and the computer systems it employs to run the various sub-systems described throughout the thesis.

## A.1 The Physical Robot

The physical robot is pictured in Figure 81. It consists of a body, a neck, a head, and a voice box. Each of these hardware units is driven by its own dedicated on-board controller, which takes serial commands via an RS-232 interface. All RS-232 communications are currently transmitted over a tether, which also provides AC power to the turret and carries CCD camera imagery off-board.

### A.1.1 Body

MAVIN's body consists of a Cybermotion K2A mobile platform equipped with a P1A mid-section assembly and a T1A application turret housing. While the base remains stationary, the upper turret rotates with the wheels in a synchro-drive configuration. Although the platform is capable of executing continuous velocity commands, it is primarily operated in an automatic mode in which it executes a path program downloaded from the host computer. It is also occasionally operated in a manual mode in which it is driven by torque commands generated using a joystick. The relevant parameters for the mobile platform are given in Table 13.

### A.1.2 Neck

Atop the turret, mounted on a cylindrical block (see Figure 82), sits a Directed Perception pan/tilt unit, which recently replaced a 5-axis robot arm capable of only position control (see Figure 83). (Note that early experiments reported in Section 7.4 were conducted using the robot arm instead of the pan-tilt unit). This unit serves as a neck for MAVIN's head. The relevant parameters for the pan-tilt unit are given in Table 14.

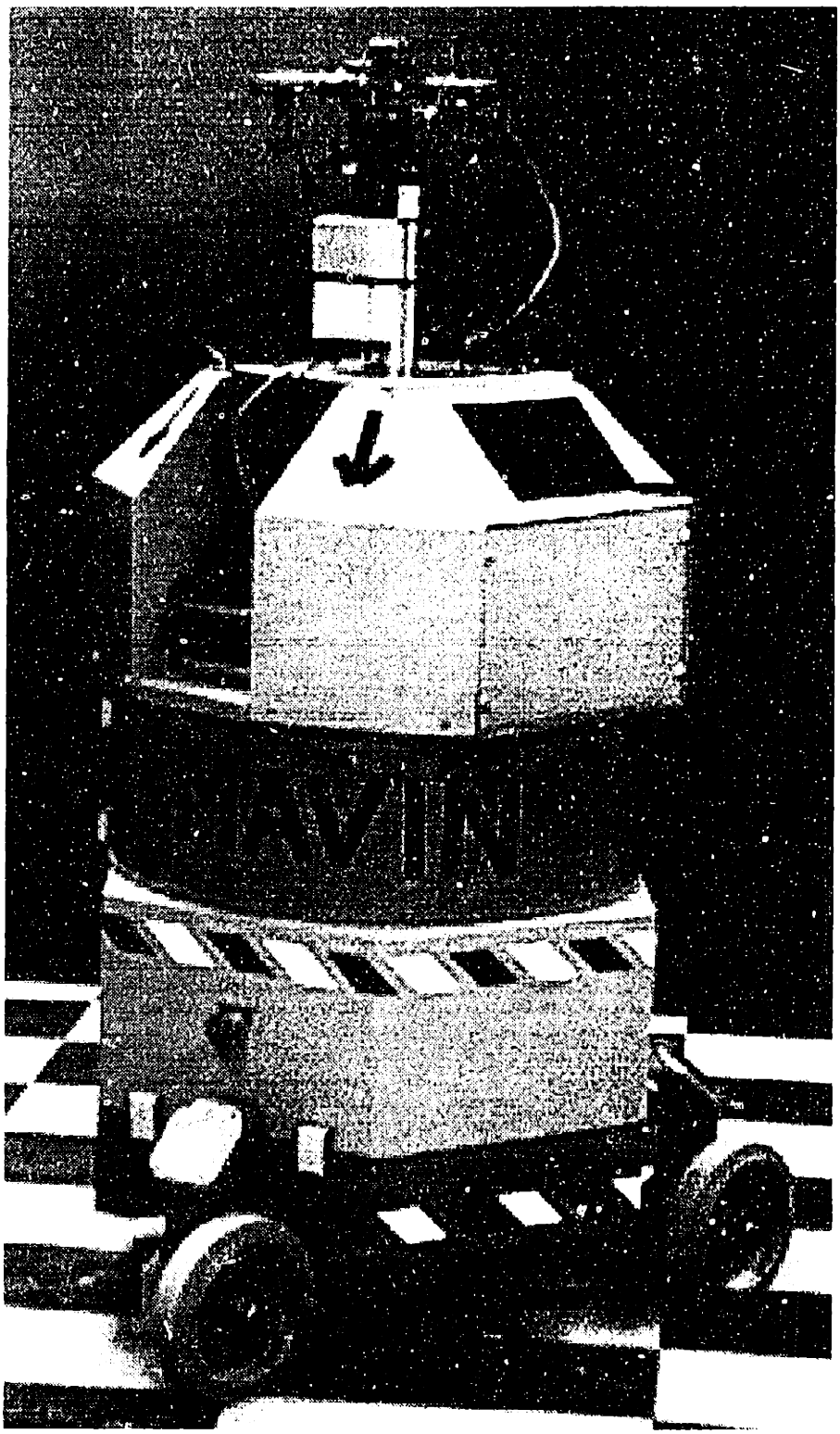

**FIGURE 81: THE MOBILE ADAPTIVE VISUAL NAVIGATOR (MAVIN).**
*MAVIN (the Mobile Adaptive VIsual Navigator) consists of a cybermotion mobile platform, upon which sits a Directed Perception pan-tilt unit serving as a neck for a binocular vergence apparatus capable of directing two Panasonic CCD cameras.*

TABLE 13: BODY PARAMETER VALUES.
*(Cybermotion K2A mobile platform with turret).*

| Parameter | Values |
|---|---|
| Turning velocity (nominal, fully loaded) | 50°/sec |
| Turning velocity (maximum, unloaded) | 250°/sec |
| Driving velocity (nominal, fully loaded) | 0.5 ft./sec |
| Driving velocity (maximum, unloaded) | 2.5 ft./sec |
| Height | 3.5 ft. |
| Diameter | 2 ft. |

## A.1.3 Head

The robot's head consists of a binocular vergence apparatus, built in-house (see Figure 82). The parameters for this vergence apparatus are given in Table 14. This apparatus controls the vergence of two identical CCD cameras, the relevant parameters of which are given in Table 16.

The vergence apparatus is calibrated by directing both cameras at an LED mounted mid way between the cameras (along the baseline). By measuring the image location of this LED for each camera at two different raw vergence angles (using a simple thresholding operation), MAVIN can determine and subsequently correct for the vergence offset and roll of each camera.

Due to play in the stepper motor gearing, the vergence angle is currently not terribly accurate. Note, however, that higher accuracy is possible by taking advantage of the friction in the gearing, which keeps the gears from rotating in the absence of motor movement. Keeping track of the vergence direction (i.e. in or out) allows one to determine at which "side" of the play range the gears reside, and therefore whether or not to add the measurement accuracy parameter value to the current vergence measurement.

## A.1.4 Voice Box

Upon the cylindrical block is mounted an Echo voice synthesizer (see Figure 82), which receives textual messages via an RS-232 interface. This synthesizer aids in demonstrating and debugging.

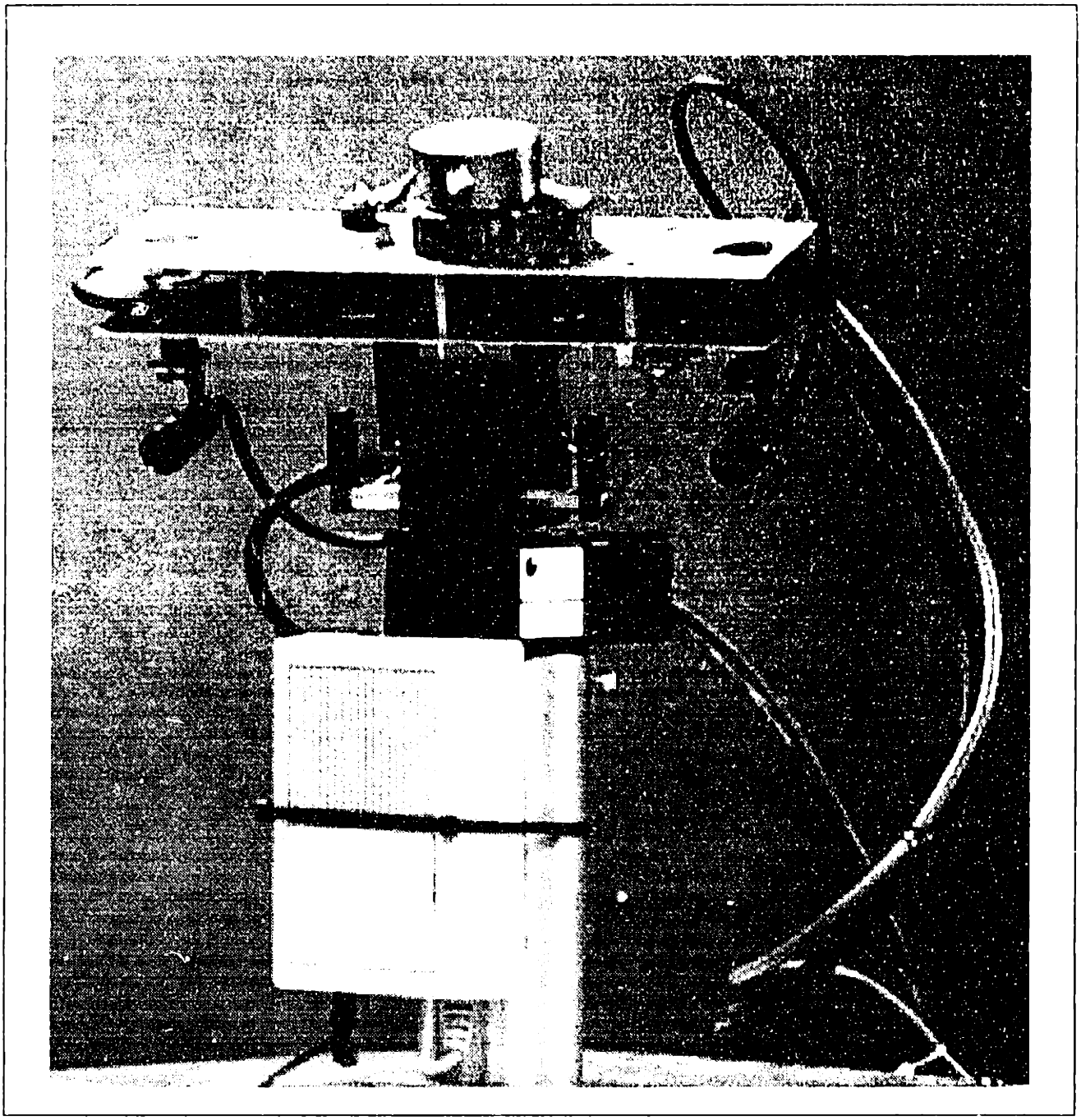

**FIGURE 82: THE HEAD/NECK SYSTEM ON MAVIN.**

*MAVIN's head consists of a binocular vergence apparatus for two Panasonic CCD cameras. The head sits upon a Directed Perception pan-tilt unit mounted atop a cylindrical base. Also mounted on the base is a voice synthesizer.*

TABLE 14: PAN-TILT PARAMETER VALUES.
*(Directed Perception PTU)*

| Parameter | Values |
|---|---|
| Pan and tilt rate (nominal, with vergence apparatus load) | 75°/sec |
| Pan and tilt rate (maximum, unloaded) | 300°/sec |
| Pan and tilt measurement accuracy | 3.086 arc-min |
| Minimum and maximum pan position (nominal range with head load) | -135°-135°. |
| Minimum and maximum tilt position (nominal range with head load) | 0°-45°. |
| Height of tilt axis above robot turret (mounted on cylindrical block) | 10 in. |

TABLE 15: VERGENCE APPARATUS PARAMETER VALUES.

| Parameter | Values |
|---|---|
| Vergence rate (nominal) | 50°/sec |
| Baseline distance | 0.75 ft. |
| Vergence measurement accuracy | 0.25° |

## A.2 Low-Level Vision

An off-board PIPE (Pipelined Image Processing Engine) computer performs real-time low-level visual processing of CCD camera imagery (see Figure 81). Designed by the National Institute of Standards and Technology (NIST), and built by Aspex, Inc., this computer can simultaneously perform local operations (such as 3x3 convolutions and morphological processing) and indirect addressing (such as global shifts) on multiple images at video rate. It can also perform histogramming, which provides a serial host computer with explicit feature lists. The relevant parameters for the PIPE computer are given in Table 16. Only the featural grouping sub-system (FGS) is implemented on this machine.

**TABLE 16: CCD IMAGING PARAMETER VALUES.**
*(Panasonic WV-CD1BW CCD camera with WV-LM7.5T wide angle lens).*

| Parameter | Values |
|---|---|
| Horizontal and vertical fields-of-view ($\Phi_x$ and $\Phi_y$). | 46.2°<br>35.5° |
| Focal length ($f$). | 7.5 mm |
| Imaging area ($I_x$ and $I_y$). | 6.4 mm<br>4.8 mm |
| Horizontal to vertical aspect ratio (each pixel). | 0.75 |

**TABLE 17: PIPE PARAMETER VALUES.**
*(Pipelined Image Processing Engine).*

| Parameter | Values |
|---|---|
| Image resolution | 256x242 pixels |
| Frame rate | 30 frames/sec |

## A.3 High-Level Processing

An off-board Sun Sparcstation 10 performs high-level computations using features extracted from the PIPE computer. This high level processing includes all neural network architecture dynamics. The Sparcstation also controls the voice synthesizer and mobile platform, neck, and vergence movements via a set of serial ports.

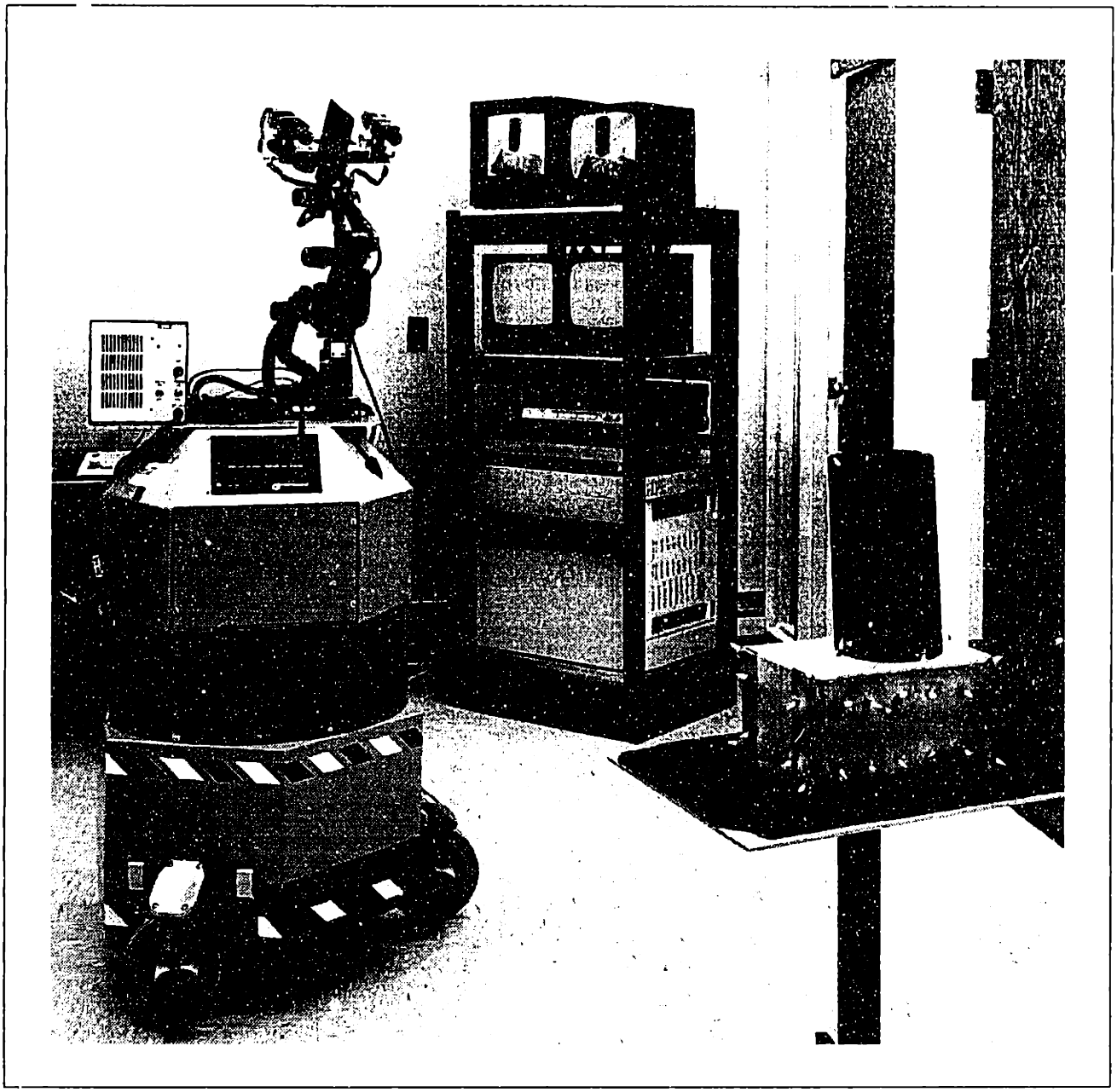

**FIGURE 83: MAVIN AND THE PIPE COMPUTER.**

*Prior to the installation of the pan-tilt unit, MAVIN's neck consisted of a 5 degree-of-freedom robot arm. Raw video imagery from MAVIN's cameras is processed by an Aspex PIPE computer, shown here in the background.*

APPENDIX B

# *Mathematical Notation*

This appendix defines the mathematical notation used to describe the various computations performed by the sub-systems, architectures, networks, and frameworks introduced in Part II and in subsequent appendices. This notation pertains to miscellaneous mathematics (Table 18), linear algebra (Table 18), probability (Table 18), and functions (Table 18).

**TABLE 18: MISCELLANEOUS MATHEMATICAL NOTATION.**

| Notation | Explanation |
|---|---|
| $x \Leftarrow x + 1$ | Reads: "$x$ becomes equal to what $x$ was plus 1." |
| $x\|_{x_{min}}^{x_{max}}$ | The value between $x_{min}$ and $x_{max}$ corresponding to $x$ in a space with periodicity defined by $x_{min}$ and $x_{max}$; i.e., $x_{min} + ((x - x_{min}) \bmod (x_{max} - x_{min}))$. |
| $(x_1 - x_2)_{x_{min}}^{x_{max}}$ | The difference between $x_1$ and $x_2$ with the smallest absolute value in a space with periodicity defined by $x_{min}$ and $x_{max}$; i.e., $(x_1 - x_2)\|_{-(x_{max} - x_{min})/2}^{(x_{max} - x_{min})/2}$ |

**TABLE 19: LINEAR ALGEBRA NOTATION.**

| Notation | Explanation |
|---|---|
| $\bar{v}$ | A vector. |
| $v_i$ | The $i$th element of vector $\bar{v}$. |
| $[x_1, x_2, \ldots, x_n]$ | A vector with elements $x_1, x_2, \ldots, x_n$. |
| $\lvert\bar{v}\rvert$ | The L1 length of $n$-dimensional vector $\bar{v}$, i.e. $\sum_{i=1}^{n} v_i$. |
| $M$ | A matrix or multiple index array. |
| $\bar{M}_i$ | A vector composed of the elements in the $i$th row of matrix $M$. |
| $M_{ij}$ | The element in row $i$ and column $j$ of matrix $M$. |
| cat($M$) | A vector obtained by concatenating all the elements of the $n_1 \times \ldots \times n_m$ array $M$ in "column" major order, i.e. $[M_{11\ldots1}, M_{21\ldots1}, \ldots, M_{n_1 1\ldots1}, M_{121\ldots1\,(1211)}, M_{221\ldots1}, \ldots, M_{n_1 21\ldots1}, \ldots, M_{n_1\ldots n_m}]$. |
| cat($\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m$) | A vector obtained by concatenating all the elements of each of the vectors $\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m$ (with respective dimensions $n_1, n_2, \ldots, n_m$), i.e. $\left[v_1^1, v_2^1, \ldots, v_{n_1}^1, v_1^2, v_2^2, \ldots, v_{n_2}^2, \ldots, v_{n_m}^m\right]$. |
| invcat($\bar{v}, n_1, \ldots, n_m$) | The $n_1 \times \ldots \times n_m$ array $M$ for which cat($M$) = $\bar{v}$. |
| conj($\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m$) | The $n_1 \times n_2 \times \ldots \times n_m$ array $M$ formed by conjunctively cross multiplying all combinations of elements in vectors $\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m$ (with respective dimensions $n_1, n_2, \ldots, n_m$). Each element $M_{i_1 i_2 \ldots i_m}$ is given by $v_{i_1}^1 \cdot v_{i_2}^2 \cdot \ldots \cdot v_{i_m}^m$. |

**TABLE 20: PROBABILITY NOTATION.**

| Notation | Explanation |
|---|---|
| $\Pr(X \mid Y, Z)$ | The probability of event $X$ given that event $Y$ and event $Z$ occur. |
| $\{X\}$ | The history or sequence of events $X$ up to but *not* including the current time. |

**TABLE 21: FUNCTION NOTATION.**

| Notation | Explanation |
|---|---|
| $G(x;c,\sigma)$ | A Gaussian of $x$, centered at $c$, with variance $\sigma$, i.e. $e^{-(x-c)^2/(2\sigma^2)}$. |
| $S(x;c,m)$ | A sigmoid of $x$ with an inflection point at $c$ with slope $m$, i.e. $1/(1+e^{-4m(x-c)})$. |
| $\text{pos}(x)$ | The result of thresholding $x$ at 0, yielding only positive values, i.e. $\text{pos}(x) = \begin{cases} x & \text{if } (x>0) \\ 0 & \text{otherwise} \end{cases}$ |

APPENDIX C

# *Coarse Population Coding*

This appendix defines, using the notation of Appendix B, the operations used to perform coarse population coding in the various neurocomputational architectures described throughout Part II of this thesis. All such population encodings employ a fixed set of nodes with uniformly spaced, uniformly sized overlapping Gaussian receptive fields.

## C.1 1D Receptive Fields

We write $\mathrm{RF}_f(x;n, x_{\min}, x_{\max})$ to denote the result of applying a population of nodes with overlapping 1D Gaussian receptive fields to an input value $x$, where $n$ is the number of nodes in the population, and $x_{\min}$ and $x_{\max}$ define the range of values the population is capable of coding. This application yields an $n$-dimensional vector,

$$\vec{r} = \mathrm{RF}_f(x;n,x_{\min},x_{\max}) \; . \tag{94}$$

Each element $r_i$, representing the activity of population node $i$, is determined using the equation

$$r_i = \mathrm{G}(x;c_i,\sigma) \tag{95}$$

where $c_i$ is the center of the Gaussian receptive field for node $i$, and $\sigma$ is the size of all receptive fields. The collective activities for all nodes are subsequently normalized via

$$r_i \Leftarrow \frac{r_i}{|\vec{r}|} \; . \tag{96}$$

To insure that the receptive fields are evenly spaced, each center $c_i$ is given by

$$c_i = \mathrm{FC}_f(i;n,x_{\min},x_{\max}) = x_{\min} + i \cdot \mathrm{FS}_f(n,x_{\min},x_{\max}) \tag{97}$$

where

$$\mathrm{FS}_f(n,x_{\min},x_{\max}) = (x_{\max} - x_{\min})/n \tag{98}$$

is the spacing between field centers. To insure that each of the receptive fields overlap sufficiently, the size $\sigma$ of each receptive field is given by

$$\sigma = \mathrm{FS}(n,x_{\min},x_{\max}). \tag{99}$$

# C.2 Periodic 1D Receptive Fields

A variation of this coarse coding results when the input space is periodic. In this case we write $\mathrm{RF}_p(x;n,x_{\min},x_{\max})$. The application again yields an $n$-dimensional vector,

$$\bar{r} = \mathrm{RF}_p(x;n,x_{\min},x_{\max}) \tag{100}$$

where

$$r_i = \max\left( \mathrm{G}\left( x|_{x_{\min}}^{x_{\max}};c_i,\sigma\right), \mathrm{G}\left( x|_{x_{\max}}^{2x_{\max}-x_{\min}};c_i,\sigma\right), \mathrm{G}\left( x|_{2x_{\min}-x_{\max}}^{x_{\min}};c_i,\sigma\right)\right) \tag{101}$$

$$r_i \Leftarrow \frac{r_i}{|\bar{r}|} \tag{102}$$

$$c_i = \mathrm{FC}_p(i;n,x_{\min},x_{\max}) = x_{\min} + i \cdot \mathrm{FS}_p(n,x_{\min},x_{\max}) \tag{103}$$

$$\mathrm{FS}_p(n,x_{\min},x_{\max}) = (x_{\max} - x_{\min})/(n+1) \tag{104}$$

$$\sigma = \mathrm{FS}(n,x_{\min},x_{\max}). \tag{105}$$

# C.3 Multi-dimensional Receptive Fields

Both periodic and non-periodic receptive field coding can be extended beyond the 1D case by taking advantage of the separability of the Gaussian; that is, by multiplicatively (conjunctively) combining the results from independent population codes for each of the input dimensions. Given a point in an $m$-dimensional space $\bar{x}$, we write $\mathrm{RF}^m(\bar{p};\bar{n},\bar{x}^{\min},\bar{x}^{\max},\Phi)$ to denote the result of applying a population of nodes with overlapping $m$-dimensional Gaussian receptive fields, where $\bar{n}$ is an $m$-dimensional vector containing the number of nodes for each dimension, $\bar{x}^{\min}$ and $\bar{x}^{\max}$ are vectors defining the range of values for each dimension, and $\Phi$ is an $m$-dimensional vector indicating which of the dimensions are infinite ($f$) or periodic ($p$). This application yields an $(n_1 \times n_2 \times \ldots \times n_m)$ array,

$$M = \mathrm{RF}^m(\bar{p};\bar{n},\bar{x}^{\min},\bar{x}^{\max},\Phi) \tag{106}$$

determined using

$$\boldsymbol{M} = \mathrm{conj}\left( \mathrm{RF}_{\varphi_1}\left( x_1; n_1, x_1^{\min}, x_1^{\max} \right), \mathrm{RF}_{\varphi_2}\left( x_2; n_2, x_2^{\min}, x_2^{\max} \right), \ \ldots, \mathrm{RF}_{\varphi_m}\left( x_m; n_m, x_m^{\min}, x_m^{\max} \right) \right). \tag{107}$$

The the center of the receptive field with indices $i_1 \ldots i_m$ is given by

$$\bar{c}_{i_1 \ldots i_m} = \mathrm{FC}^m(i_1 \ldots i_m; \bar{n}, \bar{x}^{\min}, \bar{x}^{\max}, \Phi) \tag{108}$$

determined using

$$\bar{c}^{\,i_1 \ldots i_m} = \left[ \mathrm{FC}_{\varphi_1}\left( i_1; n_1, x_1^{\min}, x_1^{\max} \right), \mathrm{FC}_{\varphi_2}\left( i_2; n_2, x_2^{\min}, x_2^{\max} \right), \ \ldots, \mathrm{FC}_{\varphi_m}\left( i_m; n_m, x_m^{\min}, x_m^{\max} \right) \right]. \tag{109}$$

APPENDIX D

# *Localist Associative Network Dynamics*

This appendix defines, using the notation of Appendix B, the dynamics used to implement the associative ANNs employed in Part II of this thesis. The use of these ANNs require localist representations of inputs and outputs; that is, each element in the input and output patterns must be tuned to a unique range of conditions (see Section 3.2.5). It also requires that the desired output changes gradually (continuously) with the input.

## D.1 Structure

The heteroassociative network consists of a set of $n_I$ input nodes, a set of $n_O$ output nodes, and an $n_I \times n_O$ weight matrix,

$$W = \begin{bmatrix} [\bar{w}^1]^T \\ [\bar{w}^2]^T \\ \cdots \\ [\bar{w}^{n_I}]^T \end{bmatrix}. \tag{110}$$

Each vector $\bar{w}^i$ of $W$ is the sparse "outstar" weight template for input node $i$, and codes for the expected activity of the output nodes that follows the activation of input node $i$. In other words, $w_j^i$ is the synaptic weight between input node $i$ and output node $j$. Initially, each of the weights are set to 0. The sparseness of these outstars, due primarily to the localist nature of the network, allows them to be represented by sparse matrices, which in turn makes implementing the network on serial computers much more time and space efficient.

The network also maintains an $n_I$-dimensional completion vector $\bar{h}$ (initially reset to 0) that indicates which templates have already been learned. This completion vector allows even more sparse representation, since only the outstars $i$ for which $h_i = 1$ must be stored explicitly.

## D.2 Prediction

The network takes as input an $n_I$-dimensional vector $I$ and an $n_C$-dimensional vector $\bar{D}$. It computes a prediction $P$, the activities of the output nodes, via the equation

$$P = \frac{W^T I}{\sum_{i=1}^{n_I} h_i I_i}. \tag{111}$$

In essence, it computes a weighted average of the learned templates. The network might also be used in the reverse direction, in order to compute the "likelihood" that each of the input elements contributed to the output, via the equation

$$I_i = \bar{w}_i \cdot \bar{D}. \tag{112}$$

This backwards propagation is similar to the bottom-up categorization performed by an ART network (see Carpenter and Grossberg, 1991), only no decision is made.

## D.3 Learning

To learn each of the weight templates, the network employs a competitive learning rule. First, it computes the index $i_{\max}$ corresponding to the maximally active element of $I$ (chooses a winner). That is,

$$i_{\max} = \max_i (I_i). \tag{113}$$

Next, it either learns the template for the winning input node using a fast (one-shot) learning rule if the weight template has not yet been learned, or adapts the template using a slow learning rule (Hebbian learning with post-synaptically gated decay) if it has. More specifically, if $h_{i_{\max}} = 0$, then

$$\bar{w}^{i_{\max}} = I \tag{114}$$

and

$$h_{i_{\max}} = 1. \tag{115}$$

Otherwise,

$$\frac{d}{dt}\left(\bar{w}^{i_{\max}}\right) = \beta I_{i_{\max}}\left(\frac{\bar{D}}{|\bar{D}|} - \bar{w}^{i_{\max}}\right) \tag{116}$$

where $\beta$ is the learning rate, and $n_D$ is the number of output nodes.

Note that it is possible to accurately learn templates in this manner only if the input $I$ is a sparse localist pattern, and only if the inputs that produce different outputs $D$ are linearly independent. The learning rule is essentially the dual of the competitive bottom-up learning rule used in an ART network. While an ART network learns a bottom-up "instar" (and a top-down "outstar") for each localist *output* category node (in the F2 layer) via Hebbian learning with *post*-synaptically gated decay, this network learns an "outstar" for each *input* node via Hebbian learning with *pre*-synaptically gated decay (see Simpson, 1990). In both cases, learning only occurs for the winning (most active) node, which helps keep interference between unrelated weights, and the ensuing unwanted re-coding, to a minimum.

## D.4 Variations

The applications for which the localist associative network are utilized in this thesis require that it be capable of handling dynamically growing input and output patterns. The growing of input patterns poses little challenge, since templates for input nodes have no impact upon prediction and learning until they undergo fast learning. The network simply increases the number of rows of its sparse weight matrix and enlarges and initializes the completion vector $h$. The growing of output patterns is a bit more tricky, because it involves enlarging each of the learned templates and initializing each of the new elements of these templates to zero.

A more sophisticated approach to the growing of output vectors is to keep a vector of counters, rather than a completion vector, for the learned templates. Each counter keeps track of the proportion of its corresponding template that has already been learned. In this fashion, the fast learning rule could be employed for the unlearned portion of the pattern, while the slow learning rule could be used to adapt the previously learned portion of the pattern.

APPENDIX E

# *Markov Decision Processes*

This appendix reviews the pertinent elements of a *Markov decision process* (MDP). For more detailed information, see White (1993) or Cassandra et. al. (1994).

## E.1 Definitions

An MDP, often called a *controlled Markov process*, is an abstract probabilistic model consisting of:

1. A finite set of states, $\{1 \ldots n_s\}$.
2. A finite set of actions, $\{1 \ldots n_a\}$.
3. A probability distribution $\bar{w}^{ij}$ over the states for each state-action pair $ij$.
4. An expected reward $r_{ij}$ for each state-action pair $ij$.

If $P_i$ stands for the statement "The process is currently in state $i$," $P_i'$ stands for the statement "The process was in state $i$ before the last action was performed," and $A_j$ stands for the statement "Action $j$ was just performed," then the probability of transitioning to state $l$ from state $i$ after having executed action $j$ is given by

$$w_l^{ij} = \Pr(P_l | P_i', A_j). \tag{117}$$

A very important assumption underlying an MDP is the *Markov property*, which states that transition probabilities only depend upon the current state and action, not upon any previous history. Another way of stating this property is by the independence relation,

$$\Pr(P_i | P_l, A_j, \{A, P\}) = \Pr(P_i | P_l', A_j). \tag{118}$$

## E.2 Optimal Policy Determination

An MDP can be used to determine an optimal action policy, which defines the best action $z_i$ for each state $i$, in the sense that following the policy achieves the highest possible reward over some specified time frame. A simple way to determine this policy for an *infinite horizon* (for all time) is using a technique called *value iteration*, a form of dynamic (tabular) programming. This technique entails computing a set of value ratings $\bar{v}$ for each of the states in the MDP, and a set of value ratings $Q$ for each of the states-action pairs (sometimes called the "Q-values" — see Watkins, 1989). The value $v_i$ is the expected sum of the rewards received in the future, given that one follows the optimal policy from state $i$, while the value $Q_{ij}$ is the expected sum of the rewards received in the future, given that one follows the optimal policy after executing action $j$ in state $i$. Assuming that it is possible to reach all states from any given starting state, the computation involves iteratively solving the following set of simultaneous equations (derived using the principal of optimality):

$$\forall (i,j) \quad Q_{ij} = \bar{w}^{ij}\bar{v} + r_{ij} \tag{119}$$

$$\forall i \quad z_i = \underset{\forall j}{\text{argmax}}(Q_{ij}) \tag{120}$$

$$\forall i \quad v_i = Q_{iz_i}. \tag{121}$$

The iteration of these equations (in this order) terminates when each of the $v_i$ values change less than some threshold $\varepsilon$. This termination condition, which is guaranteed to occur, in general requires $O(n_s^2)$ time to occur. Note that another technique, called *policy iteration*, differs from this procedure only in that equation (121) is calculated before equation (120) in each iteration. That is, the next set of values for $\bar{v}$ is determined by the current policy $\bar{z}$.

## E.3 Partial Observability

A *partially observable* MDP is an extension of an MDP wherein at any given time only a probability of each of the states, given by

$$p_i = \Pr(P_i|O) \tag{122}$$

is known, where $O$ is the current observation. This distribution is called the *belief state* (Cassandra et. al., 1994). The Markov assumption now becomes

$$\Pr(P_i|P_l', A_j, \{A, O\}) = \Pr(P_i|P_l', A_j). \tag{123}$$

In order to estimate the current belief state, one can compute, using Bayes' rule, the *a posteriori* probability distribution for the states after executing an action $a$ via the equation

$$p_i = b_i t_i / (\bar{b} \cdot \bar{t}) \tag{124}$$

where

$$\bar{t} = \sum_{k=1}^{n_s} p_k \bar{w}^{ka} \tag{125}$$

and $b_i$ is the prior probability for the current observation $O$ in state $i$ after executing action $a$, i.e.

$$b_i = \Pr(O|P_i', A_a). \tag{126}$$

Determining the optimal policy for a POMDP is similar to optimal policy determination for an MDP, but is much more complicated. Value iteration for a POMDP in general takes exponential time with respect to the number of states. More efficient algorithms exist (e.g. Cassandra et. al., 1994), but assume that there are a finite number of possible observations $O$. Note, however, that in practice satisficing (nearly optimal) solutions are often found in a relatively few number of value iterations. Furthermore, nearly optimal solutions are often found simply by treating the problem as an MDP, solving for the optimal policy $\bar{z}$, and determining the next action for the current belief state using the equation

$$a = \underset{\forall j}{\text{argmax}} \left( \sum_{i=1}^{n_s} p_i Q_{ij} \right). \tag{127}$$

This procedure is like saying that the environment becomes completely observable after the next action has been executed (see Cassandra et. al., 1994).

# *References*


Abbott, A. L. (1992). A survey of selective fixation control for machine vision. *IEEE Control Systems*, 12 (4), 25-31.

Alami, R., Chatila, R., & Freedman, P. (1990). Task-level teleprogramming for intervention robots. In *Proceedings of the IARP Workshop on Mobile Robots for Subsea Environments,* October 23-26, Monterey, CA, pp. 119-136.

Albus, J. (1985). Hierarchical control for robots and teleoperators. In *Proceedings of the IEEE Workshop on Intelligent Control*, August 26-27, Albany, NY, 39-49.

Alexander, H. L. (1990). Experiments in teleoperator and autonomous control of space robotic vehicles. i-SAIRAS'90, B31-4, 217-220.

Anderson, C. W. (1987). Strategy learning with multilayer connectionist representations. In *Proceedings of the Fourth International Workshop on Machine Learning*, 103-114.

Asada, H., & Yang, B-H (1989). Skill acquisition from human expert through pattern processing of teaching data. In *Proceedings of IEEE International Conference on Robotics and Automation,* May 14-19, Scottsdale, AZ, 1302-1307.

Asada, M. (1990). Map building for a mobile robot from sensory data. *IEEE Transactions on Systems, Man, and Cybernetics*, 37 (6), 1326-1336.

Asada, M., Fukui, Y., & Tsuji, S. (1990). Representing a global world of a mobile robot with relational local maps. *IEEE Transactions on Systems, Man, and Cybernetics*, 20 (6), 1456-1461.

Bachelder, I. A. & Waxman, A. M. (1992). Neural networks for mobile robot visual exploration. In SPIE Proceedings 1831, Mobile Robots VII, 107-119.

Bachelder, I. A. & Waxman, A. M. (1994). Mobile robot visual mapping and localization: A view-based neurocomputational architecture that emulates hippocampal place learning. *Neural Networks*, 7 (6/7), 1994 Special Issue on Models of Neurodynamics and Behavior, 1083-1100.

Bachelder, I. A., & Waxman, A. M. (1995). A view-based neurocomputational system for relational map-making and navigation in visual environments. *Robotics and Autonomous Systems*, in press.

Bachelder, I. A. Waxman, A. M., & Seibert, M. (1993). A neural system for mobile robot visual place learning and recognition. In *Proceedings of 1993 World Congress on Neural Net-*

*works*, pp. 512-518, Portland, OR; also in *Proceedings of the Artificial Neural Networks In Engineering* (ANNIE'93), pp. 343-351, St. Louis, MO.

Bachelder, I. A., Waxman, A. M., Seibert, M., & Gove, A. N. (1994). From learning objects to learning environments: Biological and computational neural systems. In *Proceedings of the ARPA Image Understanding Workshop*, November 13-18, Monterey, CA, 871-883.

Ballard, D. H., & Brown, C. M. (1982). *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Baloch, A. A., & Waxman, A. M. (1991a). Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN. *Neural Networks*, 4, 271-302.

Baloch, A. A., & Waxman, A. M. (1991b). Behavioral conditioning of the mobile robot MAVIN. In *Neural Networks, Concepts, Applications, and Implementations*, Vol IV (eds. P. Antognetti & V. Milutinovic). 162-200, Englewood Cliffs, New Jersey: Prentice Hall.

Barto, A. G., Sutton, S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 5, 834-845.

Beck, J., & Prazdny, K. (1983). A theory of textural segmentation. *Human and Machine Vision* (J. Beck, B. Hope, & A. Rosenfeld, eds), pp. 1-38, New York, NY: Academic Press.

Beer, R. D., & Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1 (1), 91-122.

Bellingham, J. G., & Humphrey, D. (1990). Using layered control for supervisory control of underwater vehicles. In *Conference Proceedings ROV 1990 IEEE Marine Technology Society*, June 25-27, Vancouver, BC, 175-181.

Benedikt, M. L. (1979). To take hold of space: isovists and isovist fields. *Environment and Planning*, B6, 47-65.

Bernardon, A. M., & Carrick, J. E. (1995). A neural system for automatic target learning and recognition applied to bare and camouflaged SAR targets. *Neural Networks, Special Issue on ATR*, 8 (November/December), in press.

Betke, M.., Rivest, R. L., & Singh, M. (1994). Piecemeal learning of an unknown environment. AI Memo 1474, MIT Artificial Intelligence Laboratory, March.

Betke, M., & Makris, N. C. (1994). Fast object recognition in noisy images using simulated annealing. AI Memo 1510, MIT Artificial Intelligence Laboratory, December.

Bourlard, H., & Wellekens, C. J. (1988). Links between markov chains and multilayer perceptrons. Technical report manuscript M-263, Phillips Research Laboratory, Brussels, Belgium.

Braunegg, D. J. (1993). MARVEL: A system for recognizing world locations with stereo vision. *IEEE Transactions on Robotics and Automation*, 9 (3), 303-308.

Brooks, M. (1989). Proposal for a pattern matching task controller for sensor-based coordination of robot motions. In *Proceedings of NATO Advanced Research Workshop: Robots and Biological Systems*, June, Il Ciocco, Tuscany, Italy, 26-30.

Brooks, R. A. (1981). Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17, 285-348.

Brooks, R. A. (1985). Visual map-making for a mobile robot. In *Readings in Computer Vision* (M. A. Fischler & O. Firschein, eds.), pp. 438-443. Los Altos, CA: Morgan Kaufmann.

Brooks, R. A. (1991). Intelligence without reason. In *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI-91), Sydney, Australia, 569-595.

Brown, C., Coombs, D., and Soong, J. (1992). Real-time smooth pursuit tracking. In *Active Vision* (A. Blake and A. Yuille, eds.), pp. 123-136. Cambridge, MA: MIT Press.

Brunnstrom, K. (1993). Active exploration of static scenes. Dissertation, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden.

Buharali, A., & Sheridan, T. B. (1982). Fuzzy set aids for telling a computer how to decide. In *Proceedings of the 1982 International Conference on Cybernetics and Society*, 82-CH-1840-8, Seattle, WA.

Burgess, N, Recce, M., & O'Keefe, J. (1995). Hippocampus: Spatial models. In *The Handbook of Brain Theory and Neural Networks* (eds. M. Arbib and P. Arbib), pp. 468-472, Cambridge, MA: MIT Press.

Cannon, D. (1992). Point-and-direct telerobotics: interactive supervisory control at the object level in unstructured human-machine system environments. PhD Thesis, Stanford University, Stanford, CA.

Caelli, T. (1985). Three processing characteristics on visual texture segmentation. *Spatial Vision*, 1 (1), 19-30.

Carpenter, G. A., & Grossberg, S. (1991). *Pattern Recognition by Self-Organizing Neural Networks* (Part III: Adaptive Resonance Theory). Cambridge, MA: MIT Press.

Cassandra, A. R., Kaelbling, L. P., & Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings AAAI-94*, Seattle, WA.

Cartwright, B. A., & Collett, T. S. (1987). Landmark maps for honeybees. *Biological Cybernetics*, 57, 85-93.

Chapman, D., & Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings IJCAI-91*, Sydney, Australia.

Chatila, R., & Laumond, J. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 138-170.

Clark, J. J., & Ferrier, N. J. (1992). Attentive visual servoing. In *Active Vision* (A. Blake and A. Yuille, eds.), pp. 137-154. Cambridge, MA: MIT Press.

Clouse, J. A., & Utgoff, P. E. (1992). A teaching method for reinforcement learning. In Machine Learning: *Proceedings of the Ninth International Conference*, pp. 92-101, San Mateo, CA: Morgan Kaufmann.

Cole, B. L., & Hughes, P. K. (1990). Drivers don't search: they just notice. In *Visual Search* (D. Brogan, ed), pp. 407-417, New York, NY: Taylor & Francis.

Colombetti, M., & Dorigo, M. (1994). Training agents to perform sequential behavior. *Adaptive Behavior,* 2 (3), 247-275.

Cunningham, R. K., & Waxman, A. M. (1994a). Diffusion-enhancement bilayer: realizing long-range apparent motion and spatiotemporal grouping in a neural architecture. *Neural Networks*, 7 (6/7), 895-924.

Cunningham, R. K., & Waxman, A.M., (1994b). Learning image motion fields of 3D objects in motion. In *Proceedings of the World Congress on Neural Networks* (WCNN'94), June 5-9, San Diego, Vol. IV, 632-637.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of National Conference on Artificial Intelligence*. MIT Press.

Davis, E. (1986). *Representing and Acquiring Geographic Knowledge*. Los Altos, CA: Morgan Kaufmann.

De Renzi, E. (1982). *Disorders of Space Exploration and Cognition*. New York, NY: John Wiley & Sons.

Dean, T., Basye, K., & Lejter, M. (1990). Planning and active perception. In *Autonomous Mobile Robots: Control, Planning, and Architecture*, Vol 2 (S. S. Iyengar & A. Elfes, eds.), pp. 197-202. Los Alamitos, CA: IEEE Computer Society Press.

DeYoe, E.A., & Van Essen, D.C. (1988). Concurrent processing streams in monkey visual cortex. *Trends in Neuroscience*, TINS 11, 219-226.

Desimone, R. (1992). Neural circuits for visual attention in the primate brain. In *Neural Networks for Vision and Image Processing* (G. A. Carpenter & S. Grossberg, eds.), pp. 343-364. Cambridge, MA: The MIT Press.

Drasic, D. (1991). Skill acquisition and task performance in teleoperation using monoscopic and stereoscopic video remote viewing. In *Proceedings of the Human Factors Society 35th Annual Meeting*, 1367-1371.

Dresher, G. L. (1987). *Made-up Minds: A Constructivist Approach to Artificial Intelligence.* Cambridge, MA: The MIT Press.

Drumheller, M. (1987). Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9 (2), 325-332.

Eichenbaum, H., & Cohen, N. J. (1988). Representation in the hippocampus: what do hippocampal neurons encode? *Trends in Neuroscience*, 11, 244-248.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3 (3), 249-265.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7, 195-225.

Engelson, S., & McDermott, D. V. (1991). Image signatures for place recognition and map construction. In *Proceedings of SPIE Symposium on Intelligent Robotic Systems: Sensor Fusion IV*, 282-293.

Ferrell, W. R. (1973). Command language for supervisory control of remote manipulation. In *Proceedings of the First National Conference*, Sept 13-15, 1972, Pasadena, CA.

Fielding, K. H., Ruck, D. W., Rogers, S. K., Welsh, B. M, & Oxley, M. E. (1993). Spatio-temporal pattern recognition using hidden Markov models. In *Neural and Stochastic Methods in Image and Signal Processing II, SPIE Proceedings, Vol. 2032*, San Diego, CA, July 12.

Funda, J., & Paul, R. P. (1992). Teleprogramming: Toward delay-invariant remote manipulation. *Presence*, 1 (1), 29-44.

Freedy, A., & Weltman, G. (1973). Remotely manned systems: exploration and operation in space. In *Proceedings of the First National Conference*, Sept 13-15, Pasadena, CA, 1972, 397-407.

Gaussier, P., and Zrehen, S. (1995). PerAc: A neural architecture for artificial animals. *Robotics and Autonomous Systems*, in press.

Gould, J. L. (1986). The locale map of honey bees: Do insects have cognitive maps? *Science*, 232, 861-863.

Grefenstette, J. J. (1992). The evolution of strategies for multiagent environments. *Adaptive Behavior*, 1 (1), 65-90.

Grimson, W. E. L. (1990). *Object Recognition by Computer.* Cambridge, MA: MIT Press.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks*, 1, 17-61.

Grossberg, S. (1994). 3-D vision and figure-ground separation by visual cortex. *Perception and Psychophysics*, 55 (1), 48-120.

Grossberg, S., Mingolla, E., & Ross, W. D. (1994). A neural theory of attentive visual search: Interactions of boundary, surface, and object representations. *Psychological Review*, 101 (3), 470-489.

Grossberg, S., and Schmajuk, N. A. (1988). Neural dynamics of attentionally modulated Pavlovian conditioning: Conditioned reinforcement, inhibition, and opponent processing. In *Neural Networks and Natural Intelligence* (S. Grossberg, ed.), Cambridge, MA: The MIT Press/Bradford Books.

Grossman, D. D. (1977). Programming a computer controlled manipulator by guiding through the motions. Technical Report, IBM T. J. Watson Research Center, March.

Haralick, R. M., & Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29, 100-132.

Hetherington, P. A., & Shapiro, M. L. (1993). A simple network model simulates hippocampal place fields: '. Computing goal-directed trajectories and memory fields. *Behavioral Neuroscience*, 107 (3), 434-443.

Heeter, C. (1992). Being there: the subjective experience of presence. *Presence: Teleoperators and Virtual Environments*, 1 (2), 262-271.

Held, R. M., & Durlach, N. I. (1992). Telepresence. *Presence: Teleoperators and Virtual Environments*, 1, 109-112.

Hinton, G. E. (1990). Mapping whole-part hierarchies into connectionist networks. *Artificial Intelligence*, 46 (1-2), 47-76.

Hirzinger, G., & Landzettel, K. (1985). Sensory feedback structures for robots with supervised learning. *IEEE International Conference on Robotics and Automation*, March, St. Louis, MO, 627-635.

Holding, D. H. (1987). Concepts in training. In *Handbook of Human Factors* (Salvendy, ed.), Wiley.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, 1-7.

Horswill, I. (1993). Specialization of perceptual processes. PhD Dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA.

Huttenlocher, D. P., & Ullman, S. (1980). Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5 (2), 195-212.

Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning - a survey. *ACM Computing Surveys*, 24 (3), 219-291.

Isermann, R., Lachmann, K. -H., & Matko, D. (1992). *Adaptive Control Systems*. New York: Prentice Hall.

Jacobs, D. W. (1989). Grouping for recognition. A.I. Memo 1117, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Jacobs, R. A., & Jordan, M. I. (1993). Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23 (2), 337-345.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16, 307-354.

Julesz, B., & Bergen, J. R. (1985). Textons, the fundamental elements in preattentive vision and perception of textures. In *Readings in Computer Vision* (M. A. Fischler & O. Firshchein, eds), pp. 243-256. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Kaelbling, L. P. (1990). *Learning in embedded systems*. Cambridge, MA: MIT Press.

Kaplan, S. (1973). Cognitive maps, human needs and the designed environment. In *Environmental Design Research*, Vol. 1 (W. F. E. Preiser, ed.), 275-283. Stroudsburg, PA: Dowden, Hutchinson, and Ross.

Kim, W. S., Ellis, S. R., Tyler, M., & Stark, L. (1985). Visual enhancements for telerobotics perspective parameters. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 807-811.

Kitchen, L., & Rosenfeld, A. (1982). Grey-level corner detection. *Pattern Recognition Letters*, 1 (2), 95-102.

Koch, C., & Ullman, S. (1985). Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4, 219-227

Koenderink, J. J., & van Doorn, A. J. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32, 211-216.

Kohler, W. (1925). *The Mentality of Apes*. New York: Harcourt Brace and World

Koza, J. R., Rice, J. P., & Roughgarden, J. (1992). Evolution of food-foraging strategies for caribbean anolis lizard using genetic programming. *Adaptive Behavior*, 1 (2), 171-199.

Kuipers, B. J. (1978). Modeling spatial knowledge. *Cognitive Science*, 2, 129-153. Reprinted in *Advances in Spatial Reasoning*, Vol. 2 (S. Chen, ed.), pp. 171-198. Norwood, NJ: Ablex Publishing, 1990.

Kuipers, B. J. (1979). Commonsense knowledge of space: Learning from experience. In *Proceedings of the 6th International Conference on AI* (IJCAI 79), Los Altos, CA Reprinted in *Advances in Spatial Reasoning*, Vol. 2 (ed. S. Chen), 199-206, Norwood, NJ: Ablex Publishing.

Kuipers, B. J. (1983). Modelling human knowledge of routes: Partial knowledge and individual variation. In *Proceedings AAAI-83*, 216-219.

Kuipers, B. J., & Byun, Y-T (1990). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Technical Report AI90-120, Department of Computer Sciences, University of Texas at Austin.

Kuipers, B. J., & Levitt, T. S. (1988). Navigation and mapping in large-scale space. *AI Magazine*, 9 (2), 25-43. Reprinted in Advances in Spatial Reasoning, Vol. 2 (ed. S. Chen). 207-251, Norwood, NJ: Ablex Publishing, 1990.

Kupferman, I. (1985). Learning. In *Principals of Neural Science* (E. R. Kandel & J. H. Schwartz, eds.), pp. 805-815. New York, NY: Elsevier.

Leonard, J., & Cox, I. J. (1990). Modelling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66, 311-344.

Levitt, T. S., & Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence*, 44 (3), 305-360.

Lin, L. (1993). Reinforcement Learning for Robots Using Neural Networks. PhD Thesis, Carnegie Mellon University.

Lippmann, R. P. (1989a). Pattern classification using neural networks. *IEEE Communications Magazine*, November, 47-63.

Lippmann, R. P. (1989b). Review of neural networks for speech recognition. *Neural Computation*, 1, 1-38.

Lowe, D. (1985). *Perceptual organization and visual recognition.* The Netherlands: Kluwer Academic Publishers.

Luo, R. C., & Kay, M. G. (1989). Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19 (5), 901-931.

Maes, P., & Brooks, R. A. (1990). Learning to coordinate behaviors. In *Proceedings AAAI-90.* Reprinted in *Autonomous Mobile Robots: Control, Planning, and Architecture*, Vol. 2 (S. S. Iyengar & A. Elfes, eds.), pp. 224-230. Los Alamitos, CA: IEEE Computer Society Press.

Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55, 311-365.

Malkin, P. K., & Addanki, S. (1990). LOGnets: A hybrid spatial representation for robot navigation. In *Proceedings of the AAAI Eighth National Conference on Artificial Intelligence* (AAAI-90), 1045-1050.

Mataric, M. J. (1991). A comparative analysis of reinforcement learning methods. A.I. Memo 1322, MIT Artificial Intelligence Laboratory, October.

Mataric, M. J. (1992). Integration of representation into goal-directed behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8 (3), 304-312.

Mataric, M. J. (1994). Interaction and intelligent behavior. TR-1495, MIT Artificial Intelligence Laboratory.

McDermott, D. (1992). Robot planning. *AI Magazine*, 13 (2), 55-79.

McGreevy, M. W. (1992). The presence of field geologists in Mars-like terrain. *Presence: Teleoperators and Virtual Environments*, 1 (4), 375-403.

McNaughton, B. L. (1989). Neuronal mechanisms for spatial computation and information storage. In *Neural Connections, Mental Computation* (L. Nadel, L. A. Cooper, P. Culicover, & R. M. Harnish, eds.), pp. 285-350. Cambridge, MA: MIT Press.

McNaughton, B. L., Barnes, C. A., & O'Keefe, J. (1983). The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely moving rats. *Experimental Brain Research*, 52, 41-49.

McNaughton, B. L., Chen, L. L., & Markus, E. J. (1991). *'Dead-reckoning,' landmark learning, and the sense of direction: A neurophysiological and computational hypothesis.* Journal of Cognitive Neuroscience, 3 (2), 190-202.

Miller, W. T., Sutton, R. S., and Werbos, P. J. (1990). *Neural Networks for Control.* Cambridge, MA: MIT Press.

Minai, A. A., & Levy, W. B. (1993a). Sequence learning in a single trial. In *Proceedings of the World Congress on Neural Networks* (WCNN'93), Portland, OR.

Minai, A. A., & Levy, W. B. (1993b). Predicting complex behavior in sparse asymmetric networks. In *Proceedings of Neural Information Processing Systems: Natural and Synthetic*, Vol. 5, 556-563.

Mishkin, M., Ungerleider, L. G., & Macko, K. A. (1983). Object vision and spatial vision: two cortical pathways. *Trends in Neuroscience,* TINS 6, 414-417.

Miyashita, Y. (1988). Neural correlates of visual associative long-term memory in the primate temporal cortex. *Nature*, 335, 817-820.

Moore, A. W. (1991). Variable resolution dynamic programming: efficiently learning action maps in multivariate real-valued state spaces. In *Eighth International Workshop on Machine Learning*, Morgan Kaufmann.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and time. *Machine Learning*, 13, 103-130.

Moravec, H. P. (1981). *Robot Rover Visual Navigation.* Ann Arbor, MI: UMI Research Press.

Morgan, D. P., & Scofield, C. L. (1991). *Neural networks and speech processing.* Boston, MA: Kluwer Academic Publishers.

Mozer, M. C., & Bachrach, J. (1991). SLUG: A connectionist architecture for inferring the structure of finite-state environments. *Machine Learning*, 7, 139-160.

Muller, R. U., Kubie, J. L., Bostock, J. S., & Quirk, G. J. (1991a). Spatial firing correlates of neurons in the hippocampal formation of freely moving rats. In *Brain and Space* (Jacques Paillard, ed.), pp. 297-333. Oxford: Oxford University Press.

Narendra, K., & Thathachar, M. A. L. (1989). *Learning Automata.* Englewood Cliffs, New Jersey: Prentice Hall.

Nelson, R. C. (1989). Visual homing using an associative memory. In *Proceedings of the DARPA Image Understanding Workshop*, 245-262.

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34, 171-175.

O'Keefe, J., & Nadel, L. (1978). *The Hippocampus as a Cognitive Map.* Oxford: Claredon Press.

O'Keefe, J, & Speakman, A. (1987). Single unit activity in the rat hippocampus during a spatial memory task. *Experimental Brain Research*, 68, 1-27.

Parker, L. E., & Pin, F. G. (1988). Man-robot symbiosis: a framework for cooperative intelligence and control. *SPIE 1006, Space Station Automation IV*, 94-103.

Pavlov, I. P. (1927). *Conditional Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex* (G. V. Anrep, trans.). London: Oxford Press.

Payton, D. W., Rosenblatt, J. K., & Keirsey, D. M. (1990). Plan guided reaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 20 (6), 1370-1382.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kauffmann

Penna, M. A., & Wu, J. (1993). Models for map building and navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 23 (5), 1276-1301.

Perrett, D. I., & Oram, M. W. (1993). Neurophysiology of shape processing. *Image and Vision Computing*, 11 (6), 317-333.

Perrett, D. I., & Oram, M. W. (1994). Modelling visual recognition from neurobiological constraints. *Neural Networks*, 7 (6/7), 945-972.

Perrett, D. I, Harries, M. H., Bevan, R., Thomas, S., Benson, P. J., Mistlin, A. J., Chitty, A. J., Hietanen, J. K., & Ortega, J. E. (1989). Frameworks of analysis for the neural representation of animate objects and actions. *Journal of Experimental Biology*, 146, 87-113.

Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning*, 7, 227-252.

Pomerleau, D. A. (1993). *Neural Network Perception for Mobile Robot Guidance.* Kluwer Academic Publishers.

Porat, S., & Feldman, J. A. (1991). Learning automata from ordered examples. *Machine Learning*, 7, 109-138.

Prepscius, C., & Levy, W. B. (1994). Sequence prediction and cognitive mapping by a biologically plausible neural network. In *Proceedings of the World Congress on Neural Networks* (WCNN'93), June 5-9, San Diego, CA, Vol IV, 164-169.

Rabiner, L. R. & Juang, B. H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3 (1), 4-16.

Rheingold, H. (1991) *Virtual Reality.* New York: Simon & Schuster.

Rimey, R. D., & Brown, C. M. (1991). Controlling eye movements with hidden Markov models. *International Journal of Computer Vision*, 7 (1), 47-65.

Rimey, R. D., & Brown, C. M. (1993). Task-oriented vision with multiple Bayes nets. In *Active Vision* (A. Blake and A. Yuille, eds.), pp. 217-236. Cambridge, MA: MIT Press.

Rivest, R. L., & Schapire, R. E. (1987). A new approach to unsupervised learning in deterministic environments. In *Proceedings of the Fourth International Workshop on Machine Learning,* 364-375.

Rivest, R. L., & Schapire, R. E. (1989). Inference of finite automata using homing sequences. In P*roceedings of the Twenty-first Annual ACM Symposium on Theory of Computing,* Seattle, WA, May, 411-420.

Rolls, E. T (1991). Functions of the primate hippocampus in spatial processing and memory. In *Brain and Space* (Jacques Paillard, ed.), pp. 353-376. Oxford: Oxford University Press.

Santini, S., & Del Bimbo, A. (1995). Recurrent neural networks can be trained to be maximum a posteriori probability classifiers. *Neural Networks*, 8 (1), 25-29.

Sarachik, K. B. (1989). Visual navigation: constructing and utilizing simple maps of an indoor environment. Technical Report 1113, MIT Artificial Intelligence Laboratory.

Sato, M., Abe, K., & Takeda, H. (1982). Learning control of finite Markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control,* 27, 502-505.

Schloerb, D. (1995). A quantitative measure of telepresence. *Presence: teleoperators and Virtual Environments*, 4 (1), 64-80.

Schmajuk, N. A., & Blair, H. T. (1993). Place learning and the dynamics of spatial navigation: A neural network approach. *Adaptive Behavior,* 1 (3), 353-385.

Schmajuk, N. A., Thieme, A. D., & Blair, H. T. (1993). Maps, routes, and the hippocampus: A neural network approach. *Hippocampus*, 3 (3), 387-400.

Schmajuk, N. A., & DiCarlo, J. J. (1991). A neural network approach to hippocampal function in classical conditioning. *Behavioral Neuroscience*, 105 (1), 82-110.

Scholkopt, B., & Mallot, H. A. (1995). View-based cognitive mapping and path planning. *Adaptive Behavior*, 3 (3), 311-348.

Seibert, M., & Waxman, A. M. (1989). Spreading activation layers, visual saccades and invariant representations for neural pattern recognition systems. *Neural Networks*, 2, 9-27.

Seibert, M., & Waxman, A. M. (1992). Adaptive 3D object recognition from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14, 107-124.

Seibert, M., & Waxman, A. M. (1993). An approach to face recognition using saliency maps and caricatures. In *Proceedings of the World Congress on Neural Networks* (WCNN 93), July 11-15, Portland, OR, Vol III, 661-664.

Seibert, M., Waxman, A. M., & Gove, A. (1995). Learning to distinguish similar objects. *Proceedings of the SPIE Conference on Applications and Science of Artificial Neural Networks*, SPIE-2492.

Servan-Schreiber, D., Cleeremans, A., & McClelland, J. L. (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7, 161-193.

Shapiro, M. L., & Hetherington, P. A. (1993). A simple network model simulates hippocampal place fields: Parametric analyses and physiological predictions. *Behavioral Neuroscience*, 107, 34-50.

Sharp, P. E. (1991). Computer simulation of hippocampal place cells. *Psychobiology*, 19 (2), 103-115.

Shen, W-M. (1994). *Autonomous learning from the environment.* New York: W. H. Freeman and Company.

Sheridan, T. B. (1988). Man-machine communication for symbiotic control. In *Workshop on Human-Machine Symbiotic Systems Proceedings* (L. E. Parker & C. R. Weisbin, eds.), Dec. 5-6, Oak Ridge, Tennessee, 21-36.

Sheridan, T. B. (1992). *Telerobotics, Automation, and Human Supervisory Control,* MIT Press: Cambridge, MA.

Simpson, P. K. (1990). *Artificial Neural Systems.* New York: Pergamon Press.

Singh, S. P. (1994). Learning to solve Markovian decision processes. PhD Thesis, Department of Computer Science, University of Massachusetts.

Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis.* New York, NY: D. Appleton Century.

Smith, A. W., & Zipser, D. (1989). Encoding sequential structure: Experience with the real-time recurrent learning algorithm. In *Proceedings of IJCNN*, Vol 1, June, Washington, DC, 645-648.

Soechting, J. F., & Flanders, M. (1992). Moving in three-dimensional space: Frames of reference, vectors, and coordinate systems. *Annual Review of Neuroscience*, 15, 167-191.

Sontag, E. D. (1995). Automata and neural networks. In *The Handbook of Brain Theory and Neural Networks* (M. A. Arbib, ed.), pp. 119-123. Cambridge, MA: The MIT Press.

Staddon, J. E. R., & Ettinger, R. H. (1989). *Learning: An Introduction to the Principals of Adaptive Behavior.* New York: Harcourt Brace Jovanovich.

Sun, G. Z., Chen, H. H., Lee, Y. C., & Giles, C. L. (1990). Recurrent neural networks, hidden Markov models and stochastic grammars. In *Proceedings of IJCNN*, Vol 1, June, San Diego, CA, 729-734.

Sutton, R. (1988). Learning to predict by method of temporal differences. *The Journal of Machine Learning*, 3 (1), 9-44.

Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. *In Proceedings, Seventh International Conference on Machine Learning*, Austin, Texas.

Swain, M. J., & Stricker, M. A. (1993). Promising directions in active vision. *International Journal of Computer Vision*, 11 (2), 109-126.

Tachi, S., & Arai, H. (1985). Study on tele-existence II: three dimensional color display with sensation of presence. In *Proceedings of the 1985 International Conference: Advanced Robotics*, Tokyo, 345-352.

Tanaka, K. (1993). Neuronal mechanisms of object recognition. *Science*, 262, 685-688.

Thompson, D. (1976). The man-robot interface in automated assembly. In *Monitoring Behavior and Supervisory Control* (T. B. Sheridan & G. Johannsen, eds.), pp. 385-391. New York, NY: Plenem Press.

Thorpe, C., Hebert, M. H., Kanade, T., & Shafer, S. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10 (3), 362-373.

Thorndike, E. L. (1911). *Animal Intelligence: Experimental Studies.* New York, NY: Macmillan.

Thrun, S. B., and Moller, K. (1992). Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems*, Vol. 4 (J. E. Moody, S. J. Hanson, & R. P. Lippmann, eds.). San Mateo, CA: Morgan Kaufmann.

Tolman, E. C. (1932). *Purposive Behavior in Animals and Men.* New York: Century.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychology Review*, 55, 189-208.

Touretzky, D. S., Wan, H. S., & Redish, A. D. (1994). Neural representation of space in rats and robots. In *Computational Intelligence: Imitating Life* (J. M. Zurada, R. J. Marks II, & C. J. Robinson, eds.), pp. 57-68, New York, NY: IEEE Press.

Treisman, A., Cavanagh, P., Fischer, B., Ramachandran, V. S., & von der Heydt, R. (1990). Form perception and attention: Striate cortex and beyond. In *Visual Perception: The Neurophysiological Foundations* (L. Spillmann & J. S. Werner, eds.), pp. 273-316. New York: Academic Press.

Ullman, S. (1987). Visual routines. In *Readings in Computer Vision: Issues, Problems, Principals, and Paradigms* (M. A. Fischler & O. Firschein, eds), pp. 298-328. Los Altos, CA: Morgan Kaufmann Publishers.

Ullman, S., & Sha'ashua, A. (1988). Structural saliency: the detection of globally salient structures using a locally connected network. A.I. Memo 1061, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Voorhees, H (1987). Finding Texture boundaries in natural images. MS Thesis, MIT Artificial Intelligence Laboratory, March.

Wang, H. H., Marks, R. L., & Rock, S. M. (1993). Task-based control architecture for an untethered, unmanned submersible. In *Proceedings of the Eighth International Symposium on Unmanned Untethered Submersible Technology*, Sept 27-29, Durham, NH, 137-147.

Watkins, C. J. C. H. (1989). Learning from delayed rewards. PhD Thesis, King's College, Cambridge, UK.

Waxman, A. M., Seibert, M., Bernardon, A. M., & Fay, D. (1993). Neural systems for automatic target learning and recognition. *MIT Lincoln Laboratory Journal*, 6 (1), 77-116.

Waxman, A. M., Seibert, M., Cunningham, R., & Wu, J. (1989). Neural analog diffusion-enhancement layer and spatio-temporal grouping in early vision. In *Neural Information Processing Systems*, Vol I (D. S. Touretzky, ed.), pp. 289-296. San Mateo, CA: Morgan Kaufmann Publishers.

Waxman, A. M., Seibert, M. C., Gove, A., Fay, D. A., Bernardon, A. M., Lazott, C., Steele, W. R., & Cunningham, R. K. (1995). Neural processing of targets in visible, multispectral IR and SAR imagery. *Neural Networks, Special Issue on ATR*, 8 (November/December), in press.

Waxman, A. M., Seibert, M., Gove, A. N., Fay, D. A., Cunningham, R. K., & Bachelder, I. A. (1994). Visual learning of objects: Neural models of shape, color, motion and space. In

*Computational Intelligence: Imitating Life* (J. M. Zurada, R. J. Marks II, & C. J. Robinson, eds.), pp. 237-251. New York, NY: IEEE Press.

Wehner, R., & Menzel, R. (1990). Do insects have cognitive maps? *Annual Review of Neuroscience*, 13, 403-414.

Wilson, M. A., & McNaughton, B. L. (1993). Dynamics of the hippocampal ensemble code for space. *Science*, 261, 1055-1058.

White, D. J. (1993). *Markov Decision Processes.* Chichester, England: John Whiley & Sons.

Whitehead, S. D. (1992). Reinforcement learning for the adaptive control of perception and action. Technical Report 406, Department of Computer Science, University of Rochester.

Yamauchi, B. M., & Beer, R. D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2 (3), 219-246.

Yarbus, A. L. (1967). *Eye Movements and Vision.* New York: Plenum Press.

Yeap, W. K. (1988). Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34 (3).

Yoerger, D. R. (1982). Supervisory control of underwater telemanipulators: design and experiment. PhD Thesis, MIT Department of Mechanical Engineering.

Zeki., S., & Shipp, S. (1988). The functional logic of cortical connections. *Nature*, 335, 311-317.

Zhang, Z., & Faugerhaus, O. (1992). A 3D world model builder with a mobile robot. *The International Journal of Robotics Research*, 11 (4), 269-285.

Zheng, J. Y., & Tsuji, S. (1992). Panoramic representation for route recognition by a mobile robot. *International Journal of Computer Vision*, 9 (1), 55-76.

Zipser, D. (1986). Biologically plausible models for place recognition and goal location. In *Parallel Distributed Processing*, Vol. 2 (D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, eds.), pp. 432-470. Cambridge, MA: MIT Press.

Zeltzer, D. (1992). Autonomy, interaction, and presence. *Presence: Teleoperators and Virtual Environments*, 1 (1), 127-132.