Biologically inspired computational structures and processes for autonomous agents and robots

by

Karthik Balakrishnan

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Computer Science
Major Professor: Vasant G. Honavar

Iowa State University
Ames, Iowa
1998

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Karthik Balakrishnan

has met the dissertation requirements of Iowa State University

Committee Member

Committee Member

Committee Member

Committee Member

Major Professor

For the Major Program

For the Graduate College

To my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Often, there comes a time in life when no words, however appropriately chosen, can do justice to one's thoughts and feelings. This is such a moment for me. I stand on the threshold of a long coveted doctorate with a myriad of emotions welling within. I know I must personally thank many a person for this accomplishment of mine, but I am concerned I will miss some. If this does indeed happen, I will have you know that it is not intentional.

Following the ancient Hindu canon, *matah pitah guru devam*, let me begin by thanking my parents for sharing with me, not only their genes, but also their keen knowledge and worldly wisdom. There is little doubt that I would not be what I am today, without their affection, support, and above all, patience (particularly through my sassy teen years!). I am indebted to my mother for her immense love and support. My modest collection of medals and awards are a direct result of her constant encouragement and faith in my abilities — curricular and otherwise. I am grateful to my late father for introducing me to the vicarious world of puzzles and for showing me the enjoyment one could derive from them. Puzzles of all kinds have since been a steady companion on my life's journey.

I owe an eternal debt of gratitude to my teachers. Not only have they taught me finer aspects of numerous subjects, but also believed in me and my abilities. Indeed, much of my academic success I owe to this trust. I would particularly like to thank my advisor, Dr. Vasant Honavar, for his patience, support, and confidence in me through my years at Iowa State. From him I have learned to enjoy research. His open-door policy and his readiness to sit down and "brainstorm", have been high points of my graduate career with him. I am also grateful to him for introducing me to an exquisite variety of artificial intelligence and machine learning techniques via carefully designed courses and seminars. I am in awe of his prolific writing talents and his ability to effortlessly transform complex thoughts into lucid prose — an inspiration for which I am extremely grateful.

I would also like to thank Dr. Albert Baker, Dr. Veronica Dark, Dr. William Robinson,

and Dr. Johnny Wong, for serving on my dissertation committee, and for their support and advice. Some of my most memorable courses at Iowa State have been with Dr. Baker and Dr. Wong. I recall, with great satisfaction, the animated discussions in the Software Engineering course with Dr. Baker. I also enjoyed Advanced Operating Systems with Dr. Wong, a course I found riveting in my early graduate career. I have had many rewarding discussions with Dr. Veronica Dark and Dr. William Robinson. Their constant encouragement and support, not to mention their interest in my career, have been a cornerstone of the later part of my dissertation work. I am very grateful to them for their detailed and invaluable feedback on my dissertation.

Apart from my teachers, I have learned valuable lessons from numerous friends and colleagues. My sisters, Usha and Sandhya, I thank for making our childhood so very interesting. I am grateful for their love and affection, and I value the close relationship we share till this day. To my friends, too numerous to mention individually, I am especially grateful for the many enjoyable hours of their company. From them I have learned much. I would especially like to thank Ravichandran, Kannan, Gowri Sankar, Siddharth, and Farid, who have not only been wonderful housemates, but have sedulously suffered through my cooking.

I would like to thank my *co-conspirators* in the Artificial Intelligence Research Laboratory at Iowa State University — Armin, Chen, Rajesh, and Jihoon, for numerous informative discussions in seminars (and otherwise). I am particularly thankful to Rajesh and Jihoon for their friendship, and for all their help during my graduate student life. I have tremendous admiration for Rajesh's thoroughness and Jihoon's gentleness, traits I hope I will have one day.

I am also grateful for the opportunity to have known and interacted with Olivier Bousquet. In a short period of a few months, he made significant contributions to the Kalman filter-based computational development of Hippocampal function. Although we worked together for only a year, I feel I have known Rushi much longer. I treasure his sense of humor and his bright quips, and miss the animated discussions that we often had (from neurobiology to cosmology and everything in between!). I am immensely grateful for his help in turning in the final copies of this dissertation.

I would like to thank the Department of Computer Science for the financial support during my graduate studies and Dr. Vasant Honavar for supporting some of my research through

grants from the National Science Foundation and the John Deere Foundation. I would also like to thank IBM for providing me with a Fellowship during the final year of my doctoral program.

The many years spent in the department would have been quite barren without the warmth and spirited humor of the office staff. I am especially grateful to have made the acquaintance of Trish Stauble, Lynn Bremer, Melanie Eckhart, and Clare Polking.

I would like to thank the Department of Computer Science for making the computing facilities available to aid my research. I am particularly thankful to Jim Schlosser for fixing numerous system problems at short notice. I am also grateful to the Scalable Computing Laboratory in Ameslab for making their computing systems available for my compute-intensive research in evolutionary systems. I would like to thank David Levine and his team for the PGAPack parallel genetic algorithm library that I used in a large portion of my research and Brian Walenz for making custom modifications to the library to support multi-objective optimization. I am also grateful to Carl Pecinovsky for developing the robot behavior animation routines. Thanks are also due to Dr. Steven LaValle for his generosity in letting me use his machines to write most of this dissertation.

I would also like to thank Gary Kerr, Tom Warden, and Joan Gilmore at the Allstate Research and Planning Center, for their patience and support while I put final touches to this dissertation.

Last but definitely not the least, I would like to thank my dear wife, Usha Nandini, for her warmth, affection, and love. Without her stolid support and patience this dissertation could not have been completed.

ABSTRACT

Recent years have seen a proliferation of intelligent agent applications: from robots for space exploration to software agents for information filtering and electronic commerce on the Internet. Although the scope of these agent applications have blossomed tremendously since the advent of compact, affordable computing (and the recent emergence of the World Wide Web), the design of such agents for specific applications remains a daunting engineering problem.

Rather than approach the design of artificial agents from a purely engineering standpoint, this dissertation views *animals* as *biological agents*, and considers artificial analogs of biological structures and processes in the design of effective agent behaviors. In particular, it explores behaviors generated by artificial neural structures appropriately shaped by the processes of evolution and spatial learning.

The first part of this dissertation deals with the evolution of artificial neural controllers for a box-pushing robot task. We show that evolution discovers high fitness structures using little domain-specific knowledge, even in feedback-impoverished environments. Through a careful analysis of the evolved designs we also show how evolution exploits the environmental constraints and properties to produce designs of superior adaptive value. By modifying the task constraints in controlled ways, we also show the ability of evolution to quickly adapt to these changes and exploit them to obtain significant performance gains. We also use evolution to design the sensory systems of the box-pushing robots, particularly the number, placement, and ranges of their sensors. We find that evolution automatically discards unnecessary sensors retaining only the ones that appear to significantly affect the performance of the robot. This optimization of design across multiple dimensions (performance, number of sensors, size of neural controller, etc.) is implicitly achieved by the evolutionary algorithm without any external pressure (e.g., penalty on the use of more sensors or neurocontroller units). When used in the design of robots with *limited battery capacities*, evolution produces *energy-efficient* robot designs that use minimal numbers of components and yet perform reasonably well. The

performance as well as the complexity of robot designs increase when the robots have access to a *spatial learning* mechanism that allows them to learn, remember, and navigate to power sources in the environment.

The second part of this dissertation develops a computational characterization of the *hippocampal formation* which is known to play a significant role in *animal spatial learning*. The model is based on neuroscientific and behavioral data, and learns place maps based on interactions of sensory and dead-reckoning information streams. Using an estimation mechanism known as *Kalman filtering*, the model *explicitly* deals with uncertainties in the two information streams, allowing the robot to effectively learn and localize even in the presence sensing and motion errors. Additionally, the model has mechanisms to handle perceptual aliasing problems (where multiple places in the environment appear sensorily identical), incrementally learn and integrate local place maps, and learn and remember multiple *goal locations* in the environment. We show a number of properties of this spatial learning model including computational replication of several behavioral experiments performed with rodents. Not only does this model make significant contributions to robot localization, but also offers a number of predictions and suggestions that can be validated (or refuted) through systematic neurobiological and behavioral experiments with animals.

1 INTRODUCTION

Humankind's limitless curiosity to understand, imitate, and improve systems and processes in nature is at the heart of perhaps every major scientific or engineering field of inquiry. For instance, while the need to count spurred the early development of the mathematical sciences, sickness and death probably provided the inspiration for the numerous biological and medical breakthroughs. This is also true of the field of *robotics*, whose origins can perhaps be traced to the animal masks and costumes worn in ancient ritual dances (Culbertson, 1957; Albus, 1981). From these humble mimicries of the living world, this field has undergone elaborate transformations, producing numerous creative artifacts along the way. The life-like behaviors demonstrated by a number mechanical and pneumatic dolls built in the middle ages (Sarton, 1936; Cohen & Drabkin, 1948; Chapuis & Droz, 1956; McCurdy, 1948; Arbib, 1966), bear testimony not only to the human curiosity to imitate nature but also the human fascination with *automata*. This fascination blossomed further in the early twentieth century, leading to electrically/electronically operated automata that could move and learn different behaviors (Walter, 1950; Walter, 1951; Shannon, 1952).

Whatever might have been the inspiration behind the early development of the field, much of the contemporary interest in robotics is largely an expression of the human need of *agency*, i.e., the need to acquire a subordinate or apprentice to execute tasks in lieu of the employer (Steels & Brooks, 1995; Valavanis & Saridis, 1992; Omidvar & van der Smagt, 1997; Zalzala & Morris, 1996). Machines in general and robots in particular, epitomize this concept of *agents*, with their potential to execute dull, repetitious, and often highly precise tasks without the usual human errors and lapses.

1.1 Intelligent Autonomous Agents

The advent of computers and computer networks have contributed to the proliferation of a slightly different breed of agents: ones where the emphasis is on *intelligence*, *autonomy*, *adaptability*, and *mobility*. For our purposes here, an agent is said to be *intelligent* if it performs actions that would be performed by normal human beings under similar circumstances. The agent is said to be *autonomous* if its behaviors are not a result of *direct external control*, but are instead, governed by its own reasoning mechanisms. Agents are said to be *adaptive* if they can modify their behaviors through their interactions with their environments, and they are *mobile* if they can move, or access, different places within their operating environments. Examples of such agents include robots for space and underwater exploration and mobile software systems for information gathering, data mining and electronic commerce. The design and development of such agents is a topic of considerable ongoing research and draws on technologies from a variety of fields like artificial intelligence, robotics, cognitive science, control systems, electrical and mechanical engineering, computer networks, distributed computing, database technologies, etc.

In very simplistic terms, an agent may be defined as an entity that *perceives* its environment through *sensors* and *acts* upon it through its *effectors* (Russell & Norvig, 1995). However, for the agents to be useful, they must also be capable of *interpreting perceptions*, *reasoning*, and choosing actions *autonomously* and in ways suited to achieving their intended *goals*. Since the agents are often expected to operate reliably in unknown, partially known, and dynamic environments, they must also possess mechanisms to *learn and adapt* to the environments they encounter. In addition, we may require the agents to be *mobile*, *persistent*, *rational*, etc. Finally, we may expect the agents to work in groups, which requires them to *collaborate* and *communicate* (Russell & Norvig, 1995; Wooldridge & Jennings, 1995; Nwana, 1996).

It can be seen that this definition of agents applies to robots that occupy and operate in physical, real-world environments, software agents (or softbots) that inhabit the electronic worlds defined by computers and computer networks, and even *animals* (or biological agents) that cohabit this planet. Robots have been used in a wide variety of application scenarios like geo, space, and underwater exploration, material handling and delivery, security patrolling, control applications in hazardous environments like chemical plants and nuclear reactors, etc., (Cox & Wilfong, 1990; Everett, 1995; Zalzala & Morris, 1996; Omidvar & van der Smagt,

1997). Softbots are being increasingly used in automated tools for diagnosis, evaluation, and optimization, information gathering and retrieval, data mining, electronic commerce, news, mail and information filtering, etc., (Bradshaw, 1997).

In general, an agent can be characterized by two elements: its *program* and its *architecture*. The agent program is a mapping that determines the actions of the agent in response to its sensory inputs, while architecture refers to the computing and physical medium on which this agent program is executed (Russell & Norvig, 1995). For example, a *mail filtering agent* might be programmed in a language such as C++ and executed on a computer, or a robot might be programmed to move about and collect empty soda cans using its *gripper arm*. Thus, the architecture of the agent, to a large extent, determines the *kinds* of things the agent is capable of doing, while the program determines *what* the agent does at any given instance of time. In addition to these two elements, the agent *environment* governs the kinds of tasks or behaviors that are within the scope of any agent operating in that environment. For instance, if there are no soda cans in the robot's environment, the can collecting behavior is of no practical use.

Given a particular agent environment the question then is, how can we design agents with the necessary behaviors and abilities to solve a given task (or a set of tasks)?

The field of Artificial Intelligence (AI) has long concerned itself with the design, development, and deployment of intelligent agents for a variety of practical, real-world applications (Rich & Knight, 1991; Winston, 1992; Tanimoto, 1995; Russell & Norvig, 1995). A number of tools and techniques have been developed for synthesizing such agent programs (e.g., expert systems, logical and probabilistic inference techniques, case-based reasoning systems, etc.) (Rich & Knight, 1991; Ginsberg, 1993; Kolodner, 1993; Durkin, 1994; Dean *et al.*, 1995; Russell & Norvig, 1995). However, designing programs using most of these approaches requires considerable knowledge of the application domain being addressed by the agent. This process of *knowledge extraction* (also called knowledge engineering) is often difficult owing either to the lack of precise knowledge of the domain (e.g., medical diagnosis) or the inability to procure knowledge owing to other limitations (e.g., detailed environmental characteristics of, say, planet Saturn). We thus require agent design mechanisms that work either with little domain-specific knowledge or allow the agent to acquire the desired information through its own experiences. This has led to considerable development of the field of *machine learning* (ML), which provides a variety of modes and methods for *automatic synthesis* of agent programs (Honavar, 1994;

Langley, 1995; Ripley, 1996; Mitchell, 1997).

In this dissertation, we adopt a slightly different approach. Instead of considering the design of artificial agents from an engineering standpoint, we look towards nature, which provides innumerable examples of animals (or biological agents) that appear well-adapted to their ecological niches. We draw inspiration from the structures and processes involved in the design of these biological agents to pursue our goal of designing artificial autonomous agents.

1.2 Design of Biological Agents

Among the processes of natural adaptation responsible for the design of biological agents, probably the most crucial is that of *evolution*, which excels in producing agents designed to overcome the constraints and limitations imposed by their environments. For example, *flashlight fish (photoblepharon palpebratus)*, inhabits deep-waters where there is little surface light. However, this is not really a problem since evolution has equipped these creatures with *active vision* systems. These fishes produce and emit their own light, and use it to detect obstacles, prey, etc. This vision system is unlike any seen on surface-dwelling organisms (Grier & Burk, 1992).

Apart from evolution, *learning* is another biological process that allows animals to successfully adapt to their environments. For instance, animals learn to respond to specific stimuli (*conditioning*) and ignore others (*habituation*), recognize objects and places, communicate, engage in species-specific behaviors, etc., (Mackintosh, 1983; Gallistel, 1990; Grier & Burk, 1992; McFarland, 1993). Since most animals inhabit spatial environments and must navigate in order to fulfill basic needs of finding food and mates, avoiding predators, finding their way home after foraging expeditions, etc., they require the *crucial* ability of *spatial learning* which enables them to *acquire, represent*, and *use* information pertaining to the spatial attributes of their environments. Although some of these spatial abilities are *in-born* or genetically programmed, others must be learned. For instance, it is well known that marine turtles (e.g., *Chelonia mydas*) lay their eggs on tropical beaches (e.g., Ascension Island in the Atlantic ocean). When the eggs hatch, the young automatically crawl to the water without any kind of parental supervision. They appear to be genetically programmed to interpret cues emanating from large bodies of water or patches of sky over them (Grier & Burk, 1992). Animals are also capable of learning the spatial attributes of *novel* environments. For instance, as will become apparent

in Chapter 6, rodents have an immense capacity to learn and successfully navigate through complex mazes (Tolman, 1948; O'Keefe & Nadel, 1978).

These processes of natural adaptation, namely, evolution and learning, play a significant role in shaping the behaviors of biological agents. They differ from each other in some significant aspects including the spatio-temporal scales at which they operate. While learning operates on individuals, evolution works over entire populations (or species). Further, learning operates during the lifetime of the individual and is presumably aided by long lifespans, while evolution works over generations, well beyond an individual's effective lifespan (Ackley & Littman, 1991).

Despite these apparent differences, evolution and learning work synergistically to produce animals capable of surviving and functioning in diverse environments. While the architectures of biological agents (e.g., digestive, respiratory, nervous, immune, and reproductive systems) are shaped by evolution, the agent programs (e.g., behaviors like foraging, feeding, grooming, sleeping, escape, etc.) are affected and altered by both evolutionary and learning phenomena. In such cases, evolution produces the *innate* (or genetically programmed) behaviors which are then modified and contoured to the animal's experiences in the specific environment to which it is exposed. Thus, by equipping the agents with good designs and instincts, evolution allows them to survive sufficiently long to learn the behaviors appropriate for the environment in question.

Although the processes of evolution and learning are reasonably well detailed, there are still many gaps to be filled before a complete understanding of such processes can be claimed. Ongoing research in neuroscience, cognitive psychology, animal behavior, genetics, etc., is providing new insights into the exact nature of the mechanisms employed by these processes — the structures they require and the functions they compute. This has led to many new hypotheses and theories of such mechanisms. *Computational modeling* efforts complement such research endeavors by providing valuable tools for *testing* theories and hypotheses in controlled, simulated environments. Such modeling efforts can potentially identify and suggest avenues of further research that can help fill in the gaps in current human understanding of the modeled processes (Churchland & Sejnowski, 1992).

Drawing inspiration from the biological processes at work in the design of biological agents, this dissertation explores artificial analogs of such processes in the design of artificial agents.

1.3 Contributions of this Dissertation

In brief, this dissertation considers the role of evolution in the design of artificial neural networks and sensory systems for robots, and develops a biologically inspired computational model of spatial learning to aid mobile robot localization. Although the work is presented in the context of robots, the principles apply equally to the design of software agents. The primary contributions of the dissertation are summarized below.

1.3.1 Evolutionary Design of Neural Architectures

Evolutionary algorithms (EAs), loosely inspired by biological evolutionary processes, have gained considerable popularity as tools for searching vast, complex, and deceptive design (or solution) spaces using little domain-specific knowledge (Holland, 1975; Goldberg, 1989; Koza, 1992; Mitchell, 1996). They have been successfully applied to a variety of difficult optimization problems including the synthesis of artificial neural networks for specific applications (see (Balakrishnan & Honavar, 1995a) for a bibliography). In this dissertation research, we have made the following contributions to the field of evolutionary design of neural architectures.

1. A *taxonomy* of any field of research is a very useful tool since it identifies *characteristics* or *attributes* of the research domain that can be used to perform fair comparisons between different approaches that have been adopted. Although a number of approaches had been developed for the evolutionary synthesis of neural architectures, the field lacked a taxonomy. Based on our knowledge of the field, we suggested a *preliminary taxonomy* that characterized each research endeavor along *four axes* defined by: *encoding scheme*, *network topology*, *variables of evolution*, and *application domain* (Balakrishnan & Honavar, 1995a). The *encoding scheme* refers to the genetic representation chosen for encoding and decoding the neural networks (e.g., weight matrix, rewrite rules, LISP programs, etc.), the *network topology* refers to the physical properties of the networks evolved by the system (e.g., feed-forward, recurrent, locally-connected, etc.), while *variables of evolution* identify the precise aspects of the neural networks that were subject to evolution (e.g., topology, weights, activation function, learning algorithm, etc.). The final category, *application domain*, tagged each of the approaches with the kinds of problems that were addressed using the approach (e.g., toy problems such as XOR and encoder-decoder, face

recognition, traveling salesperson problem, etc.) (Balakrishnan & Honavar, 1995a). Not only did this taxonomy allow us to distinguish between different approaches, but also helped us identify potential research directions. For instance, we found that no one had addressed the evolution of activation functions in neural architectures. This allowed us to explore this area fruitfully (Juedes & Balakrishnan, 1996).

2. The *genetic representation* chosen in an EDNA system not only dictates the kinds of target neural networks that the system can possibly evolve, but also determines the amount of *resources* expended in this effort. For instance, one would like the genetic search to proceed as *efficiently* as possible with little wastage of system resources like time and space. One way of doing this might be to design genetic representations and operators that are guaranteed to produce *valid* networks under all possible matings (a neural network might be considered invalid if it has, say, no signal pathways from its inputs to its outputs). Genetic representations that always produce valid networks are said to have the property of *closure*. Previous characterizations of genetic representations lacked precise specification, and further, only considered the genetic encoding without taking into account the associated decoding process (Collins & Jefferson, 1990; Gruau, 1994). We have developed reasonably precise notions of properties of genetic representations that characterize the genetic encoding as well as the decoding process. The properties we have identified include: *completeness, closure, compactness, scalability, multiplicity, ontogenetic plasticity, modularity, redundancy,* and *complexity* (Balakrishnan & Honavar, 1995b). These are described in Section 2.5.1.

1.3.2 Evolution of Sensory and Behavior Systems for Robots

Evolutionary algorithms have also been used successfully in the design of *behavior controllers* (LISP programs, neural networks, etc.) for simulated and real robots (Colombetti & Dorigo, 1992; Harvey *et al.*, 1992; Cliff *et al.*, 1993a; Cliff *et al.*, 1993b; Husbands *et al.*, 1993; Nolfi *et al.*, 1994b; Reynolds, 1994b; Teller, 1994; Yamauchi & Beer, 1994; Nolfi *et al.*, 1994a; Miglino *et al.*, 1994; Jakobi *et al.*, 1995; Walker, 1995; Nolfi & Parisi, 1995; Miglino *et al.*, 1995). Typically, the robot tasks considered in these cases have been *passive* in nature, i.e., tasks wherein the robot actions do not *significantly* affect its operating environment. Common examples of such tasks include: obstacle avoidance, wall or corridor following, approaching or

avoiding a light source or some specific object, finding food or energy sources, homing, exploring, etc. Further, most of the simulation-based approaches make unrealistic assumptions about the sensory and motor abilities of the robots. For instance, it is often assumed that the simulated robot has no range limitations and is capable of sensing its environment *completely* and *reliably* (Nolfi *et al.*, 1994b; Walker, 1995).

In our work we have made use of a robot task proposed by Teller (1994). Here, the robot is introduced into a room (assumed to be decomposed into $N \times N$ cells), littered with boxes, and has the task of pushing the boxes to the enclosing walls (Teller, 1994). As the actions of the robot affect the distribution of the boxes in the environment, this task is more *active* than obstacle avoidance or wall following behaviors. In addition, the robots in this task are assumed to be constrained by a number of elements that make the design of effective box-pushing behaviors rather hard. For instance, each robot has a limited *time* within which it has to execute its room-clearing task and limited *sensory ranges* (the robot can only sense one *cell* in each of eight directions around it). Also, the robot cannot push more than one box at a time and is incapable of detecting futile moves (e.g., moving into a wall, pushing a box into a wall or against another box, etc.). The fitness of the robot is the number of boxes resting against walls at the end of its simulation time, with an extra point for each box pushed into a corner. The goal, then, is to evolve high fitness box-pushing behaviors for the robot. Although the constraints on the behavior and sensory systems make this robot task more realistic, they make the design of appropriate behaviors correspondingly harder. The contributions of our evolutionary design approach to this simulated robot task are summarized below:

1. Although we used extremely simple neural networks (equivalent in computational power to *finite state machines* rather than Turing machines), we were able to evolve neurocontrollers of comparable performance to the LISP programs of Teller (1994). Importantly, unlike Teller, we were able to *analyze* the evolved neural structures and determine precisely *how* our agents achieved their high fitnesses. This led to an important insight into the kinds of behaviors that would *intuitively* work well given the constraints of the robot task (Balakrishnan & Honavar, 1996a). These results are presented in Sections 3.3.2 and 3.3.3.

2. We also showed that these high-fitness behaviors are truly characteristic of the properties

and constraints of the environment. This was done by manipulating the *coding of the outputs* of the neurocontrollers, which changed the way the robots interpreted their output actions. We found that in each case evolution discovered neurocontroller structures that were functionally (behaviorally) equivalent to the earlier designs (Balakrishnan & Honavar, 1996a). These results appear in Section 3.3.6.

3. By relaxing the environmental constraints in controlled ways, we also demonstrated that evolution produces high fitness designs that exploit these environmental changes (Balakrishnan & Honavar, 1996a; Balakrishnan & Honavar, 1996c). For instance, when provided with a simple form of feedback, robots engaged in futile pushing demonstrate radically different behaviors and *exploit* the feedback mechanism to improve their performance on the box-pushing task (Balakrishnan & Honavar, 1996a). These results are presented in Section 4.1.

4. We also showed that artificial evolution can be used in the design of robot sensory systems by allowing it to choose the numbers, placement, and ranges of the sensors (Balakrishnan & Honavar, 1996b). An important, and somewhat surprising, discovery was the fact that having more sensors was detrimental to the fitness of the robot, possibly due to sensory conflicts or confusion caused by too much sensory information. The use of evolutionary algorithms in the design of sensory systems automatically leads to the discarding of unnecessary sensors, even without any explicit penalties on the use of more sensors, as shown in Section 4.2. This is reminiscent of the *feature-subset selection* problem often encountered in the data-mining community, where the goal is to discard redundant, uninformative, or useless features (inputs, variables, attributes, etc.) and choose a subset of them for analysis (Yang & Honavar, 1998).

5. Evolution is also capable of discovering robust and noise-tolerant designs. For instance, when the sensors were assumed to be faulty (as is the case with many contemporary robot sensors (Everett, 1995)), evolution discovered designs with multiple sensors in key sensing directions around the robot (Balakrishnan & Honavar, 1996c). We also found that evolution often exploits noise to overcome other system limitations that contribute to low fitnesses. For instance, evolution exploits noise to improve the fitnesses of feed-forward neurocontrollers, as explained in Section 4.3.

6. In general, the design of any system can be thought of as a search in a multi-dimensional space defined by system requirements and performance constraints (Simon, 1983; Honavar, 1994). Often, these dimensions work at cross-purposes, requiring the design process to perform appropriate *tradeoffs* between them. For instance, by capturing temporal information, multiple hidden units in the neural network allow the robots to display *sequentially correlated* behaviors. However, these extra units are a drain on the limited power resources of the robot. One way to perform such tradeoffs is to map or evaluate the different objectives in terms of a *common currency*, for example, *utility* (Keeney & Raiffa, 1976), and choose options that *maximize the expected utility* (von Neumann & Morgenstern, 1944). We have adopted a similar, but simpler, approach by mapping the different components of our agent into the common currency of *power consumption* (McFarland & Bosser, 1993). Thus, the robot has a limited amount of energy which is slowly lost through power consumption by the sensors, neurocontroller units, and the effectors. Given these constraints, instead of using decision-theoretic mechanisms, we let evolution optimize agent designs across these multiple dimensions. Indeed, evolution produces high fitness agents with a few key sensors and almost no hidden units in their neurocontrollers (Balakrishnan & Honavar, 1999). We present these results in Section 5.6. To the best of our knowledge, we are the first to adopt this approach in the evolutionary robotics area.

7. We have also evolved *spatially adaptive* agent designs. Although these robots have limited energy reserves, they have mechanisms to learn, remember, and navigate to power sources in the box-pushing environment. With this ability to approach power sources and charge their batteries, these robots evolve to have rather different designs. They employ significantly more numbers of sensors and units, and demonstrate effective box-pushing behaviors. These results are presented in Section 5.9.

1.3.3 Hippocampal Model of Spatial Learning

We mentioned the evolution of box-pushing behaviors in spatially adaptive agents that possess in-built spatial learning abilities. But how are these spatial learning abilities acquired by animals and robots? We have summarized some contemporary results in robot spatial learning in Sections 6.3 and 6.5. But how do animals engage in spatial learning? Would a study of the structures and processes involved in animal spatial learning have any implications

for robots? In the remainder of the dissertation we have explored answers to these questions.

Based on a considerable body of data from neurophysiological, neuroanatomical, and behavioral experiments with animals (primarily rodents), we have developed a characterization of *hippocampal function* in animal spatial learning and navigation (Bousquet *et al.*, 1998; Balakrishnan *et al.*, 1997). The proposed model is a computational realization of the *cognitive-mapping* theory of (Tolman, 1948) who suggested that animals learn a *metric* representation of space, and is based on the thesis of (O'Keefe & Nadel, 1978) who suggested that the *hippocampal formation* (a part of the brain) is involved in the formation of such cognitive maps. Although a number of models of hippocampal function have been suggested in the cognitive psychology and neuroscience literature (see (Burgess *et al.*, 1995; Trullier *et al.*, 1997) for surveys), none have provided explicit mechanisms to handle uncertainty in the information streams. This dissertation makes the following contributions.

1. We have provided a computational (and implementable) realization of the *locale system* hypothesis of (O'Keefe & Nadel, 1978). Further, this model differs from other metric models, primarily (Wan *et al.*, 1994; Redish & Touretzky, 1996; Redish & Touretzky, 1998), in *explicitly* handling uncertainty in the information sources. This is a significant development, since both the sensory inputs and the animal's dead-reckoning[1] system are known to entertain considerable errors (e.g., recognition errors, uncertainty in the estimation of distances to sensed objects, drifts and errors in animal motion, etc.). To the best of our knowledge, none of the metric hippocampal models address this issue of information fusion and localization from uncertain sources.

2. We have also drawn a parallel between the posited hippocampal function and probabilistic localization approaches used in contemporary robotics (e.g., Kalman filter). Based on this parallel, we have developed a Kalman filtering framework for hippocampal spatial localization, with update expressions that can be proven to be *stochastically optimal.*

3. We have extended the computational framework to support *incremental* learning of local place maps, and have developed mechanisms to consistently *integrate or merge* these local maps into global ones.

[1]Dead-reckoning (or path-integration) is the process of updating an estimate of one's own position based on knowledge of direction, speed, and time of self-motion

4. We have also developed a computational mechanism for learning, remembering, and updating position estimates of encountered *goal locations*. This system is also capable of learning and representing *multiple* goal locations, and has been augmented with algorithms for goal selection and goal-directed navigation.

5. The model has been used to simulate a number of behavioral experiments, primarily the gerbil experiments of (Collett *et al.*, 1986) and the water-maze task of (Morris, 1981). In either case, the behaviors demonstrated by our animats[2] are largely similar to those observed by the researchers in their experiments with rodents. In addition to a computational validation of these behaviors, our model also makes it feasible to propose and study new hypotheses of animal spatial learning and navigation. Indeed, based on our simulations we have suggested a number of further experiments that can be performed by cognitive scientists and animal behaviorists in order to better understand these processes in animals.

6. Our characterization of hippocampal spatial learning also generates many testable predictions for neuroscientists and cognitive scientists. For instance, we suggest that the CA1 layer of the hippocampus is involved in distinguishing between perceptually similar places. The model also suggests that animals explore in slowly expanding trajectories from their point of entry in novel environments.

7. Our work contributes to the literature on robot navigation by providing a *place-based* extension of Kalman filtering used in robotics. It also provides a mechanism for distinguishing between perceptually similar places in the environment (also known as *perceptual aliasing* in robotics), by using the *Mahalanobis distance* and the robot's dead-reckoning position estimates (Balakrishnan *et al.*, 1997).

1.4 Summary

In this dissertation we have drawn inspiration from structures and processes employed in the design of animals, and have tried to use them in the design of artificial agents and robots. We have used the processes of evolution and spatial learning operating on structures that include artificial neural networks and a quasi-neural model of the hippocampal formation.

[2] Animat: an artificial automaton simulating the behavior of an animal

Apart from the specific contributions listed earlier, this dissertation also contributes to the science and technology of designing agent architectures and programs via the use of evolutionary, neural, and statistical approaches. Viewed from a computational modeling standpoint, this dissertation provides implementations of two biological processes: evolution and spatial learning. In addition, each of these frameworks can be easily employed for testing new computational theories or hypotheses concerning evolutionary and spatial learning phenomena.

2 EVOLUTIONARY NEURO-ROBOTICS

In the previous chapter we outlined our interest in the design of artificial agents and robots for specific tasks. We also introduced the three broad components of agent design, namely environment, architecture, and control program. In this chapter we will consider the design of control programs for robot-like agents. We will allude to the different representational paradigms for robot control programs, and argue for the use of *artificial neural networks* for robot control. We will then outline the difficulties associated with the design of such neurocontrollers and demonstrate the use of *artificial evolution* in the discovery of appropriate network architectures and functions. We will also develop a set of *properties* that can be used to characterize the *genetic representations* used in such evolutionary approaches, in particular, in the evolution of neural architectures.

2.1 Programs for Robot Control

As we mentioned earlier, the agent control *program* is primarily responsible for the behavior of the agent or robot in a given environment. The control program determines the sensory inputs available to the agent at any given moment in time, processes the inputs in ways suited to the goals (or functionalities) of the agent, and determines appropriate agent actions to be performed.

In order to be used in practice, these control programs have to be *represented* within the agents using an appropriate *language* and the agents must possess mechanisms to *interpret* and *execute* them. One of the earliest control program representations involved the use of *circuits and control systems* that directly sensed input events, processed them via appropriate *transfer functions*, and directly controlled the output behaviors of the robot (Lewis *et al.*, 1993). Examples of such representations include PI, PD, and PID controllers used extensively in the control of industrial robots, robotic arms, etc., (Anand & Zmood, 1995). However, these

approaches require the transfer function to be known and appropriately implemented, which is often difficult in practice. In addition, these control mechanisms are rather inflexible and reprogramming the robot for a different behavior or task may entail extensive changes.

The advent of computers and their ability to be effectively *reprogrammed*, have made them extremely attractive propositions in the control of modern-day robots. In these cases, robot control programs are written in some computer language (e.g., C++), interpreted using compilers, and executed by the computer. It is assumed that the computer program acquires the necessary inputs from the robot sensors, processes them appropriately to determine robot actions to be performed, and controls the robot actuators to realize the intended behaviors. The primary advantage of such computer based control program representations is their role in making the robot *adaptive*, i.e., the robot behaviors can be easily changed by altering the control program.

The question then is, how can we develop a control program that will make the robot exhibit a desired behavior?

2.1.1 Designing Control Programs for Specific Behaviors

Many contemporary robots make use of programs that are *manually* developed. This is a daunting task, given the fact that the robot-environment interactions exhibit a host of unpredictable effects like sensing errors (e.g., occlusion, specular reflections, shadows, etc.) and motion errors (e.g., friction, wheel slippage, uneven tire inflation, etc.). These effects lead to robot behaviors that are often sizable deviants of the behaviors actually designed and programmed. Thus, manually developing control programs that are relatively immune to these interaction errors is an extremely difficult task.

In addition, complex robot behaviors often involve tradeoffs between multiple competing alternatives. For example, suppose a robot has the task of clearing a room by pushing boxes to the walls. Let us also assume that the robot has limited sensing ranges that prevent it from observing the contents of the entire room and it does not have any means to remember the positions of boxes it has observed in the past. Suppose this robot currently observes two boxes. Which one should it approach and push? This decision is critical as it directly affects the subsequent behaviors of the robot. We may program the robot to approach the closer of the two boxes, but can we be sure that such a decision made at the local level will indeed lead

to any kind of globally optimal behavior? Manually designing control programs to effectively address such competing alternatives is an equally challenging proposition.

We thus need mechanisms to *automatically* design robot behaviors that are not only robust to the unpredictable environmental effects but also appropriately balance competing objectives inherent in the robot task. In recent years two kinds of automatic design approaches have met with much success: *discovery* and *learning*. Approaches belonging to the former category typically include some mechanism to *efficiently search* the space of robot control programs in the hope of finding or discovering a good one. Each control program found during this search is evaluated on the robot task and the best one is retained. Some discovery approaches (e.g., evolutionary search) use these evaluations to *guide* or *focus* the search procedure, making the process more efficient. The latter category includes approaches that allow the robot behaviors to be modified based on the experiences of the robot, i.e., the robot *learns* the correct behaviors based on its experience in the environment.

In the remainder of this chapter we will introduce two paradigms that aid in the automatic design of appropriate robot behaviors. While *artificial neural networks* offer a host of advantages that make them excellent choices for robot control programs, (e.g., their ability to learn from experience), *artificial evolution* provides a powerful tool to automatically discover effective robot behaviors based on little domain-specific knowledge. As this dissertation employs both these paradigms, we treat them in more detail in the following sections.

2.2 Artificial Neural Networks for Robot Control

Artificial neural networks offer an attractive paradigm of computation for many applications including robot and process control, pattern recognition, system identification, cognitive modeling etc. This is largely due to the numerous advantages offered by these networks, including their potential for *massively parallel computation*, *robustness* in the presence of noise, *resilience* to the failure of components, amenability to *adaptation* and *learning* via the modification of computational structures, etc. For these reasons, primarily their innate ability to tolerate noise and learn from experience, they have found increasing use as behavior controllers for robots.

2.2.1 What are Artificial Neural Networks?

Artificial neural networks are models of computation that are inspired by, and loosely based on, the nervous systems in biological organisms. They are conventionally modeled as massively parallel networks of simple computing elements, called *units*, that are connected together by *adaptive links* called *weights*, as shown in Figure 2.1. Each unit in the network computes some simple function of its inputs (called the *activation function*) and propagates its outputs to other units to which it happens to be connected. A number of activation functions are used in practice, the most common ones being the *threshold*, *linear*, *sigmoid*, and *radial-basis* functions. The weights associated with a unit represent the strength of the synapses between the corresponding units, as will be explained shortly. Each unit is also assumed to be associated with a special weight, called the *threshold* or *bias*, that is assumed to be connected to a constant source of +1 (or a -1). This threshold or bias serves to modulate the firing properties of the corresponding unit and is a critical component in the design of these networks.

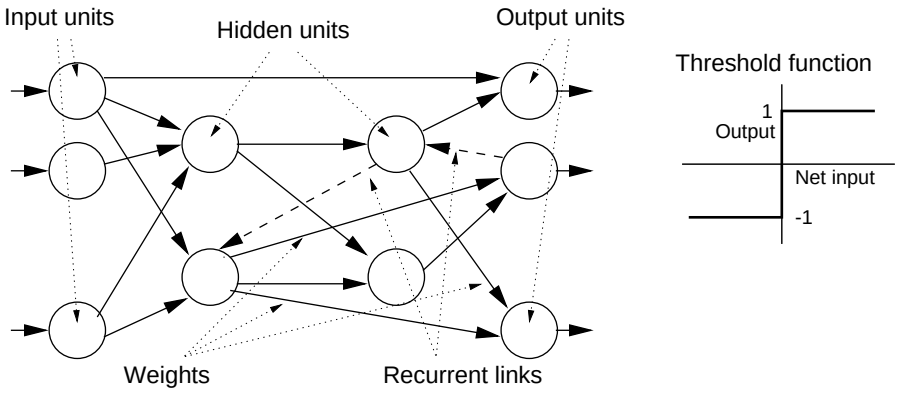

Figure 2.1 Artificial neural network (left) and the bipolar threshold activation function (right).

The input to an n-input (including the bias) unit is typically represented by a pattern vector $\mathbf{X} \in \mathcal{R}^n$ or in the case of binary patterns, by a binary vector $\mathbf{X} \in [0,1]^n$. The weights associated with an n-input unit i are typically represented by an n-dimensional weight vector $\mathbf{W}_i \in \mathcal{R}^n$. By popular convention, the first element of the weight vector usually represents the threshold (or bias). The input activation of a unit i, represented by A_i, in response to a pattern $\mathbf{X}$ on its input links is usually given by the vector dot product: $A_i = \mathbf{W}_i.\mathbf{X}$. The

output of the unit is a function of A_i and is dictated by the activation function chosen. For example, the *bipolar threshold* activation function, shown in Figure 2.1, produces: $O_i = 1$ if $A_i = \mathbf{W}_i.\mathbf{X} \geq 0$ and $O_i = -1$ otherwise.

Units in the network that receive input directly from the environment are referred to as *input units*, while the units that provide the environment with the results of network computations are called *output units*. In conventional neural network terminology, the set of input and output units are said to reside in *input* and *output layers*. In addition to these kinds of units, the networks can also have other units that aid in the network computations but do not have a direct interface to or from the external environment. Such units are referred to as *hidden units*. Often, the hidden units critically determine the kinds of mappings or computations the networks are capable of performing.

In a typical neural network the activations are propagated as follows. At any given instance of time, the input *pattern* is applied to the input units of the network. These input activations are then propagated to the units that are connected to these input units, and the activations of these second layer units are computed. Now the activations of these units are propagated via their output links to other units, and this process continues until the activations reach the units in the output layer. Once the computations of the output layer units are complete, the resulting firing pattern across the output layer units is said to be the output of the network in response to the corresponding input pattern. Thus, in a typical neural network, activations enter the input units, propagate forward through links and hidden units, and produce an activation in the units of the output layer.

A wide variety of artificial neural networks have been studied in the literature. Apart from differences stemming from the activation functions used, neural networks can also be distinguished based on their topological organization. For instance, networks can be single-layered or multi-layered; sparsely connected or completely connected; strictly layered or arbitrarily connected; composed of homogeneous or heterogeneous computing elements, etc. Perhaps the most important architectural (and hence functional) distinction is between networks that are simply *feed-forward* (where their connectivity graph does not contain any directed cycles) and *recurrent* (where the networks contain feedback loops). Feed-forward networks can be trained via a host of simple learning algorithms and have found widespread use in pattern recognition, function interpolation, and system modeling applications. In contrast to feed-forward

networks, recurrent networks have the ability to remember and use past network activations through the use of recurrent (or feedback) links. These networks have thus found natural applications in domains involving temporal dependencies, for instance, in sequence learning, speech recognition, motion control in robots, etc. For further details regarding artificial neural networks and their rather checkered history, the reader is referred to any of a number of excellent texts (Dayhoff, 1990; Hertz *et al.*, 1991; Levine, 1991; Gallant, 1993; Kung, 1993; Haykin, 1994; Ripley, 1996).

2.2.2 Advantages Offered by Neural Networks

As we have mentioned earlier, artificial neural networks have a number of properties that make them particularly attractive for many applications in pattern classification, prediction, and control (Kung, 1993; Haykin, 1994; Ripley, 1996; Zalzala & Morris, 1996; Omidvar & van der Smagt, 1997). These networks can *learn* target functions from examples, *generalize* to situations not seen during learning, *resist noise* and other *perturbations* of the system components, operate *robustly* in the face of failures of units and links, and importantly, their massively parallel structure provides considerable performance gains when implemented in hardware.

Further, it can be shown that artificial neural networks are equivalent to other models of computation such as computers, Post productions, Turing machines, Lambda-calculus, etc., (Uhr & Honavar, 1994; Minsky, 1967; Lewis & Papadimitriou, 1981). Given this equivalence of computational models, the choice of one program representation over another will be dictated by the particular advantages offered by the chosen representational media, which, in light of the many properties of artificial neural networks, makes them particularly attractive for representing control programs for agents and robots. Additionally, since agents and robots can be thought of as artificial (albeit oversimplified) analogs of human information processors, using artificial neural networks for representing agent programs allows us to experiment with mechanisms *conceptually* similar to the neuronal networks of biological agents. For these reasons the rest of this dissertation will make use of neural networks for representing control programs for agent and robot behaviors.

2.2.3 Model of Neural Computation Used in Our Work

As will become evident in later chapters, we use both feed-forward and recurrent threshold networks in our research. However, our neural model employs a slightly different form of activation propagation. The rationale for this is explained below.

As we described earlier, the activation propagation mechanism commonly used in feed-forward networks applies the input activations to input units of the network and computes the activations of the first layer units. Once this is done, the activations of the second layer units are computed, and so on, until the activations of the output layer units have been computed. It should be noted that in such feed-forward networks, once the activations of, say, the first layer units have been computed, they remain unchanged while the activations of units in the subsequent layers are being computed.

This does not hold in recurrent networks. For instance, suppose a recurrent link exists between unit i in the second layer and unit j in the first layer. Now, once the activations of the first layer units have been computed unit j has some activation that was influenced by the activation of unit i. When the activations of the second layer units are computed, it is possible that the new activation of unit i is different from its earlier one. In this case, due to the recurrent link between i and j, the activation of unit j also changes. This new activation of unit j may change the activation of unit i (if there is a forward link between them). Thus, changes in the activations of two (or more) units in a recurrent network can affect each other in a cyclic manner and it may take a while for the activations to settle down to some *equilibrium* values. However, there is no *a-priori* way of knowing how long the network will take to settle down to such an equilibrium. In addition, there is no guarantee that an equilibrium state even exists, and it may so happen that the network activations will keep changing indefinitely. This is referred to as the *oscillation* problem in recurrent networks, since the network oscillates between states without settling to an equilibrium.

In order to avoid problems associated with such oscillations, a special form of activation propagation is often used in recurrent networks. This mechanism is called the *bounded-time* network computation. Here, the network is supplied with its inputs, and the activations of the network units are computed in a *near-synchronous* manner, i.e., rather than compute the activations of the units in a layer-by-layer fashion, all the units in the network compute their activations simultaneously. This process of updating the activations of the units is repeated

for an *a-priori fixed* interval of time called the *bound*. At the end of this time interval, the activations of the output units are taken to be the response of the network (Hertz *et al.*, 1991).

In our work, we use this mode of activation propagation for both feed-forward and recurrent networks, with the fixed time-bound being the delay associated with the computation of the activation of one unit. Thus, the inputs are applied and all the units in the network compute their activations in a near-synchronous fashion. Since all the units cannot compute their activations in a fully synchronized manner (owing to a number of real-world limitations), we can avoid some undesirable effects by associating two *activation buffers* with each of the units. While the buffer labeled IN contains the current activation of a unit and is used by the other units, buffer OUT denotes the place where the new activation of the corresponding unit will be computed. Once all the units in the network have computed their new activations in their respective OUT buffers using activations from the IN buffers of units that they happen to be connected to, the activations in the OUT buffers of the output units are taken to be response of the network. Once this has been done the buffers are updated for the next computation step. Each buffer IN is updated with the new activation computed in the previous step and stored in the buffer OUT. This process repeats. As can be observed, not only does this mode of computation bypass problems associated with oscillations in recurrent networks, but also allows feed-forward networks to capture limited amounts of temporal dependency (since activations propagate through one layer in each time step).

2.2.4 Design of Artificial Neural Networks

As may be inferred, the input-output mapping realized by an artificial neural network is a function of the numbers of units, the functions they compute, the topology of their connectivity, the strength of their connections (weights), the control algorithm for propagating activations through the network, etc., (Honavar, 1994). Thus, to create a neural network with a desired input-output mapping, one has to appropriately design these different components of the network. Not surprisingly, *network synthesis* is an extremely difficult task because the different components of the network and their interactions are often very complex and hard to characterize accurately.

Much of the research on neural network synthesis has focused on algorithms that modify the weights within an otherwise fixed network architecture (Gallant, 1993). This essentially

entails a search for a setting of the weights that endows the network with the desired input-output behavior. For example, in a network used in classification applications we desire weights that will allow the network to correctly classify all (or most of) the samples in the training set. Since this is fundamentally an *optimization* problem, a variety of optimization methods (gradient-descent, simulated annealing, etc.) can be used to determine the weights. Most of the popular learning algorithms use some form of error-guided search (e.g., changing each modifiable parameter in the direction of the negative gradient of a suitably defined error measure with respect to the parameter of interest). A number of such learning algorithms have been developed, both for *supervised* learning (where the desired outputs of the network are specified by an external teacher) and *unsupervised* learning (where the network learns to classify, categorize, or self-organize without external supervision). For details regarding these learning paradigms, the reader is referred to (Gallant, 1993; Kung, 1993; Hassoun, 1995; Ripley, 1996).

Although a number of techniques have been developed to adapt the weights within a given neural network, the design of the neural architecture still poses a few problems. Conventional approaches often rely on human experience, intuition, and rules-of-thumb to determine the network architectures. In recent years, a number of *constructive* and *destructive* algorithms have been developed, that aid in the design of neural network architectures. While constructive algorithms incrementally build network architectures one unit (or one module) at a time (Honavar, 1990; Honavar & Uhr, 1993; Parekh, 1998; Yang *et al.*, 1998), destructive algorithms allow arbitrary networks to be pruned one unit (or one module) at a time. Thus, not only do these approaches synthesize network architectures, but also entertain the possibility of discovering *compact* (or minimal) networks. A number of such constructive and destructive learning algorithms have been developed, each offering its own characteristic bias. Some of these algorithms are discussed in (Parekh, 1998).

In addition to these approaches, *evolutionary algorithms* (to be described shortly) have also been used to search the space of neural architectures for near-optimal designs (see (Balakrishnan & Honavar, 1998) for a bibliography). This evolutionary approach to the design of neural network architectures has been adopted in this dissertation research and is described in Section 2.4.

2.3 Evolutionary Algorithms

Evolutionary algorithms, loosely inspired by biological evolutionary processes, have gained considerable popularity as tools for searching vast, complex, deceptive, and multimodal search spaces (Holland, 1975; Goldberg, 1989; Mitchell, 1996). Following the metaphor of biological evolution, these algorithms work with populations of individuals, where each individual represents a point in the space being searched. Viewed as a search for a *solution* to a problem, each individual then represents (or encodes) a solution to the problem on hand. As with biological evolutionary systems, each individual is characterized by a *genetic representation* or *genetic encoding*, which typically consists of an arrangement of *genes* (usually in a string form). These genes take on values called *alleles*, from a suitably defined domain of values. This genetic representation is referred to as the *genotype* in biology. The actual individual, in our case a solution, is referred to as the *phenotype*. As in biological evolutionary processes, phenotypes in artificial evolution are produced from genotypes through a process of *decoding* and *development*, as shown in Figure 2.2. Thus, while a human being corresponds to a phenotype, his/her chromosomes correspond to the genotype. The processes of nurture, growth, learning, etc., then correspond to the decoding/developmental processes the transform the genotype into a phenotype.

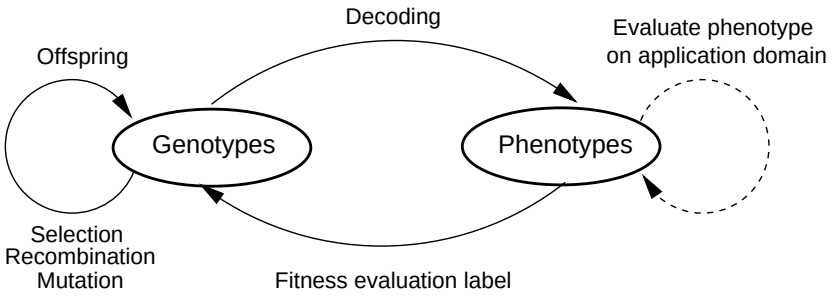

Figure 2.2 The functioning of an evolutionary algorithm.

In artificial evolution (also referred to as simulated evolution), solutions represented by the phenotypes are evaluated based on the target problem for which solutions are being sought. This evaluation of the phenotypes assigns differential fitness labels to the corresponding genotypes. Processes akin to natural selection then preferentially choose genotypes of higher fitness

to participate in probabilistically more numbers of matings. These matings between chosen individuals leads to offsprings that derive their genetic material from their parents via artificial genetic operators that roughly correspond to the biological operators of *recombination* and *mutation*. These artificial genetic operators are popularly referred to as *crossover* and *mutation*. The offspring genotypes are then decoded into phenotypes and the process repeats itself. Over many generations the processes of selection, crossover, and mutation, gradually lead to populations containing genotypes that correspond to high fitness phenotypes. This general procedure, perhaps with minor variations, is at the heart of most evolutionary systems.

The literature broadly distinguishes between four different classes of evolutionary approaches: *genetic algorithms*, *genetic programming*, *evolutionary programming*, and *evolution strategies*. While genetic algorithms typically use *binary* (or bit) strings to represent genotypes (Holland, 1975; Goldberg, 1989), genetic programming evolves *programs* in some given *language* (Koza, 1992). Both these paradigms perform evolutionary search via genetic operators of crossover and mutation. Evolutionary programming, on the other hand, allows complex structures in the genotypes but only uses a mutation operator (Fogel, 1994). Evolution strategies are typically used for parameter optimization (Schwefel, 1981; Bäck *et al.*, 1993). They employ recombination and mutation, and also permit *self-learning* (or evolutionary adaptation) of strategy parameters (e.g., variance of the Gaussian mutations). In recent years, the distinctions between these different paradigms have become rather fuzzy with researchers borrowing from the strengths of different paradigms. For instance, we use complex data structures for representing genotypes and employ both recombination as well as mutation operators to perform the evolutionary search. In this regard our approach may be described as a combination of evolutionary programming and genetic algorithms. For these reasons we prefer to use the generic term, *evolutionary algorithms*, to describe our approach to the use of artificial evolution.

As each population member represents a potential solution, evolutionary algorithms effectively perform a *population-based* search in solution space. Since this is equivalent to exploring multiple regions of the space in parallel, evolutionary algorithms are efficient search tools for vast spaces. In addition, the population-based nature of evolutionary search often helps it overcome problems associated with *local maxima*, making it very suitable for searching multimodal spaces. Further, the genetic encoding and genetic operators can be chosen to be fairly

generic, requiring the user to only specify the decoding function and the *fitness* or *evaluation* function. In most cases these functions can be specified using *little* domain-specific knowledge. Thus, one does not necessarily have to understand the intricacies of the problem in order to use an evolutionary approach to solve it.

2.3.1 Evolutionary Robotics

In addition to their application in a variety of optimization problems, evolutionary algorithms have also been used to design control programs (e.g., artificial neural networks, LISP programs, etc.) for a wide variety of robot tasks (Floreano & Mondada, 1994; Reynolds, 1994a; Harvey *et al.*, 1994; Reynolds, 1994b; Yamauchi & Beer, 1994; Colombetti & Dorigo, 1992; Menczer & Belew, 1994; Walker, 1995; Koza, 1991; Collins & Jefferson, 1991; Lewis *et al.*, 1992; Miglino *et al.*, 1994; Nolfi *et al.*, 1994b; Cecconi *et al.*, 1995; Lund *et al.*, 1997). In such cases, evolutionary search operates in the space of robot control programs, with each member of the population representing a robot *behavior*. By evaluating these behaviors on the target robot task and performing fitness-proportionate reproduction, evolution discovers robot behaviors (control programs) that lead to effective execution of the robot's task. Some researchers have also used artificial evolution to design robot sensors and their placements, tune sensor characteristics, and even evolve robot body plans. Widespread interest in the use of artificial evolution in the design of robots and software agents has given birth to a field that is increasingly being referred to as *evolutionary robotics*. A large portion of this dissertation research concerns itself with evolutionary robotics.

2.4 Evolutionary Design of Neural Architectures

In an earlier section we alluded to the difficulty of synthesizing artificial neural networks that possess specific input-output mappings. Owing to the many properties of evolutionary algorithms, primarily their ability to search vast, complex, and multimodal search spaces using little domain-specific knowledge, they have found natural applications in the automatic synthesis of artificial neural networks. Several researchers have recently begun to investigate evolutionary techniques for designing such neural architectures (see (Balakrishnan & Honavar, 1995a) for a bibliography).

Probably the distinguishing feature of an evolutionary approach to network synthesis is that unlike neural network learning algorithms that typically determine weights within *a-priori* fixed architectures, they can co-design the neural architecture as well as the network weights. Also, in contrast to constructive/destructive algorithms that design network architectures based on some local heuristic measure, the population-based nature of evolutionary search potentially endows it with a more global heuristic. Further, the evolutionary algorithm may be easily extended to automatically adapt other parameters of the network like the individual activation functions of the units (Juedes & Balakrishnan, 1996), rates of mutation (Schwefel, 1987) and learning (Salomon, 1991), and even the learning algorithm (Chalmers, 1990). In addition, by appropriately modifying the fitness function, the same evolutionary system can be used to synthesize vastly different neural networks, each satisfying different task-specific performance measures (e.g., accuracy, speed, robustness, etc.) or user-specified design constraints (e.g., compactness, numbers of units, links and layers, fan-in/fan-out constraints, power consumption, heat dissipation, area/volume when implemented in hardware, etc.). Evolutionary algorithms also allow these networks to be optimized along multiple dimensions either *implicitly* (see Chapter 5) or *explicitly* via the use of different multi-objective optimization approaches (Fonseca & Fleming, 1995; Horn & Nafpliotis, 1993).

2.4.1 An Example of Evolutionary Synthesis of Neural Networks

A number of researchers have designed evolutionary systems to synthesize neural networks for a variety of applications. Here we will present the approach adopted by Miller et al. (1989).

In their system, Miller et al., *encode* the topology of an N unit neural network by a *connectivity constraint matrix* C, of dimension $N \times (N+1)$, as shown in Figure 2.3. Here, the first N columns specify the constraints on the connections between the N units, and the final column codes for the connection that corresponds to the *threshold or bias* of each unit. Each entry C_{ij}, of the connectivity constraint matrix indicates the nature of the constraint on the connection from unit j to unit i (or the constraint on the threshold bias of unit i if $j = N+1$). While $C_{ij} = 0$ indicates the *absence* of a *trainable* connection between units j and i, a value of 1 signals the *presence* of such a trainable link. The rows of the matrix are concatenated to yield a bit-string of length $N \times (N+1)$. This is the genotype in their evolutionary system.

The fitness of the genotype is evaluated as follows. First, the genotype is *decoded* into

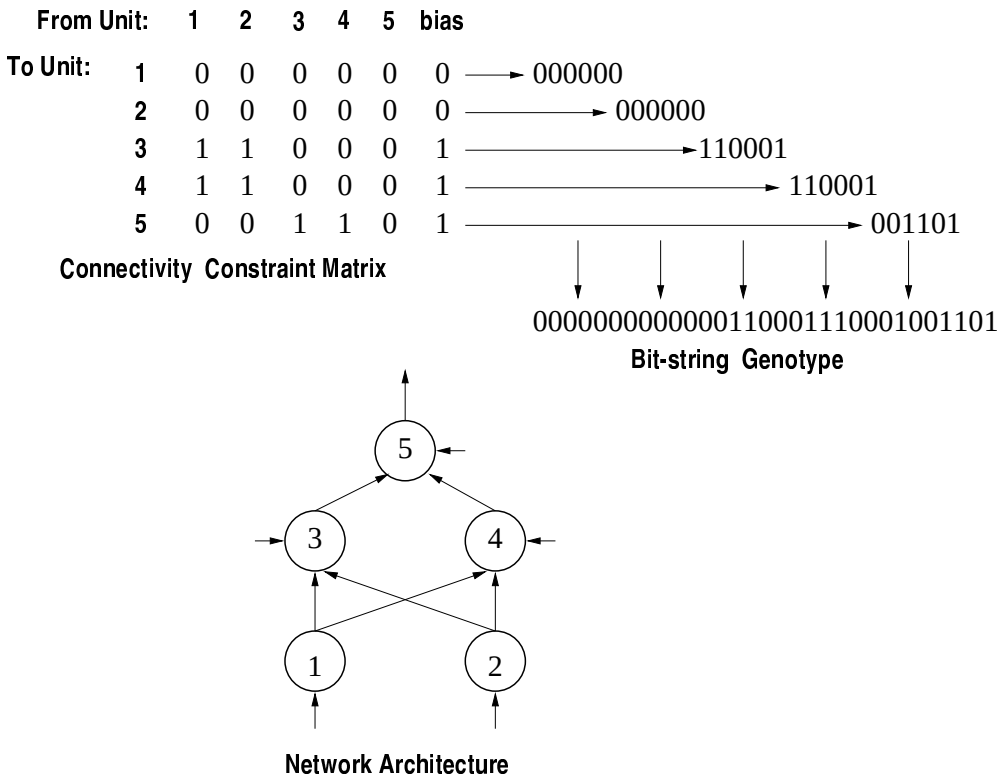

Figure 2.3 An example of an evolutionary approach to the synthesis of neural networks.

the corresponding neural network (or the phenotype). This decoded network has connections (or weights) between units that have a 1 in the corresponding position in the connectivity constraint matrix (or the genotype), as explained earlier. Even though feedback connections can be specified in the genotype, they are ignored by the decoding mechanism. The system thus evolves purely feed-forward networks. Next, all the connections in the network are set to small random values and trained for a fixed number of epochs on a given set of training examples, using the *back-propagation* algorithm (Rumelhart & McClelland, 1986). The *total sum squared error* ($\mathcal{E}$) of the network, at the end of the training phase, is used as the *fitness* measure, with *low* values of $\mathcal{E}$ corresponding to better performance and hence a higher fitness label for the corresponding genotype.

The system maintains a population of such genotypes (bit-strings), and uses a *fitness-proportionate* selection scheme for choosing parents for reproduction. The genetic operator *crossover* swaps rows between parents while *mutation* randomly flips bits in the genotype with

some low, pre-specified probability. The researchers used this evolutionary system to design neural networks for the *XOR, four-quadrant* and *pattern-copying* problems (Miller *et al.*, 1989).

2.5 Genetic Representations of Neural Architectures

Central to any evolutionary design system is the choice of the *genetic representation*, i.e., the encoding of genotypes and the mechanism for decoding them into phenotypes. The choice of the representation is critical since it not only dictates the kinds of phenotypes (and hence solutions) that can be generated by the system, but also determines the amount of *resources* (e.g., time, space, etc.) expended in this effort. Further, the genetic operators for the system are also defined based largely on the representation chosen. These factors contribute directly to the *efficiency* (e.g., time, space, etc.) and the *efficacy* (e.g., quality of solution found, etc.) of the evolutionary search procedure. Thus, a careful characterization of the properties of genetic representations as they relate to the performance of evolutionary systems is a necessary and useful venture.

Some authors have attempted to characterize *properties* of genetic representations of neural architectures (Collins & Jefferson, 1990; Gruau, 1994). However, most such characterizations have been restricted to a specification of the properties of the encoding scheme without considering in detail the associated decoding process. This is an important oversight because the decoding process not only determines the phenotypes that emerge from a given genotype, but also critically influences the resources required in the bargain. For instance, a cryptic encoding scheme might be compact from a storage perspective but might entail a decoding process that is rather involved (both in terms of time and storage space). If we were blind to the effects of the decoding process, we might be enamored by this encoding scheme and use it as our genetic representation. This would have severe consequences on the performance of the evolutionary system.

For these reasons it is imperative that we consider the encoding and the decoding process as a closely related pair while characterizing different properties of genetic representations. Given this motivation, we have identified and precisely defined a number of key properties of genetic representations of neural architectures, taking both the encoding and the decoding processes into account (Balakrishnan & Honavar, 1995b). However, it must be pointed out that this is only a preliminary characterization and we expect these definitions to get more refined as we

examine a large variety of evolutionary systems more closely. The following section describes these properties.

2.5.1 Properties of Genetic Representations of Neural Architectures

In order to develop formal descriptions of properties of genetic representations, we need the following definitions.

- $\mathcal{G}_\mathcal{R}$ is the space of genotypes representable in the chosen genetic representation scheme $\mathcal{R}$. $\mathcal{G}_\mathcal{R}$ may be explicitly enumerated or implicitly specified using a grammar Γ whose language $L(\Gamma) = \mathcal{G}_\mathcal{R}$.

- $p = \mathcal{D}(g, \mathcal{E}_D)$, where $\mathcal{D}$ is the decoding function that produces the phenotype p corresponding to the genotype g possibly under the influence of the environment $\mathcal{E}_D$ (e.g., the environment may set the parameters of the decoding function). A value of λ for $\mathcal{E}_D$ denotes the lack of direct interaction between the decoding process and the environment. It should be borne in mind that $\mathcal{D}$ may be *stochastic*, with an underlying probability distribution over the space of phenotypes.

- $p_2 = \mathcal{L}(p_1, \mathcal{E}_L)$, where the learning procedure $\mathcal{L}$ generates phenotype p_2 from phenotype p_1 under the influence of the environment $\mathcal{E}_L$. The environment may provide the training examples, set the free parameters (e.g., the learning rate used by the algorithm) etc. We will use $\mathcal{L} = \lambda$ to denote the absence of any form of learning in the system. In the following discussion we will use the term *decoding function* to refer to both $\mathcal{D}$ and $\mathcal{L}$. This slight abuse of notation allows the following properties to apply to genetic representations in general, even though they are presented in the context of evolutionary design of neural architectures.

- $\mathcal{P}_\mathcal{R}$ is the space of all phenotypes that can be constructed (in principle) given a particular genetic representation scheme $\mathcal{R}$. Mathematically, $\mathcal{P}_\mathcal{R} = \{p / \exists g \in \mathcal{G}_\mathcal{R}[(p_1 = \mathcal{D}(g, \mathcal{E}_D)) \wedge (p = \mathcal{L}(p_1, \mathcal{E}_L))]$

- $\mathcal{S}$ is the set of *solution networks*, i.e., neural architectures or phenotypes that satisfy the desired performance criterion (as measured by the fitness function π) in a given environment $\mathcal{E}_\pi$. If an evolutionary system with a particular representation $\mathcal{R}$ is to

successfully find solutions (even in principle), $\mathcal{S} \subseteq \mathcal{P}_{\mathcal{R}}$, or, at the very least, $\mathcal{S} \cap \mathcal{P}_{\mathcal{R}} \neq \emptyset$. In other words, there must be at least one solution network that can be constructed given the chosen representation $\mathcal{R}$.

- $\mathcal{A}$ is the set of *acceptable* or *valid* neural architectures. For instance, a network may be deemed invalid or unacceptable if it does not have any paths from the inputs to the outputs. In general, $\mathcal{A}$ may be different from $\mathcal{P}_{\mathcal{R}}$. However, it must be the case that $\mathcal{A} \cap \mathcal{S} \neq \emptyset$ if a particular evolutionary system is to be useful in practice.

We now identify some properties of genetic representations of neural architectures. Unless otherwise specified, we will assume the following definitions are with respect to an *a-priori* fixed choice of $\mathcal{E}_D, \mathcal{L}$, and $\mathcal{E}_L$.

1. **Completeness:** A representation $\mathcal{R}$ is *complete* if every neural architecture in the solution set can be constructed (in principle) by the system. Formally, the following two statements are equivalent definitions of completeness.
   - $(\forall s \in \mathcal{S})(\exists g \in \mathcal{G}_{\mathcal{R}})[(p_1 = \mathcal{D}(g, \mathcal{E}_D)) \wedge (s = \mathcal{L}(p_1, \mathcal{E}_L))]$
   - $\mathcal{S} \subseteq \mathcal{P}_{\mathcal{R}}$

   Thus, completeness demands that the representation be capable of producing all possible solutions to the problem. Often, this may be hard to satisfy and one may have to choose between *partially complete* representations. In such cases, another figure of merit called *solution density*, denoted by $\frac{|\mathcal{S} \cap \mathcal{P}_{\mathcal{R}}|}{|\mathcal{P}_{\mathcal{R}}|}$, might be useful. One would then choose representations that correspond to higher solution densities, since this implies a higher likelihood of finding solutions. It should be noted that if the solution density is very high, even a *random search* procedure will yield good solutions and one may not have much use for an evolutionary approach.

2. **Closure:** A representation $\mathcal{R}$ is *completely closed* if *every* genotype decodes to an acceptable phenotype. The following two assertions are both equivalent definitions of closure.
   - $(\forall g \in \mathcal{G}_{\mathcal{R}})[(p_1 = \mathcal{D}(g, \mathcal{E}_D)) \wedge (\mathcal{L}(p_1, \mathcal{E}_L) \in \mathcal{A})]$
   - $\mathcal{P}_{\mathcal{R}} \subseteq \mathcal{A}$

   A representation that is not closed can be transformed into a closed system by *constraining* the decoding function, thereby preventing it from generating invalid phenotypes. Additionally, if the genetic operators are designed to have the property of closure, then one

can envision *constrained closure* wherein all genotypes do not correspond to acceptable phenotypes, however, closure is guaranteed since the system *never* generates the invalid genotypes. Closure has bearings on the *efficiency* of the evolutionary procedure as it determines the amount of effort (space, time, etc.) wasted in generating unacceptable phenotypes.

3. **Compactness:** Suppose two genotypes g_1 and g_2 both decode to the same phenotype p, then g_1 is said to be more *compact* than g_2 if g_1 occupies less *space* than g_2:
   • $(p_1 = \mathcal{D}(g_1, \mathcal{E}_D)) \wedge (p = \mathcal{L}(p_1, \mathcal{E}_L)) \wedge (p_2 = \mathcal{D}(g_2, \mathcal{E}_D)) \wedge (p = \mathcal{L}(p_2, \mathcal{E}_L)) \wedge \mid g_1 \mid < \mid g_2 \mid$
   where $\mid g \mid$ denotes the size of storage for genotype g.

   This definition corresponds to *topological-compactness* defined by Gruau (1994). His definition of *functional-compactness* – which compares the genotype sizes of two phenotypes that exhibit the same behavior, can be expressed in our framework (for solution networks) as
   • $(p_1 = \mathcal{D}(g_1, \mathcal{E}_D)) \wedge (\mathcal{L}(p_1, \mathcal{E}_L) \in \mathcal{S}) \wedge (p_2 = \mathcal{D}(g_2, \mathcal{E}_D)) \wedge (\mathcal{L}(p_2, \mathcal{E}_L) \in \mathcal{S}) \wedge \mid g_1 \mid < \mid g_2 \mid$

   Compactness is a useful property as it allows us to choose genetic representations that use space more efficiently. However, compact and cryptic representations often require considerable decoding effort to produce the corresponding phenotype. This is the classic space-time tradeoff inherent in algorithm design. Hence, the benefits offered by a compact representation must be evaluated in light of the increased decoding effort before one representation can be declared preferable over another.

4. **Scalability:** Several notions of scalability are of interest. For the time being we will restrict our attention to the *change* in the *size* of the phenotype, measured in terms of the numbers of units, connections, or modules. This change in the size of the phenotype manifests itself as a change in the *size* of the encoding (space needed to store the genotype), and a corresponding change in *decoding time*. We can characterize the relationship in terms of the *asymptotic order of growth* notation commonly used in analyzing computer algorithms — $O(\cdot)$.

   For instance, let $n_{N,C} \in \mathcal{A}$ be a network (phenotype) with N units and C connections (the actual connectivity pattern does not really matter in this example). We say that the representation is $O(K)$*–size-scalable with respect to units* if the addition of *one* unit

to the phenotype $n_{N,C}$ requires an increase in the size of the corresponding genotype by $O(K)$, where K is some function of N and C. For instance, if a given representation is $O(N^2)$ size-scalable with respect to units, then the addition of one unit to the phenotype increases the size of the genotype by $O(N^2)$. Size-scalability of encodings with respect to connections, modules, etc., can be similarly defined.

The representation is said to be $O(K)$– *time-scalable with respect to units* if the time taken for decoding the genotype for $n_{N+1,C}$ exceeds that used for $n_{N,C}$ by no more than $O(K)$. Similarly, time-scalability with respect to the number of connections, modules, etc., can also be defined.

Scalability is central to understanding the space-time consequences of using a particular genetic representation scheme in different contexts. In conjunction with completeness and compactness, scalability can be effectively used to characterize genetic representations.

5. **Multiplicity:** A representation $\mathcal{R}$ is said to exhibit *genotypic multiplicity* if multiple genotypes decode to an identical phenotype. In other words, the decoding function is a many to one mapping from the space of genotypes to the corresponding phenotypic space.

   • $(\exists n \in \mathcal{P}_{\mathcal{R}})\ (\mid \{g \in \mathcal{G}_{\mathcal{R}} / (p = \mathcal{D}(g, \mathcal{E}_D)) \wedge (n = \mathcal{L}(p, \mathcal{E}_L))\} \mid > 1)$

   Genotypic multiplicity may result from a variety of sources including the encoding and decoding mechanisms. If a genetic representation has the property of genotypic multiplicity, it is possible that multiple genotypes decode to the same *solution* phenotype. In such cases, if the density of solutions is also high, then a large fraction of the genotypic space corresponds to potential solutions. This will make the evolutionary search procedure very effective.

   A representation $\mathcal{R}$ is said to exhibit *phenotypic multiplicity* if different instances of the same genotype can decode to different phenotypes. In other words, the decoding function is a one to many mapping of genotypes into phenotypes.

   • $(\exists g_1, g_2 \in \mathcal{G}_{\mathcal{R}})[(p_1 = \mathcal{D}(g_1, \mathcal{E}_D)) \wedge (n_1 = \mathcal{L}(p_1, \mathcal{E}_L))) \wedge (p_2 = \mathcal{D}(g_2, \mathcal{E}_D)) \wedge (n_2 = \mathcal{L}(p_2, \mathcal{E}_L))) \wedge (g_1 = g_2) \wedge (n_1 \neq n_2)]$

   Phenotypic multiplicity may result from several factors including the effects of the environment, learning, or stochastic aspects of the decoding process. If the density of

solutions is low, then the property of phenotypic multiplicity increases the possibility of decoding to a solution phenotype.

6. **Ontogenetic Plasticity:** A representation $\mathcal{R}$ exhibits *ontogenetic plasticity* if the determination of the phenotype corresponding to a given genotype is influenced by the environment. This may happen as a result of either environment-sensitive developmental processes (in which case $\mathcal{E}_D \neq \lambda$), or learning processes (in which case $\mathcal{L} \neq \lambda$).

   Ontogenetic plasticity is a useful property for constraining or modifying the decoding process based on the dictates of the application domain. For instance, if one is evolving networks for a pattern classification problem, the search for a solution network can be dramatically enhanced by utilizing a *supervised* learning algorithm for training individual phenotypes in the population. However, if such training examples are not available to permit supervised learning, one will have to be content with a purely evolutionary search.

7. **Modularity:** Gruau's (1994) notion of *modularity* is as follows: Suppose a *network* n_1 includes several instances of a subnetwork n_2 then the encoding (genotype) of n_1 is *modular* if it codes for n_2 only once, with instructions to copy it that would be understood by the decoding process. Modularity is closely tied to the existence of *organized structure* or regularity in the phenotype that can be concisely expressed in the genotype in a form that can be used by the decoding process. Other notions of modularity dealing with *functional* modules, *recursively-defined* modules etc., are also worth exploring.

   It can be observed that the property of modularity automatically results in more compact genetic encodings and a potential *lack* of *redundancy* (described below). In modular representations any change in the genotypic encoding of a module, either due to genetic influences or errors, affects all instances of the module in the phenotype. Non-modular representations, on the other hand, are resistive to such complete alterations. It is hard to decide *a-priori* which scenario is better, since modular representations benefit from benign changes while non-modular representations are more robust to deleterious ones.

8. **Redundancy:** Redundancy can manifest itself at various levels and in different forms in an evolutionary system. Redundancy often contributes to the robustness of the system in the face of failure of components or processes. For instance, if the reproduction and/or decoding processes are error-prone, an evolutionary system can benefit from *genotypic*

redundancy (e.g., the genotype contains redundant genes) or *decoding redundancy* (e.g., the decoding process reads the genotype more than once). If the phenotype is prone to failure of components (e.g., units, connections, sub-networks, etc.), the system can benefit from *phenotypic redundancy*. Phenotypic redundancy can be either *topological* (e.g., multiple identical units, connections, etc.) or *functional* (e.g., dissimilar units, connections, etc., that somehow impart the same function).

It is worth noting that genotypic redundancy does not necessarily imply phenotypic redundancy and vice versa (depending on the nature of the decoding process). This simply reiterates the importance of examining the entire representation (encoding as well as decoding) when defining properties of evolutionary systems. Also note that there are many ways to realize both genotypic as well as phenotypic redundancy: by replication of identical components (structural redundancy) or by replication of functionally identical units, or by building in modules or processes that can dynamically restructure themselves when faced with failure of components etc. (von Neumann, 1956).

9. **Complexity:** Complexity is perhaps one of the most important properties of any evolutionary system. However, it is rather difficult to characterize satisfactorily using any single definition. It is probably best to think of complexity using several different notions including: *structural complexity* of genotypes, *decoding complexity*, *computational* (*space/time*) *complexity* of each of the components of the system (including decoding of genotypes, fitness evaluation, reproduction, etc.), and perhaps even other measures inspired by *information theory* (e.g., entropy, Kolmogorov complexity, etc.) (Li & Vitanyi, 1997).

   Although it is clear that one would like to use a genetic representation that leads to lower system complexities, the many interacting elements of the evolutionary system, genetic representations and their properties, and the existence of many different kinds of complexities, make it hard to arrive at *one* scalar measure that would satisfy all.

This list of properties, although by no means complete, is nevertheless relevant in an operationally useful characterization of evolutionary systems in general, and the design of neural architectures in particular. Table 2.1 illustrates a characterization of the evolutionary system proposed by Miller et al. (1989), that was described in Section 2.4.1.

Table 2.1 Properties of the genetic representation used by Miller et al.

Property	Satisfied?	Comments
Completeness	$\surd$	With respect to feed-forward networks.
Closure	$\times$	Invalid networks can result.
Topological Compactness	$\surd$	Determined by back-propagation.
Functional Compactness	$\surd$	Also possible.
Space Scalability	$\surd$	$O(N)$ with respect to units.
Time Scalability	$\surd$	$O(N)$ with respect to units.
Genotypic Multiplicity	$\times$	No genotypic multiplicity.
Phenotypic Multiplicity	$\surd$	Dictated by back-propagation.
Ontogenetic Plasticity	$\surd$	Back-propagation used for training.
Modularity	$\times$	Genotype only specifies connections.
Genotypic/Decoding Redundancy	$\times$	One gene for each connection.
Phenotypic Redundancy	$\surd$	Units and modules, but not connections.
Space Complexity	$\surd$	Dictated by genotype size.
Time Complexity	$\surd$	Dictated by GA and back-propagation.

2.6 Discussion

In this chapter we have introduced artificial neural networks as a representational media for robot control programs. The suitability of such neural networks for robot control stems from a number of advantages offered by these networks, including their ability to absorb moderate amounts of noise, generalize and respond to scenarios not encountered earlier, function reliably in the face of modest failures and malfunctions of system components, and importantly, the possibility of learning robot behaviors through experience. We presented a brief discussion of neural networks and outlined some approaches to synthesizing such networks including algorithms that modify weights within *a-priori* fixed network architectures and algorithms that attempt to design the network architectures themselves.

We also introduced evolutionary algorithms as loose artificial models of biological evolutionary processes that work in nature. Such evolutionary processes are becoming increasingly popular owing to a number of advantages offered by them, particularly their ability to efficiently and effectively search vast, complex, and multimodal solution spaces using little domain-specific knowledge. This has immediate applications in the synthesis of network architectures, since such evolutionary approaches allow many different elements of the networks to be co-designed or co-evolved (e.g., the network architecture, connection weights, unit activation functions,

rates of mutation and learning, and even the learning algorithm). In addition, by appropriately designing the fitness function used in the evolutionary algorithm, one can use the same system to design neural architectures satisfying different sets of performance and user-specified constraints. Motivated by some of these reasons, we decided to adopt an evolutionary approach to the design of neural network controllers for robot behaviors.

Since the efficiency and efficacy of any evolutionary design system is critically governed by the encoding mechanism chosen for specifying the genotypes and the decoding mechanism for transforming them into phenotypes, extreme care must be taken to ensure that these two mechanisms are designed with the application problem in mind. To aid this process, we identified and formalized a number of properties of such genetic representations. To the extent possible, we have tried to characterize each property in unambiguous mathematical terms. It is our hope that these properties will help identify good choices of genetic representations for different applications. For instance, suppose we need to design neurocontrollers for robots that have to operate in hazardous and *a-priori* unknown environments. Examples of such applications include exploration of unknown terrains, nuclear waste cleanup, geo and space exploration, etc. Since robots in such environments are required to plan and execute sequences of actions (where each action in a sequence may be dependent on previous actions performed as well as the sensory inputs), a recurrent neural network is probably needed. Further, if the system is to be used to design robots capable of functioning in different, *a-priori* unknown environments, the robot controllers must have ontogenetic plasticity, i.e., the robots must be capable of learning from their experiences in the environment. The hazardous nature (e.g., in nuclear waste cleanup) or remoteness of the environment (e.g., in the case of robots used to explore distant planets) makes it desirable that the controllers operate robustly in the face of component failures etc., which calls for phenotypic redundancy of some form (e.g., duplication of units, links, or modules of the neurocontroller). In addition, implementation technology and cost considerations might impose additional constraints on the design of the controller. For instance, hardware realization using current VLSI technology would benefit from locally connected, modular networks built from simple processors. Also extended periods of autonomous operation might require designs that are efficient in terms of power consumption, etc.

In order to design a robot controller satisfying these multiple performance constraints, one

might resort to an evolutionary design approach. In such cases, a number of these constraints translate into properties that we have identified in Section 2.5.1. Using these, one can choose an appropriate genetic representation that can be used to evolve appropriate robot behaviors. In the following chapters we present an evolutionary approach to the synthesis of robot behaviors for a box-pushing robot task, where we choose a genetic representation based on the properties we have identified in this chapter.

3 A BOX-PUSHING ROBOT TASK

In this chapter we will present a task that requires a robot to clear a room by pushing boxes in the room to the walls. Our goal is to design an appropriate neural network controller that will allow the robot to push as many boxes to the walls as possible, within a limited amount of time. We will outline the constraints associated with this task and argue for the need for an evolutionary approach to the design of neurocontrollers. We will also compare this task to a number of others commonly pursued in evolutionary robotics, and show that this box-pushing task offers many more challenges. Finally, we will present simulation results of a number of experiments in the evolutionary synthesis of neurocontrollers and demonstrate the ability of evolution to produce designs well adapted to the constraints and limitations of the task.

3.1 Box-Pushing Agents of Teller

Teller (1994) proposed an interesting task for studying the evolution of control behaviors in artificial agents and robots. The task environment consisted of a square room of dimension $N \times N$ cells, which was littered with M boxes, as shown in Figure 3.1. The room had impenetrable, delimiting walls.

The robot (or agent) had the task of clearing the room by pushing the boxes to the enclosing walls. The robot had eight sensors capable of detecting boxes, walls, and empty spaces. These sensors were placed to sense one cell in each of the eight directions around the robot's current position, as shown by the shaded region in Figure 3.2. Thus, the robots were *sensorily handicapped*, with the robot being *blind* beyond one cell in each direction. The sensors were also assumed to be fixed to the robot and hence were assumed to turn with the robot, as shown in Figure 3.2.

Based on its sensory inputs and memory, the robot could choose to perform one of three actions: forward move, left turn through 90 degrees, or right turn through 90 degrees. The

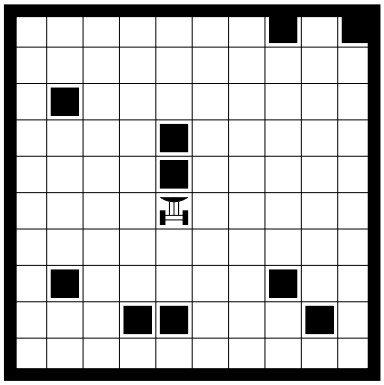

Figure 3.1 The box-pushing environment.

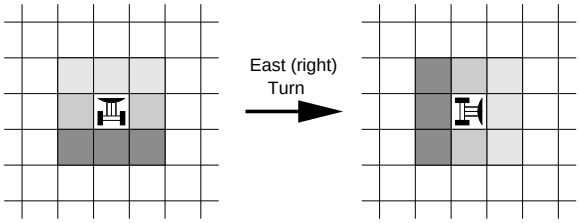

Figure 3.2 Sensors have a range of one cell and are fixed to the robot.

robot was thus incapable of moving diagonally through its environment and could only move in directions parallel to the walls. The robot was assumed to turn *in place*, i.e., it remained in the same cell after executing a turn although it was now facing a different direction. Forward moves, on the other hand, took the robot from one cell to the next, in the direction in which it was currently facing. Turn actions executed by the robot were always considered successful, although forward moves could fail under two circumstances. First, if there was a wall in front of the robot, it could not move into the wall (or push boxes through them). Second, the robot could not push more than one box at the same time. For instance in Figure 3.1, if the robot wanted to move forward, the move would fail because there are two boxes in front of the robot and it is only capable of pushing at most one. The first constraint was something we would expect, however, the second constraint was critical to the performance of the robot since it prevented the robot from collecting multiple boxes and pushing them to the walls together. The robots were thus forced to move boxes to the walls one at a time.

What made the robot task even harder was the inability of the robot to *detect* such failed movements. For instance, if the robot happened to be against a wall and attempted to move forward, the move would fail. However, owing to its inability to detect such failures, the robot would consider the move successful. For obvious reasons this had undesirable repercussions on the behaviors of the robot.

These constraints, namely: limited sensor ranges, restricted movements, limited box-pushing ability, and the inability to detect failed actions make the task very hard, and the design of an appropriate robot controller immensely challenging.

3.1.1 Genetic Evolution of Robot Control Programs

Teller used a Genetic Programming framework to evolve robot behavior programs for the box-pushing task. His important contribution was the addition of *state* information to standard Genetic Programming. In particular, his evolved agents had access to 20 *indexed memory elements* which they could use to remember past information. The evolved programs could make use of 10 predefined functions for reading sensory and memory data, performing mathematical operations on them, comparing numerical quantities, branching, and writing to indexed memory elements.

Teller used environments of size 6×6 cells with six boxes randomly placed in the inner 4×4 grid. Each agent was introduced in a random cell in the arena (facing a random direction), and allowed a maximum of 80 simulation time steps within which to move the boxes to the walls. In each time step, the agent sensors sensed the corresponding cell to which they were tuned, returning a value of 0, 1, or 2 corresponding to an empty cell, a box, or a wall respectively. Using these sensory inputs and its state (index memory) information, the agent behavior program determined an appropriate action to perform and the agent was made to execute the action. At the end of the simulation period, each box against a wall was awarded one point while boxes pushed into corners earned an extra point. Thus, the maximum fitness attainable in this environment was 10 (all six boxes along walls with four of them in corners). Each agent was introduced into 40 such random environments and its average performance over them was declared its fitness. For further details regarding this experiment, the reader is referred to (Teller, 1994).

Teller performed a number of evolutionary experiments with different kinds of agents. He

found that without access to indexed memory the evolved programs performed rather poorly, causing the box-pushing agent to obtain an average fitness of less than 0.5 points per test. In the second experiment, Teller provided the agents with a fourth action that enabled the agents to perform random-walks. While the first three actions were the same as before, the fourth action, when chosen by the control program, allowed the agent to randomly choose and perform one of the three actions. With this *noise source*, evolution produced programs (and hence agents) with improved fitnesses, albeit with average fitnesses less than 1.0 point per test.

Teller also evolved *mental* agents that had the ability to access and use indexed memory. He found considerable improvement in fitness over agents without such memory abilities. In particular, agents with memory, but without automatically defined functions (ADFs), obtained an average fitness of about 3.0 points per test, while memory agents using ADFs performed much better, with average fitnesses of 4.25 points per test (Teller, 1994). The best agent program discovered by evolution resulted in agent fitness of 4.4 per fitness test.

By switching off specific memory elements and sensors, Teller also identified the role played by these in the fitnesses of the mental agents. He found that damage of memory cells led to a marked reduction in agent fitness. He also observed that the effect of sensor damage on agent fitness depended on the position of the corresponding sensor. In particular, damage to sensors located to sense cells diagonally across from the robot position did not affect the fitness of the agent. However, sensors located to sense cells to the North, South, East, and West directions (with respect to a North facing robot) played a critical role in determining the agent fitness and their damage led to a significant drop in agent performance.

Although Teller was able to identify that *memory* was critical for successful box-pushing behaviors, he was unable to *analyze* the evolved robot control programs to determine *how* the robots used their memories to achieve their fitnesses. He was thus unable to *characterize agent behaviors* in the box-pushing environment. As will become clear later, through our analysis we have been able to characterize successful box-pushing behaviors.

3.1.2 Comparison to Other Tasks in Evolutionary Robotics

Most of the robot tasks used in *evolutionary robotics* studies have been simple variants of basic navigation behaviors like obstacle avoidance (Floreano & Mondada, 1994; Reynolds, 1994a), goal approaching (Harvey *et al.*, 1994), wall following (Reynolds, 1994b; Yamauchi

& Beer, 1994), light or target following (Colombetti & Dorigo, 1992), feeding (Menczer & Belew, 1994; Walker, 1995), homing, maze or trail learning (Koza, 1991; Collins & Jefferson, 1991), simple exploration (Lewis *et al.*, 1992; Miglino *et al.*, 1994), etc. In contrast, the box-pushing task described above has a number of salient properties that make it significantly more interesting and challenging.

Firstly, the box-pushing task is *dynamic*, in the sense that the robot can *alter* its own environment and hence effect its *fitness landscape*. For instance, suppose the robot is introduced in an environment containing six boxes, where no two boxes are together. Given this state of the environment, the maximum fitness attainable by the robot is 10 (all six boxes against the walls, with four of them in corners). Now suppose the robot moves in such a way that four boxes are somehow collected together in the middle of the arena in the form of a closed square. Since the robot cannot push more than one box at a time, it cannot push any of these boxes to the wall. The maximum attainable fitness has now dropped to 4 (two remaining boxes, both in corners). Thus, the behavior of the robot dynamically alters its fitness landscape. It is much harder to find good control functions (or behaviors) in such dynamic environments, which makes the box-pushing task more challenging than behaviors like obstacle avoidance, wall following, etc., where the environments are static.

Secondly, the robots of Teller have limited sensory and pushing abilities (probably overly limited). In any case, this scenario is more realistic (given the state of contemporary robotics technology) than the assumptions made in a number of other evolutionary robotics tasks. For instance, a number of approaches assume that the robots have infinite sensor range (bounded only by the limits of the environment) (Nolfi *et al.*, 1994b; Walker, 1995). Since all biological and artificial sensors are physically constrained to operate within specific limits (range, resource usage, processing time, etc.), such assumptions are quite unrealistic and will lead to problems if the designs are ever to be implemented on actual robots. In contrast, the sensorily and operationally-constrained robots of Teller offer more realistic conditions to work with.

Finally, the box-pushing task *subsumes* a number of other behaviors mentioned above. For instance, to function well in their environment Teller's robots have to move and explore their environment, approach and push boxes, identify and avoid walls, etc. These primitive behaviors must be interleaved and mixed together in appropriate ways for the robot to excel in pushing many boxes to the walls. Interleaving these primitive behaviors involves multiple tradeoffs.

For instance, the robot should be capable of *exploring* different regions of its environment to find boxes to push. In this process, it must identify and avoid bumping into walls. Once it finds boxes, it must decide whether to push them to walls (at the risk of deviating from its exploratory behavior), or continue with its exploratory behavior and find other boxes to move. This is reminiscent of the *exploration versus exploitation* dilemma of search algorithms, i.e., should it exploit what is has found (push box) or should it explore more (find other boxes). Assuming that the robot somehow performs this tradeoff and decides to push a box, it also has to decide when to stop pushing the box and continue its exploration. This is rather difficult in the current setup since the limited sensory range of the robot prevents it from detecting walls ahead of the box which it happens to be pushing. As the robots are also incapable of detecting failed actions, the robot really has no way of knowing when the box it is pushing comes against a wall.

As may be surmised, the behaviors required by robots in the box-pushing task must be well-balanced combinations of the primary behaviors of exploration, approach, and avoidance, modulated in appropriate ways by the constraints associated with the task. For these reasons it is extremely difficult, if not impossible, to *manually* design controllers for these box-pushing robots. In contrast, it is rather easy to manually develop controllers to approach and avoid obstacles, follow walls, or navigate in open spaces.

These aspects set the box-pushing task apart from most of the others used by researchers in the evolutionary robotics community. The challenges inherent in this task offer a multitude of opportunities for the evolution of robots with interesting behaviors, and for these reasons, we use the box-pushing robot task in our research. We also employ neural networks to represent robot control programs and evolutionary algorithms to search the space of neurocontroller designs, for reasons outlined in Chapter 2.

3.2 Simulation Details

We have performed a number of experiments in the evolution of neurocontrollers for the box-pushing robot task. In this section we present details of our simulation setup.

3.2.1 Neural Network Structure and Output Coding Strategies

The neurocontrollers in our simulations used eight input units, each deriving input from one robot sensor. The robot sensors (and hence the input units) provided a value of 0, 1, or -1 to the neurocontroller, in response to an empty cell, box, or wall respectively. While some experiments did not permit the use of hidden units, others allowed up to 10 hidden units to be used. These hidden units, when used, computed *bipolar threshold* functions, i.e., they produced an output of +1 when the net input activation was greater than zero and a -1 otherwise, as described in Section 2.2.1.

The number of output units in the neurocontroller (along with their activation functions), were dictated by the *output coding strategy* used, i.e., the mechanism used to translate the neurocontroller outputs into robot actions. For instance, the *Left-Forward-Right* (LFR) output coding strategy requires the use of three output units, one corresponding to the left-turn action, one for forward moves, and one for right-turns. In the LFR scheme, the output units compute their input activations and then engage in a *winner-take-all* computation. This simply means that the unit with the largest activation is declared the winner and it produces an output of +1 while the remaining two units produce outputs of -1. The robot then performs the action associated with the unit that is the winner. This output coding strategy is shown in Table 3.1.

Table 3.1 Left-Forward-Right output coding strategy.

Unit L	Unit F	Unit R	Robot Action
+1	-1	-1	Turn Left
-1	+1	-1	Move Forward
-1	-1	+1	Turn Right

Some of our experiments have also used an output coding strategy suggested by Braitenberg (Braitenberg, 1984). This strategy, labeled *Braitenberg* (BR), uses two output units that compute bipolar-threshold functions. The two output units are considered to be directly connected to the two wheels of the robot (assuming the robot has one wheel on either side). In this scheme, if an output unit produces a +1, the corresponding wheel turns in the forward direction and vice-versa. Appropriate combinations of the two wheels then lead to forward and backward moves, and to left and right turns, as shown in Table 3.2. As we do not allow

Table 3.2 Braitenberg's output coding strategy.

Unit L	Unit R	Robot Action
-1	-1	Move Forward
-1	+1	Turn Left
+1	-1	Turn Right
+1	+1	Move Forward

backward moves in our simulations, we interpret that combination (-1, -1), as a forward move.

We have also experimented with another output coding strategy called *Action-Direction* (AD), which makes use of two output units that compute bipolar-threshold functions. The output of the first unit is interpreted as the *action* to be performed (+1 to move forward and -1 to turn) while the second unit indicates the direction of *turn* (+1 to turn left and -1 to turn right). This scheme is illustrated in Table 3.3.

Table 3.3 Action-Direction output coding strategy.

Unit A	Unit D	Robot Action
-1	-1	Turn Right
-1	+1	Turn Left
+1	-1	Move Forward
+1	+1	Move Forward

As can be observed, the output coding strategy dictates the number and interpretation of the neurocontroller outputs.

3.2.2 Genetic Representation

The genetic representation used in our experiments is a hybrid of a number of representations used in the evolutionary robotics community. It was carefully designed to possess a number of properties listed in Section 2.5.1 (although this is not necessarily the most optimal genetic representation). As shown in Figure 3.3, our representation explicitly encodes the connectivity (or topology) of the neurocontroller. Each *gene* in the representation corresponds to the input connectivity of a hidden or output unit. This gene (or input connectivity) itself

corresponds to a number of *connections* between units. For instance, the gene corresponding to hidden unit 0 in Figure 3.3 contains three connections, while the gene corresponding to output unit F contains six connections (the LFR output coding strategy being used in this case). Each connection is specified as a 3-tuple represented by: (LayerID, UnitNo, Weight). Here LayerID is I, H, O, or B corresponding to input, hidden, or output layers, with B denoting the bias (threshold) of the unit. UnitNo identifies the specific unit in LayerID which serves as the *source* of that connection and Weight, an integer in [-100, +100], denotes the strength of the connection.

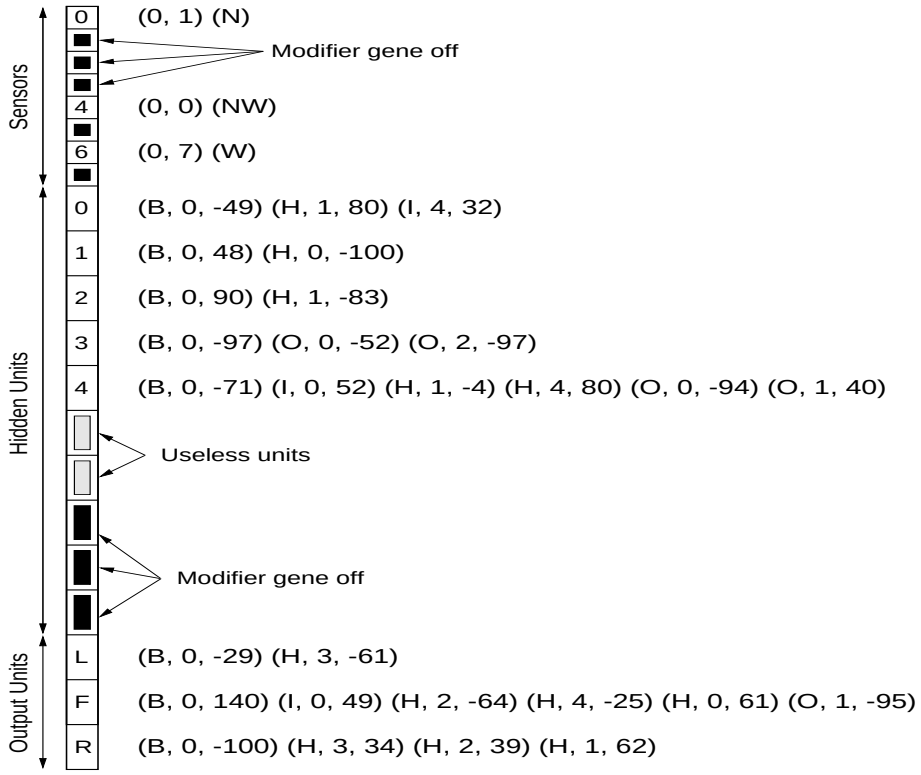

Figure 3.3 Genetic representation used in our simulations.

For instance, hidden unit 0 has a bias value of -49, a *recurrent* connection from hidden unit 1 of strength 80, and a connection from input unit 4 with a weight of 32. In our experiments, we have restricted the number of connections to any single unit (also called the *fan-in* of the unit) to a maximum of 10.

In addition to the input connectivities of the neurocontroller units, the genetic represen-

tation also supports the evolution of robot sensors. Although the experiments presented in this chapter do not allow the evolution of sensors, later experiments do so. If those cases, the genetic representation additionally contains genes that correspond to robot sensors. Each such sensor gene encodes the *position* of the corresponding robot sensor. Note that this representation implicitly allows the evolution of not only the placement of the robot sensor but also the tuning of its range. For instance, if a sensor is placed to sense a cell two units away, its range is automatically tuned to two units.

Figure 3.3 shows the genetic encoding of three sensors. While sensor numbered 0 is placed to sense the cell immediately ahead of the robot, sensor 4 observes the cell forward and to the left, and sensor 6 senses the cell to the left of the robot. For convenience, we will label sensors based on the cells they observe when the robot *faces north*. Thus, a sensor that observes the cell immediately ahead of the robot is labeled *N*, while the sensor observing the cell to the left of the north-facing robot is labeled *W*. It should be noted that even though the genetic representation in Figure 3.3 has a W sensor, it is effectively useless as none of the neurocontroller units derive input from it.

The encoding of the sensors, i.e., how (0, 1) translates to sensor N and (0, 7) translates to W, will be explained in Chapter 4. In the experiments reported in this chapter the sensor positions do not evolve. The neurocontrollers thus derive input from eight sensors that are fixed to sense the eight cells immediately around the robot.

Our genetic representation also assumes the existence of *second-level* or *modifier genes* which control the *expression* of entire sequences of other genes (Dasgupta & McGregor, 1992). In our representation, these modifier genes control the expression of hidden units and robot sensors. For instance, in Figure 3.3 only the modifier genes corresponding to sensors 0, 4, and 6 are ON. The rest of the genes are OFF, as shown by the black squares in the corresponding gene positions. Thus, the robot using this neurocontroller has access to only three sensors. Similarly, although 10 hidden units are allowed, the representation suggests that only seven of them are expressed. Further, only five of these seven hidden units are actively used, with the other two (shown in grey), not being connected to either the inputs or the outputs (and hence being useless). These modifier genes allow evolution to easily manipulate the size of the evolved neurocontroller or the number of sensors at the robot's disposal, thereby entertaining the possibility of discovering *minimal* (or optimal) designs.

Figure 3.4 shows the neurocontroller that corresponds to the genetic representation of Figure 3.3. This network has three sensors and seven hidden units. However, the network effectively makes use of only five hidden units and two sensors, with the other units and sensors not being connected to inputs, outputs, or other hidden units. It can also be noticed that the network contains a number of recurrent links.

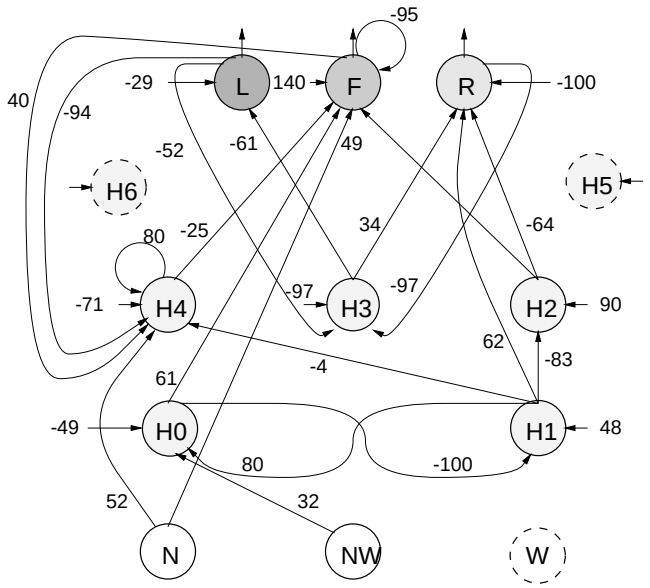

Figure 3.4 Neurocontroller for which the genetic representation is shown in Figure 3.3.

3.2.2.1 Properties of this Genetic Representation

The genetic representation we have chosen to use in our experiments has a number of properties described in Section 2.5.1. It easily supports the evolution of both feed-forward and recurrent neurocontrollers and hence can be used equally well in temporal and non-temporal domains. However, each weight in our network is restricted to be an integer in [-100, 100]. It would appear that networks with *arbitrary* weights between units cannot evolve in our system. However, as our representation allows *multiple connections* between units, arbitrarily large weights between units can be easily realized via appropriate combinations of multiple connections. This endows our representation with the important property of *completeness.*

The same genotype in our representation can be decoded into a feed-forward or a recurrent

network by simply changing the decoding mechanism to disable or enable the expression of recurrent connections. Similarly, the expression of hidden units, recurrent links, etc., can also be controlled during decoding by altering the expression of modifier genes. Thus, multiple phenotypes *can* result from the same genotype, which leads to the property of *phenotypic multiplicity*. Also, since the order of the connections in a given gene (input connectivity of a hidden or output unit) does not matter, the exact same neurocontroller can be encoded using multiple genotypes. Thus, our representation exhibits *genotypic multiplicity*, with *different* genotypes decoding to the same phenotype.

We have also mentioned that in our genetic representation a given unit (target) can have multiple connections from another unit (source). This allows multiple links between two units. This feature corresponds to *genotypic redundancy* as well as *phenotypic redundancy*. Genotypic redundancy allows the system to maintain backup copies of good genes, which is of considerable value if the genetic operators are disruptive and/or the decoding process is error prone (e.g., it does not read entire gene sequences, etc.). Phenotypic redundancy is a useful feature in scenarios where phenotypic components fail. For instance, if our robots operate in hazardous environments that cause neurocontroller links and units to fail, phenotypic redundancy can compensate for such effects through robust and fault-tolerant designs that make use of multiple identical units and links.

Since our representation supports genotypic multiplicity (multiple genotypes decode to the same phenotype), we can easily bias our evolutionary system to preferentially choose *compact representations*. As described in Section 2.5.1, notions of *topological* as well as *functional compactness* can be incorporated in our system by appropriately defining *cost-functions* to characterize compactness. One such approach will be presented in Chapter 5.

As can be seen, the genetic representation used in our experiments possesses a number of interesting properties that make it suitable for the evolution of neurocontrollers for the box-pushing robot task.

3.2.3 Miscellaneous Details

Our simulations used populations of size 500 and the evolutionary runs lasted 100 generations. We used *binary tournament selection* to choose parents for mating in each step (Goldberg, 1989). Our experiments made use of two genetic operators: crossover and mu-

tation. We used *uniform crossover* with the probability of crossover set at 0.5. In uniform crossover each gene in the offspring has a uniform chance of coming from either of its parents (Syswerda, 1989). This requires a random coin-toss at each of the gene positions to determine the parent the offspring inherits that particular gene from. Some commonly used crossover strategies are contrasted in Figure 3.5.

Figure 3.5 Comparison of commonly used crossover strategies.

In our implementation, crossover respected gene boundaries, i.e., offsprings inherited entire genes *intact* from their parents (subject to mutation). Since genes in our representation encoded input connectivities of neurocontroller units or positions of sensors, crossover had the effect of producing offspring with a mix of units and sensors available in the parents. Mutation, on the other hand, operated within genes. Each neurocontroller unit and robot sensor was mutated with probability 0.1. Further, once a unit (or sensor) was chosen for mutation, either the modifier gene bit was flipped with probability 0.1 or the gene (input connectivity or sensor position) was mutated. For neurocontroller units, mutation involved a random modification of either the LayerID, UnitNo, or the Weight of one randomly chosen connection. For sensors, this gene mutation resulted in a random change of the sensor position.

As with the experiments of Teller, each individual in the population (a neurocontroller for a robot) was evaluated in 40 random box-pushing environments and its average performance was used as a measure of fitness for selection. In each experiment, we performed 50 evolutionary runs, each starting with a different random seed. This was done to ensure statistical significance of the results obtained. Further, in order to directly compare the fitnesses of the

different neurocontrollers, we computed *standardized* fitnesses for each of them. This was done by evaluating the performance of the neurocontroller over a fixed set of 1000 box-pushing environments. Unless otherwise mentioned, the fitnesses reported in the following experiments refer to the standardized fitnesses.

3.3 Evolution of Neurocontrollers for the Box-Pushing Robot

In this section we present results of experiments in the evolutionary synthesis of neural network controllers for the box-pushing robot task described earlier.

3.3.1 Evolution of Behaviors and Improvement in Fitness

Figure 3.6 shows a plot of the course of one evolutionary run. In this case the system was evolving recurrent networks with no hidden units and three output units. The LFR output coding strategy was employed.

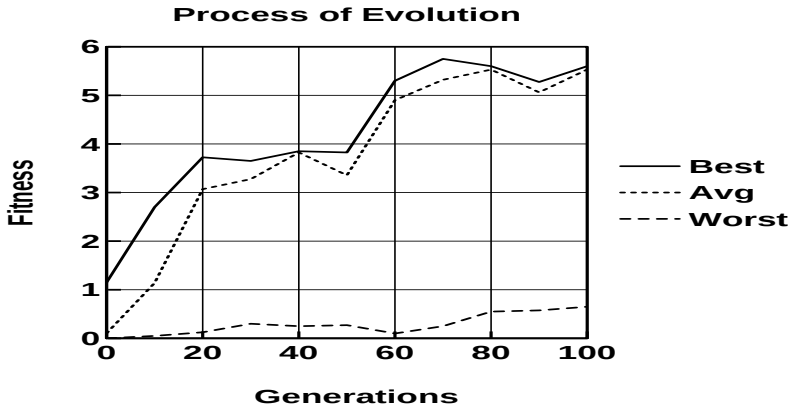

Figure 3.6 From initial populations consisting of highly unfit robot controllers, evolution produces better, fitter controllers.

As can be observed, the networks in the initial population (generation 0) are highly unfit. This is to be expected given that initial populations are randomly created. The robots endowed with these networks exhibit poor navigation skills and either keep spinning in the same position or attempt to move into walls. As a result, the best network in the initial population had a fitness of 1.2 and the population, an average fitness of 0.15.

From this modest beginning, the fitnesses can be seen to improve over generations, with the best network in generation 100 having a fitness of 5.6 and the population an average fitness of 5.5. Notice that the fitness of the worst member of the population remains consistently low (< 1) over the generations. This is due to the disruptive effects of the genetic operators which modify parts of the network crucial to the survival of the robot.

The networks of generation 100 demonstrate effective navigation behaviors, particularly when compared to the primitive capabilities of their ancestors. The robots in generation 100 address the box-pushing task well; they move about purposefully, avoid walls and push boxes. Thus, evolution discovers neurocontroller designs that give rise to effective box-pushing behaviors.

3.3.2 Evolution of Feed-forward and Recurrent Networks

In our experiments, we evolved both feed-forward and recurrent neurocontrollers. As explained earlier, we conducted 50 complete evolutionary runs to present results of statistical significance. Figure 3.7 shows a comparison of the fitnesses of feed-forward and recurrent networks that were evolved without any hidden units. In this figure, AFF refers to the fitness of the best feed-forward network produced in each of the 50 evolutionary runs, averaged over the runs, while BFF denotes the fitness of the best feed-forward network discovered by evolution. ARR and BRR have denote similar fitness measures, but for recurrent networks. These fitness values are *standardized*, i.e., they denote the fitness of the robot over a fixed set of 1000 box-pushing environments and hence are directly comparable.

From Figure 3.7 one can observe that the fitness of the best feed-forward networks, averaged over the 50 evolutionary runs and denoted by AFF, was approximately 1.65, while the best feed-forward network discovered by evolution (denoted by BFF) had a fitness of 2.0. On the other hand, the average of the fitnesses of the best recurrent networks evolved (ARR) was approximately 4.0, with the best recurrent network (BRR) having a fitness of 5.46. Some of these recurrent neurocontrollers thus have considerably higher fitnesses than the control programs evolved by (Teller, 1994).

One can also observe that the recurrent networks are over twice as effective as feed-forward ones. The reason for this difference lies in the ability of recurrent networks to remember their *past actions*, and their ability to exploit this *memory* in determining current actions

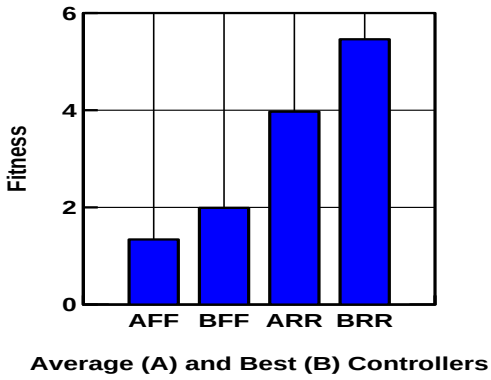

Figure 3.7 Comparative performances of feed-forward and recurrent neurocontrollers. Recurrent networks contain limited-depth memory, and hence do well on the task; feed-forward networks perform rather poorly.

to perform. In contrast, feed-forward networks are constrained to make action choices based solely on current sensory inputs. But how does the ability of to remember past actions provide recurrent networks with an edge over their feed-forward counterparts? We offer the following explanation.

The networks used in our experiments are *deterministic*, i.e., they always produce the same action in response to a given input. In such cases, if the sensory inputs available at a given place trigger a robot action but the robot action fails, then the robot remains at the same location indefinitely. This is because the sensory inputs remain the same at subsequent time steps (since the robot does not move) and the deterministic controller produces the same action which keeps failing. This leads to a *permanently* stuck robot. This might happen, for instance, if the robot wanted to push a box and the box happened to be against a wall or another box. This would lead to failed actions and a permanently stuck robot. All the feed-forward networks in this experiment suffered from this affliction.

3.3.3 Evolved Networks and Behaviors

We have shown earlier that the fitness of the neurocontrollers is rather low in the initial populations but then improves considerably as evolution progresses. Further, we also showed that recurrent networks have much higher fitnesses than their feed-forward counterparts. Why

are the robots in the initial populations less fit and why is it that recurrent networks outperform feed-forward ones? What behavior must the robot possess to do well in this box-pushing environment?

First, let us consider why the networks might have low fitnesses in this environment. One probable cause, outlined earlier, is the possibility of the robot getting into situations in which it remains *stuck* till the end of simulation. For example, the robot might keep spinning in the same location, try to (unsuccessfully) move into a wall, attempt to push two or more boxes at the same time, etc. If the robot is unable to detect and escape from such potential traps, it will indeed have little time to do useful box-pushing. This results in low fitnesses.

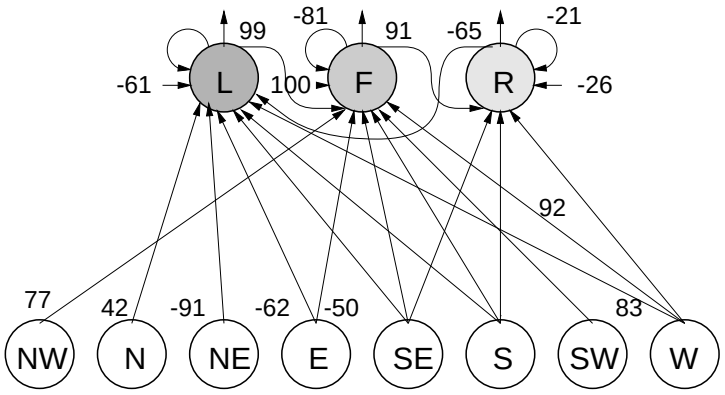

Figure 3.8 The best recurrent (RR) neurocontroller (without any hidden units) that was discovered by evolution. A strong negative self-loop at the forward move (F) unit makes the robot *alternate* move and turn actions.

Since one of the constraints of the box-pushing task of Teller is the inability of the robot to detect failed moves, the above scenarios frequently arise in our simulations. What, then, would be the most effective strategy under such circumstances? Obviously, the key is to avoid getting stuck. Since repeated actions of the same kind (e.g., turning in place or continually moving forward and eventually into the wall), lead the robot to stuck states, they must somehow be avoided. One way of doing this would be for the robot to remember its previous action and choose to perform an action that is different from it. For instance, if the robot moved forward in the previous time step, it would be inclined to turn now.

A careful analysis of the structure of the recurrent networks produced in our evolutionary

experiments indicate that over generations these networks develop structures that permit them to do just that. These networks evolve a *strong negative self-loop* at the output unit that codes for forward moves (unit F), as shown in Figure 3.8 (unit F has a recurrent link with weight -81).

Such structures force the robot to *interleave* or *alternate* forward moves with turns. Thus, if the current action of the network is to move forward the output of the F unit is a 1. The strong negative self-loop at the F unit automatically *biases* the robot towards a turn action at the next time step. Of course, strong sensory inputs can override this automatic bias, still, it provides an effective safeguard mechanism against getting into states in which it might be stuck forever. Feed-forward networks cannot exploit this feature since self-loops of this sort qualify as recurrent links. Consequently, these networks perform poorly in comparison.

This observation is further reiterated by Figure 3.9, which shows a typical plot of the actions performed by the robot in the course of one simulation run.

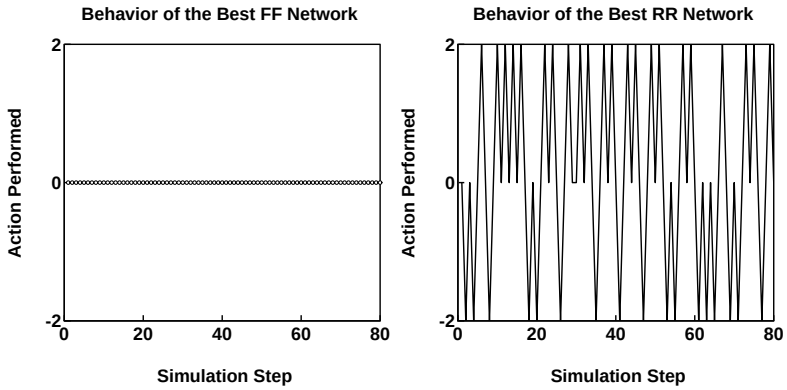

Figure 3.9 Actions performed by the robots using the best feed-forward (FF) and recurrent (RR) neurocontrollers discovered by evolution. Here, 0, 2, and -2 represent forward moves, right turns, and left turns respectively. The robot behavior on the right was produced by the recurrent neurocontroller shown in Figure 3.8.

The ordinate axis in these figures codes for the action performed by the robot, with 0 representing forward moves and 2 and -2 denoting right and left turns respectively. The figure on the left shows the behavior of the robot with the best feed-forward network evolved. The relatively poor fitness of the robot is a direct consequence of its inability to change actions, in this particular case, its inability to determine when the box it was pushing actually came

against a wall. As a result, the robot continues (at least tries to) move ahead with the box little realizing that it is in fact against the wall. It remains stuck in this state till the end of simulation, obtaining a fitness of merely 1.

The figure on the right shows the behavior of the robot with the best recurrent network found (shown in Figure 3.8). This network frequently switches actions, interleaving motions of moving forward and turning. It does not choose one box and push it all the way to the side, rather, it pushes many different boxes by steps. This behavior reduces its chances of getting stuck indefinitely, thereby contributing to its higher fitness. This neurocontroller obtains a fitness of 6 in the same environment.

A few other observations can be made about the neurocontroller shown in Figure 3.8. Notice that the bias (or thresholds) of the L and R units are negative values while that of the F unit is a large positive value. Thus, in the absence of any sensory input this robot is biased to move forward. We have already pointed out the strong negative self-loop at the F unit that biases the robot to interleave moves and turns. In addition to this mechanism, this neurocontroller also possess other safeguard mechanisms. Consider the large weight (99) between units L and F. This attempts to equalize the numbers of moves and turns. For instance, if the robot did not turn left, the output of the L unit is a -1. This lowers the chances of the robot choosing to move forward at the next time step. A similar role is played by the link between F and R units, which has a weight of 91.

Let us now consider the effect of the right sensor (E). If it detects a box (sensor value of 1), it provides negative input values to L and F units, thereby biasing the robot to chose R as the winner (and consequently perform a right turn). The left sensor (W) on the other hand, biases the robot towards a left turn by providing the L unit with a large positive value. By analyzing the evolved neurocontrollers we have been able to identify the roles played by the different network components, and have characterized their influence on the behavior of the robot.

It must be pointed out that though Teller was able to evolve behavior programs and calculate the fitnesses of the evolved agents, he was unable to analyze the programs to decipher the behaviors that were evolved (Teller, 1994). His main conclusion was that *memory* is important for this task. In contrast, our experiments have allowed us to not only concur with his conclusion regarding the need for memory, but also show the exact role played by memory

in high fitness box-pushing behaviors.

3.3.4 Importance of Recurrent Links

We just argued that recurrent networks lead to higher fitnesses in the box-pushing task owing to their ability to remember past actions and choose current actions accordingly. In particular, our analysis showed that recurrent links allow the robot to interleave actions of moving forward and turning, thereby leading to higher fitnesses in the box-pushing task.

In this section we use a crude but simple experiment to demonstrate the importance of the recurrent links and their effect on the fitnesses of the robots. We take the best recurrent network produced in each of the 50 evolutionary runs, and determine its standardized fitness with and without the recurrent links. Figure 3.10 shows the result of this experiment.

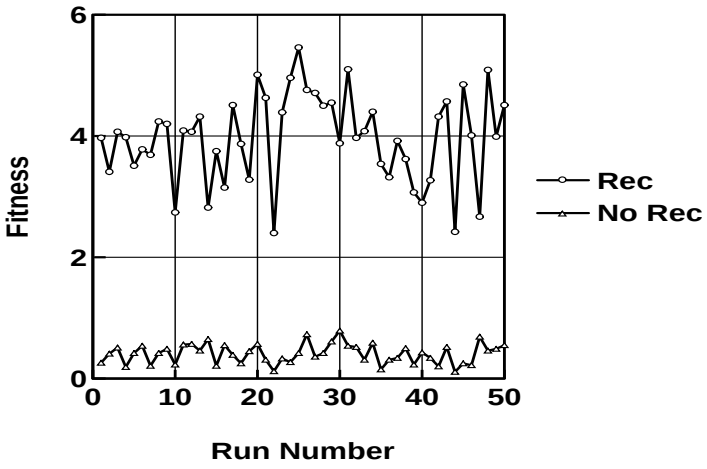

Figure 3.10 The importance of recurrent links can be demonstrated by comparing the fitnesses of the best robots with and without their recurrent links. Removal of the recurrent links can be seen to drastically reduce robot fitness.

It can be observed the fitness of the neurocontroller drops significantly when the recurrent links are removed. For instance, run 25 produced a network with fitness 5.46 (shown in Figure 3.8). Without its recurrent links the fitness of this network drops to 0.42. This shows the reliance of the neurocontroller on the recurrent links and the role of the links in the fitness of the box-pushing robot.

3.3.5 Effect of Hidden Units

We showed earlier that the recurrent networks manage higher fitnesses than their feed-forward counterparts by evolving an intuitively appealing strategy that involves the use of recurrent links to remember past actions. However, even with recurrence, the robot's memory is only *one* time-step long, i.e., the robot remembers the action performed in the previous time step and nothing more. It is natural to wonder if a robot with a longer memory will perform better, since in that case its decision can be expected to benefit from its history of past input activations and output actions. This motivates us to explore networks with hidden units, since in our model of neural computation (Section 2.2.3), a network with two hidden layers can remember network activations two time steps earlier.

In order to study the effect of hidden units in the evolution of box-pushing behaviors, we allowed evolution to choose up to 10 hidden units. As explained in Section 3.2.2, the genetic representation used in our simulations allowed arbitrary connections to be formed between units. However, the *fan-in* or the input connectivity of any unit was restricted to a maximum of 10, as explained earlier. In feed-forward networks, connections between input-output, input-hidden, and hidden-output units were permitted. However, the connectivity between hidden layer units were constrained in such a manner as to allow lower numbered units to connect to higher numbered ones but not vice versa. Recurrent networks, on the other hand, could have connections between arbitrary pairs of units. In both kinds of networks, connections *into* input units were not allowed.

As can be observed from Figure 3.11, hidden units improve the performance of both kinds of networks. While the improvement is only marginal for recurrent networks, feed-forward networks benefit tremendously from the addition of hidden units. In fact, the average and best fitnesses of feed-forward networks almost double (a 100% improvement). Note that the best feed-forward network (BFF) evolved with hidden units has a fitness of almost 4.0, which is nearly equal to the average fitness of the best recurrent networks (ARR) without hidden units. It is clear that the hidden units somehow help the feed-forward neurocontrollers avoid or escape from situations where the robots used to get stuck earlier.

Just as we did with recurrent links earlier, we can perform a simple experiment to demonstrate the importance of hidden units. We simply take the best neurocontroller produced in each of the evolutionary runs and compute its fitness with and without its hidden units. Fig-

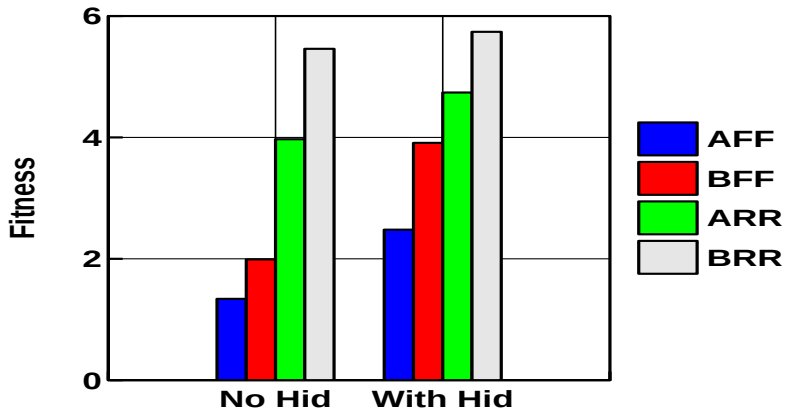

Figure 3.11 Robot fitnesses when hidden units are allowed in the neurocontrollers. Hidden units can be seen to significantly enhance the fitnesses of feed-forward networks.

ure 3.12 shows the result of this experiment for the feed-forward neurocontrollers. It can be observed that in every case the fitness drops alarmingly when the hidden units are disabled. Feed-forward networks thus appear to rely heavily on the hidden units.

Figure 3.13 demonstrates the reliance of recurrent networks on their hidden units. Although fitnesses often decrease by significant amounts when hidden units are disabled, there are evolutionary runs wherein the robot fitnesses are not affected by much. For instance, the neurocontrollers produced in runs 24 and 41 have little change in fitness when hidden units are removed. Based on these results and earlier ones, we can conclude that recurrent networks perform well on the box-pushing task by making use of recurrence or hidden units, depending on what is available. Thus, when there are no hidden units available, evolution relies on recurrence. However, when hidden units are available, the tendency of evolution is to exploit them to attain higher fitnesses.

Figure 3.14 shows the behavior of the best feed-forward network evolved with hidden units, on *one* box-pushing environment. Unlike feed-forward networks evolved earlier (Figure 3.9), this neurocontroller can be observed to switches effectively between forward moves and turns. In particular, one can easily notice that this robot either moves forward or turns right. It should also be noted that unlike the recurrent networks shown earlier, this robot does not

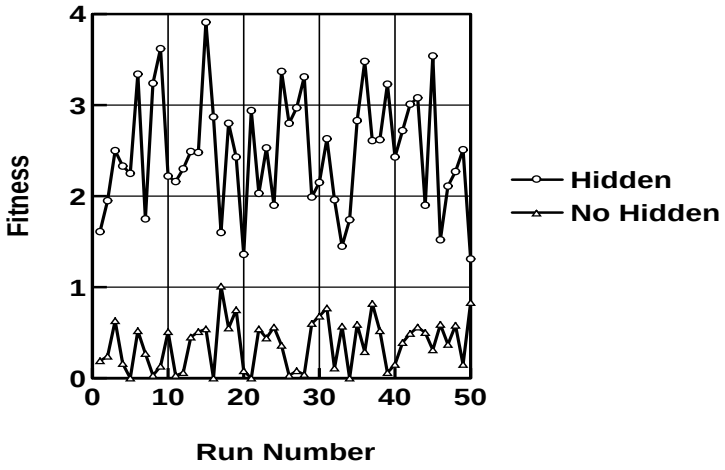

Figure 3.12 Performance of the best feed-forward neurocontrollers from each of the evolutionary runs, with and without hidden units.

necessarily *alternate* actions. For instance, between simulation steps 6 and 9, the robot moves forward on *four* consecutive time steps before turning right. It displays this behavior many times during the course of its box-pushing exercise, as can be confirmed from Figure 3.14. We have argued earlier that such behaviors allow the robot to escape from states in which it might otherwise be stuck indefinitely. This contributes to the increase in fitness of feed-forward neurocontrollers.

Figure 3.15 shows the best feed-forward network (with hidden units), that was discovered by evolution. It may be noted that though the evolved neurocontroller has 7 hidden units, only three of them are in *active* use. From the thresholds of the three output units and the three hidden units, one can infer that in the absence of other sensory inputs the robot is biased towards forward moves. This is because the activation at the output unit corresponding to forward moves, F, is given by 127 (62+14+51), which is significantly greater than the activations at the other two output units. It can also be noticed that a box detected by the N sensor biases the robot towards a forward move by suppressing the right turn (through a weight of -62), while a box detected by the SE sensor biases the robot towards a right turn by increasing the activation of the R unit (through a weight of 91).

The hidden units H0, H2, and H4 play a critical role in preventing the robot from pushing a box too long. This is achieved as follows. Suppose the units in the network are at their

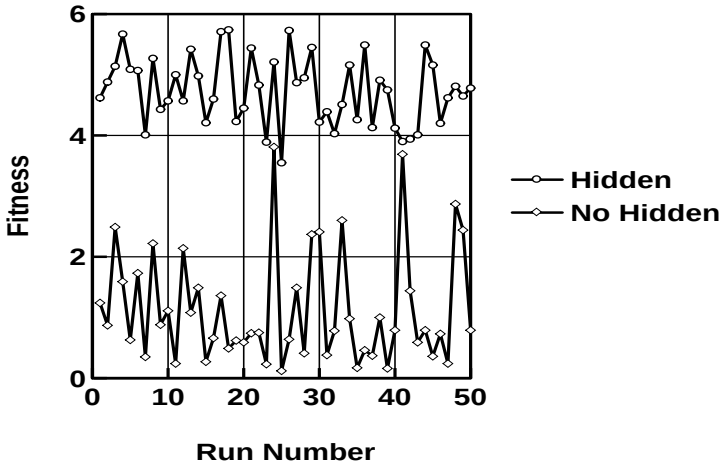

Figure 3.13 Performance of the best recurrent neurocontrollers from each of the evolutionary runs, with and without hidden units.

default activations dictated by their thresholds. Hence, H0=+1, H2=-1, H4=-1, L=-25, F=127 (62+14+51), and R=90 (34+56). Since the output unit F has the highest activation, the robot moves forward. Suppose the robot now finds a box immediately ahead and there are no other boxes or walls within its sensory range. This box is detected by the sensor N, which provides an input of +1 at the next step. It can be easily verified that this causes the following activations in the network units: H0=+1, H2=+1, H4=+1, L=-55, F=127, and R=28. Hence the robot moves forward again, possibly pushing the box (if it is not against another box or wall).

At the next time step, assuming the robot does not sense any additional boxes or walls, the changed activations of H2 and H4 cause the output activations to become: L=-55, F=-3 (62-14-51), and R=28 (34-62+56). Now the unit R is declared the winner and the robot turns right.

Thus, hidden units enable the feed-forward neurocontrollers to remember previous sensory *inputs* (although they still cannot remember past actions owing to the unavailability of recurrent links). With such hidden units the networks can choose wise actions by taking past observations into account. For instance, in the above example, the hidden units H2 and H4 record the fact that the robot has sensed a box ahead (N=1), by changing their activations from -1 to +1. Once the robot turns away from the box, the activations of these units return to -1.

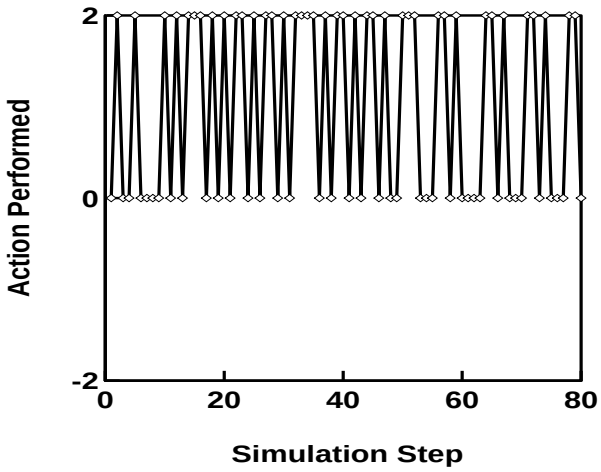

Figure 3.14 Behavior of the best feed-forward network with hidden units, that was discovered by evolution. Here 0, 2, and -2 represent forward moves, right turns, and left turns respectively.

Figure 3.16 shows the numbers of hidden units used by the evolved networks. As can be seen, on an average both feed-forward and recurrent networks (AFF and ARR) make use of approximately six hidden units. While the best feed-forward network uses seven hidden units, the best recurrent network uses six units, as shown in Figure 3.15. It must be stressed that though the neurocontrollers were allowed to use up to ten hidden units, evolution preferred to use fewer than ten in every instance. This is interesting considering the fact that our evolutionary system did not impose any explicit pressure towards the use of lesser numbers of units. If, in fact, some penalty or cost had been attached to the use of hidden units, would evolution have discovered designs with even fewer units? We explore such issues in Chapter 5.

3.3.6 Comparison of Different Output Coding Strategies

Our results thus far have shown that evolution can be used to design neurocontrollers that result in high fitnesses box-pushing behaviors. We observed that robots with recurrent networks attain high fitnesses by interleaving actions of moving forward and turning. We also showed that such behaviors arise through the use of hidden units that remember past inputs. However, are such behaviors merely artifacts of the LFR output coding mechanism chosen or are they truly characteristic of the constraints of the robot and its environment? In this section

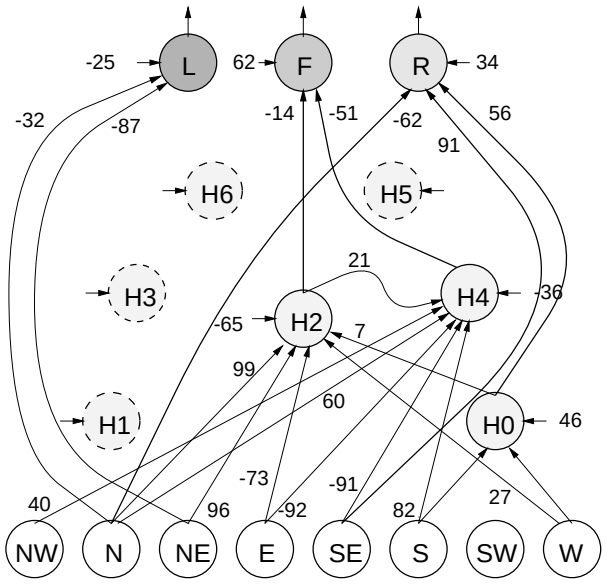

Figure 3.15 The best feed-forward network with hidden units, discovered by evolution. Although this network has seven hidden units, it effectively uses only three.

we attempt to answer this question.

One way to absolve the output coding strategy from the primary role in the evolved behaviors is to evolve robots with *different* output coding mechanisms, analyze the resulting behaviors, and show that the behaviors are qualitatively similar. In order to do this, we repeated the evolutionary experiments described above, with both Braitenberg's output interpretation and the Action-Direction coding scheme described in Section 3.2.1.

As shown in Figure 3.17, there is little difference between the robot fitnesses using the three schemes. A careful analysis of the structures evolved using Braitenberg and Action-Direction output coding strategies confirms our earlier observation that high fitnesses on the box-pushing task are realized by interleaving or alternating forward moves and turns. The networks using the Braitenberg strategy evolve *large recurrent links* at the two output units. While one link is *positive*, the other is *negative*. Thus, the output of one unit is biased towards remaining constant, while the output of the other unit is biased to change at every time step. In the Braitenberg strategy this leads to alternating moves and turns.

The neurocontrollers using the Action-Direction strategy consistently evolve a large *negative*

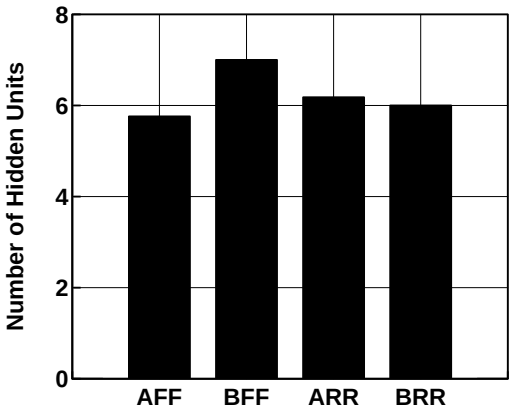

Figure 3.16 Number of hidden units used by the evolved networks.

recurrent link at the output unit coding for action. This mechanism makes the robot alternate between moves and turns at each time step. A number of results with the Action-Direction output coding strategy are presented in (Balakrishnan & Honavar, 1996a; Balakrishnan & Honavar, 1996b; Balakrishnan & Honavar, 1996c). It thus appears that the behavior of interleaving moves and turns is characteristic of the task environment and is consistently discovered by evolution irrespective of the output coding mechanism used.

These results lead us to believe that evolution automatically comes up with ways to effectively counter the constraints and limitations of the task environment. Given the box-pushing task and its associated constraints, evolution uses the neurocontroller structures at its disposal to sculpt behaviors that involve alternating moves and turns.

3.3.7 Baseline Experiments: Random Walk and Random Search

When we presented the box-pushing task, we argued that it was a challenging environment that made the design of appropriate behaviors rather hard. We then used evolutionary algorithms to design neurocontrollers for box-pushing robots and presented many instances of effective behaviors. However, one might be tempted to ask for some quantitative measure of the *difficulty* of achieving good fitnesses on this task. Although we cannot provide such a measure, we can show that the box-pushing task cannot be effectively addressed by simple

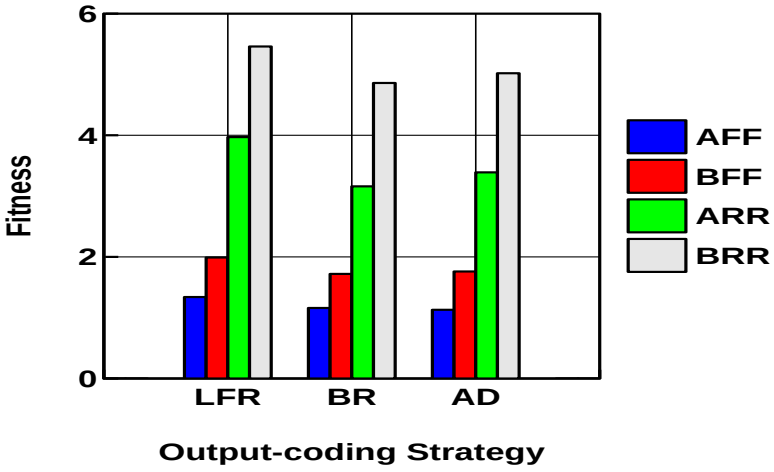

Figure 3.17 Performance of the robots with different output coding strategies.

behaviors like *random walk*. We can easily demonstrate this by conducting simulations where the robots randomly choose an action to perform at each step.

We have also alluded to the fact that evolution is capable of effectively searching vast, multimodal, and complex search spaces, using little domain-specific knowledge. But can we show that the space of neurocontrollers for this box-pushing task is indeed vast and complex? Can we show that other algorithms for searching this neurocontroller space will indeed falter? Although we cannot really *prove* that other search algorithms will fail to effectively search the neurocontroller space, we can show that *simple* search algorithms like exhaustive or random search, will be inefficient.

First, let us consider the size of the search space. Assuming that the neurocontrollers do not have any hidden units, the search procedure must determine appropriate connectivities for the three output units (assuming an LFR output coding scheme). Since there are eight input units, three output units, and a threshold for each unit, the search algorithm must determine (8+3+1 = 12) parameters for each of the output units. Now, since the weights are restricted to be integers in the range [-100, +100], there are 201 possible values for each weight. Thus, the *total size* of the neurocontroller space, i.e., the space of possible values for the weights of the three output units, is $(201)^{36}$, which is a truly large quantity. Remember that this figure is for a neural network without any hidden units. Adding hidden units leads to an exponential increase

in the size of the search space. This vast space plays havoc with simple search algorithms. While an exhaustive search of this space is *infeasible* and *impractical*, random search is confronted with a *needle-in-a-haystack* situation. Thus, simple search algorithms like exhaustive and random search are unsuitable for searching the space of neurocontrollers for the box-pushing task.

Figure 3.18 compares the fitnesses of neurocontrollers produced by evolution with two *baseline* experiments. The first involves the average fitness of a random walking robot (RW) and the second is a random search of the space of neurocontrollers. As can be seen, random walk results in extremely poor performance in this environment, with the average random walk (over 10,000 such environments), producing a fitness of a mere 1.0 point and the best random walk producing a fitness of approximately 1.6.

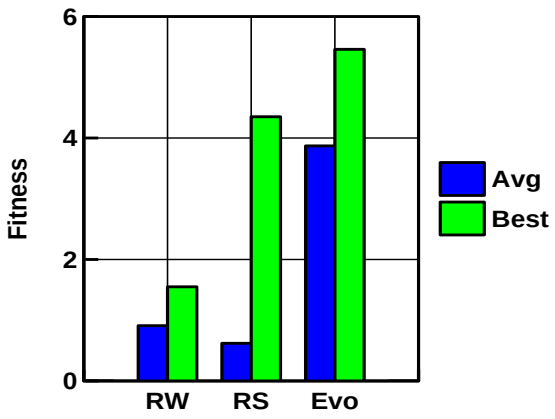

Figure 3.18 Baseline experiments to compare random walk, random search, and evolutionary search.

Figure 3.18 also shows the best and average fitnesses of 10,000 randomly created neurocontrollers. As can be observed, this random search yields a best fitness of 4.2 and an average fitness less than 1.0. In contrast, the evolutionary approach produces individuals of fitness over 4.5 within 20 generations (i.e., total evaluation of $500 \times 20 = 10{,}000$ neurocontrollers). Thus, with the same effort (measured in terms of the number of neurocontrollers evaluated), evolutionary search finds better neurocontrollers than random search.

Further, when continued to 100 generations, the evolutionary approach produces neuro-

controllers with a best fitness of 5.46 with *all* the neurocontrollers produced (and evaluated) yielding an average fitness of approximately 3.9. Thus, searching the neurocontroller space using population-based evolutionary algorithms has a higher likelihood of discovering highly fit designs.

One of the factors that make the box-pushing task hard is that the boxes in the room are randomly placed for each trial. Also, the robot does not start from a single fixed position in the room, rather, it starts from randomly chosen places. This makes the task particularly hard because the robot has to develop behaviors that work well across all the random environments. What would happen if the robot always found the boxes in the same locations every time? In addition, if the robot were to be introduced into the room at the same place in every trial, would evolution produce robot behaviors of higher fitnesses? It should be noted that with the box positions fixed and the robot starting at the same location every time, the box-pushing task acquires the same flavor as that of *trail-following* or *maze-running* tasks.

Figure 3.19 shows the results of such a modified experiment. TR1 refers to the *trail-like* box-pushing task (with box and robot start positions fixed across trials). In this case the robots used feed-forward neurocontrollers without any hidden units. As can be seen, the best feed-forward neurocontroller attains a fitness of 4.0 on this task, which is twice the fitness of the best feed-forward neurocontroller evolved for the regular box-pushing task (Section 3.3.2). When hidden units are permitted, evolution discovers feed-forward networks with an average fitness of 8.2 and a peak fitness of 9.0, as shown by TR2 in Figure 3.19. Contrast this with the average and peak fitnesses of 2.6 and 3.9 achieved by feed-forward neurocontrollers on the regular box-pushing task (Figure 3.15). TR3 shows the fitnesses of recurrent networks *without* any hidden units. While the average fitness of these networks is 8.2, the best recurrent network produced by evolution has a fitness of 10, which is the maximum fitness attainable in this box-pushing environment. Thus, evolution discovers neurocontrollers with optimal behavior in the modified, trail-like environment.

Contrast these results with the best fitnesses observed in the regular box-pushing environment (where the box positions and the robot start locations are randomly chosen). The label RD in Figure 3.19 shows the average and peak fitnesses of the recurrent networks discovered by evolution. It can be observed that the peak fitness of 5.46 is much lesser than the maximum attainable fitness of 10.

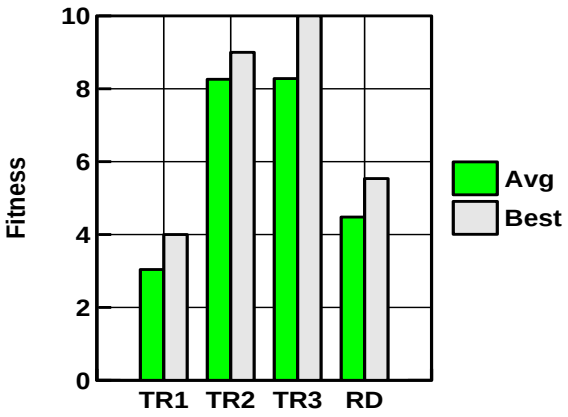

Figure 3.19 Comparison of trail-like behaviors (TR1, TR2, and TR3), with regular box-pushing behaviors (RD).

Thus, by relaxing some constraints, the regular box-pushing task can be transformed into a trail-following one. Given the fact that evolution appears perfectly capable of discovering designs (and behaviors) of maximum fitness in trail-following environments, its failure to do so in the regular box-pushing environment may be construed as an indication of the inherent difficulty of the box-pushing task.

These results show that it is rather difficult to determine good box-pushing behaviors in general. They also demonstrate the ability of artificial evolution to efficiently and effectively search the space of neurocontroller designs.

We now present some work in evolutionary robotics that is closely related to our work described earlier.

3.4 Related Approaches for the Evolution of Neurocontrollers

Floreano and Mondada (1994) evolved a neural network controller for the Khepera robot (Mondada *et al.*, 1993) to demonstrate navigation and obstacle avoidance behaviors. They used a neurocontroller with 8 inputs and two outputs. The output units computed a sigmoid activation function and could have recurrent links between themselves. The input units were directly connected to the eight infrared sensors on the Khepera and the output units directly

controlled the two stepper-motors associated with the robot wheels. They used a roughly circular corridor of external size 80×50 cm. Using a fitness function that rewarded speed, direction of robot motion, and distance from obstacles, they evolved neural networks with good navigation and obstacle avoidance behaviors. Importantly, all their evaluations were performed on the robot, i.e., each member of each population was evaluated on the Khepera. Although this approach led to the discovery of *real* behaviors, the time penalty associated with it was immense. On an average, the evaluation of each generation took approximately 39 minutes. Thus, 100 evolutionary generations required about 65 hours.

Miglino et al. (1994) evolved neurocontrollers for a *wandering* robot. In their work, the robot was introduced into a square arena containing 26 × 26 cells. 20 × 20 cells in the center of the arena were white in color, while the three cells at the periphery of the arena were black. The goal of the robot was to visit as many white cells as possible, while avoiding the black ones. Thus, the robot effectively performed a variation of navigation and wall-avoidance tasks. The robot was capable of detecting the color of the cell in front of, and immediately behind, its current position. White cells provided an input value of 0 to the corresponding input unit of the neurocontroller, while black cells provided a value of 1. The neurocontrollers contained two input units, two hidden units, one memory unit, and two output units. The units computed a threshold activation function. The binary output produced by the network was interpreted into four robot actions: forward/backward move, or left/right turn through 45 degrees. The researchers evolved effective exploration behaviors, analyzed them, and implemented the evolved behaviors on a Lego robot. However, they found considerable differences between robot behaviors in simulation and reality. They also found that by evolving designs for simulated robots that had *noisy* actions, the simulated and actual behaviors matched rather well.

Lund and Parisi (1995) attempted to study the effect of environmental changes in the behavioral inclinations of evolving agent populations. In their simulations, agents lived in separate environments of size 40 × 40 cells that contained 15 randomly distributed food elements. These food elements were of three types: A, B, and C, with 5 numbers of each. Evolution produced agent behaviors to approach and consume these food elements. These behaviors were realized using a neural network with 5 input, 9 hidden, and 2 output units. While two input units provided the distance and angle of the *nearest* food element, the other three units

signaled the *type* (A, B, or C) of this food element. The two neurocontroller outputs were interpreted as a turn angle and forward move step size for the robot. Thus the robot could directly approach specific food elements. The researchers *co-evolved* the fitness function, i.e., the amount of energy that each food type provided to the robot was individual-specific and was determined by evolution. This led the robots to develop preferences for specific food types. The researchers also found that after the emergence of such preferences, the robots quickly adapted themselves to alternate food types if the preferred food type suddenly disappeared from the environment.

In other related work, Miglino et al. (1995) evolved neural networks for obstacle avoidance behaviors. They used simple feed-forward neurocontrollers with 8 input units and two output units. Their simulations used a model of the Khepera robot, which has 8 infrared sensors and two stepper-motor driven wheels. Instead of performing their evolutionary experiments directly on the Khepera, they built a *simulation environment* by placing the Khepera at various positions and orientations in the real environment and recording its sensory inputs. They also built a *motor model* by executing specific motion commands on the Khepera and observing its displacements. This simulation model was then used to evolve robot behaviors in simulation. In their experiments, three obstacles of diameter 5.5 cm were placed in a rectangular environment of size 60 $\times$ 35 cm. Neurocontrollers were evolved to successfully avoid these obstacles in simulation and were then tested on actual Khepera robots. Despite using a simulation model built from a real Khepera, the researchers found considerable differences between fitnesses in simulation and on real robots. As with (Miglino *et al.*, 1994), the researchers found that the addition of *conservative noise* in the simulations led to robust designs that performed well even when transferred onto real robots.

Walker (1995) studied a variation of a foraging task where the robots had to locate and approach *radiative* energy sources. These robots used extremely simple feed-forward networks with two input units and two output units. Each input unit provided the robot with a *measure* of the *energy field* (field strength of each energy source, normalized by its distance from the corresponding robot sensor). The two output units directly controlled the speed of the corresponding wheel of the robot. The researchers evolved behaviors for approaching energy sources, avoiding walls, etc. A number of related evolutionary robotics approaches are discussed in Section 5.10.

It may be noted that most of these approaches involve relatively simple robot tasks (exploration, approach, avoidance, etc.). As we have argued earlier, such behaviors are often easy to program manually. In contrast, the constraints and limitations of the box-pushing task make it hard to manually develop effective behaviors. Secondly, some of these approaches assume that the robot sensors have unlimited ranges (Lund & Parisi, 1995; Walker, 1995). This design choice is unrealistic for large environments. The box-pushing task, on the other hand, makes use of sensorily-constrained robots. Importantly, all of these approaches evolve neurocontrollers of fixed size. In contrast, our approach uses modifier-genes to evolve neurocontrollers of different sizes, providing evolution with the opportunity to automatically discover minimal designs. Finally, in most of these approaches, the *robot behaviors* have not been analyzed in detail. In contrast, we have evaluated the evolved behaviors and developed intuitively compelling explanations for high fitness box-pushing behaviors.

3.5 Discussion

In this chapter we have introduced a task that requires the robot to push boxes to walls within an allocated amount of time (Teller, 1994). There are a number of constraints associated with this box-pushing task, which make it very hard to design good control programs for the robots. In particular, the box-pushing robots are constrained by sensor ranges of only one cell in each direction around the robot. In addition, the robots can only move in directions parallel to the walls of the room. Thus, they cannot directly approach a box that is diagonally across from them. The robots are also constrained by limited pushing ability. They are incapable of pushing more than one box at a time. In fact, if such a scenario arises, the robot actions fail. What makes the task even harder is the fact that such failures cannot be detected by the robot. These constraints make the box-pushing task very challenging. Although Teller (1994) proposed this task and evolved programs to control robot behaviors, he was unable to characterize the behaviors of the evolved programs or analyze them to determine how the fitnesses were achieved.

We evolved simple neural networks to control robot box-pushing behaviors. In addition to evolving networks with higher fitnesses than the programs of Teller, we were also able to analyze the structures of the evolved neurocontrollers and identify mechanisms by which the robots attained high fitnesses on this task. In particular, we found that recurrent neurocontrollers led

to higher fitnesses compared to feed-forward ones, because they allowed the robot to remember its previous action and choose actions different from it. We argued that this mechanism leads to interleaved or alternating actions of moving forward and turning, which minimizes the chances of the robot getting into states where it might be stuck indefinitely. We suggested that feed-forward neurocontrollers attain much lower fitnesses since they cannot avail themselves of such information from the past.

However, when the neurocontrollers were allowed to make use of hidden units, we found that the fitnesses of feed-forward networks increased remarkably. By analyzing the behavior of such networks, we showed that the hidden units remember past *input activations* and use them to prevent the robot from persisting with one action. Although these robots do not necessarily alternate moves and turns, they evolve mechanisms that prevent them from constantly moving forward and getting stuck in the process.

We also performed a number of experiments to show that this behavior of alternating or interleaving moves and turns was characteristic of the environment. We showed this by choosing different output coding strategies for the neurocontroller. In each case, we analyzed the neurocontrollers discovered by evolution and showed that the primary behavior of the robot, that of interleaving moves and turns, remained the same.

Finally, we also showed a measure of the difficulty of the box-pushing task. We showed that a random-walk behavior on this task leads to extremely poor fitness. We showed that the neurocontroller space is large and cannot be efficiently searched through exhaustive means. We also showed that a random search of the neurocontroller space is ineffective and only finds neurocontrollers of mediocre fitness. In contrast, an evolutionary search strategy finds neurocontrollers of higher fitness with the same search effort (number of neurocontrollers evaluated). These results validate the choice of evolution in the design of neurocontrollers for the box-pushing task.

In another set of baseline experiments, we showed that the box-pushing task is hard to solve due to the unpredictability of the box positions. Indeed, in environments where the boxes are always placed at the same locations, we showed that evolution discovers trail-following behaviors that are optimal (attain the maximum fitness possible). It thus appears that trail-following or maze-running tasks commonly used in evolutionary robotics, are relatively easy to address, and the true challenge to the power of evolutionary search lies in tasks such as

box-pushing.

We also compared the box-pushing task to a number of others commonly used in evolutionary robotics. We argued that the box-pushing task subsumes a number of them as it involves aspects of navigation, exploration, approach, and avoidance. Further, the box-pushing task presents many instances of the exploration (find another box) versus exploitation (push box) dilemma, which makes it all the more interesting and challenging.

However, a number of constraints of the box-pushing task appear unrealistic. For instance, the box-pushing robots are incapable of detecting failed actions. Since most animals and contemporary robots have mechanisms to detect motion (or its failure thereof), they can easily determine whether they have moved or not. Then why not the box-pushing robots? Suppose the box-pushing robots had access to such a mechanism, what kinds of behaviors would evolve? Will the robots still interleave actions or will other behaviors emerge? In such a scenario, will feed-forward networks still be disadvantaged or will they perform as well as recurrent ones? Will hidden units help or hurt? A number of such interesting questions can be formulated and studied.

In addition, a careful analysis of the neurocontrollers evolved in our experiments suggests that most networks do not make use of all the sensors. Indeed, some sensors, particularly the ones located behind the robot, are rarely used. Since sensors on real robots consume energy, it is a good idea to only make use of sensors that are necessary. Discarding (or switching off) useless sensors can thus lead to minimal or optimal designs. But then, how do we identify the sensors that must be retained and the ones which must be discarded? One possibility is to use evolution to design the robot sensory system in conjunction with the neurocontroller.

Finally, the simulations in this chapter have assumed that there is absolutely no noise in the robot components and in the robot-environment interactions. For instance, sensors return accurate values and the robot motions, if successful, are always error-free. However, this is in contrast to real-world scenarios where a number of sensory errors manifest themselves (e.g., improper lighting, specular reflections, etc.). Robot motions in the real-world are also error prone due to a number of factors including friction, wheel-slippage, uneven tire inflation, etc. If such conditions exist in the real-world, designs evolved in simulation have little hope of doing well on real robots. Thus, these errors must somehow be taken into account in the simulations. Although it is hard to model these errors accurately, researchers have found that even the use

of random noise in simulations promotes robust designs that work well when transferred to real robots.

These issues and extensions of the box-pushing task form the subject matter of the next chapter.

4 EXTENSIONS OF THE BOX-PUSHING TASK: FEEDBACK, SENSOR DESIGN, AND NOISE

In the previous chapter we presented a box-pushing robot task that was proposed by Teller (1994). We showed the constraints and limitations associated with the task, and the challenges it offers to the design of good box-pushing behaviors. We also presented results from an evolutionary approach to the synthesis of neurocontrollers for this task. We showed that evolution requires little domain-specific knowledge, yet discovers high fitness designs that are well adapted to the constraints of the task environment. Though the constraints on this task make it rather challenging, we argued that some of them are not reasonable. In this chapter we relax or modify some of these constraints and present results from an evolutionary approach to the design of robot neurocontrollers and behaviors. We also analyze the precise ways in which the changes in constraints are exploited by evolution in sculpting robot designs and behaviors.

4.1 Detecting Self-Motion

In Teller's box-pushing task the robots were incapable of detecting failed actions that arose when the robot attempted to move into walls, push boxes into walls, or push more than one box. This constraint made the design of effective box-pushing behaviors rather hard since the robots often got into such futile pushing states. It is our contention that this limitation is unrealistic since both animals and contemporary robots have mechanisms to detect *self-motion*.

There is a wealth of literature in the biological and behavioral sciences that indicate that animals possess a variety of mechanisms to measure *linear* and *angular* accelerations. In fact, these mechanisms allow them to not only *detect* self-motion, but also compute their displacement from some point of reference, by appropriately *integrating* these acceleration signals twice over time. This process of maintaining an estimate of one's own position based on knowledge of distance, direction, and time of self-motion is referred to as *dead-reckoning*

or *path-integration* (Gallistel, 1990). As will be clarified in Section 6.5, there is considerable evidence that animals use such dead-reckoning mechanisms to learn and navigate between places (Gallistel, 1990).

In addition to animals, contemporary robotics also offers a variety of devices that enable robots to detect motion and perform dead-reckoning (Everett, 1995). Odometric techniques that make use of inexpensive optical and magnetic wheel encoders are popular mechanisms for obtaining *rough* estimates of robot position. Other more expensive (and accurate) approaches are described in Appendix A.

Thus, both animals as well robots are capable of detecting self-motion. In light of this fact, one reasonable modification of Teller's box-pushing task would be to provide the agents with a mechanism to detect self-motion. In the next section we discuss the addition of a simple self-motion detection mechanism and its consequences for effective box-pushing behaviors.

4.1.1 Adding a Simple Feedback Mechanism

Suppose the box-pushing task was modified such that the robots were capable of detecting self-motion. For instance, let us assume that the robots possess a mechanism that provides a *feedback* when actions *fail*. This feedback may be used by the robot in a variety of ways. To keep things simple, let us assume that the feedback signal is used by the robot to choose an action different from the one that led to the failure. Thus, if the robot attempted to move forward into a wall and the action failed, in the next time step it would avoid forward moves and turn instead. What would happen if such a feedback were to be available? How would the relaxation of this constraint affect the kinds of behaviors and neurocontrollers required for successful box-pushing?

Figure 4.1 shows the best and average fitness of *feed-forward* networks (without hidden units) evolved in the presence of the simple feedback mechanism described above. It can be observed that for neurocontrollers using the LFR output coding strategy, the average fitness of the best neurocontrollers (AFF) discovered by evolution increases to 5.27, while the best neurocontroller (BFF) fitness increases to 5.82. Notice that other output coding mechanisms like BR and AD also lead to similar increases when compared to feed-forward networks evolved without feedback (Figure 3.17).

These fitnesses are much larger than the best results obtained with recurrent networks

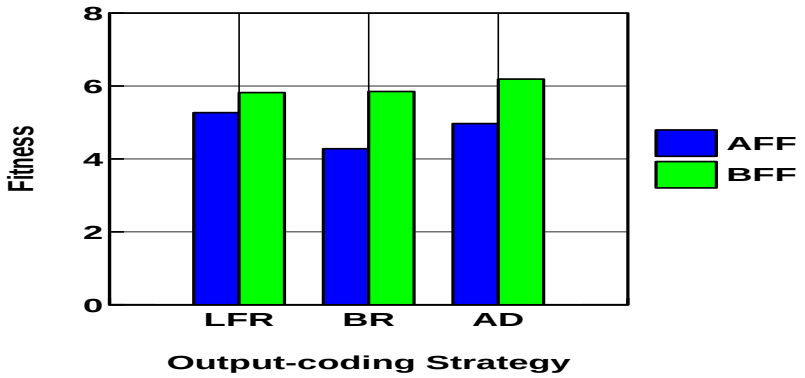

Figure 4.1 Performance of robots with feed-forward neurocontrollers with a simple feedback mechanism.

in Section 3.3.2. The feedback mechanism thus transforms even feed-forward networks into effective control strategies. Figure 4.2 shows the best network discovered by evolution. One can notice that this network simply makes the robot move forward blindly if it happens to detect a box ahead. If the robot gets stuck moving or pushing such a box, either against a wall or against another box, the feedback mechanism allows it to escape from the situation.

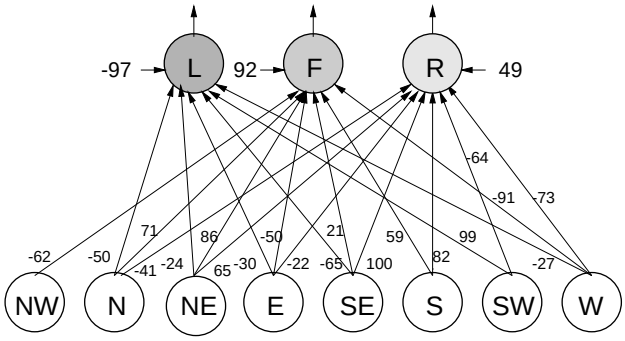

Figure 4.2 The best feed-forward neurocontroller evolved with the feedback mechanism.

Figure 4.3 shows the behavior of the robot with this neurocontroller. As can be observed, the robot often moves ahead on successive time steps, e.g., on steps 6, 7, and 8, or steps 14, 15, and 16, etc. However, if this behavior leads to situations where the robot actions fail, the feedback mechanism forces the robot to turn at the next time step.

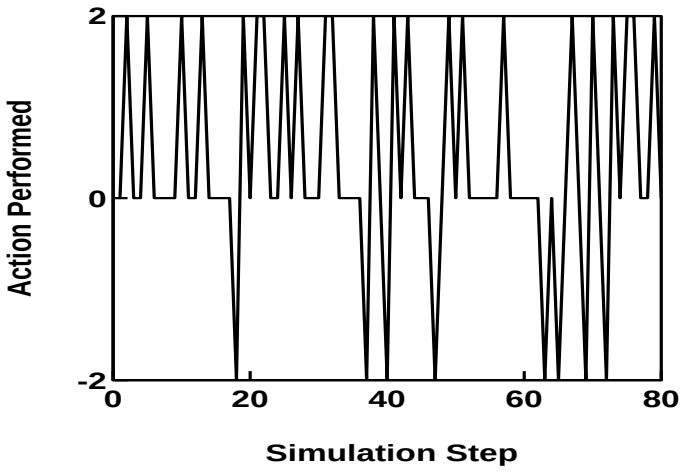

Figure 4.3 Behavior of the robot with the best feed-forward neurocontroller and a simple feedback mechanism.

Thus the feedback mechanism critically affects the fitnesses and behaviors of the box-pushing robots. With even a simple feedback mechanism at their disposal the robots attain rather high fitnesses using just feed-forward neurocontrollers. As we showed earlier, the factor severely limiting the robot fitnesses is their inability to detect failed actions. Under such circumstances we showed how evolution exploits recurrent networks to alternate actions of moving forward and turning. Now, with an explicit, albeit simple feedback mechanism even feed-forward networks lead to worthy controllers. Similar results using the AD output coding mechanism are presented in (Balakrishnan & Honavar, 1996a; Balakrishnan & Honavar, 1996c).

4.2 Sensory System Design

Biological organisms employ a variety of sensory systems like mechanoreceptors, photoreceptors, thermoreceptors, electroreceptors, chemoreceptors, etc., (Grier & Burk, 1992). One possible reason for this variety is that each such sensory modality endows the organism with a survival advantage in the niche that it occupies. For example, the vision system of most animals is *passive*, designed to detect natural light reflected from objects. However, certain species of deep-water marine life, for example *flashlight fish (photoblepharon palpebratus)*, possess an *active* vision system (Grier & Burk, 1992). They generate and emit their own light, and

detect objects by appropriately sensing the reflected light. Since surface light hardly reaches the ocean depths that these fish inhabit, evolution seems to have found this novel design to enable these fishes to overcome this severe handicap. Specific environmental constraints or behaviors thus impose their own demands on the kinds of sensors required for successfully surviving in that environment.

Not only are specific sensor traits critical to the behavior produced, interactions between different sensors are also highly important. For example, *echolocation* in bats is a successful behavior because the ultrasonic emitters and receptors act in tandem. One without the other would prove disastrous for the otherwise blind bats (McFarland, 1993). Thus, determining the right combination of sensors is an equally hard, yet important issue. Another dimension in sensory system design is the number and placement of the sensors. For example, two eyes are a common configuration in animals. As regards their placement, eyes located at the front provide binocular vision thereby allowing stereo vision and depth perception. But this results in a smaller visual field compared to eyes located on either side of the head (monocular vision). Depending on the habitat of the organism, one placement might have an evolutionary advantage over the other.

This reasoning also explains the differences in the design of the same sensory organ in different animals. For instance, the functions of vision in diurnal animals differ from those in nocturnal species. For nocturnal animals, sensitivity is at a premium and the eye must collect as much light as possible. This requires a big lens. However, to maintain the image in focus on the retina, increased lens size requires a corresponding increase in the lens curvature. Thus, many nocturnal animals have large lenses which produce small, bright images, while diurnal animals have smaller lenses which produce larger retinal images (McFarland & Bosser, 1993).

Finally, the physical environment provides many raw materials that can be used to make up signals: molecules, light waves, electrical and mechanical fields, vibrations, etc., (Grier & Burk, 1992). However, in order to exploit some or all of these signals, the organism should not only have sensors capable of detecting them, but the sensors should also be *tuned into* these signals. For example, noctuid moths are preyed upon by bats as they fly about at night. The auditory system of these moths consists of two extremely simple ears. These ears are specifically tuned to recognize the *ultrasonic* cries of hunting bats. Further the neural circuitry is so arranged as to resort to different evasive measures depending on the proximity of the approaching bat

(McFarland & Bosser, 1993). Without their systems tuned to detect ultrasonic frequencies, the moths would lose their survival edge.

What can we learn from the design of sensory systems of organisms? Given the role evolution seems to have played in developing organisms with varied behaviors realized through the *combined or co-evolved* design of sensory and neural information processing systems, it only seems natural to presume that the field of robotics or any enterprise that aims to produce autonomous agents of a certain behavioral repertoire, would benefit from similar co-evolutionary design considerations. Since evolutionary techniques are already being used quite successfully in the design of neurocontrollers for robots, we would like to explore the use of evolution in the design of the sensory systems (Balakrishnan & Honavar, 1996b). The experiments reported in this section are an effort in this direction.

4.2.1 Multi-Resolution Representation of Sensory Inputs

As in our earlier studies, the experiments in the following sections assume that the robot has visual sensors. These sensors are assumed to produce a *multi-resolution* representation of sensory information, i.e., with increasing distance from the robot the sensors return readings over larger areas, but at a coarser resolution. Inspired largely by the approximately *conical visual fields* in most animals, this sensory input representation allows regions closer to the robot to be represented at higher resolutions than those further away.

In the experiments to follow, each sensor is characterized by a 2-tuple (r, p). Here, r refers to the range to which the corresponding sensor has been tuned, measured in terms of the number of intervening cells along the shortest path from the cell occupied by the robot and the cell to which the sensor is tuned. For instance, sensors that observe cells immediately around the robot's position, as in the experiments described in Chapter 3, have range $r = 0$. Sensors with range $r = 1$ allow the robot to observe cells one cell *beyond* the robot's current position. The element p refers to the placement (or position) of the corresponding sensor, thereby dictating the *directionality* of the sensor. Our experiments assume that the robot sensors are positioned to sense along eight possible directions around the robot. We use $p = 0$ to refer to the direction 45 degrees to the left of the robot, $p = 1$ for the direction directly ahead of the robot, $p = 2$ for the direction 45 degrees to the right, and so on, as illustrated in Figure 4.4.

Given this interpretation of sensor positions, a sensor specified by $(r = 0, p = 3)$ (denoted

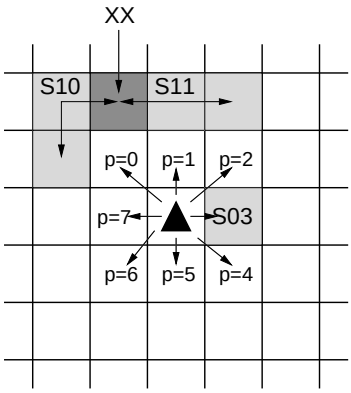

Figure 4.4 Multi-resolution representation of sensory inputs.

S03), will sense the cell immediately to the right of the robot, while the sensor $(r = 1, p = 0)$ (denoted S10) will sense a cell diagonally left and 2 cells ahead of the robot, as shown in Figure 4.4. In line with our multi-resolution representation of space, we assume that each sensor tuned to a sensing range of r cells from the robot, responds to objects over $2r + 1$ cells, $2r$ cells on either side of the target cell (r, p). Thus, a sensor with a sensing range of $r = 0$ responds to $2r + 1 = 1$ cell, while a sensor with a range of $r = 1$ responds to 3 cells, and so on. For instance, the sensors S10 and S11 in Figure 4.4 have range $r = 1$, and hence each responds to three cells shown by the arrows.

It should be noted that sensor ranges are measured in terms of the number of intervening cells and not the Euclidean metric. It can also be observed that the multi-resolution representation of sensory information leads to overlaps in sensory fields, with more than one sensor often responding to the same cell. For instance, sensors S10 and S11 have overlapping ranges in Figure 4.4 since they both respond to objects appearing in cell marked XX. These overlaps confer the sensory system with the important property of *robustness*, which is useful if some sensors happen to malfunction or fail.

4.2.2 Results in Sensor Evolution

In order to co-design the sensory system and the robot neurocontroller, we used the genetic representation described in Section 3.2.2, which specifies not only the network topology but also the placement and ranges of the sensors. Our first goal was to determine whether eight

sensors, as specified in Teller's description of the box-pushing task, were really needed or whether the task could be solved with fewer sensors. The motivation behind this study was that real sensors on real robots have acquisition and operational costs associated with them, and one would want to use the minimum number necessary for the task. A related goal was to identify the sensors that play a crucial role in the performance of the box-pushing robots.

As in our earlier experiments, we used populations of size 500 and the evolutionary runs lasted 100 generations. In order to make statistically meaningful interpretations, we performed 50 evolutionary runs, each starting with a different random seed. Further, based on our earlier results, we only evolved recurrent neurocontrollers. Since the sensors in Teller's experiments could only sense cells immediately around the robot, we set the maximum range of our sensors to $r = 0$. Our results indicate that the average fitness of the best neurocontrollers discovered over the 50 evolutionary runs is 4.63, while the best neurocontroller has a fitness of 6.32. Further, this neurocontroller only employs *three* sensors, as shown in Figure 4.5.

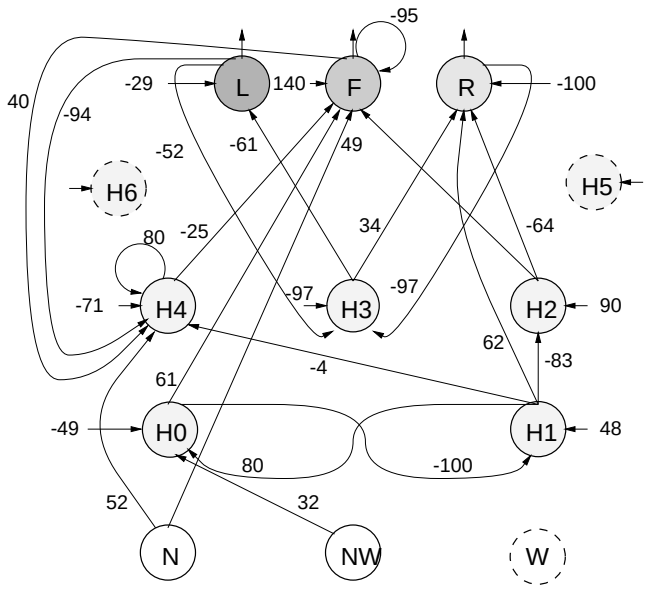

Figure 4.5 Best network with sensor evolution.

Not only does this robot use 62.5% fewer sensors than the number allowed ($\frac{8-3}{8}$) but also its fitness is 16% more ($\frac{6.32-5.46}{5.46}$) than the best neurocontroller with 8 sensors (Section 3.3.2). Although the neurocontroller in Figure 4.5 has three sensors and seven hidden units, it effectively uses only two sensors and five hidden units. It should be noted that this robot uses its

sensors to sense cells in the front, in particular, the cell immediately ahead (labeled N) and the cell diagonally to the left in front (labeled NW). Thus, there are no sensors that provide input from behind the robot.

It can be observed that owing to its strong positive bias (90), the hidden unit H2 always produces an output of 1. This changes the bias on units F and R to 76 and -61 respectively. Given these biases, it is clear that the robot is inclined towards forward moves in the absence of any sensory inputs. Further, the strong negative loop (-95) makes the robot interleave moves and turns. However, the hidden units lead to more complex dynamics. For instance, suppose the robot moved forward (F=1, L=R=-1) and encountered a wall immediately ahead. This causes the N and NW sensors to produce a -1 at the next time step, which leads to values of +1 and -1 at the hidden units H3 and H0 respectively. Now, the action performed by the robot depends on its past history, in particular on the activation of unit H1. If H1 happens to be in state +1, the output activation becomes: L = -29 -61 = -90, F= 76-95-49 = -68, and R = -61+34+62=35, which makes the robot turn right. However, if H1 happens to be in state -1, the activation of the R unit becomes -61+34-62=-89, which makes the robot move forward. Although this causes the robot to bump into the wall, it can be easily verified that at the next time step H1 produces an output of +1 (since the output of H0 becomes a -1), making the robot turn right.

Our empirical results indicate that on an average, the robots evolved in this study employ only 4.58 sensors out of the maximum possible 8. Further, the best designs found in *each* of the 50 evolutionary runs make use of fewer than 8 sensors, thereby suggesting that 8 sensors are possibly superfluous for this box-pushing task. As shown in the design in Figure 4.5, even two appropriately positioned sensors appear to be sufficient for successful box-pushing behaviors. Also remember that this design leads to higher fitness than designs evolved with 8 sensors.

What is most noteworthy is that the sensors are automatically discarded *without* the imposition of any explicit penalty. For instance, we do not penalize robots that make use of their 8 sensors; nor do we favor robots that use fewer than 8 sensors. The only evolutionary pressure appears to be the effect of sensory information on the robot behavior. We strongly suspect that having more numbers of sensors leads to *sensory inundation* and *conflict*, making the robot perform actions that are eventually *self-defeating*. Such conflicts possibly take the robot away from optimal behaviors. Our results indicate that this discarding of superfluous

sensors also leads to improvements in robot fitness.

Given that eight sensors are not necessary for box-pushing behaviors, can we identify the sensor placements that indeed contribute to the robot fitness? Are certain sensor placements absolutely critical to robot performance? We can answer these questions by analyzing data from our 50 evolutionary runs and observing the distribution of sensors used by successful agents, as shown in Figure 4.6. As can be seen, 48.3% (almost half) of the sensors used by the evolved robots are placed so as to sense cells ahead of the robot (N, NE, and NW). Further, almost 75% of the sensors are used by the evolved robots to observe cells on the front (N, NE, and NW) and sides (E and W). Few sensors, if any, are tuned to the three cells behind the robots (S, SE, and SW).

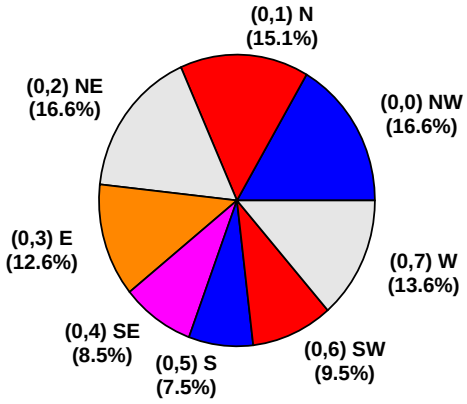

Figure 4.6 Distribution of sensors in the best robots produced in each of the evolutionary runs.

Thus, sensor evolution leads to effective designs that in *all* cases employ fewer than the maximum permissible 8 sensors. Further, the placement of sensors is largely biased towards the region ahead and to the sides of the robot, which makes intuitive sense since the robot cannot directly respond to sensors behind it. For instance, if the S sensor detects a box, the robot has to turn *twice* before it can get into a situation where it can push the box. We have observed similar results with the use of BR and AD output coding strategies. Results using the latter approach appear in (Balakrishnan & Honavar, 1996b), which also discusses the effect of increasing the range of the sensors.

This sensory design approach is reminiscent of the *feature selection problem*, where a number

of input features are available and the reasoning system has to choose to use a subset of them for a variety of performance related constraints. In our case, the robot is allowed to use many sensors, but the evolutionary design system chooses to use just a few that somehow translate into performance gains on the box-pushing task. Thus, evolutionary approaches are attractive options for the feature selection problem, as has also been demonstrated by (Yang & Honavar, 1998).

4.3 Role of Noise

Noise plays a very important role in robotics research largely because of the noise inherent in real-world environments and robot components. For example, visual sensory readings corresponding to the same object may be quite different at different times owing to a number of environmental effects like lighting, reflection, interference, occlusion, etc., or due to defects and malfunctions of the sensory apparatus. Similarly, robot actions are often sizable deviants of their intended ones, with factors like friction, improper tire-inflation, battery power, motor manufacturing defects, gradients, etc., causing marked changes from the expected trajectories of mobile robots. If robots are to be built to operate in such real-world environments, they must possess mechanisms for dealing reliably with such noises.

A further complication arises from the use of robots in unknown and often hazardous environments like space exploration, nuclear waste clean-up, geo-exploration, etc. In these operating domains, the components of the robot themselves might be prone to failure due to radiation, heat, moisture, and a host of other factors. The goal then is to design robust systems that can operate reliably in the face of such failures and malfunctions — an extension of the principle of reliable computing proposed by von Neumann (von Neumann, 1956).

Limitations, primarily time and cost, often force developers to design robots in simulation before actually building them as physical entities. In such cases, it is imperative that the simulated environments model reality as closely as possible. However, in practice, this is often hard to achieve. For example, it is easy to move a robot opposed by frictional forces in the real world but very hard to characterize the forces through closed-form differential equations to be used in the simulation. This is an extremely hard problem with no apparent solutions. Some researchers have found a compromise approach that involves adding random noise to the simulated environments. They have found that the robot behaviors designed in such

environments work rather well when transferred onto real robots (Miglino *et al.*, 1994; Miglino *et al.*, 1995).

In addition to the use of noise in developing robust behaviors in simulation, noise in the form of non-deterministic or probabilistic actions has other applications in robotics. For instance, Culbertson suggested the use of noisy actions in overcoming problems associated with robots getting caught in fixed-cycle paths (Culbertson, 1963). For example, a robot might be moving towards a goal and might deviate from its path owing to the presence of an obstacle. This deviation might take it to the beginning of its original path, thereby making it follow the same unsuccessful trajectory over and over again, until it runs out of power. This problem may be addressed by introducing probabilistic actions for the robot, i.e., the robot might perform a preferred action with high probability, but could also ignore the preferred action and perform some other action with a low probability. Culbertson suggested ways of designing artificial neural circuitry for generating non-deterministic behavior from deterministic neural components. Extensions of this idea are used in contemporary *reinforcement learning* systems (Sutton & Barto, 1998), often applied to robot learning (Connell & Mahadevan, 1993a).

In the following sections, we study the evolution of robust and reliable robot behaviors that overcome (or even exploit) noise in the robot components.

4.3.1 Kinds of Noise

In our work, we are concerned with manifestations of noise in the system components. In particular, we would like to study the effects of noise caused by two sources: sensor faults and errors in action determination. These failures and malfunctions may be a result of factors intrinsic to the sensors and neurocontroller elements (e.g., manufacturing defects, etc.), or they may be caused by environmental features like excessive concentration of radiation or corrosive substances, moisture, etc. Whatever might be the cause, we expect the robots to function reliably in their presence.

4.3.1.1 Noise Caused by Sensor Faults

In general, sensor noise can be modeled in a number of ways. For instance, sensors could either *fail* and thereby not sense at all, or they might return sensory inputs corrupted by noise. This noise in sensing may be random white Gaussian, or characterized by some probability

distribution based on the properties of the operating domain. In addition, sensors may either fail completely, i.e., for the entire duration of simulation, or may simply be unoperational for a few time-steps of the trial. In our work, we assume that each sensor has a certain *a-priori* probability of being faulty, where a fault is modeled by the sensor reading a value of 0 instead of what it currently senses. This is tantamount to saying that each sensor (with a pre-specified probability) confuses boxes and walls with empty spaces. Our simulations were performed with a 10% probability of each sensor being faulty. Thus, at each time-step, each sensor *independently* determines (with probability 0.1), whether to provide a 0 or the sensed value to the neurocontroller. Henceforth, the abbreviation *NS* will be used to characterize an experiment performed with such noisy sensors.

4.3.1.2 Noise Caused By Errors in Action Determination

We also study the effect of noise caused by errors in determining the action to be performed by the robot. For instance, under normal circumstances, a given set of sensory inputs may cause the neurocontroller to choose a particular action to perform. However, residual errors in the system might force the neurocontroller to choose a different action. Such errors in action determination are another source of noise. In our system, these action determination errors are modeled as follows.

At any given instance of time, sensory inputs are applied and propagated through the neurocontroller, resulting in the input activation of the three output units (assuming an LFR output coding strategy). But instead of the usual *winner-take-all* computation to determine the action to be performed, these units now engage in a *probabilistic winner-take-all* computation. This works as follows. The unit with the highest activation is declared the winner with probability 0.9. With probability 0.09 the unit with the second largest activation is deemed the winner and with the remaining probability (0.01), the least activated unit is chosen. Once the winner unit has been determined, its output is set to a +1 while the outputs of the other two units are set to -1. The action associated with the winner unit is then performed by the robot. Thus, even though the neurocontroller might ideally suggest a particular action to be performed in response to the sensory inputs, the robot might end up performing a different action, albeit with a relatively small probability. The net effect of this mechanism is to provide noise in the actions being chosen by the robot. However, this noise can be potentially benefi-

cial, as argued by Culbertson (Culbertson, 1963), with the robots using such alternate actions to escape from fixed-cyclic paths and stuck states. We will abbreviate such experiments as *NO* (for noisy outputs).

4.3.2 Results

In this section we present results from the evolution of box-pushing robot designs that are robust to the two forms of noises described earlier. In these experiments, the robots could use up to 10 hidden units in their neurocontrollers and up to 8 different sensors. Further, evolution was allowed to co-design the neurocontroller and the sensory system of the box-pushing robot. The experimental details were same as in Section 4.2.2.

4.3.2.1 Only Sensor Noise

When the sensors were assumed to be noisy, we found that evolution discovered robot designs that used 4.46 sensors on average. Of these, 24.7% of the sensors were placed to sense the cell immediately ahead of the robot (labeled N), as shown in Figure 4.7.

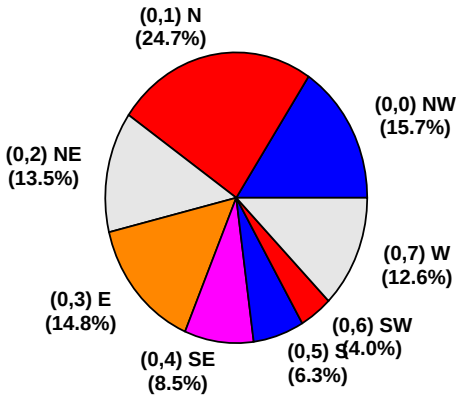

Figure 4.7 Distribution of sensors in robots evolved with sensor noise.

Thus, on an average, each evolved robot design has *at least* one sensor (actually 1.1) placed to sense the cell immediately ahead of the robot (labeled N). Further, 28% of the evolved robots have *two or more* N sensors, while 8% of the robots use three N sensors. Since the sensors are faulty, one may infer that evolution discovers robust designs that involve *duplication* of the faulty resource, namely sensors, in the critical sensing position in front of the robot. We

have shown similar results with the Action-Direction output coding strategy (Balakrishnan & Honavar, 1996c).

In addition to this design feature, it can also be observed that 81.3% of the robot sensors (3.63 sensors) are tuned to cells on the front and sides of the robot, as shown in Figure 4.7.

In our experiments the average fitness of the evolved designs (over 50 evolutionary runs), was found to be 4.53 while the best evolved network had a fitness of 5.64. Interestingly, the best evolved network made use of two N sensors, two NE sensors, and one E sensor. It should be noted here that though sensory noise affects the robot fitnesses, they are not adversely affected. This is captured in Figure 4.8, which shows the *decrease* in the fitness of the robot with and without sensory noise. This plot was obtained by evaluating the fitness of the best neurocontroller produced in each of the 50 evolutionary runs, on a set of random environments with sensory noise and then reevaluating the neurocontrollers on the same set of environments without the noise. This *difference*, δ, is shown in Figure 4.8.

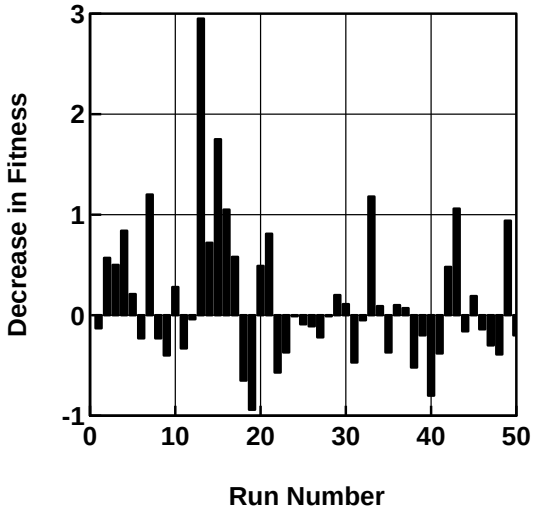

Figure 4.8 Decrease in fitness with the removal of sensory noise.

Here, a positive value of δ represents a *decrease* in fitness with the removal of sensory noise, while a negative δ value represents an *increase* in fitness. In Figure 4.8, we find that evolutionary runs produce equal numbers of designs that benefit and hurt from the removal of sensor noise. This is rather surprising since one would expect sensor noise to be harmful.

However, a closer inspection of the evolved neurocontrollers indicates that evolution often discovers designs that *exploit*, or benefit from, the sensor faults. In such cases the robots use these sensor faults to break away from their fixed-cycle paths. It is no surprise then that the removal of sensor noise causes a decrease in the fitness of these designs.

We used our data to test the hypothesis that the removal of sensor noise *decreases* the robot fitness, i.e., $\delta > 0$. We formulated the *null hypothesis* as: $H_0 : \delta \leq 0$, and the *alternative hypothesis* as: $H_1 : \delta > 0$. Given the sample size of 50 (from 50 evolutionary runs), we used the *large-sample* test (Freund, 1992; Johnson & Bhattacharyya, 1996) to compute $Z = \frac{\overline{\delta}-0}{s_\delta/\sqrt{N}}$, where $\overline{\delta}$ represents the sample mean and s_δ represents the sample standard deviation, and N the sample size. We fixed the *level of significance* at $\alpha = 0.05$ and computed $Z_\alpha = 1.645$ from the standard normal distribution tables. Based on our data we computed $\overline{\delta} = 0.28$, $s_\delta = 1.22$, and $Z = 1.628$. Since $Z = 1.628 \ngeq Z_\alpha = 1.645$, we conclude that we do not have sufficient evidence to reject the null hypothesis. Thus, our data does not provide reliable support for the claim that removal of sensor noise decreases the robot fitness.

4.3.2.2 Only Output Noise

When the sensors are assumed to be reliable but there is output noise (errors in action determination as described earlier), evolution produces designs with average fitness 4.0 and best fitness a mere 5.01. Thus, errors in action determination appear to have a critical effect on the robot fitness. This is to be expected, since these output errors make the robot ignore preferred actions and perform alternate ones with non-zero probability. This interferes with effective box-pushing behaviors and leads to a lower fitness. Thus, though output noise hurts the performance of recurrent neurocontrollers, we will show later that it is very useful for feed-forward neurocontrollers. This is because the primary benefit of output noise is in preventing the robot from performing an indefinite sequence of failed actions. As we have shown earlier, recurrent neurocontrollers make use of recurrent links to avoid getting into permanently stuck states. Thus, they do not really need the benefits offered output noise. Feed-forward neuro-controllers, on the other hand, do not have mechanisms to help them escape from such states. They exploit the output noise to attain high fitness on the box-pushing task.

But how do these evolved designs fare when they are reevaluated in environments without output noise? Figure 4.9 shows a plot of δ — the *decrease* in the fitnesses of the best robots

produced in each of the 50 evolutionary runs. As in the earlier experiments, positive values of δ represent a decrease in fitness while negative values signal a fitness increase. It can be easily observed that compared to Figure 4.8 there are more positive values of δ in Figure 4.9. This leads us to suspect that the removal of output noise actually causes a reduction in robot fitness.

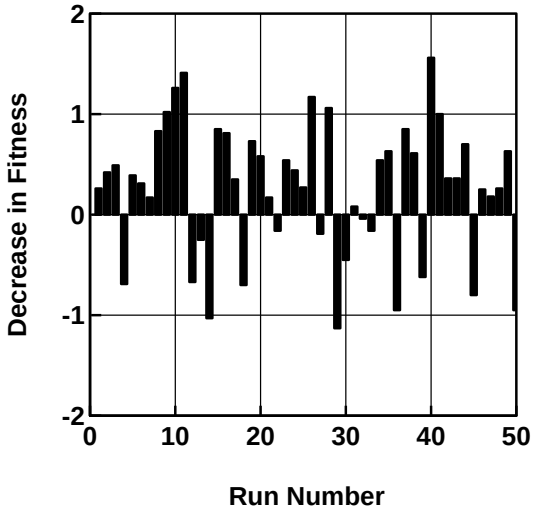

Figure 4.9 Decrease in fitness with the removal of output noise.

We can easily test such a hypothesis based on our experimental data, as was done in Section 4.3.2.1. We found that $\overline{\delta} = 0.26$, $s_\delta = 0.66$, and $Z = 2.75$. Since $Z = 2.75 > Z_\alpha = 1.645$, we reject the null hypothesis ($H_0 : \delta \leq 0$) at a 0.05 level of significance. In fact, the P-value of this test is 0.0035, i.e., there is considerable support for rejecting H_0. Thus, based on our experimental data it appears that the removal of output noise decreases the fitness of the evolved robots. It is rather surprising that these recurrent neurocontrollers rely more on output noise rather than their recurrent links to escape from potentially stuck situations. This is probably because output noise is not in their control and the preferred action is always ignored with probability 0.1. Evolution then adapts the controller designs to these circumstances, producing robots that have a strong tendency to move forward and that turn due to errors in action selection.

These evolved designs employ 4.76 sensors on an average with 3.84 of them located to sense

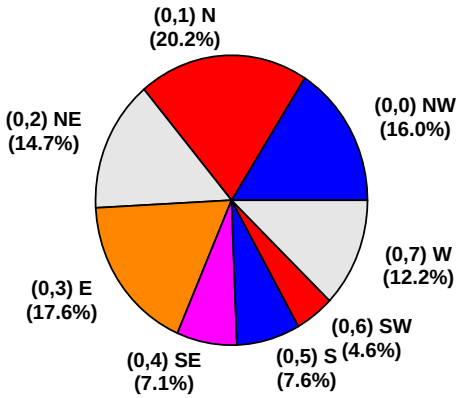

Figure 4.10 Distribution of sensors in robots evolved with noisy outputs.

cells on the front and sides of the robot. Further, each robot employs 20.2% or 0.96 sensors to observe the cell directly ahead of the robot, as shown in Figure 4.10.

4.3.2.3 Noisy Sensors and Output Noise

When the robots have to contend with both sensory as well as output noise, one would expect the robot fitnesses to be rather poor. Our experiments indicate that the average fitness of the best evolved recurrent neurocontrollers is only 3.93, while the best recurrent neurocontroller has a relatively low fitness of 4.97. The explanation for this observation has to do with the nature of sensory and output noise. It must be noted that the way we have set up things, these two noises are not correlated. This leads to situations where the two noise sources conflict with each other, often undoing what the other has done or intends to do. For instance, suppose the robot senses a box directly ahead of itself and attempts to push it. Let us assume that the box is against a wall, which causes the action to fail. Suppose the sensor that detected the box fails at the next time step. Let us assume that this change in sensory input leads to an activation of the unit corresponding to a right turn. Thus, if the robot follows the dictates of the neurocontroller, it can escape from its futile pushing. However, let us assume that the output determination mechanism also faults at this time step and chooses the forward move action over the right turn. Thus, the robot, which could have used sensor noise to escape from its undesirable state, is again relegated to its earlier fate.

Although this reasoning might suggest that the robots will have much higher fitnesses if

one or both sources of noise are removed, our results indicate otherwise. The fitnesses of the 50 recurrent neurocontrollers with and without the noises, are shown in Figure 4.11. As can be observed, the removal of both forms of noise lead to a decrease in fitness. Also, there appear to be instances where the removal of one form of noise actually leads to an increase in fitness. However, these fitness increases appear to be rather small in magnitude.

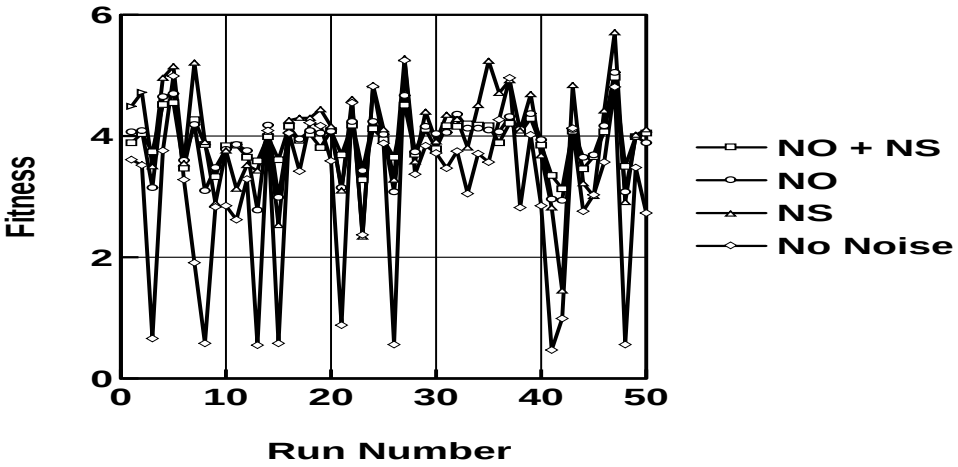

Figure 4.11 Performance of evolved recurrent networks evaluated with and without noises.

We can support these observations by statistically analyzing our results. For instance, we tested the hypothesis that the robot fitnesses decrease when both noise sources are simultaneously removed. We found that the data supported this hypothesis at a significance level $\alpha \ll 0.0001$. Hence with immense confidence we can conclude that the evolved robot designs rely heavily on both forms of the noise, and their joint removal leads to a statistically significant decrease in robot fitness.

These evolved designs use 4.46 sensors on average. As with our earlier experiments, these robots generally prefer to use sensors located in front and on the sides of the robot (85.7% of the sensors) as shown in Figure 4.12.

The above results were obtained with recurrent neurocontrollers. We mentioned earlier that recurrent neurocontrollers do not need the benefits of noise (e.g., their non-determinism) to escape from futile situations. Instead, they can judiciously use their recurrent links to interleave moves and turns. Indeed, the fitnesses of the evolved robots bear testimony to

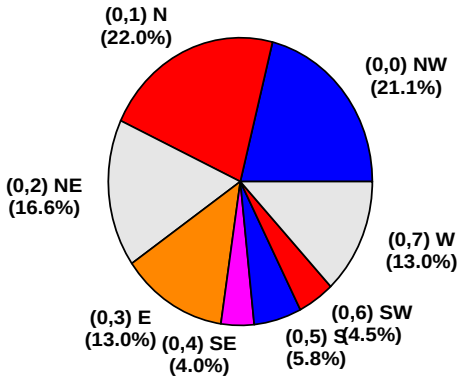

Figure 4.12 Distribution of sensors in robots evolved with noisy sensors and noisy outputs.

the fact that recurrent networks evolved without noise perform much better than those that have to deal with noise. Thus, noise hampers recurrent neurocontrollers, interfering with the box-pushing abilities of such robots.

What about feed-forward networks? Will these networks benefit from the addition of noise? Figure 4.13 shows the fitnesses of the best feed-forward neurocontrollers (with no hidden units) that were found in each of the 50 evolutionary runs. One can clearly see the decrease in fitness when one or both noise sources are removed. In fact, it appears that the removal of output noise causes a marked decrease in fitness, while the removal of sensory noise has little effect. When both noise sources are removed, the fitness of each neurocontroller drops to approximately 0.5.

These differences are clear even to the naked eye. A statistical analysis of our data allows us to conclude with great confidence ($\alpha \ll 0.00001$) that the removal of one or both noise sources results in a decrease in fitness. Thus, noise plays a critical role in the fitness of feed-forward neurocontrollers and its removal results in a statistically significant decrease in fitness.

Our experiments thus show that evolution produces designs that overcome or appropriately exploit the constraints imposed by the robot and its environment. For instance, when sensors are faulty, evolution discovers robust designs that involve duplication of sensors along key dimensions around the robot. When the action selection is assumed to be noisy, evolution discovers neurocontroller designs that are quite different. For example, the evolved recurrent neurocontrollers do not use recurrent links to interleave moves and turns. Instead, they rely

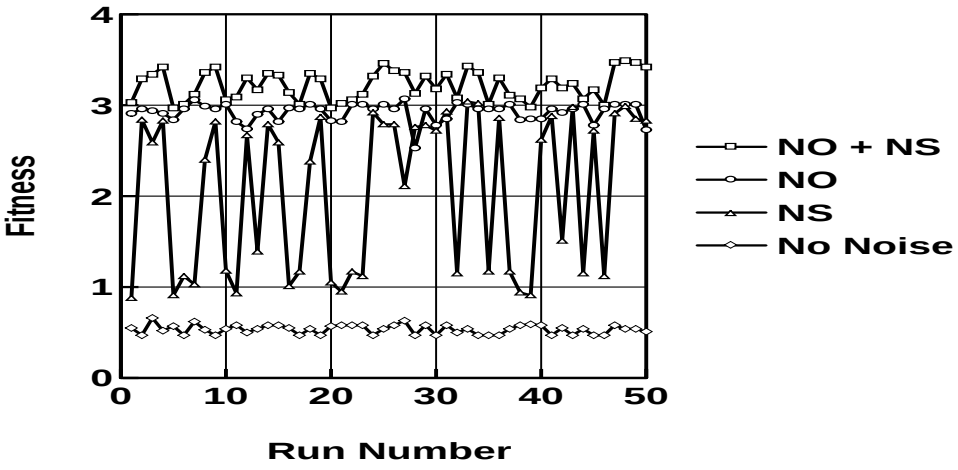

Figure 4.13 Performance of the best evolved feed-forward networks (with no hidden units) evaluated with and without noises.

on the action determination errors to escape from potentially stuck situations. Similarly, feed-forward neurocontrollers exploit system noise to attain relatively high fitnesses on the box-pushing task.

4.4 Discussion

In this chapter we have extended Teller's box-pushing task along several important dimensions which, in our opinion, makes the task more *realistic*. First we studied the effect of feedback in the design of effective box-pushing behaviors. Thus, in contrast with the box-pushing robots of Chapter 3, these robots have a simple feedback mechanism that signals them if their actions fail in the environment. Such situations arise in three different ways in this environment and include attempts made by the robot to move into a wall, push a box into a wall, or push two or more boxes at the same time. When these situations arise and the action chosen by the robot fails, the feedback mechanism forces the robot to turn in the next time step. Thus, the presence of the feedback mechanism makes the robots more efficient in their execution of the task since they waste less time pursuing futile moves. Our results indicate that when such a mechanism is available, even feed-forward neurocontrollers demonstrate highly effective box-pushing behaviors. In such cases, once a box is detected, the

robots keep pushing it until an action failure is signaled by the feedback mechanism. When this happens, the robots simply turn and move away. This behavior is in stark contrast to the behaviors we observed without the feedback mechanism, which involved complicated ways of interleaving moves and turns. Further, even recurrent neurocontrollers evolved with a feedback mechanism have rather different structures from the ones evolved earlier, and appear largely inclined towards moving forward. Thus, not only does evolution design behaviors in line with the constraints of the environment, but it also finds novel ways of adapting to changes in the environmental constraints.

We also explored the use of evolution in the design of the robot sensory system. In particular, we evolved the number, placement, and range of the robot sensors, in addition to designing the neurocontroller. This approach produced a number of interesting results. We found that the box-pushing task did not require 8 sensors and each of our evolutionary runs produced robot designs that made use of fewer than 8 sensors. It should be pointed out that there was no explicit pressure towards using fewer sensors and the optimization of the numbers of sensors was an *emergent property* of this system. We believe that having fewer sensors leads to lesser sensory conflict and hence is favored by evolution. In this regard, the sensory design problem is simply the *feature selection problem* in a different guise. This study also confirmed our intuition that sensors located to sense cells ahead and to the sides of the robot probably play a more significant role in their performance than sensors located behind the robot.

Finally, we considered the design of robust robot behaviors in the presence of two kinds of noise. Since noise is omnipresent in the real-world, this consideration makes the box-pushing task a little more realistic. We introduced two forms of noise in our system, one caused by sensor faults and the other caused by errors in the determination of the robot action. In each case we found that evolution discovers designs that are well adapted to effectively handle (or exploit) the noise. For instance, if the sensors are faulty, evolution produces robots that employ two (sometimes three) sensors to sense the cell immediately ahead of the robot. In effect, evolution discovers robust designs that involve the *duplication* of the *faulty resource* (namely the sensors) in key directions around the robot. Unlike sensory noise which can be countered by means of resource duplication, output noise cannot directly be countered by evolution. In this case, evolution discovers robots that are very strongly biased towards moving forward. These designs rely heavily on the output noise to stop them from incessantly moving forward

or repeating futile forward moves. This dependence is easily illustrated via statistical studies that showed that the removal of output noise results in a significant decrease in the fitness of the corresponding robot. When both kinds of noise are present in the system, evolution discovers designs that are tailored to exploit them. This is particularly evident in feed-forward neurocontrollers that use this noise to escape from futile pushing situations.

In this chapter we have used the evolutionary approach to design neurocontrollers and sensory systems for box-pushing robots. In each case we have analyzed the empirical results and have found that evolution discovers novel ways of exploiting the environmental constraints and tailoring the designs to adapt perfectly to the environments in question. Although the extensions that we have explored here make the task more realistic, there is another important dimension that we have ignored. We have assumed that the robots have an inexhaustible source of energy that allows them to survive through their entire simulation time. This is in contrast to most real-world situations where the robots must be designed keeping such energy considerations in mind. For instance, real-world sensors require energy to operate. In such a scenario, unnecessary or irrelevant sensors must be discarded (or switched off) by the robot. Similarly, useless units must be discarded from the neurocontrollers. A number of robot designs presented in this chapter include examples of such excesses where sensors and units are retained by the robot, but are never used in any fruitful computation. We need mechanisms to circumvent such designs faults. In Chapter 5 we will pursue one such approach that imposes energy penalties on the robot components and uses evolution to discover energy-efficient designs.

5 ENERGY-EFFICIENT BOX-PUSHING ROBOTS AND THE NEED FOR SPATIAL LEARNING

The robots and behaviors evolved in Chapters 3 and 4 were not designed to be efficient users of energy. Instead, it was implicitly assumed that the box-pushing robots always had enough energy to help them see through their quota of box-pushing time. In this chapter we consider the case where the robots have a limited amount of energy and must make judicious use of their limited reserves. We study the role of evolution in the design of such energy-efficient robots. We also study the evolution of robot designs when power sources are present in the environment and the robots have mechanisms to charge their batteries by approaching them. Finally, we explore the kinds of behaviors that emerge when the robots have a built-in mechanism for learning, remembering, and navigating to the locations of power sources in their environments.

5.1 Effects of Energy Constraints on Robot Design

Teller's specification of the box-pushing task (Teller, 1994) implicitly assumed that the robots had enough energy reserves to take them through their stipulated box-pushing time. The design of robot behaviors in such environments were then influenced only by the environmental constraints and other limitations on the abilities of the robot. However, for this task to be realistic, one must also worry about the amount of energy consumed by the robot. In fact, energy can turn out to be a key factor in determining the features the robot can possess and the kinds of behaviors it can exhibit. For instance, if a robot uses sophisticated equipment for sensing, processing, and acting, it may require substantial amounts of energy. Additionally, if this robot has to operate autonomously for relatively long periods of time, it needs a battery with a rather large capacity. Assuming the use of conventional battery technology, larger capacities typically entail larger on-board volume to stow the battery and a proportional

increase in the robot weight. Not only does this affect the dexterity, speed, and efficiency of the robot, but also restricts the kinds of robot behaviors that can emerge.

Alternatively, the robots might carry smaller batteries on-board and instead compromise on the sensors, processing, and actuators they support. Such robots must be *energy-efficient* in terms of their use of sensors and processing. Given our design parameters, these robots must only use as many sensors as are absolutely necessary for the task on hand. Similarly, they should possess neurocontrollers that make use of minimal numbers of units and links. These considerations may severely restrict the abilities and performance of the robot.

However, if there are power sources in the environment (e.g., power outlets in a room) and the robots have mechanisms to approach these power sources to charge their batteries, the robot designs can afford to be a little more extravagant. In such cases, the robot designs might indulge a little more in the numbers of sensors and neurocontroller units. This would work in environments where the locations of the power sources remain constant. However, if the robot has to operate in novel environments, it needs additional mechanisms to learn and remember the locations of power sources in this new environment, an ability we refer to as *spatial learning*. With such spatial learning mechanisms at their disposal, robots can learn the locations of power sources in novel environments, and navigate to them when in need of power.

In what follows, we describe our attempts to study the evolution of energy-efficient robot sensory designs and behaviors. We will study the effects of the presence or absence of power sources on the designs evolved, and attempt to characterize the influence of spatial learning. with and without power sources and spatial learning. But first, we describe an important change in the sensory apparatus of our robots.

5.2 Object Sensors

In the task specification of Teller (Teller, 1994), it was assumed that the robot sensors were capable of reliably recognizing and differentiating between boxes, walls, and empty spaces. Thus, if the robot had a sensor placed to detect the cell immediately in front of it, the sensor was assumed to return different values depending on the occupancy of that cell. Although this is largely true of biological sensors (e.g., a human eye can detect and aid in the recognition of different kinds of objects), contemporary robot sensor technology dictates otherwise. For instance, sonar sensors can be used quite reliably to differentiate between walls and empty

spaces (barring some problems posed by specular reflections) (Everett, 1995). However, differentiating between walls and boxes is much harder, and may even be impossible using simple sonar sensing. One may need to use other aspects like shape, color, surface texture, reflection properties, etc., to perform this distinction. In any case, it is possible that the detection and recognition of different kinds of objects might require different types of sensors and entail their own respective costs (in terms of time required, energy consumed, reliability and accuracy of recognition, etc.). It is thus fruitful to think of *logical or virtual* sensors that detect (or recognize) *specific* kinds of objects at particular positions relative to the robot. For instance, we can think of a logical wall sensor that is placed so as to detect the presence of walls immediately ahead of the robot. This sensor is assumed to be incapable of detecting boxes or empty spaces. These sensors are logical in the sense that this does not imply the robot *physically* has a sensor positioned to sense the corresponding cell. Rather, using the different sensors that are available to the robot, it is somehow able to determine the presence or absence of walls in the cell under question. By delinking the logical from the physical, this view of sensors permits the design of systems that can be easily implemented on different robots with vastly different sensory equipment, as long as the physical sensors on the robot can be used to yield the inputs required by the logical sensors. In the experiments that follow, we use logical sensors that detect walls (W), boxes (B), and power sources (P). When these sensors detect an object of their type, they provide a value of $+1$ to the neurocontroller. Otherwise, they provide a value of 0.

5.3 Energy Consumption and Energy Acquisition Models

In the experiments in this chapter, we assume that the robot sensors and neurocontroller units consume *one unit* of energy every time they are used or are involved in a computation. Thus, if a robot has four sensors and five neurocontroller units, the drain on the robot's battery in each time step of operation is nine energy units. In addition, robot motion also consumes energy. While turns and moves only result in a loss of five energy units, pushing a box takes an additional toll of five energy units. The choice of these numbers is rather *ad-hoc*, with the only guiding principle being to ensure that box-pushing is an energy-draining operation. More realistic values can be chosen depending on the task environment and the physical characteristics of the robots employed.

Given this scenario the system is *a-priori* biased *against* pushing boxes since that is the most efficient use of its limited energy. However, as will become evident soon, effective box-pushing behaviors emerge since the reproductive potential of the robots is governed by their fitness (number of boxes against the wall) rather than by their ability to survive their quota of simulation time.

The robots in these experiments also possess mechanisms to charge their batteries when they encounter power sources. This happens as follows. When the robot comes into contact with a power source head-on, its battery level increases by a fixed amount, referred to as unitCharge, in every time step. Thus, if the robot spends 5 time units facing a power source, its battery level increases by the amount: $5\times$ unitCharge. We also assume that the robot battery has a maximum capacity (maxEnergy) which limits the maximum amount to which the robots charge. Although power sources are normally located along the walls, some of our experiments assume that power sources can appear anywhere in the arena. Although this makes it easy for the robot to charge up at a power source by approaching it from multiple directions, it often impedes the robot's execution of its box-pushing task – power sources appearing on the path of a fully-charged robot are obstacles that the robot must avoid. In addition, boxes may end up next to the power sources, either due to their random placement or as a result of the box-pushing behaviors of the robot, and may block access to power sources. These factors may cause robot fitnesses to actually decrease, contrary to what we might expect. Evolution must contend with these possibilities while designing effective robot behaviors.

5.4 Simulation Details

In order to make the box-pushing task a little more challenging, we considered rooms of size 10×10 containing 10 boxes. The robots were given 500 time steps to clear the room. The robots could use up to 20 sensors and 10 hidden units. As we mentioned earlier, each sensor was capable of detecting only one type of object (B, W, or P), and the sensor ranges were restricted to a maximum of two cells beyond the robot. Each member of the population in a particular generation was evaluated on the same set of 25 random environments, while different generations were evaluated on different sets of environments. The decision to use a small number of environments to evaluate the robots was made keeping in mind the increased time complexity of the robot evaluations (500 simulation steps per environment). The average

performance of the robot over the 25 environments was said to be its fitness, and was used by the binary tournament selection procedure.

Following our results in Chapter 4, the robots in these experiments were assumed to have a feedback mechanism that informed them of failed actions. This mechanism made sure that the robots did not repeat an action that led to failure in the previous step. To keep the task realistic, we assumed that the robot sensors were noisy, as described in Section 4.3.2.1. We also assumed that errors existed in choosing robot actions, as described in Section 4.3.2.2. We used evolutionary algorithms to search the combined design space of robot sensory systems and recurrent neural network controllers. The following sections present results from these experiments.

5.5 Inexhaustible Battery Power and No Power Sources

In this experiment, the robots were assumed to possess a battery with inexhaustible energy. In this regard they were similar to the robots in Chapters 3 and 4. This experiment also assumed that there were no power sources in the environment. Results of our evolutionary experiments are summarized in Table 5.1

Table 5.1 Results with inexhaustible battery power.

	Average	Best
Fitness	10.25	11.48
Survival Time	500.00	500.00
Wall Bumps	12.07	8.44
Hidden Units	4.70	0.00
Sensors	11.70	16.00

It can be easily observed that evolution discovers robot designs that perform very well on the box-pushing task. The best robot attains a fitness of 11.48, which is approximately 82% of the maximum fitness (14) that could be attained by the robot in this environment. This is a significant increase from the performance of the best robot discovered in our earlier experiments, which had a fitness of 6.32 (approximately 63% of the maximum fitness of 10 in the 6×6 environment). The primary reason for this difference is the increase in the simulation

time. Even though the robots in the earlier experiments worked in smaller environments (6× 6), they only had 80 time steps for box-pushing when compared to the 500 time steps allowed for the robots in the current experiment. We have shown elsewhere (Balakrishnan & Honavar, 1996a) that an increase in the time allowed to push boxes, leads to an increase in robot fitness. However, we also showed that fitness does not increase indefinitely, and the fitness curve flattens out after approximately $O(N^3)$ of simulation time (where N × N is the size of the arena). We also provided an intuitive explanation for this observation (Balakrishnan & Honavar, 1996a).

It can be observed from Table 5.1 that though these robots demonstrate high fitness behaviors, they are rather extravagant in their use of resources. For instance, the average neurocontroller uses 4.7 hidden units while the average robot employs 11.7 sensors. The best robot design can be seen to employ 16 sensors. This conflicts with the results of our earlier experiments in sensor evolution (Section 4.2), which indicated that the box-pushing task could be well addressed with just a handful of sensors. In particular, evolution discovered robot designs that used as few as two sensors. Even though the sensors in the current experiment are different from the ones used earlier (the current sensors are object-specific while the earlier sensors could detect multiple kinds of objects), 16 is still a large number of sensors to use. Why does this happen?

There are two possible explanations for this observation. First, it may be the case that a number of these sensors and hidden units are useless i.e., even though they are available, they may not affect the robot behavior in any way. This may happen because the sensors and hidden units are not connected to other elements of the neurocontroller. Second, our experiments assume that the sensors are noisy. As we illustrated in Section 4.3.2.1, this leads to robot designs that employ multiple sensors to detect cells in key directions around the robot. Such fault-tolerant designs may also be contributing to the number of sensors used by robots in this experiment.

A careful analysis of the evolved designs tells us that these explanations are correct. We observe that many sensors and hidden units, although available to the robot, do not influence the behavior of the robot. For instance, Figure 5.1 shows the best neurocontroller evolved under these conditions.

It can be observed that though this robot has 16 sensors at its disposal, only 11 provide any form of input to the neurocontroller (notice that multiple sensors appear in three locations).

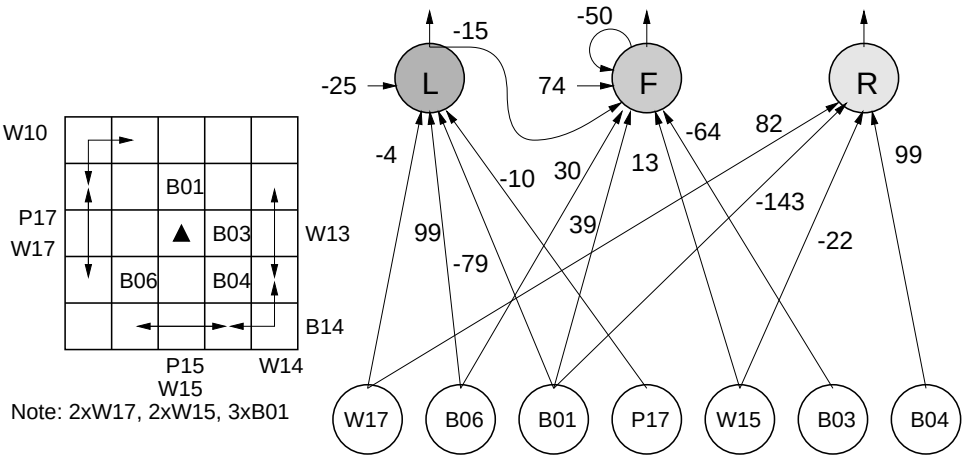

Figure 5.1 Best robot design evolved with inexhaustible battery power.

The remaining 5 sensors are not used at all. For instance, the robot does not make use of sensors W13 and P15. In addition, the robot appears to use a power source sensor (P17). But, as there are *no* power sources in this environment, this sensor always provides a value of 0 to the neurocontroller. Thus, even this sensor, though connected to the neurocontroller, is completely useless.

Also notice that this design has multiple sensors of the same type tuned to certain cells (e.g., 3 box sensors at (0, 1), 2 wall sensors at (1, 7), etc.). Since the sensors are assumed to be faulty, this design might offer benefits of fault-tolerance by duplicating the faulty resource.

Figure 5.2 shows the distribution of the sensor types at the different possible sensor positions, averaged over the designs produced in each of the 50 evolutionary runs. For instance, the evolved robots in this experiment, on average, use 1.25 box sensors positioned to sense the cell immediately ahead of the robot (0, 1). They also use 0.5 wall sensors, on average, to detect walls two cells ahead and to the left of the robot (1, 0). Further, since the average values are non-zero for all sensors and positions in Figure 5.2, evolution appears to discover robot designs that make use of every type of sensor in every possible sensor position. In addition, power sensors appear to be used often in these designs even though there are no power sources in this environment. Thus, these robot designs seem to be extremely wasteful of system resources.

What would happen to these robots if they were to now operate with limited battery power? No doubt their fitness would decrease, but how bad would their fitness become? Can

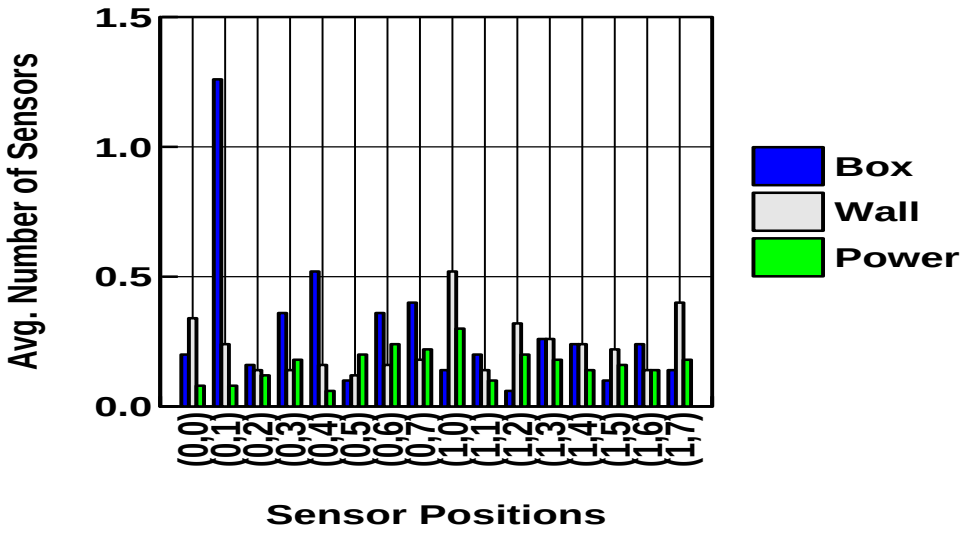

Figure 5.2 Distribution of sensors in robots evolved with inexhaustible battery power.

we improve the robot designs to help them operate in energy-constrained environments? We explore answers to these questions in the following sections.

5.6 Limited Battery Power and No Power Sources

Suppose the robots had batteries of limited capacity, say maxEnergy = 1000. This energy is depleted by the sensors, neurocontroller units, and robot actuators, as described in Section 5.3. Under these circumstances, how would the robots designed in Section 5.5 fare? We would naturally expect the fitness to decrease, but by how much? How adversely would the energy limitations affect the robot design evolved earlier?

When the best neurocontroller, shown in Figure 5.1, was reevaluated with maxEnergy = 1000, its fitness decreased from 11.48 to a mere 1.74 (an 85% drop in fitness). Further, its survival time changed drastically, from 500 time steps to just 46.07 (a drop of 91%). The changed environmental constraints, in this case limited energy reserves, thus play havoc with the box-pushing performance of the robot. However, if robot designs were to be evolved in environments with these energy constraints, would evolution find more effective designs?

Table 5.2 shows the result of evolving *energy-efficient* robot designs in environments without

Table 5.2 Results with limited battery power and no power sources.

	Average	Best
Fitness	3.50	3.69
Survival Time	150.67	152.41
Wall Bumps	1.50	1.13
Hidden Units	0.00	0.00
Sensors	1.10	1.00

any power sources. It can be observed that evolution discovers designs that better fit the new environmental constraints. In this case, not only are the robots more fit on average (3.5 versus 1.74 earlier), but also employ fewer sensors and hidden units (1.10 and 0.00 respectively).

An analysis of the best robots produced in each of the 50 evolutionary runs shows that *none* of the robots use box sensors. This curious design is presumably the result of multi-objective tradeoffs inherent in the robot's task. For instance, box sensors are useful because they can detect the presence of boxes and focus the robot in appropriate directions. However, in our system the box sensors may often fail to dictate the actions of the robots. This is because our system is constrained by sensor and output noise, which may either cause boxes to go undetected or ignore actions suggested by the box sensor. Further, box sensors are not critical to box-pushing since the robots can push boxes by simply running into them without even detecting them. In addition, a box sensor consumes one unit of energy in each step of its operation. Given these evaluations and the fact that the robot has a limited *non-replenishable* energy source, evolution presumably determines that the disadvantages of having a box sensor outweigh the advantages. Similarly, these robot designs do not make use of any hidden units.

Figure 5.3 shows the best neurocontroller discovered by evolution in this setting. It has a fitness of 3.69, and employs one wall sensor and no hidden units.

Notice that the thresholds on the output units bias the robot towards forward moves in the absence of sensory inputs. We can also observe that this robot uses a sensor to detect walls two cells in front of the robot. This has interesting consequences both for avoiding bumping into walls and also for automatically determining when to quit pushing a box. For instance, if the robot is pushing a box, the wall sensor detects a wall the instant the box comes against the wall, producing a sensory input of W11=+1. This causes the activations of the output units

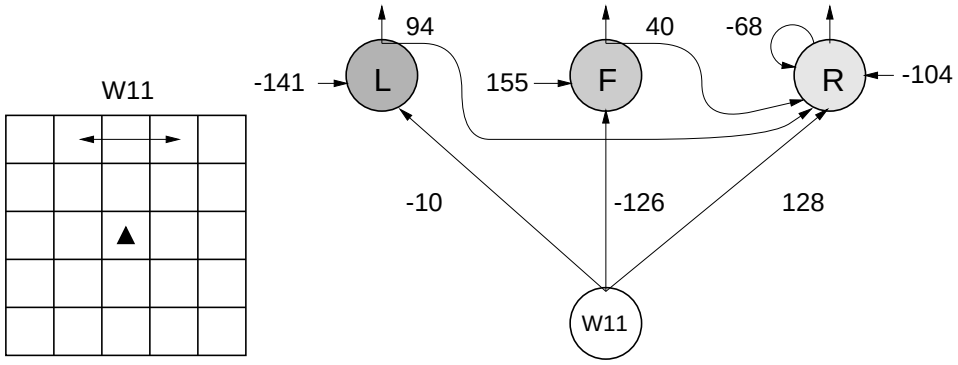

Figure 5.3 Best robot design evolved with limited battery power and no power sources.

to become: L=-141-10=-151, F=155-126=29, and R=-104+68+40+94+128=226. Hence the robot turns right with high probability, thereby leaving the box against the wall. Notice that the robot could have used its feedback mechanism to realize the same behavior, i.e., its attempt to push the box into the wall would have caused it to turn at the next time step. However, this causes the robot to waste an action step, which is of premium to these energy-constrained robots. Evolution thus discovers designs that save the robots this wasted effort.

Figure 5.4 shows the distribution of sensors in the evolved robots. It can be noticed that these robots primarily use wall sensors that are critically placed to detect walls two cells ahead of the robot, i.e., in positions (1, 0), (1, 1), or (1, 2). As we mentioned earlier, these sensor placements allow the robot to not only avoid walls, but also determine when to stop pushing a box. In contrast to Figure 5.2, one should notice that sensors in Figure 5.4 are placed to sense choice locations around the robot. For instance, none of the evolved robots have sensors to observe cells *behind* the robot.

5.7 Power Sources in Fixed Locations

In this experiment we used robots with limited battery reserves as before, but assumed that there were power sources in the environment. The locations of these power sources were fixed across environments. For instance, in our experiments the environments had two power sources located along the walls, as shown in Figure 5.5.

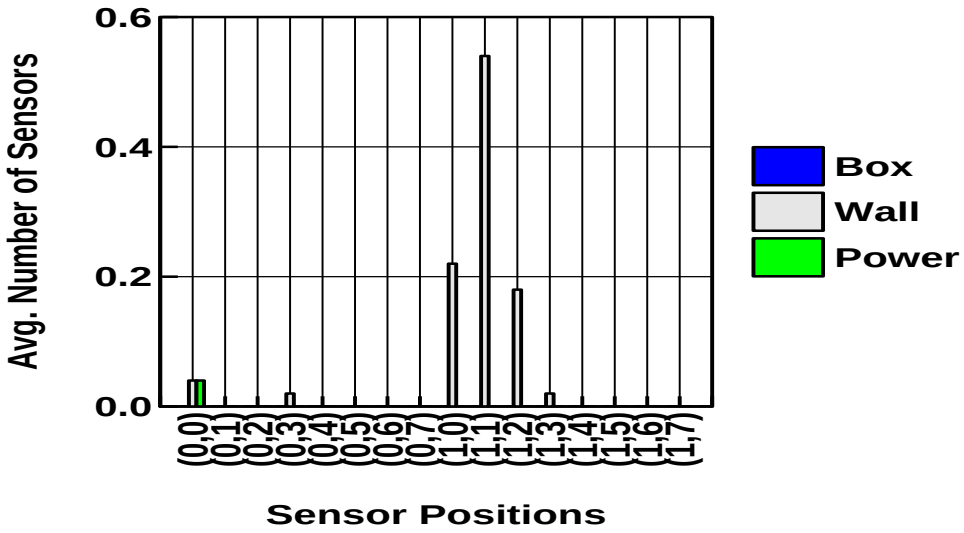

Figure 5.4 Distribution of sensors in robots evolved with limited battery power and no power sources.

Table 5.3 summarizes the results of our evolutionary experiments in this environment. It can be observed that these results are very similar to those obtained in environments without any power sources (Table 5.2). The only difference between results in the two environments appears to be in the number of wall bumps. An analysis of the results suggests that the robots in environments with fixed power source locations bump into walls more often than the robots in environments without power sources. However, this is a direct result of the locations of the power sources. Since power sources are assumed to be attached to the walls, the robots bump into walls when they approach the power sources to charge their batteries.

Table 5.3 Results with two power sources in fixed locations.

	Average	Best
Fitness	3.54	3.73
Survival Time	152.72	156.26
Wall Bumps	2.44	1.25
Hidden Units	0.06	0.00
Sensors	1.08	1.00

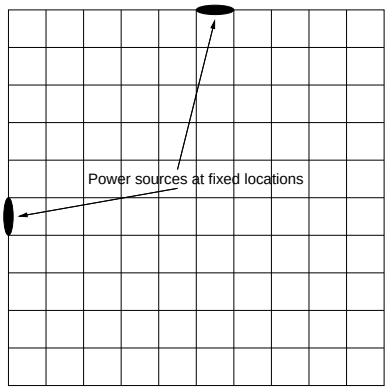

Figure 5.5 Locations of the two fixed power sources.

But why do robots perform so poorly in this environment? Why is their survival time no different from those of robots in environments without power sources?

The explanation again lies in the locations of the power sources. As we mentioned earlier, the power sources are located along walls, and in order to charge up, the robots must approach walls. But this behavior conflicts with the ability of the robot to detect and avoid walls, and determine when to quit pushing a box. In order for both kinds of behaviors to co-exist, the robot must be capable of sensing both power sources and walls, and determining the appropriate behavior in that context. However, operating multiple sensors involves energy costs. Further, it is quite possible for the power source to be obscured by a box that has been pushed against it. This makes the power source inaccessible. In light of these considerations, evolution appears to choose designs that *do not* make use of power sensors. Further, the designs evolved are extremely similar to the ones in environments without any power sources (Section 5.6).

This similarity is clearly evident in Figure 5.6, which bears a striking resemblance to Figure 5.4. As can be observed, almost all of the sensors used by the evolved robots are wall sensors that are placed to detect walls two cells ahead of the robot.

Figure 5.7 shows the design of the best robot evolved in these circumstances. As can be observed, this robot employs one sensor critically placed to detect walls two cells ahead of the robot. In the absence of walls near the robot, the thresholds on the output units bias the robot towards forward moves. However, when the robot encounters a wall two cells ahead,

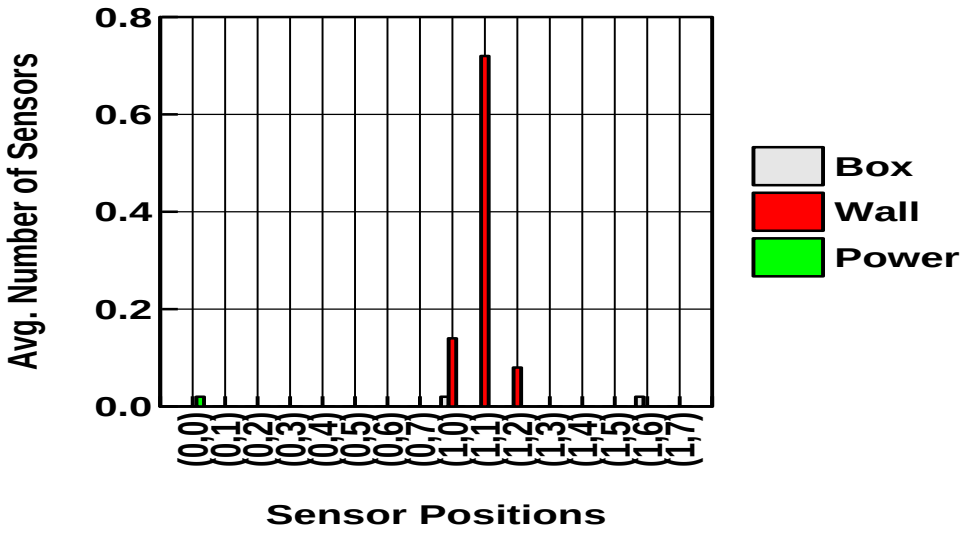

Figure 5.6 Distribution of sensors in robots evolved with power sources in fixed positions along two walls.

the wall sensor provides a value W11=+1 to the neurocontroller. This leads to the following activations in the output units: L=-42+88+50+91=187, F=142+2-86=58, and R=-98. Thus the robot quits pushing the box and turns left with very high probability.

For the reasons mentioned earlier, our results indicate that the robots are unable to really exploit the power sources, even though their positions are completely predictable (fixed locations). However, we argued that this was a direct result of the locations chosen for the power sources. We suggested that power sources located on the walls of the arena imposed conflicting requirements on the robots, forcing them to approach walls rather than avoid them. This makes us wonder what would happen if the power source locations became unpredictable? How would the robots fare if different environments had power sources located in different positions? The following experiment provides answers to these questions.

5.8 Power Sources in Random Locations

Suppose the box-pushing robot was being evaluated in different environments (e.g., different rooms in a building). It is quite possible that the power sources might appear in different locations in each of these environments. How does this unpredictability of power source locations

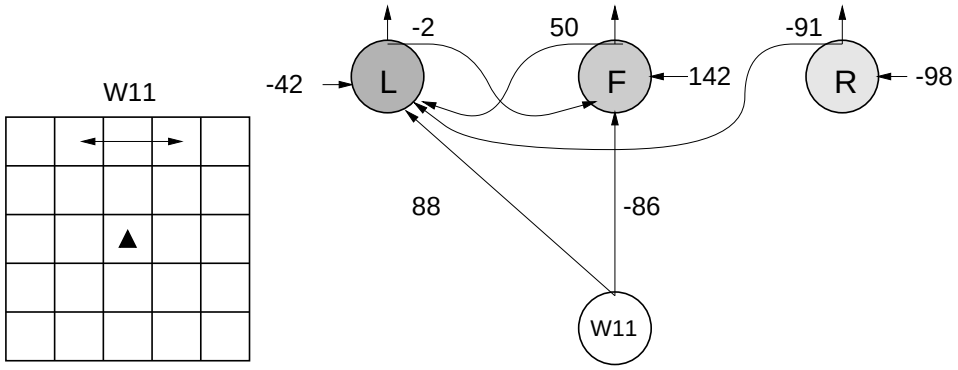

Figure 5.7 Best robot design evolved in environments with power sources in fixed positions along two walls.

affect the evolved behaviors?

In our experiments, each robot environment had two power sources that were placed at *random* locations. These locations could be on the walls or even within the arena. As described earlier (Section 5.3), power sources located in the arena allow the robot to charge up from multiple directions. However, like walls, they too obstruct robot motion and their box-pushing behaviors.

Table 5.4 shows the results of our experiments in this setting. It can be observed that though these robots survive longer than the robots in environments with fixed power sources, they have slightly lower fitnesses. In fact, their fitnesses are even lower than the ones evolved in environments without any power sources (although the magnitude of the difference is small, e.g., 2.95 versus 3.5, the results are statistically significant). It would thus appear that the robots do better in environments without any power sources than in ones where the power sources are in unpredictable locations. Are these power sources not used at all then?

It should be noted that these robots survive much longer (approximately 55% longer) than the robots in Sections 5.6 and 5.7. This is largely the result of their ability to detect power sources and approach them. In fact, we observed that most of the robots evolved in this experiment, made use of power sensors, unlike the designs in the earlier experiments. If the robots are indeed using power source sensors to survive longer, how is their fitness much lower than before? This can be explained by the fact that often power sources appear within the arena. In such cases, the power sources often obstruct good box pushing behaviors as they

Table 5.4 Results with two power sources in random locations.

	Average	Best
Fitness	2.95	3.21
Survival Time	193.73	241.70
Wall Bumps	3.46	2.69
Hidden Units	0.00	0.00
Sensors	1.12	2.00

happen to be on the path of a robot pushing a box to a wall. This leads to lower fitnesses.

Figure 5.8 shows the distribution of the sensors in the robots evolved in this experiment. It can be observed that, on an average, almost all the wall sensors on any evolved robot are placed to detect walls at a distance of two cells in front of the robot. As we explained earlier, this design choice allows the robots to detect boxes against walls, thereby allowing them to push boxes in an optimal fashion. It should be noted that, on an average, the robots in this experiment also use one power source sensor positioned to detect power sources to the immediate right. The robots in the earlier experiments did not have this feature.

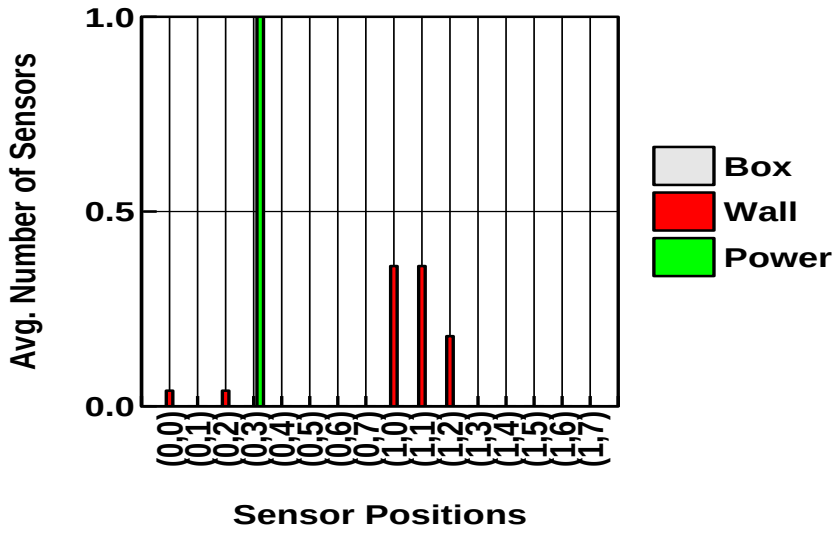

Figure 5.8 Distribution of sensors in robots evolved in environments with random power source locations.

Figure 5.9 shows the design of the best robot discovered by evolution. Interestingly, this robot employs two sensors: one to detect walls and the other to detect power sources. This robot demonstrates many interesting behaviors. It should be noted that in the absence of sensory inputs the robot is biased towards forward moves. However, if the robot moves forward and its sensors detect a wall two cells away (W12=+1), the neurocontroller activations become: L=13+81+49=142, F=81, and R=32-19=13, which causes the robot to turn left. When this robot encounters a power source located within its environment, it can be easily determined that the robot turns right towards it. Given the location of the power source sensor, this right turn action directly leads to a charging position (since the power source is now directly ahead of the robot).

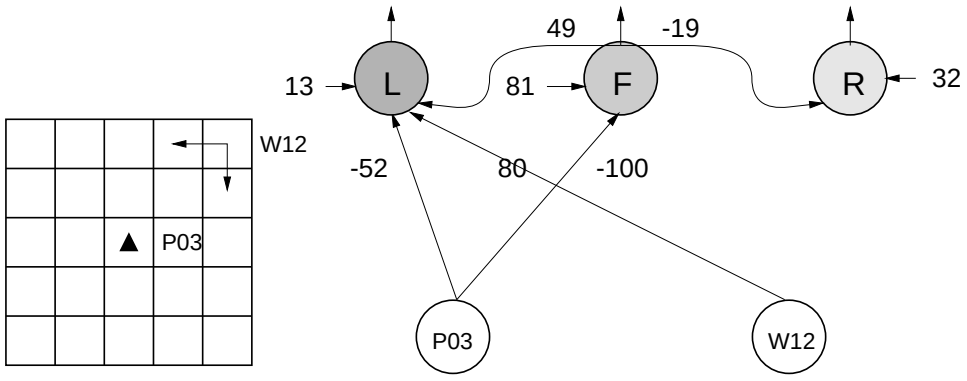

Figure 5.9 Best robot design evolved in environments with random power source locations.

A really interesting behavior arises when the robot turns (left or right) and detects a wall (W12=+1) and a power source (P03=+1) at the same time. This leads to network activations of L=13-52+80-49=-8, F=81-100=-19, and R=32+19=51. The robot then turns right, towards the power source. It can also be verified that at the next time step the robot attempts to move forward into the power source. Since power sources can possibly be on walls, this leads to a situation where the otherwise wall-eluding robot bumps into the wall. It is no surprise then that this robot bumps into walls an average of 2.69 times, i.e., once every 90 time steps ($\frac{241.7}{2.69} = 90$). On an average, the robots in this experiment bump into walls more often than their counterparts in the earlier experiments.

It thus appears that even though the power sources are in unpredictable locations, evolution

often retains power source sensors in the robot designs to allow them to detect and respond to power sources. However, the robots can only use this feature to approach power sources if they have an appropriately positioned power source sensor to detect them. What would happen if the robots had additional mechanisms to remember the locations of power sources they encountered? The following experiments attempt to address this question.

5.9 Need for Spatial Learning

Although our experiments have shown that evolution can discover designs well adapted to the constraints of the task environment, it cannot anticipate the dynamic (or unpredictable) aspects of the environment. For instance, in Section 3.3.7 we demonstrated the inability of evolution to anticipate or predict the locations of the boxes, and its consequences on box-pushing fitnesses. Similarly, if the robots have to function in different environments, power source locations might also be unpredictable. We have argued earlier that such unpredictable or dynamic environments require the agents to possess abilities to learn and adapt. In the context of using power sources effectively, we require the robots to possess *spatial learning ability*, i.e., the ability to learn, remember, and navigate to the locations of power sources found during the performance of their task.

In order to test whether spatial learning ability adds to the box-pushing performance of the energy-constrained robots, we evolved robots that had a spatial learning mechanism built-in. As we will describe shortly, robots were born with a spatial learning mechanism that they could use to learn, remember, and navigate to the power sources in their environments.

5.9.1 Spatial Learning Model

The experiments described in the following sections assume that the robots have a *spatial learning* mechanism built-in. For reasons that will become clear in Chapter 7, this spatial learning model requires the robot to have a dead-reckoning mechanism. As we mentioned earlier, dead-reckoning is the ability of animals (and robots) to maintain and update estimates of their position in the environment by integrating linear and angular acceleration and velocity signals generated by their motion (Gallistel, 1990; Everett, 1995). Hence we provided our box-pushing robots with the ability to maintain estimates of their position. These estimates

were assumed to be specified in terms of Cartesian coordinates, which effectively correspond to cells in the box-pushing environment.

In addition, the spatial learning model required the robots to possess *power source sensors* (P) since power sources could only be detected (or sensed) by these sensors. With these features the robots could learn the locations of power sources as follows. Suppose the robot was executing its task and a power sensor detected a power source. Since the robot was assumed to have accurate estimates of its own position in the environment and it knew the placement and range of the power source sensor, it could easily compute the position of power source in the environment. The spatial learning model then simply memorized the location of this power source.

The robots made use of two thresholds – *low battery threshold* (LBT) and *high battery threshold* (HBT), which dictated the robot behaviors at any given instance of time. If the robot's battery level fell *below* LBT, the robot switched from its box-pushing behavior to a *power seek* behavior. In this mode the robot simply consulted its spatial memory to identify the location of the *nearest* power source. Once this was determined, the robot switched off all its sensors (to conserve energy) and navigated to the remembered location of the power source. Since the box-pushing robots operate in a grid-world, navigating to power sources was realized through a modified *X-Y* routing mechanism. For instance, suppose the target power source was at the location (x_G, y_G) and the current position of the robot was (x_R, y_R). The robot turned appropriately and first navigated from its current row (x_R) to the row that contained the goal location (x_G). Once at the correct row, the robot turned and navigated towards the correct column (y_G). In this navigation phase the robot did not employ any of its usual sensors and we assumed that it used simple *touch sensors* to avoid obstacles on its path. This mechanism reduced the drain on the robot's already depleted battery.

It was quite possible that the navigation path followed by a power-seeking robot contained obstacles such as boxes, walls, or even other power sources. Our navigation algorithm assumed that the robot could detect these obstacles via its touch sensors. Once detected, the robot avoided them by performing two to five random actions and then reinitiating the X-Y routing algorithm. In addition to being simple, this navigation mechanism only required *local* knowledge; it not need any extra information pertaining to the robot world like the positions of other boxes, power sources, etc., which would be required by other approaches to path or

motion-planning.

Once the robot reached the target power source, it maneuvered itself into a charging position (directly facing the power source). Once in this position, the robot battery charged by unitCharge at every time step. The value chosen for HBT determined the amount of time spent by the robot in this charging position. In effect, the robot spent enough time at the power source to charge its battery up to HBT. Once this was done, the robot resumed its box-pushing behavior (from its current location rather than the location where it had suspended box-pushing to embark on a power seek behavior).

It must be noted that the magnitudes of LBT and HBT are rather critical. A low value of LBT may lead to a situation where the robot runs out of power while navigating to the power source, while a high LBT value will lead to a robot that frequently engages in power seek behaviors. Neither of these situations are favorable. Similarly, since the value of HBT governs the amount of time spent at the power source, it critically affects the distribution of the robot's limited time between box-pushing and power seek behaviors. High HBT values force the robot to spend a considerable fraction of their valuable time at the power source. However, it is not clear that low HBT values are preferable. Indeed, if the robot does not charge its batteries enough, it will demonstrate power seek behaviors with nagging frequency. Thus, LBT and HBT values must be carefully chosen. In the following we present results from two experiments, one where the LBT and HBT values are chosen by us, and the other where evolution determines appropriate values for them.

5.9.2 Fixed Thresholds in Spatial Learning

As in the earlier experiments, the robots in this experiment had a limited battery capacity of maxEnergy = 1000, which they could replenish at a power source by charging at a rate of unitCharge = 50 every time step. These robots also had the spatial learning mechanism described in the previous section, with the two thresholds set to: LBT=100 and HBT=1000. These values were chosen for the thresholds based on the following arguments.

We reasoned that LBT should be low enough to allow the robot to spend a large portion of its time in pushing boxes. However, it should not be so low as to cause the robot to run out of power before it reaches a power source. Since the robots switch off all their sensors before navigating to the remembered position of the power source, each robot step in this

mode consumes only 5 units of energy (only the robot motion consumes energy). As the robots operate in 10× 10 environments and cannot move diagonally, the farthest possible power source (remember there are two) is less than 20 steps away. Thus, the robot requires 20×5=100 units of energy to reach the power source (without any obstacles on the way). This motivated the choice of LBT=100. Further, if the robot indeed decides to seek power, we reasoned that it should charge to the battery's full capacity in order to reduce the likelihood of future power seek behaviors. Hence we chose HBT=maxEnergy=1000.

Table 5.5 shows the results of our experiments. It is immediately obvious that the robot fitnesses have increased substantially. In fact the best robot in this experiment is 68% more fit than the best robot evolved in this environment without spatial learning (Section 5.8). The survival time can also be seen to increase dramatically. This shows that the ability to learn, remember, and navigate to power sources critically affects the performance of the robots. A side effect of this ability to maintain energy for a relatively long period of time is reflected in the average number of sensors used by these robots (3.1), which is much more than those used by robots without the spatial learning mechanism. In fact, the best robot discovered by evolution employs five sensors.

Table 5.5 Results with fixed thresholds in spatial learning.

	Average	Best
Fitness	5.03	6.26
Survival Time	394.93	362.00
Wall Bumps	6.47	3.52
Hidden Units	0.30	0.00
Sensors	3.10	5.00

Figure 5.10 shows the distribution of sensors in the evolved robots. It is easy to note that these robots often employ power source sensors. This is natural given the constraints of our spatial learning model, which cannot learn power source locations without power source sensors for detecting them. It can also be noticed that if a robot employs power sensors, it is likely to have that sensor placed to sense cells (1, 1) or (1, 5). While (1, 1) is two cells in front of the robot, (1, 5) is two cells directly behind the robot. This is an interesting design because it allows robots to also detect power sources that happen to be behind them.

Similarly, it can be observed that wall sensors are likely to be positioned at either (1, 0) or (1, 2), both two cells ahead of the robot. Unlike the designs evolved in the earlier experiments in this chapter, the robots in this experiment often use box sensors. As may be observed in Figure 5.10, these sensors are typically placed to detect boxes immediately ahead of the robot (0, 1). Our data also indicate that on an average, 1.18 power source sensors, 1.1 wall sensors, and 0.68 box sensors are used by each evolved robot. Thus, with a spatial learning mechanism at their disposal, not only do the robots evolve to use more numbers of sensors, but also employ more power source sensors than wall or box sensors. This is quite different from the designs presented earlier in this chapter where the robots largely made use of wall sensors. Thus any change in the environmental properties, in this case the ability of the robot to learn locations of power sources, is quickly exploited by evolution to produce designs of better adaptive value.

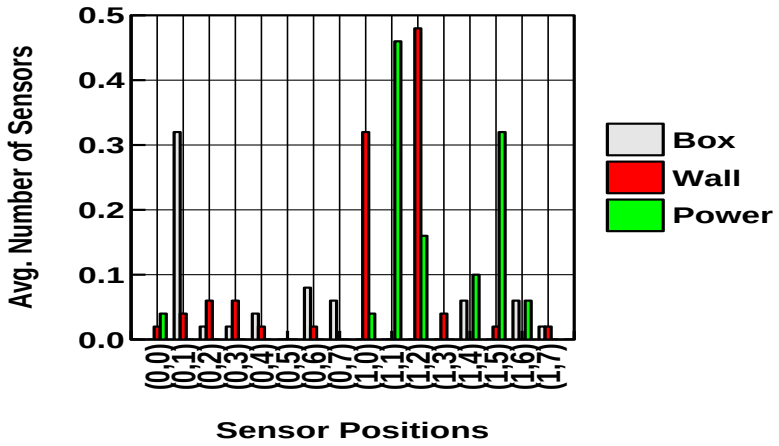

Figure 5.10 Distribution of sensors in robots evolved with the spatial learning mechanism and fixed thresholds.

The best robot discovered by evolution has a fitness of 6.26 and is shown in Figure 5.11. This robot makes use of 5 sensors, including 3 box sensors, 1 power source sensor, and 1 wall sensor. It should be noted that the power source sensor looks for power sources two cells behind the robot. It can also be verified that the box sensors exert direct control over the robot behavior. Thus, while B07 coerces the robot into a left turn, B01 moves the robot forward, and B04 induces a right turn. It can also be observed that boxes detected by B01 exert the maximum influence, and the robot moves forward in such cases even if other box sensors detect

boxes. As with the earlier designs, this robot too responds to walls detected by W10, turning right in such cases. Although sensory inputs from P15 are used by the neurocontroller, we believe that the true evolutionary value of these sensors stem from their critical role in spatial learning, namely, the inability of the robots to learn power source locations without detecting them using power source sensors (Section 5.9.1).

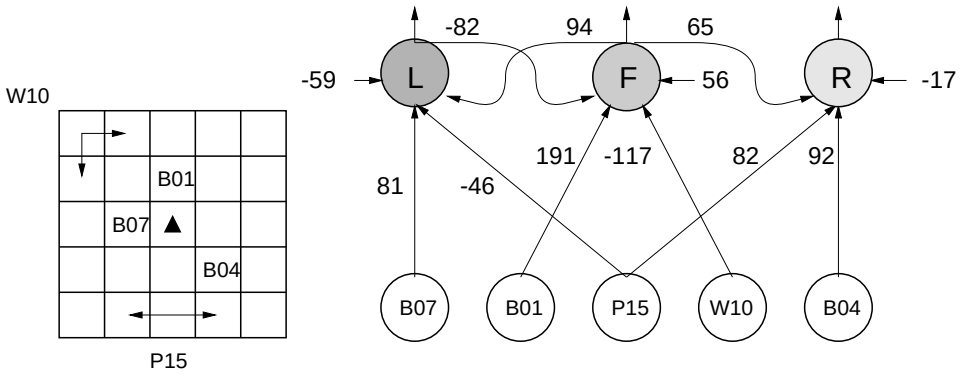

Figure 5.11 Design of the best robot evolved with fixed thresholds in the spatial learning mechanism.

5.9.3 Evolving Thresholds for Spatial Learning

In the experiments described in the previous section, the thresholds LBT and HBT were fixed at *a-priori* chosen values. Although we put some thought into choosing appropriate values, the question remains whether these are the best choices possible. Given the ability of evolution to co-evolve designs optimized across multiple dimensions, there is reason to suppose that it can optimize these thresholds to work well with individual robot sensory systems and neurocontrollers. In order to do this, we augmented the genetic representation described in Section 3.2.2, with two integer genes encoding the thresholds LBT and HBT respectively.

These new genes were also subject to the effects of crossover and mutation. While crossover allowed the offsprings to inherit both thresholds from one parent or one from either parent, mutation altered the thresholds by a random amount between 0 and 1000. The genetic operators were constrained to ensure that HBT was always greater than LBT.

Results of these evolutionary experiments are summarized in Table 5.6. From the table it can be readily inferred that though the robot fitnesses only improve marginally (about 5%)

over robots with fixed thresholds, there is a substantial increase in survival time (about 18%). However, these robots still do not survive through their quota of 500 time steps. An analysis of our experimental data indicates that in many environments the robots do indeed survive their entire quota. However, there are environments where the box-pushing behavior of the robot leads to situations where the power sources become inaccessible (e.g., both power sources on walls and boxes pushed against them). This leads to extremely short-lived robots in such trials. As a result, the robots only survive for approximately 466.30 time steps on average.

It should also be noted that the average LBT values of the evolved robots is 384.36 and that of the best robot is 326. These are quite different from the LBT value of 100 that we used in the previous experiment. Similarly, the evolved HBT values are different from the value of 1000 we had chosen earlier. Since the only difference between the two experiments was in the choice of thresholds (fixed versus evolved), the ability of the robots to survive longer in this experiment, can be directly attributed to the evolved threshold values.

Table 5.6 Results with evolved thresholds in spatial learning.

	Average	Best
Fitness	5.25	6.39
Survival Time	466.30	445.19
Wall Bumps	12.35	5.29
Hidden Units	0.04	1.00
Sensors	2.42	5.00
LBT	384.36	326.00
HBT	793.10	918.00

The best evolved robot has a fitness of 6.39, uses 5 sensors and 1 hidden unit, and is shown in Figure 5.12. As the sensors only provide values of +1 and 0 to the neurocontroller, the hidden unit always produces an output of +1. This simply has the effect of modifying the threshold of F and R units to 47 and -111 respectively, biasing the robot towards forward moves in the absence of other sensory inputs. Notice that this robot does not have any means to detect walls two cells away. So how does this robot detect boxes against walls? How does it know when to stop pushing a box? It turns out that in this case the robot does not have any automatic means to quit pushing the box. It relies on the feedback mechanism described

in Section 4.1.1, to guide it out of trouble.

The role of the power sensor P10 should also be noted. This sensor does not directly affect robot behavior as it does not provide any form of input to the neurocontroller. However, this sensor is not useless as it contributes to the detection and learning of power source locations.

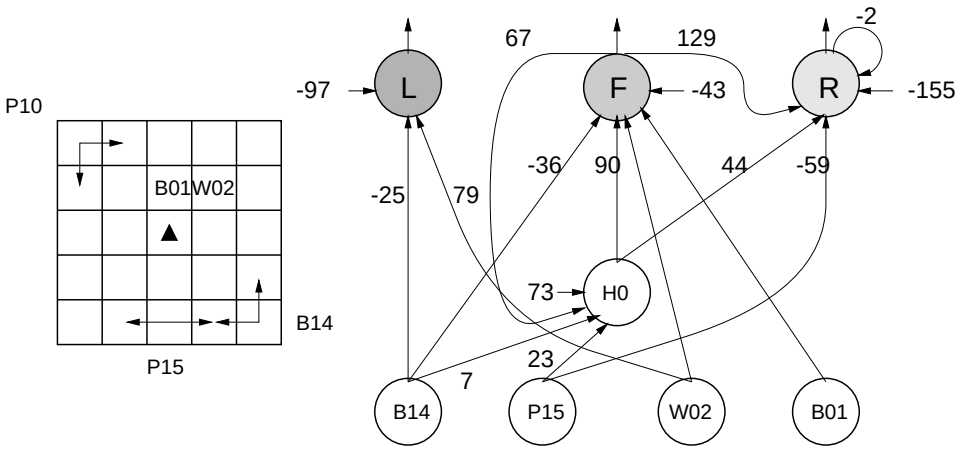

Figure 5.12 Best robot design co-evolved with the thresholds. LBT=326 and HBT=918.

The distribution of sensors in the evolved robots, shown in Figure 5.13, can be seen to parallel that of robots in Section 5.9.2. Again the most likely placement of power source sensors appears to be either position (1, 1) or (1, 5) (two cells ahead or behind the robot). Similarly the preferred position of wall sensors appears to be (1, 2), while box sensors, although few in number, are preferentially placed at (0, 1). On an average, these robot designs employ 1.16 power source sensors, 1.02 wall sensors, and a mere 0.22 box sensors. Thus, given the environmental constraints, the robots appear to benefit from power source and wall sensors rather than box sensors.

5.10 Related Work

Nolfi et al. (1994) described a simulation of a foraging task where the agent (or robot) had the task of finding food. However, in addition to this foraging task, the agents also *predicted* the next position of food based on the present position of the robot and its intended movement. Like the robots in the box-pushing task, these robots too, occupied a two-dimensional grid and

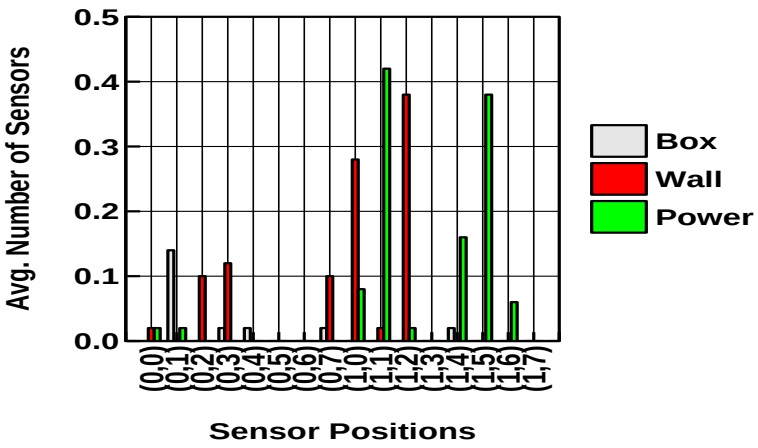

Figure 5.13 Distribution of sensors in robots with evolved thresholds.

were capable of moving forward or turning right or left. The robots had two sensors that provided them with information regarding the distance and angle of the *nearest* food element. The food elements themselves were randomly distributed in the environment and occupied entire cells. When the robot moved into a cell occupied by a food element, it was said to have consumed the food and the food disappeared. The robot behaviors were generated by a *strictly feed-forward* network that contained four input units, seven hidden units, and two or four output units. While two input units provided sensory information (distance and direction of the nearest food element), the other two units provided the network with the currently *planned* and as yet *unexecuted* robot action. Based on these inputs the network produced an output action, which was the action that the robot would execute at the *next* time step. Thus, this was the *planned* action, while the action executed in the current time step was the one that was planned at the previous time step. Thus, in their system, the robot had a planned action for time step t_0. It obtained sensory inputs at t_0 and determined a planned action for t_1. It then executed the action planned for t_0, which led to new sensory inputs at t_1. And so on.

The network had four output units, two corresponding to the action being planned for the next time step and the other two corresponding to the prediction of sensory inputs at the next time step. Based on the error between predicted and observed sensory inputs, the network

was trained using the back-propagation algorithm. Since the subnetworks contributing to sensory prediction and action generation shared hidden units and links, learning to predict also influenced the planning of future actions.

The researchers found that evolution and learning affected each other dynamically, with populations having the ability to learn to predict food locations, demonstrating better evolutionary increase in the average ability to find food in the environment. They also found that learning to predict improved the ability to find food in the environment, and that evolution produced individuals that were predisposed to learn.

The box-pushing task differs from this work in a few critical ways. As we have argued earlier, the box-pushing task is more complex than food-approaching. Also, the box-pushing robots had extremely limited sensory ranges, unlike the foraging robots in the experiments of (Nolfi *et al.*, 1994b). Indeed, if the location of the nearest food element is directly available, a simple, manually developed program can guide the robot to the food element. Such programs are very hard to develop manually for the box-pushing task owing to the many constraints inherent in the task formulation. Another related observation concerns the *lack* of significant difference in the peak fitnesses observed with and without learning in their experiments. In fact, it appears that the best non-learning network performs better than the best learning network. This result seems to suggest that there is little need for learning in their task, an observation that must be explored in more detail.

In another related effort, Cecconi et al. (1995) used the *Latent Energy Environment* (LEE) of (Menczer & Belew, 1994; Menczer & Belew, 1995) to study the interactions between evolution and *imitative learning*. LEEs provide a tightly controlled environment for studying and characterizing evolved behaviors. The creatures (or agents) in LEE move around in their environments. Atoms of different kinds appear randomly at different cells in the environment. These atoms produce differing amounts of energy when consumed by the agents. The agents possess ambient as well as contact sensors, that provide real-valued inputs to a neural network. The network produces binary outputs, that are interpreted into actions of moving forward, turning left or right, and stand-still.

The network used in the simulations of Cecconi et al. contained 4 input units, 7 hidden units, and two output units. The four input units derived sensory inputs from one contact sensor immediately ahead of the agent, an ambient sensor of range 5 units to sense in front of

the agent, and two ambient sensors observing 3 units to the left and right of the agent. In their study with imitative learning, the offsprings produced by the agents were taken care of by their mothers until they came of age or *matured*. This maturity age was governed by a separate gene. Thus, individuals with differing maturity ages evolve. When an offspring was born, its mother carried it on her shoulder. The offspring then learned to associate specific actions with sensory inputs, based on the actions it observed its mother perform. This learning was implemented using back-propagation. This imitative learning continued until the offspring matured, at which point it separated from its mother and went its own way. Immature offsprings were incapable of moving on their own or reproducing.

With such a set up, the researchers studied the evolution of a maturing age that was controlled by the maturity gene. It must be noted that since the learning of the agents took place only during immaturity, short maturity age interfered with learning while long ones left little room for effective reproduction and propagation of genes. Their results suggested that *delayed maturation* evolved due to its benefits of learning, even though it imposed a *cost* on the mother (Cecconi *et al.*, 1995).

Although a very interesting study, this work implemented an extremely simple form of neural learning. In contrast, the need for spatial learning in our box-pushing agents is much more complex. Further, as we have mentioned earlier, effective box-pushing behaviors are hard to develop manually, unlike the survival task used in the experiment of (Cecconi *et al.*, 1995).

5.11 Discussion

In this chapter we have explored a number of avenues related to the design of energy-efficient robots. In contrast to the robots evolved in earlier chapters, these robots have to make do with a battery of limited capacity. Robot designs evolved with these constraints demonstrate a flair for minimality with the robots barely using one sensor and the neurocontrollers discarding all their hidden units. Even though pushing a box is expensive in terms of energy required, these robots develop box-pushing skills and compromise, instead, on the use of sensors and neurocontroller units.

When the robot environments contain power sources and the robots have mechanisms to charge their batteries at these sources, evolution produces robot designs that sometimes use power source sensors to detect and approach these sources. However, if the power sources

appear at different locations in each of the robot environments, evolution cannot predict the locations of these power sources and the robots are unable to survive any longer than their counterparts operating in environments without power sources.

In order to cope with such unpredictable and dynamic environments, the robots need spatial learning mechanisms. When the robots had access to a mechanism to detect, learn, remember, and navigate to power sources, their fitnesses improved considerably. So did their survival time. This was made possible by their ability to support more numbers of sensors and hidden units, thereby making them capable of displaying more complex behaviors.

This chapter demonstrates the ability of evolution and learning to complement each other in useful ways. For instance, evolution helps in the discovery of effective robot designs and behaviors while spatial learning allows the robot to learn and respond to novel spatial environments. Together they work better than either one alone. But how do animals and robots actually learn aspects of their spatial environments? How do they *localize*, i.e., how do they recognize where they are in their environments? How do they use learned spatial information to navigate? The remaining chapters in this dissertation explore answers to these fundamental questions regarding animal spatial learning and navigation, and their implications for mobile robots.

6 SPATIAL LEARNING IN ANIMALS AND ROBOTS

The ability to acquire a representation of the spatial environment and the ability to localize within it are essential for successful navigation in *a-priori* unknown and dynamic environments. Since animals and robots both occupy largely similar kinds of environments (natural or man-made), they must both possess mechanisms to learn, localize, and navigate purposefully within their spatial habitats. Considerable research effort has been expended on issues dealing with spatial learning, representation, and navigation in animals and mobile robots. In this chapter, we outline the basic processes involved in purposeful, goal-directed navigation, and show how contemporary robots and animals (primarily rodents) address these issues. This provides a context for comparing and evaluating robot and animal spatial learning behaviors, and sets the stage for exploring useful synergies of these behaviors.

6.1 Introduction

The ability to successfully navigate in a wide range of natural environments is essential to the survival of animals. Mobile robots need to be equipped with similar capabilities in order to perform the tasks expected of them in natural or man-made environments. This requires them to be able to acquire and use adequate representations of their spatial environments. Animals offer compelling existence proofs of such capabilities that have evolved in nature (Anderson, 1983; Schone, 1984) and challenge us to explore information processing mechanisms and computational architectures that can match their functionality, although they might be realized using different physical substrates and perhaps different design and performance constraints.

There is a large body of neuroanatomical, neurophysiological, and behavioral data on the possible role of different parts of the brain in general, and the *hippocampal formation* in particular, in spatial learning and navigation in animals (O'Keefe & Nadel, 1978; Churchland & Sejnowski, 1992). This has led to the formulation of a number of computational models and

accounts of these spatial behaviors. *Computational models*, although often simplified caricatures of their biological counterparts, provide an attractive approach to organizing, analyzing, abstracting, and exploring the implications of such data. In addition to suggesting new experiments designed to fill the gaps in our understanding of the systems being modeled, they are also very useful as sources of ideas for building artificial systems with comparable abilities. Against this background, a number of biologically-inspired models of spatial learning, localization, and navigation have been proposed and implemented in robots and other artificial automata (Kuipers & Byun, 1991; Mataric, 1992; Kortenkamp, 1993; Kuipers *et al.*, 1993; Bachelder & Waxman, 1994; Wan *et al.*, 1994; Nehmzow, 1995; Blum & Abbott, 1996; Redish & Touretzky, 1996; Recce & Harris, 1996; Sharp *et al.*, 1996).

On the other hand, those involved in the design of autonomous robots are necessarily faced with multiple design and performance constraints imposed by the available technology and the task environments. Attempts to address the attendant engineering challenges in the design of such systems have led to the development of a broad range of mathematical and computational tools. These include information-theoretic characterizations of sensory and system complexities (Elfes, 1995; Koenig *et al.*, 1995), algorithms for the integration and use of noisy sensory data from multiple sensors (Ayache & Faugeras, 1987; Moutarlier & Chatila, 1989; Leonard & Durrant-Whyte, 1992; Hummel, 1995; Pagac *et al.*, 1995), and probabilistic localization approaches for mobile robots (Smith *et al.*, 1990; Crowley, 1995; Hebert *et al.*, 1995). Use of such tools to analyze biologically inspired models can often yield new insights into the capabilities and limitations of the underlying information processing structures and processes (Levy, 1989; Linsker, 1990; Treves & Rolls, 1991; Treves & Rolls, 1992).

The preceding discussion suggests that biologically inspired modeling efforts and the design of autonomous mobile robots can each benefit from the results and tools developed by the other. Against this background, we have developed a biologically-inspired model of spatial learning and localization and analyzed it using information fusion and probabilistic localization tools from robotics. The resulting model contributes to both rodent spatial learning and robot navigation, as will be clarified in the following sections and chapters. Before we proceed with the development of the model, we will pause briefly to consider different aspects of spatial learning, localization, and navigation in animals and robots.

6.2 Spatial Learning, Localization, and Navigation

Both animals and autonomous mobile robots need mechanisms to navigate *purposefully* in *a-priori* unknown or partially-known environments. However, for such behaviors to be possible, they must be endowed with mechanisms capable of answering the following questions (adapted from Levitt and Lawton (1990)):

1. Where am I? *Localization*
2. Where are other places relative to me? *Spatial map*
3. Where is the goal? *Goal determination*
4. How do I get to the goal from here? *Path planning*
5. How do I acquire places and goals? *Spatial learning*

As might be expected, the answers to these rather straightforward questions are intricately intertwined. The first question is concerned with the identification or recognition of the current place, a problem commonly referred to as *localization* in robotics. To aid localization, places must be represented and remembered in terms of sensory features that allow the animal or robot to quickly and *unambiguously* recognize different places. Quite often this is a problem owing to *perceptual aliasing* in the environment. An artifact of the environment and/or sensor limitations, perceptual aliasing causes multiple places in the environment to appear sensorily identical, thus interfering with map learning and localization. Any mechanism that aids in the resolution of such ambiguities is thus of interest, as will be clarified later.

The second question deals with the representation of the spatial environment, which we refer to as a *spatial map*. This map encodes the relationship between places in the environment and could contain *topological*, *metric*, or *directional* information. It must be clarified that the map need not be *topographic* (e.g., maps in an atlas), as long as it *functionally* captures the relationships between places. Often it is also possible (and beneficial) for the map to encode multiple kinds of relationships (e.g., topological, metric, etc.) at differing levels of abstraction.

The third question is concerned with *goal determination* and includes the acquisition, representation, and future identification of goals. Indeed, if the desired goal cannot be chosen based on the current information state (sensory and memory representation) of the animal

or robot, goal-directed navigation is not possible. Goals may be represented in a number of ways, for instance, using visual cues, proximity relationships to significant landmarks, metric positions in a coordinate space, etc., (Schone, 1984; Trullier *et al.*, 1997).

The computation of the navigation trajectory is the answer to the fourth question, which is referred to as *path planning* in the robotics literature (Latombe, 1991; Hwang & Ahuja, 1992). The navigation trajectory depends on the nature of information encoded in the map (topological, metric, etc.), the specification of the goal, and importantly, on the kinds of computational mechanisms that the robot or animal can afford. The trajectory may also be optimized for constraints such as the minimum distance to the goal, the safest path to the goal, the least travel time to the goal, etc. Assuming the availability of appropriate computational resources, each representation scheme used in the spatial map entails a different goal navigation strategy (Schone, 1984; Trullier *et al.*, 1997). For instance, if goals are represented by their positions in an underlying coordinate frame and the animal or robot knows its current position in this frame, the shortest trajectory to the goal is simply a vector-difference of the two positions.

If the animal or robot has reliable answers to these four questions, it can navigate in a purposeful manner to arbitrary goals. Further, if the environments are *a-priori* known and static, spatial maps can be pre-specified and goals can be pre-determined, making purposeful navigation a simple matter of *genetically programmed behaviors* in animals and *pre-wired control* in robots. However, the real-worlds occupied by animals and most robots are dynamic and at best *partially-known*. In such cases, animals and robots must possess mechanisms to *learn* spatial maps of the environments they encounter and *adapt* to dynamic changes within them. They must be capable of recognizing, learning, representing, and updating places and goals, i.e., they must possess *spatial learning* ability.

In our research we have only focused on spatial learning (including building of spatial maps) and localization aspects of goal-directed navigation. Where necessary, we have augmented the system with simple algorithms for learning, representing, choosing, and navigating to goals. We now briefly consider our current understanding of how these processes of spatial learning and localization are realized by contemporary robots and animals.

6.3 Representation of Spatial Information in Robots

Contemporary robots represent spatial information in one of two broad ways: *location-based (metric)* or *relation-based* (Moutarlier & Chatila, 1989), although some recent approaches have attempted to combine merits of the two (Levitt & Lawton, 1990; Kuipers & Byun, 1991; Kortenkamp, 1993; Thrun, 1996). In location-based schemes places (or locations of objects) are all represented in the *same* coordinate frame with a *global origin*. This origin may be absolute in the space or chosen appropriately by the robot. In either case the robot learns and represents the locations of places and objects with respect to this origin. With such a scheme, the direct short-cut path between any two arbitrary places can be easily determined by simply comparing their metric locations in the common frame. However, local relationships between adjacent places are much harder to extract and require some computation on their metric representations. For instance, if we need to know the place that lies north-west of place P_1, we practically have to search through the entire map to identify the place that comes closest to being $P_1's$ north-west neighbor. Thus, location-based representations make it easy to determine the relationship between two given places, but make it hard to determine places that are in specific local relationships.

Relation-based approaches, on the other hand, represent *local relationships* between places, thereby easily capturing adjacency relations. With these schemes, it is quite straightforward to determine the place that is in a specific local relationship to another (e.g., which place is north-west of place P_1). However, since there is no global coordinate frame representation, the relationship between two arbitrary places can only be determined in terms of the relationships of the intermediate places. Depending on the size and form of the spatial map, this might require cumbersome computations.

Let us provide examples to illustrate these two spatial representation schemes.

6.3.1 Occupancy Grid Representation of Metric Spatial Information

An example of metric spatial representation in robots is the *occupancy grid* (Moravec & Elfes, 1985; Elfes, 1989) which encodes the environment using a grid-like decomposition. Each grid cell corresponds to a portion of the environment and adjacent grid cells represent adjacent physical regions in the environment. The grid cells are associated with a figure of merit called

the *probability of occupancy*, which is updated based on sensory inputs and appropriate sensor models (Elfes, 1989). Since grid cells correspond to regions of physical space, the probability of occupancy is an indicator of the presence or absence of objects in the corresponding region of physical space. These occupancy grids are useful for determining *free-space* for uninhibited robot navigation and can often be easily learned from *sonar* data. *Inference grids* are recent generalizations of occupancy grids that can be used to learn, represent, and manipulate different kinds of *features* (e.g., shape, color, texture, etc.) in addition to just occupancy (Elfes, 1992; Elfes, 1995). Since the grids provide a complete map of objects and obstacles in the explored portion of the environment, an obstacle-free navigation trajectory can be computed to arbitrary goal locations on the map. However, since the grid represents the entire environment and not just the significant places, space is stored in an inefficient manner. Further, after each sensory measurement, the robot must update the occupancy probabilities of the entire grid (feature probabilities in the case of inference grids), making it computationally expensive to maintain these maps. Also, increasing the grid-cell resolution (smaller grid cells) improves the accuracy of representation but results in a quadratic (for 2D maps) increase in the number of grid cells required for the same Decreasing the grid-resolution, on the other hand, reduces the computational burden on the update algorithm but results in a loss of information and a consequent increase in uncertainty (Elfes, 1992).

6.3.2 Topological Maps for Representing Spatial Information

A popular example of relation-based spatial representations is the *topological map* which only represents *distinctive places* in the environment along with the local relationships between them (Kuipers, 1978). This map can thus be thought of as a graph where nodes represent distinct places and the edges denote the relationship between them (Brooks, 1985). Usually, distinctive places are characterized using *sonar signatures* (Kuipers & Byun, 1991; Mataric, 1992; Kortenkamp, 1993), *image signatures* (Kortenkamp, 1993; Kortenkamp & Weymouth, 1994; Engelson, 1994), *panoramic views* (Tsuji & Li, 1993), *local occupancy grids* (Langley & Pfleger, 1995; Yamauchi & Langley, 1997), etc., depending on the kinds of sensors the corresponding robots possess. Local relationships usually represented in such maps include the *directional* relationship between places (Nehmzow, 1995) or the directional relationship compounded with *distance* information (Kuipers & Byun, 1991). Some topological schemes also

represent metric information pertaining to the distinctive place (e.g., length), which is useful if the place represents a wall or a corridor (Mataric, 1992). Such relation-based approaches produce a compact world representation since they only represent distinctive places and not the entire environment. Further, they are robust to global movement errors as they only represent local relationships between places (Brooks, 1985). Although these topological graphs can be easily learned through robot exploration, navigation to a goal requires a search through the graph to determine a *route* from the current robot location to the goal, which can be computationally expensive for large graphs.

6.3.3 Other Issues in Robot Spatial Learning

Given the attendant advantages and limitations of the two spatial representation schemes, *hybrid* approaches have been developed that complement each other in useful ways. This has led to the development of models that encode spatial information of multiple kinds at multiple levels of abstraction (Levitt & Lawton, 1990; Kuipers & Byun, 1991; Kortenkamp, 1993; Thrun, 1996). Some of these models are described in more detail in Section 9.3.

Although the spatial representations discussed above (and their many variants) are extensively used in contemporary robotics, a number of other issues have to be addressed before such representations are possible. An important question that must be addressed concerns the aspects of the environment to represent. Is a spatial scene sensed and represented simply as some form of a *signature* (e.g., a photograph-like snapshot) or is it represented as a *composite* of the individual objects (also called *landmarks, cues* etc.) making up the scene? Further, are *all* the landmarks in the scene treated equally or are some landmarks given priority over others? If they are indeed prioritized, what is the mechanism used to assign these priorities? What roles do landmark size, distinctiveness, function, stability, reliability, etc., play in spatial representation? Is the representation of space a single-form, monolithic entity or is it a distributed entity supporting spatial representations of multiple forms and at multiple levels of granularity?

These issues are intricately connected with (and influenced by) the sensory systems and the computational mechanisms available to the robot. In contemporary robotics these questions are largely resolved by human design choices based on the dictates of the sensors, their models, and the processing capabilities of the robot in question. For instance, *sonar sensors* emit an

ultrasonic pulse of energy and measure the time required for the pulse to reach a reflecting object and echo back to the receiver (Everett, 1995). Although these sensors can detect the presence of objects they are not capable of differentiating between (or recognizing) different kinds of objects. Further, they can only obtain rough estimates of the distances to objects, this estimate often being corrupted by *specular* (or false) reflections (Everett, 1995). Thus, a robot equipped only with sonar sensors cannot identify individual objects in a scene, rather it is forced to learn the boundaries of *objects* in the environment. Consequently, such a robot can only process its sensory inputs in the form of a *signature* (Mataric, 1992) and its spatial representation can only use information of this form.

Since animal spatial learning must also contend with similar issues, it is natural to ask how animals acquire, represent, and use spatial information. The following section provides some insight into these processes in animals.

6.4 Spatial Learning and Representation in Rodents

Animal learning in general, and spatial navigation in particular, have been the subject of intense study for many decades now (Tolman, 1932; Tolman, 1948; Hull, 1943; Hull, 1952; Mackintosh, 1983; Anderson, 1983; Schone, 1984; Gallistel, 1990). A large fraction of this effort has been devoted to the study of rodent navigation because experiments have indicated that space plays a dominant role in their behavior, making them relatively more willing subjects in spatial learning tasks (Hebb, 1949). An important aspect of rodent spatial learning appears to be their tendency to learn *places* rather than individual *stimuli*. For instance, Hebb (1949) trained rats to run to a dish of food located at the edge of an open table. Once the rats were sufficiently trained, the table (and the dish) were rotated by 90^o. It was found that the rats ran (at least once) to the prior location of the dish relative to the room even though the dish itself was visibly at a different position. In another experiment, rats were trained to jump from one platform to another to obtain a food reward. Once trained, the rats were found to jump into space when the second platform was moved to a different location (Hebb, 1949). Similar results confirming *place-learning* (as opposed to *cue-learning*) have also been observed by (O'Keefe & Nadel, 1978).

Experiments have also revealed that rats are capable of *detour behaviors* and *latent learning*. In detour behavior, rats trained to follow a particular trajectory to a goal show themselves

capable of adopting alternate trajectories (that are often *novel* and *optimal*) when the original trajectory is blocked or otherwise unavailable (Tolman & Honzik, 1930; Tolman *et al.*, 1946). Latent learning refers to the ability of rodents to acquire a spatial representation of the environment in the *absence* of explicit goals (Tolman, 1932; Tolman *et al.*, 1946).

Rats have also been found to successfully swim to a submerged platform in a milky pool of water, thereby demonstrating an ability to compute trajectories to *hidden goal locations* (Morris, 1981). Other related experiments have established that the rats appear to know how visual scenes are transformed by locomotion and are capable of computing approach trajectories using *inverse transformations* (Keith & McVety, 1988). Experiments with gerbils have led to the suggestion that these animals compute and store *vectors* to landmarks from the goal location. Further, *independent* vectors appear to be computed for each of the landmarks (Collett *et al.*, 1986). This has led to the suggestion of a vector-based representation of space, according to which a direct vector to a goal location can be computed by subtracting a vector from the goal location to a landmark from a vector to the same landmark from the current location (Collett *et al.*, 1986; McNaughton *et al.*, 1995).

Experiments have also provided many insights into the processing of sensory stimuli. For instance, it appears that rats give more importance to *remote sensory cues than to local ones*, possibly because remote cues are the *least-variable* objects in the environment (Hebb, 1949). It has also been found that the *stability* of stimuli (landmarks) is critical for spatial learning, i.e., even if a goal is constantly and reliably associated with a landmark, rats fail to capture this relationship if the landmark (and the goal) are *unstable*, i.e., they are moved to different locations in each of the learning trials (Biegler & Morris, 1996). Further, simply increasing the number of landmarks in the environment also does not help. The researchers concluded that spatial learning is critically influenced by *stable* relationships rather than merely the number or salience of landmarks (Biegler & Morris, 1996). Similar results have also been reported by (Bennett, 1993a). Other experimental manipulations have led to findings that *short* landmarks are often not remembered by the animals possibly because they tend to become *obscured* by intervening taller objects (Bennett, 1993b).

6.4.1 Theories of Animal Spatial Learning and Navigation

It is clear from the above discussion that many aspects of sensory stimuli processing have been studied in rodents via controlled experiments and their consequent effect on behavior. However, it is much harder to isolate, identify, or characterize the *representation* of these elements in the animal's brain and their use in navigation. Nevertheless, a number of theories of spatial learning, representation, and navigation have been put forth. Possibly the earliest such theory was Thorndike's suggestion that learning, in general, consists of an *association* between stimuli *(S)* and responses *(R)* (Thorndike, 1898). Thorndike also proposed the *law-of-effect*, suggesting that these stimulus-response *(S-R)* associations increase whenever the responses are followed by *satisfaction* and decrease with *discomfort* (Thorndike, 1898). Although this definition of learning applies to almost all major forms of learning phenomena (including classical and instrumental conditioning), in the context of spatial navigation it suggests that animals *learn* to perform specific responses *(R)* at particular places *(S)*, based on reward feedback. With such Thorndikian learning spatial navigation is then simply a matter of recognizing the place (given by sensory stimuli S) and performing the associated response R.

6.4.1.1 Cognitive Maps of Tolman

Based on extensive experiments with rodents Tolman concluded that animals demonstrate abilities for detour behaviors and latent learning (Tolman, 1948). He pointed out that Thorndikian framework of spatial learning falls short of explaining these behaviors in rodents. For instance, Thorndikian spatial learning requires a reinforcement at the goal location in order to learn the S-R associations. Hence this will not work in latent learning environments where there are no goals. Similarly, in Thorndikian learning animals learn to execute specific responses at particular places. However, if these responses happen to be unsuccessful, the animal cannot perform alternate responses because they have not been associated.

Tolman developed an alternate view of spatial representation and suggested that animals learn *expectancies* of the form $S_1 - R - S_2$ which captures the transformation of stimuli based on responses (Tolman, 1948). Unlike Thorndikian learning, Tolman argued that rewards are not necessary for learning these associations, which are simply learned based on *temporal contiguity*. Further, Tolman proposed the existence of an *inference process* that could combine and collate a large number of these expectancies into a semantic structure which he called a *cognitive*

map (Tolman, 1948). Since Tolman's thesis was that learning of space progresses devoid of rewards, the cognitive map theory implicitly supports latent learning. Further, it is clear from the structure of these expectancies that the animals not only associate responses with stimuli but also learn the *effect* of responses on stimuli. Thus, the cognitive map encodes *relationships* between places in the animal's environment. With such a representation, alternate routes can be easily identified if usual ones happen to be blocked. Although an important conceptual advance, the cognitive mapping theory was a non-rigorous, verbal specification, and Tolman provided little detail about the properties of such cognitive maps or the way animals build them.

6.4.1.2 Habit-Family Hierarchy of Hull

In contrast to Tolman's cognitive maps, Hull developed a rigorous and non-verbal mathematical extension of Thorndikian learning and used it to characterize maze learning in rodents (Hull, 1951). He proposed a further extension of this formalism that could explain detour behavior. According to this theory rats used Thorndikian learning to associate specific responses with stimuli, thereby learning one *habit*. Hull's primary contribution was the suggestion that rats learn a *family* of such habits, with different habits being initiated by the same stimulus. In short, the habit-family hierarchy suggests that rats learn to perform *multiple* responses at any given place, each leading to a possibly different behavior. With such habit hierarchies animals can not only represent routes *actually* followed but also alternate routes that could potentially lead to the goal (Hull, 1934a; Hull, 1934b). With such a spatial representation rats can easily choose alternate routes (detour behavior) when familiar routes happen to be blocked. Hull also provided an explanation for *latent learning*, suggesting that when the rats are allowed to explore the maze without any reward provided in the *goal box*, they still receive some minimal reinforcement when they are removed from the goal box. This, he claimed, was sufficient to learn S-R associations, although rather weakly. Later, when a reward is presented in the goal box, the *incentive motivation* changes abruptly, improving the goal-directed navigation behavior of the rat (Hull, 1951).

6.4.1.3 Locale and Taxon Hypotheses of O'Keefe and Nadel

O'Keefe and Nadel (1978) suggested that rats possibly use two spatial representation schemes (along with their associated navigation mechanisms): the *locale system* and the *taxon system*. The locale system corresponds to a cognitive map and represents spatial information in Euclidean terms. The locale system thus contains a spatial representation *"which does not depend for its existence on particular objects but which serves as a framework for relating these objects to each other independent of the observer"* (O'Keefe & Nadel, 1978). Since Euclidean space is metric in nature, the locale system captures a metric map of the spatial environment. The *taxon system*, on the other hand, is associated with stereotypic behaviors like route following and other orientation behaviors (O'Keefe & Nadel, 1978). O'Keefe and Nadel associate the locale system with Tolman's cognitive map hypothesis and the taxon system with sensory-motor relationships that underlie Thorndikian/Hullian spatial learning.

6.5 Path-Integration in Rodent and Robot Navigation

Path-integration, popularly known as *dead-reckoning*, refers to the process of updating an estimate of one's own position based on the knowledge of direction, speed, and time of self-motion (Gallistel, 1990; Everett, 1995). This usually involves integrating acceleration signals over time to obtain velocities, and the integration of velocity signals over time to obtain displacement vectors (Gallistel, 1990; Everett, 1995).

There is substantial evidence for path-integration in rodents, primarily in the use of dead-reckoning for *homing* behaviors (Etienne, 1992; Gallistel, 1990). For instance, gerbils have been found to return on a direct bearing to their nest after circuitous search trajectories, even in *complete darkness* (Mittelstaedt & Mittelstaedt, 1980; Mittelstaedt & Mittelstaedt, 1982). Further, based on observations from experimental manipulations like rotation of the entire arena or a linear displacement of the nest, the researchers concluded that path-integration is based on *inertial* directional information from the vestibular system and *ideothetic* linear information involving proprioception and/or efference copy from the animal's self-generated motion (Mittelstaedt & Mittelstaedt, 1980; Mittelstaedt & Mittelstaedt, 1982).

Similar experiments have also been performed by other researchers on golden hamsters (Etienne, 1985; Etienne, 1992; Etienne *et al.*, 1996), which demonstrate that without frequent

visual corroborations, path-integration systems rapidly accumulate errors. This has led to the suggestion that in the absence of visual information, path-integration appears to be useful only for short exploratory excursions from a known site (Etienne *et al.*, 1996). It has also been shown experimentally that in conflict situations between distant but familiar visual references and path-integration, the animals appear to give priority to stable visual references (Etienne *et al.*, 1990).

Researchers have also found some evidence for the grounding of internal spatial representations in a *locomotion-based metric system* in non-homing behaviors (Collett *et al.*, 1986). It is strongly suspected that these metric estimates are provided by the dead-reckoning system (McNaughton *et al.*, 1995; McNaughton *et al.*, 1996).

Dead-reckoning mechanisms are also used in robot navigation and a wide range of devices have been developed for this purpose (Everett, 1995). As with animals, dead-reckoning is usually involved in *homing* behaviors, where the robot returns to its *home-base* after executing its spatial task. However, dead-reckoning input has also been used in the building of metric spatial maps, with the dead-reckoning system providing a Cartesian coordinate representation of space. For instance, approaches for fusing *stochastic information* (e.g., Kalman filtering) have been used for robot localization and world modeling (Ayache & Faugeras, 1987; Moutarlier & Chatila, 1989; Crowley, 1995). In these approaches, dead-reckoning is used to provide estimates of robot position which are subsequently corrected based on observations. Recently, Yamauchi and Beer (1996) have defined a spatial representation scheme based on adaptive place networks where Cartesian position estimates derived from dead-reckoning are used to represent regions of physical space.

6.6 Discussion

In this chapter we have considered issues related to the processing and representation of spatial information in animals and contemporary robots. Since animals and robots both inhabit similar kinds of environments (natural or man-made), the environmental constraints operating on them remain largely the same. For example, physical phenomena like gravity, magnetism, friction, occlusion, light, etc., and their effect on perception and action, remain largely the same for navigating animals and robots. While animals appear to have evolved a variety of mechanisms to counter or handle these constraints effectively and efficiently, those

involved in the design of mobile robots are still grappling with many of these issues. Given this scenario, if we identify and understand the mechanisms involved in animal spatial learning and navigation, the possibility remains that we can develop equivalent mechanisms for mobile robots.

However, identifying and understanding biological structures and processes in animals is not an easy task. These difficulties have led to the development of modeling techniques which use biological data to build computational models that are mathematically or algorithmically specified and usually implemented on a computer. Such modeling approaches provide valuable tools not only for validating experimental observations but also as a source of ideas for designing biological experiments to further our understanding. A number of engineering and mathematical tools like linear algebra, topology, calculus, uncertainty modeling, probabilistic information fusion, entropy-based analysis, etc. have found their way into the domain of computational modeling and have contributed significantly to its development.

Although such a marriage between biological experimentation and computational modeling techniques can lead to useful insights and advances (as will become evident later in this thesis), it must be borne in mind that many a time such approaches may fail. This is largely because of the differences in the implementation media; the neuronal substrates in animal nervous systems and the largely silicon-based worlds of computers. There are also differences in the way the computations are inherently realized: animal nervous systems implement highly parallel and distributed electro-chemical computations while computers execute sequences of electronic instructions. These inherent differences may present hurdles in the complete understanding of biological processes and mechanisms, and in their realization in robots and other artificial automata. However, there is little doubt that much will be learned in the process.

7 HIPPOCAMPAL INVOLVEMENT IN RODENT SPATIAL LEARNING

We mentioned in the previous chapter that animals, particularly rodents, demonstrate a keen sense of space and are capable of learning novel environments and localizing accurately within them. How do they do it? Which part of their nervous system allows them to realize these behaviors? In this chapter we pool data from numerous sources to localize the spatial learning function (at least in rodents) to an area of the brain called the *hippocampal formation.* We summarize anatomical, neurophysiological, and behavioral data concerning the hippocampal formation and its role in animal spatial learning. We also discuss the suggestion that the hippocampal formation is the site of the Tolmanian cognitive map introduced in the previous chapter (O'Keefe & Nadel, 1978). The chapter concludes with a discussion of different computational models of hippocampal spatial learning that have been proposed and a brief examination of hippocampal involvement in memory in general.

7.1 Role of the Hippocampus in Rodent Spatial Learning

The hippocampal formation in animals has been strongly implicated in spatial learning based on evidence provided by two broad sources: *lesion studies* and *cellular recordings* from hippocampal cells. While lesion studies typically demonstrate the inability of hippocampus-lesioned animals to learn tasks of a spatial nature (e.g., mazes, object-place tasks, etc.), cellular recordings show correlated firings of hippocampal cells during the execution of such tasks. For instance, in the water-maze task (where rats have to swim to a submerged platform in a milky pool of water), rats with hippocampal damage are incapable of learning to navigate directly or efficiently to the hidden platform, although they appear perfectly capable of navigating accurately to a *visible* one (Morris *et al.*, 1982; Sutherland *et al.*, 1982). A number of other lesion studies are detailed in (O'Keefe & Nadel, 1978; Jarrard, 1993; Fenton & Bures, 1994). We

will focus more on the anatomical and physiological properties of the hippocampal formation since they shed more light on the neural structures and processes used in rodent spatial learning and localization.

7.1.1 Anatomy of the Hippocampal Formation

The hippocampal formation is an association area of the brain that receives highly processed sensory information from the major associational areas of the cerebral cortex (Cohen & Eichenbaum, 1993). As shown in Figure 7.1, these inputs arrive at a convergence area called the *entorhinal cortex (EC)*, which itself is a part of a major convergence area called the *parahippocampal cortical* area (Squire *et al.*, 1989). The hippocampal formation is composed of the *dentate gyrus (Dg)*, and areas *CA3* and *CA1* of Ammon's horn.

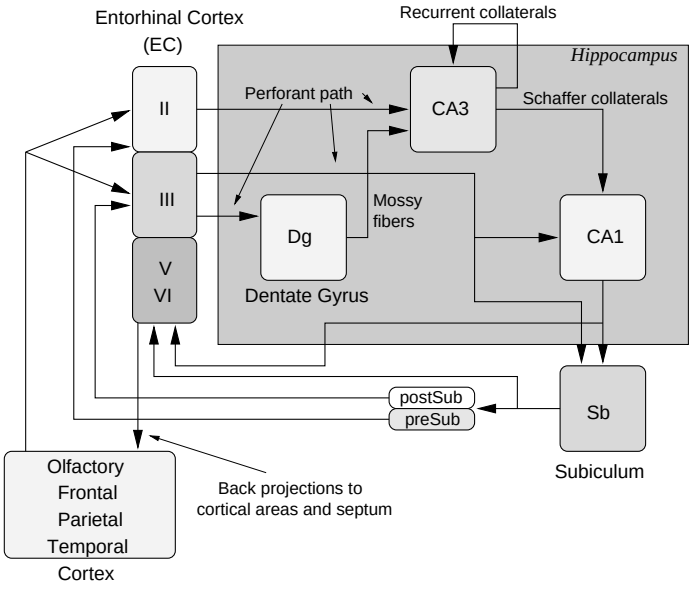

Figure 7.1 Anatomy of the hippocampal formation.

The dentate gyrus contains *granule cells* that receive input from the entorhinal cortex via the *perforant path*, and output to the CA3 via the *mossy fibers*. The mossy fiber synapses between the dentate gyrus and CA3 cells are *sparse* and *strong*, which has led researchers to suggest that the dentate gyrus provides a *context* (Rolls, 1990) or *reference frame* (O'Keefe, 1989) for spatio-temporal associations in the CA3.

The CA3 region of the hippocampus primarily contains *pyramidal* (or complex-spike) cells along with inhibitory interneurons like *basket cells, chandelier cells, and mossy cells* (Traub & Miles, 1991). These CA3 cells receive inputs from the entorhinal cortex through the perforant path, the dentate gyrus via mossy fibers, and recurrent inputs from other CA3 pyramidals. The CA3 cells are believed to represent spatio-temporal events or episodes by appropriately associating the sensory input available at the entorhinal cortex layer (O'Keefe, 1989; McNaughton & Nadel, 1989; Rolls, 1990). However, there are two schools of thought with regard to the role of the CA3 recurrent collaterals. While some have likened this structure to an *auto-associative* memory that serves as a pattern completion device (Marr, 1971; McNaughton & Nadel, 1989; Rolls, 1990; Rolls, 1996), others have ascribed a *hetero-association* function, suggesting that the recurrent collaterals *predict* future activations of CA3 units based on current activation and animal motion (Levy, 1989; Minai & Levy, 1993; Prepscius & Levy, 1994; Jensen & Lisman, 1996; Skaggs & McNaughton, 1996).

The CA1 region of the hippocampus too contains pyramidal cells and interneurons. However, unlike CA3 cells, CA1 cells do not project to other levels of CA1. The CA1 pyramidals receive inputs directly from the entorhinal cortex via the perforant path and from the CA3 pyramidals through the *Schaffer collaterals*. Axons from the CA1 pyramidal neurons project via the *alveus* to the *subiculum (Sb)* and also back to the entorhinal cortex. As will be explained in the following section, the CA1 pyramidal cells in rats have been observed to code for specific spatial locations visited by the rat. This has prompted researchers to label these cells as *place cells* since they appear to fire maximally when the rat is in a particular region or place (O'Keefe, 1976). In addition to such *cell effects*, the CA1 region also demonstrates two population effects in rats: *theta waves* and *sharp waves*. In rats, theta waves (low amplitude 4-8 Hz) have been observed during exploratory behaviors like sniffing, walking, rearing, and during REM sleep (dreaming). Sharp waves, on the other hand, are observed when the rats are sitting quietly, drinking, eating, grooming, or deeply asleep (Buzsaki, 1989).

The subiculum receives input from the entorhinal cortex and projects to the *pre* and *post-subiculum*, the deep layers of the entorhinal cortex, and to the hypothalamus, septum, anterior thalamus and the cingulate cortex (Churchland & Sejnowski, 1992). Recently, place cells have also been discovered in the subiculum (Sharp, 1996), although not much is known about them yet. There is also some evidence that the entorhinal cortex projects back to many of the

cortical association areas from which it receives input and mediates information storage there (Rolls, 1996). However, these mechanisms are not well understood.

We will now consider some physiological properties of hippocampal cells, that have been identified through intricate cellular recordings. Some of these properties are critical for understanding the design choices made in our computational model.

7.1.2 Physiological Properties of Hippocampal Cells

Cellular recordings from many regions of the brain, including the hippocampus, have revealed crucial properties of the underlying neuronal mechanisms. For instance, in their recordings from CA1 pyramidal cells of a rat hippocampus, O'Keefe and Dostrovsky found that the neurons were selectively active in particular regions of the environment of the moving rat (O'Keefe & Dostrovsky, 1971). These cells thus appear to code for specific places and have been accordingly named *place cells* (with *place fields* denoting the corresponding regions over which they are active) (O'Keefe, 1976). Since their initial discovery, cells with such location-specific firing have been found in almost every major region of the hippocampal system, including the entorhinal cortex (Quirk *et al.*, 1992), the dentate gyrus (Jung & McNaughton, 1993), the hippocampus proper (O'Keefe & Dostrovsky, 1971; O'Keefe, 1976), the subiculum (Barnes *et al.*, 1990; Sharp & Green, 1994), and the postsubiculum (Sharp, 1996).

In addition to place cells, *head-direction cells* have also been discovered in the hippocampal region. These cells appear to respond to particular directions of the animal's head, irrespective of its location in the environment. Each cell fires only when the animal's head faces one particular direction (over an approximately 90 degree range) in the horizontal plane, and their relative directional tuning appears to be independent of the pitch and roll of the head. Thus, these cells function as some sort of an *in-built compass*. These cells were first discovered in the postsubicular area of the hippocampal formation (Ranck, 1984; Taube *et al.*, 1990a; Taube *et al.*, 1990b). Since then, such directional cells have also been discovered in the retrosplenial cortex (Chen *et al.*, 1994a; Chen *et al.*, 1994b), the anterior thalamus (Taube, 1995; Blair & Sharp, 1995), and the laterodorsal thalamus (Mizumori & Williams, 1993).

A number of experiments have been performed in order to determine the properties of the place and head-direction cells. For instance, it is now known that the spatial representation in the place cells is not *grid-like*, i.e., adjacent neurons are as likely to represent distant portions of

the environment as close ones (O'Keefe, 1976; Muller *et al.*, 1987; O'Keefe & Speakman, 1987; O'Keefe, 1989; Wilson & McNaughton, 1993). Additionally, place cells are active in multiple places in the environment (O'Keefe & Speakman, 1987) and also in multiple environments (Kubie & Ranck, 1983; Muller *et al.*, 1987; Muller & Kubie, 1987). Further, places appear to be represented in the hippocampus using an *ensemble code*, i.e., a set of place cells appear to code for a place (Wilson & McNaughton, 1993).

Experiments have also revealed that when the animal is introduced into a familiar environment, place fields are initialized based on visual cues and landmarks in the environment (Muller & Kubie, 1987; Muller *et al.*, 1987; Sharp *et al.*, 1990). Once initialized, the place fields have been found to persist even if the visual cues are removed in the animal's presence (O'Keefe & Speakman, 1987), implying that place cell firing must also be maintained by a source other than visual stimulus. CA1 cell firings have also been found to be conserved in darkness, provided the animal is first allowed some exploration of the apparatus under illuminated conditions (McNaughton *et al.*, 1989; Quirk *et al.*, 1990). This has led to the suggestion that place fields are maintained by ideothetic (self-motion) mechanisms, in particular, by the dead-reckoning system.

Similarly, head-direction cells have also been found to be responsive to visual inputs and demonstrate a number of properties similar to the place cells described above.

Experiments have shown that the motor system plays a critical role in the firing of place and head-direction cells. For instance, with restraints on active motion, both hippocampal place cell activity (Foster *et al.*, 1989) as well thalamic head-direction cell activity (Knierim *et al.*, 1995) have been observed to cease. Since the dead-reckoning system presumably receives input from the motor system (e.g., in the form of a motor efference copy) these experiments further implicate dead-reckoning in place and head-direction cell firing (McNaughton *et al.*, 1996). Rats have also been found to develop stable, unique associations between visual stimuli and the cells of the path-integration system, which presumably allows them to realign the dead-reckoning system when mismatches occur (Knierim *et al.*, 1995).

In summary, place cells and head-direction cells respond to sensory as well as dead-reckoning inputs. These cells are active in multiple environments and also active in multiple places in the same environment. The firing of these cells is conserved in darkness, provided the animal is first allowed to orient itself under lighted conditions. Finally, the firing of these cells is directly

related to the motor system and any restraint on active motion ceases cell firing.

7.2 Cognitive Map Theory of Hippocampal Function

Based on a corpus of neuroscientific and cognitive data, O'Keefe and Nadel (1978) differentiated between two spatial representation schemes: *routes* and *maps*. In their view, routes were specified in terms of successive $S_1 - R - S_2$ instructions that led the animal from one place to another via the appropriate responses. A chain of such instructions then took the animal from the start to the goal. Importantly, such a set of instructions implied a *goal*, since following the instructions led the animal to it. They called this the *taxon system hypothesis*. They further differentiated taxon behaviors into *guidance* and *orientation* behaviors. When the emphasis of the $S_1 - R - S_2$ instruction lay in the resulting stimulus S_2, they suggested that the animal automatically produced responses that generated the resulting S_2. This was termed *guidance* and the net response learned by the animal was said to be of the form $S_1 - S_2$. On the other hand, if the emphasis of the instruction was on the response R instead, the animal would always produce the same response R given stimuli S_1. This led to associations of the form $S_1 - R$ and was termed *orientation*. Despite this difference, it is clear that taxon systems lead to *stereotypic behaviors* (e.g., approaching cues in the same sequence or performing the same sequence of responses) and *autonomous habits* (e.g., producing automatic responses even in the absence of stimuli).

In contrast to this was the *locale hypothesis* which supported a *map-like* representation of space that was composed of a set of places systematically related to each other by a group of spatial transformation rules (O'Keefe & Nadel, 1978).. O'Keefe and Nadel suggested that this *flexible* representation could be used for multiple purposes (e.g., representing and approaching multiple goals from multiple points) and for determining alternate paths between places (detour behavior). They also hypothesized that the locale representation arises as a result of the interaction between the *sensory* and *dead-reckoning* input streams. They argued that this spatial representation occurs in an Euclidean space which does not depend, for its existence, on the presence or absence of particular objects. According to their theory, animals build and update such Tolmanian cognitive maps using their locale systems. Importantly, they hypothesized that the *locale* system resides within the hippocampus and is affected by hippocampal lesions, while the *taxon* system lies largely outside the hippocampal area. This viewpoint is

now popularly referred to as the *cognitive map theory of hippocampal function.* (Nadel, 1996).

They also outlined the possibility of parallel learning by the locale and taxon systems, with the former rapidly learning flexible representations and the latter slowly converging to relatively inflexible ones. They suggested that if animals were to be exposed to repetitive tasks (like running the same maze over and over again), the taxon systems would slowly gain favor over the locale ones, leading to inflexible stereotypical behaviors.

O'Keefe and Nadel also ascribed specific roles to the different hippocampal regions in the construction of the cognitive map. In their view, the dentate gyrus organized inputs transmitted by the taxon systems into a schema (format) required by the cognitive mapping system and combined simple stimuli into more complex ones. The CA3 region was believed to represent places and the relations between them, while the CA1 region played the role of refining the map representation and directing the learning of places. The CA1 region was suggested to achieve this using a *misplace system* that signaled the presence of something new or the absence of something old, in the current sensory content. The motor circuits of the animal were believed to be driven by the outputs of the misplace detectors, which allowed the animals to not only explore novel areas, but also approach familiar objects and goals (O'Keefe & Nadel, 1978).

In summary, the locale hypothesis or the cognitive map hypothesis of O'Keefe and Nadel suggests that the hippocampal formation is involved in novelty or discrepancy driven exploratory behaviors, neural representation of spatial maps and episodes (or events) in specific environments or contexts, the updating of these representations based on detected discrepancies, and the use of these spatial maps in the generation of routine or novel trajectories (Nadel, 1996). Although O'Keefe and Nadel justified this hypothesis by pooling extensive data from cellular and lesion studies, they did not provide any computational implementation of this cognitive map theory. In Chapter 8 we will develop a computational characterization of this theory and in Chapters 9 and 10 we will provide an implementation of this model. But before proceeding to the development of our model, we will briefly discuss some computational models of hippocampal function that have been proposed in the literature.

7.3 Computational Models of Hippocampus-Based Animal Navigation

Following O'Keefe and Nadel's assertion regarding the involvement of the hippocampus in animal navigation, a number of computational models of hippocampal spatial learning and navigation have been proposed. These have been implemented in a variety of artificial automata, including computer simulations and actual robots. In what follows, we will use the term *animat* to refer to the artificial entities (e.g., computer programs, robots, etc.) used to realize or simulate animal behaviors.

The models of hippocampus-based navigation can be primarily distinguished based on the nature of spatial representation used and the form of navigation behaviors supported by them. These models fall into three broad categories: (1) Place-response based navigation; (2) Topological navigation; and (3) Metric navigation, with models in these three categories differing from each other in significant ways. Since most of these models have been formulated with the express intention of explaining hippocampus-aided spatial learning and navigation, how they be so different?

The resolution of this concern lies in the realization that these broad approaches (and the specific models therein) are motivated by *different* aspects of neurophysiological, neuroanatomical, and behavioral data. Thus, place-response based methods are inspired by the existence of place cells in the hippocampus and are influenced by the Thorndikian or Hullian spatial learning theories described in Sections 6.4.1 and 6.4.1.2. In contrast, topological navigation approaches are based on the cognitive mapping theory of Tolman, presented in Section 6.4.1.1. These models assume that the CA3 recurrent collaterals encode a topological description of the environment. Finally, the metric approaches discussed here are either explicitly or implicitly influenced by the locale hypothesis of (O'Keefe & Nadel, 1978) and their suggestion that the hippocampus learns place maps in terms of dead-reckoning based position estimates.

Although we have developed a computational characterization of the locale system hypothesis (to be described in the next chapter), it is our belief that the hippocampus represents information of multiple kinds including, directional (as found in place-response based systems), topological, as well as metric. We also believe that the navigating animal has access to these different kinds of information and it chooses an appropriate navigation strategy, or some combination of them, in ways suited to its current navigational needs.

We now provide a brief description of these hippocampus-based animal navigation models.

For a more detailed analysis, the reader is referred to Trullier et al. (1997).

7.3.1 Place-Response Based Navigation

In models belonging to this category, each place is associated with a navigation response that the animat performs when it is in that place. These responses are typically ones that move the animat towards a goal. Since these approaches only require recognition of a place, the animat's response can be determined quickly and such behaviors are presumably fast to execute. For instance, Zipser proposed a neural model in which each *place cell* was tuned to the distance, bearing, and identities of three landmarks (Zipser, 1986). Place cells were also associated with a *goal cell*, which encoded a vector to the goal location from that place. Incoming sensory inputs activated appropriate place cells, signaling recognition of the current place. Once place recognition was complete, the animat moved in the direction given by the *weighted average* of the goal vectors associated with the active place cells.

In the model of Brown and Sharp (1995), place cells and head-direction cells converged onto two clusters of cells in the *nucleus accumbens*. Each cluster was associated with a turn direction (left or right), and depending on the activity in the cluster, the animal moved forward by a small amount while also turning in the corresponding direction. Intra-cluster inhibition allowed only one cell to be active in any cluster, while inter-cluster inhibition made the animal turn in only one direction. Importantly, *synapses* between the head-direction cells and the nucleus accumbens cells, that were recently active, were modified based on rewards obtained at the goal site. Thus the animats learned to execute specific responses at particular places, and learned to execute sequences of actions to reach a goal.

Blum and Abbott (1996) proposed a model where locations were represented by place cell activity. Importantly, the location coded by the place cell activity could shift based on the experiences of the animat. This was achieved through a form of LTP (long term potentiation) that modified the synapses between the place field that the animat was in earlier (P_1) and the place field that the animat was currently in (P_2). These synaptic changes had the effect of shifting the location coded by place cell activity, i.e., if the animat visited P_1 at a later time, the place cell activations would lead to an excitation of place field P_2. Navigation was then just a matter of moving from the current place field towards the location coded by the place cell activity.

As can be surmised, these approaches encounter two kinds of problems. First, they cannot represent multiple goal locations without major extensions. Secondly, latent learning or learning a spatial map in the absence of goals is difficult to achieve with these models.

The model of Burgess and O'Keefe, on the other hand, was a place-response based navigation system that addressed both these problems (Burgess *et al.*, 1994; Burgess & O'Keefe, 1996). Their model included a specification of not only the hippocampal formation but also the adjacent cortical activity of entorhinal cells, subicular cells, and goal cells. Their model implemented competitive learning within place and subicular cell layers, and a form of Hebbian learning between layers. They also used a *spiking* model of neuronal activity, with multiple firing phases, to model the EEG θ-rhythm. Importantly, they hypothesized the existence of goal-cells downstream of the hippocampus, which were assumed to be tuned to different goals and updated based on reinforcements obtained at the goal locations. Their model allowed directed navigation, including the ability to devise *short-cuts*. Importantly, it supported the representation of multiple goals and latent learning.

7.3.2 Topological Navigation

In topological navigation approaches, the animats learn a map of places in their environment along with the motion required to get from one place to the other. Though these models are very similar to the topological graph approaches used in robotics (discussed in Section 6.3.2) they differ in the important aspect that these models are directly inspired by (and based on) neuro-cognitive data. In their purest form, these approaches provide one realization of Tolman's $S_1 - R - S_2$ expectancies discussed earlier, thereby providing one implementation of his cognitive map idea (Tolman, 1948).

These schemes allow for latent learning, demonstrating an ability to learn topological place maps in the absence of explicit goals. Once such relationships have been captured, goal-directed navigation reduces to determining a path from the current place to the place that houses the goal. This procedure typically searches the topological graph to identify a path satisfying possible constraints like the shortest path-length, least travel time, etc. The primary advantage of these models is their ability to determine multiple alternate routes to the goal, thereby allowing for effective detour behaviors.

A number of hippocampal models of topological navigation have been proposed. The model

of Muller et al. (1991) encoded the spatial map as a *cognitive graph*. In their model, place cells were assumed to exist *a-priori* and were connected together in a network fashion via modifiable links. The animat was assumed to move at a constant speed between places, which resulted in the firing of successive place cells with temporal delays proportional to the distance between the corresponding place fields. Their model then updated the weights between places such that links between places closer together were enhanced more. This approach led to weights that were inversely proportional to the distance between the corresponding place fields. The shortest path to arbitrary goals could then be easily determined by identifying a path from the current node to the goal that had the greatest sum of the weights.

The model of Schmajuk and Thieme (1992) also made similar assumptions regarding *a-priori* specification of place cells and their possible neighbors (called a *view*). Based on animat motion, a Hebbian learning procedure systematically associated place nodes with their corresponding view nodes thereby creating a topological map of the environment. Their model supported two kinds of dynamics. While *slow* dynamics allowed the animat to predict the next place based on the knowledge of the current place and its motion, *fast* dynamics used the topological map to determine routes to goals through a form of *mental rehearsal*.

Scholkopf and Mallot (1995) proposed a similar model where the spatial map was encoded using a neural network with a layer of interconnected place cells. The inputs to the network consisted of visual (or sensory) inputs as well as information pertaining to the motion of the animat. The sensory inputs led to the excitation of specific place cells, while the motion information gated the connections between place units, thereby learning a topological description of the environment. Once the map was learned, goal-directed navigation was realized though an interesting mechanism. First, the sensory inputs corresponding to the current place were applied to the network, which resulted in place cell firing. Then, different movement commands were systematically applied to the network and the resulting predictions of places were recorded. Different movement commands were then applied to each of these predictions thereby generating second degree predictions, and so on. This process was repeated until a goal prediction was obtained, at which point the appropriate sequence of movements could be executed by the animat to reach the goal.

These topological navigation schemes have been validated through a variety of simulation experiments. In addition to these, a few other topological navigation models inspired by

hippocampal or cognitive mapping theories have been implemented on physical robots (Kuipers & Byun, 1991; Mataric, 1992; Bachelder & Waxman, 1994; Recce & Harris, 1996). These are discussed in Section 9.3.2.

7.3.3 Metric Navigation

In addition to the topological realizations of Tolman's cognitive mapping theory, a number of models inspired by the hippocampal cognitive map theory of O'Keefe and Nadel (1978), have also been proposed. The important difference between these two classes of models is that metric navigation approaches learn places in terms of their position in some metric framework computed by the animat (usually Euclidean in nature). If goals are also represented in terms of metric positions in the same framework, goal-directed navigation comes down to determining the current metric position of the animal, the position of the goal, and a simple computation (vector subtraction) to determine the approach vector from the current position to the goal.

Using an extension of Zipser's model (1986), Prescott (1992) developed a metric space representation in terms of local *frames* defined by groups of three landmarks. By representing the position of a fourth landmark in the local frame, his model allowed the creation of a database of relational landmark locations. This model implicitly encoded the metric positions of the places, since any arbitrary goal location could be determined by a set of intervening local frame transformations that mapped the goal frame to the frame that the animat was in. Navigation to a goal required the activation of frames that contained the goal location. Each of these active frames predicted the relative position of a fourth landmark. These landmarks were then added to the list of visible (or available) landmarks and all frames activated by this new set of landmarks were retrieved from the relational database. This process was repeated until a frame containing the animat's current location was activated. At this point the set of frame transformations from the current frame to the goal frame could be used by the animat to navigate. Prescott also extended this approach to automatically determine the *shortest* trajectory to the goal.

Another related theory of hippocampal function was forwarded by Worden (1992), who used *fragments* to encode metric relationships between three prominent landmarks in each place (along with other discriminatory non-geometric properties such as smell, color, etc.). Navigation in this theory required *fitting together* a map by translating and rotating fragments

to appropriately align the landmarks common to the different fragments. This map-building process terminated when the map contained both the current location as well as the goal, at which point a direct trajectory to the goal location could be computed.

Another interesting metric model of hippocampal navigation was proposed by Redish and colleagues (Wan *et al.*, 1994; Redish & Touretzky, 1996; Redish & Touretzky, 1998). As this model is very similar in principle to our work, we will describe this model in detail in Section 8.6.

7.4 Role of the Hippocampus in Memory

The term *memory* has been historically used to refer to the ability to recall prior experiences into present consciousness (James, 1890). In general, memory refers to the *persistence* of acquired information, usually through learning, in a state that can be revealed at a later time (Squire, 1986). Our discussions so far have focused on the role of the hippocampus in the realization of one form of memory, namely *spatial memory*. However, it has been proposed that the hippocampus is involved in a wide variety of other memory processes as well. Given this situation, it is only appropriate that we discuss, at least briefly, some alternate viewpoints of hippocampal function in *memory*.

1. **Attentional theories**

   Attentional theories of hippocampal function suggest that the hippocampus controls (or mediates) information stored in the brain by regulating the *attention* (or amount of processing) awarded to each stimulus. The hippocampus is believed to achieve this by associating stimuli with reinforcement (or non-reinforcement), and reducing attention from non-reinforced stimuli and shifting attention to possibly novel ones (Douglas & Pribram, 1966; Kimble, 1968; Douglas, 1972). Thus, these theories predict that hippocampal lesions will affect the ability of the animal to detect novel stimuli and adversely affect its tendency to explore.

2. **Cognitive map theory**

   As described in Section 7.2, the cognitive map theory (O'Keefe & Nadel, 1978) asserts that the hippocampus participates in: (1) exploratory behaviors; (2) building of neural spatial maps encoding places and events; (3) the update of these cognitive maps using

misplace detectors; (4) the use of these maps in the generation of routine or novel trajectories; and (5) the use of spatial and episodic representation in non-spatial domains. Thus, this theory predicts that hippocampal lesions or damage will result in deficits in exploration, place learning and behavior, reaction to novelty, and the ability to recognize contexts and episodes (Nadel, 1996).

3. **Working memory theory**

   This memory hypothesis forwarded by Olton (1986)suggests that the hippocampus is involved in learning and remembering spatio-temporal aspects of the environment that *vary* from trial to trial. In this sense, the hippocampus captures and updates transient aspects of the environment, while more permanent aspects are said to reside in *reference memory*. According to this theory, working memory is characterized by flexibility, memory interference (owing to the transient aspects of the environment), and the ability to capture information pertaining to the temporal order of the stimuli. A direct prediction of this theory is the inability of hippocampal lesioned animals to localize or operate reliably in transient environments.

4. **Hippocampus as a comparator**

   According to this viewpoint, the hippocampus compares external environmental information with stored information. If the predicted and observed events are the same, then behavior is maintained. However, mismatches detected by the hippocampus lead to changes in behavior (and a consequent shift in attention) (Smythies, 1966; Vinogradova, 1975). In reality, this view of hippocampal function is general enough to fit into any of the other theories (attentional, cognitive map, etc.). For instance, mismatches could trigger increased attention or spatial exploration.

5. **Episodic memory theory**

   This theory forwarded by Rolls (1990) suggests that the hippocampal formation plays a critical role in learning and representing spatio-temporal *episodes*. These episodes correspond to *event sequences* in the temporal domain and *spatial scenes* in purely spatial contexts, with most real-world environments leading to hybrid spatio-temporal representations (Rolls, 1996). This theory has been mapped to a sequence of computational models of the hippocampus, which include specific roles assigned to the different hip-

pocampal regions (Rolls, 1989a; Rolls, 1989b; Treves & Rolls, 1992; Treves & Rolls, 1994; Rolls, 1996). This theory predicts that hippocampal lesions will affect the learning and recall of episodic information, thereby hindering place recognition and localization.

6. **Declarative memory theory**

   According to this theory the hippocampus plays a critical role in the accumulation of facts and data derived from the learning experiences of the animal (Cohen & Eichenbaum, 1993). Importantly, the nature of this representation is *relational*, i.e., the hippocampus represents relationships between objects such as relative size, color, texture, shapes, positions, etc., as well as higher-order temporally-contiguous relationships between objects and events. Although relational memory includes spatial maps, it also emphasizes the role of the hippocampus in the learning of arbitrary pairings (e.g., face with a name, etc.). According to this theory, hippocampal lesions should lead to deficits in the ability to acquire relational information about the environment, but should leave acquisition of perceptual-motor skills largely intact.

7. **Configural association theories**

   Rudy and Sutherland (1989) proposed that the hippocampus is involved in the acquisition and storage of *configural associations*. According to this theory, animals form *representations* of stimuli in the environment and form *associations* between these representations. In addition to *simple* associations between elemental representations, animals also form higher-order unique representations of *combinations* of elemental stimuli. These higher-order representations form the basis of the configural association theory with the claim that the hippocampus achieves these configural associations. Just like the declarative theory, configural representations can encode a range of possible relationships between objects. However, the difference lies in the formation of higher-order associations, which are said to involve a *fusion* of elements in configural theory while they are only *compositional* in declarative theory.

As can be observed, these different theories of hippocampal involvement in memory share many common features. Importantly, they all agree that there exist multiple learning and memory systems and that the hippocampus participates in one such. Most of the theories also agree that the representations formed by the hippocampus are relational, flexible, and permit

creative behavior in response to novelty. Finally, all the theories, either explicitly or implicitly, support the role of the hippocampus in spatial learning.

7.5 Discussion

In this chapter we have provided a brief description of hippocampal anatomy and have outlined the roles played by the different hippocampal regions in spatial memory. We also summarized physiological properties of hippocampal cells, primarily place cells and head-direction cells, that are directly related to animal spatial learning abilities.

Importantly, we outlined the hippocampal cognitive map theory of O'Keefe and Nadel (1978). According to this theory the hippocampus participates in the representation of a Tolmanian cognitive map by associating sensory inputs with position estimates derived from the animal's dead-reckoning system, thereby learning a metric spatial map of the environment. We also summarized related computational models of hippocampal spatial function and pointed out their significant features and differences.

There is also considerable evidence for the involvement of the hippocampal formation in the learning and representation of non-spatial tasks. A number of theories have been forwarded regarding hippocampal involvement in different types of memories. We presented a brief summary of some of the significant theories of hippocampus as a memory system.

In the following chapters we will develop and implement a computational characterization of the hippocampal cognitive map theory (or the locale hypothesis) of O'Keefe and Nadel (1978).

8 A COMPUTATIONAL MODEL OF SPATIAL LEARNING AND LOCALIZATION

As we have mentioned earlier, O'Keefe and Nadel (1978) were the first to suggest that the cognitive map of Tolman (1948) resides in the hippocampus. Based on extensive lesion and neurophysiological data, they suggested that the hippocampal place cells integrate information from two sources: the sensory inputs and the animal's self-generated dead-reckoning estimates, thereby learning a *metric* map of the environment. However, these two information streams provide uncertain information owing to errors in recognition, estimation of distances and directions to objects, drifts in dead-reckoning, etc. We have developed a computational specification of the cognitive map theory that explicitly handles uncertainty in the information streams and provides robust mechanisms for learning and localizing in their presence. This is achieved by drawing a parallel with Kalman filter based robot localization approaches. In this chapter we present this computational model and its extensions to support incremental map learning and goal directed navigation.

8.1 A Computational Model of Hippocampal Spatial Learning

Based on neurophysiological, anatomical, and behavioral data, O'Keefe and Nadel (1978) suggested that the hippocampal formation is the site of the cognitive map proposed by Tolman (1948). In what they termed the *locale hypothesis*, they suggested that the hippocampal formation learns a map-like representation of space. Importantly, they hypothesized that the hippocampal cells integrate and associate sensory inputs with dead-reckoning information generated by the animal, thereby encoding the map using a metric representation scheme (O'Keefe & Nadel, 1978).

We have developed a computational specification of this locale hypothesis that learns, represents, and updates a metric spatial map (Balakrishnan *et al.*, 1997; Bousquet *et al.*, 1998;

Balakrishnan *et al.*, 1998c). The system learns a map of *distinct* places in the environment and labels the *center* of each place with metric position estimates derived from dead-reckoning. This fusion of sensory and dead-reckoning information takes place in a functional model of the hippocampal formation, as shown in the sketch of the model in Figure 8.1.

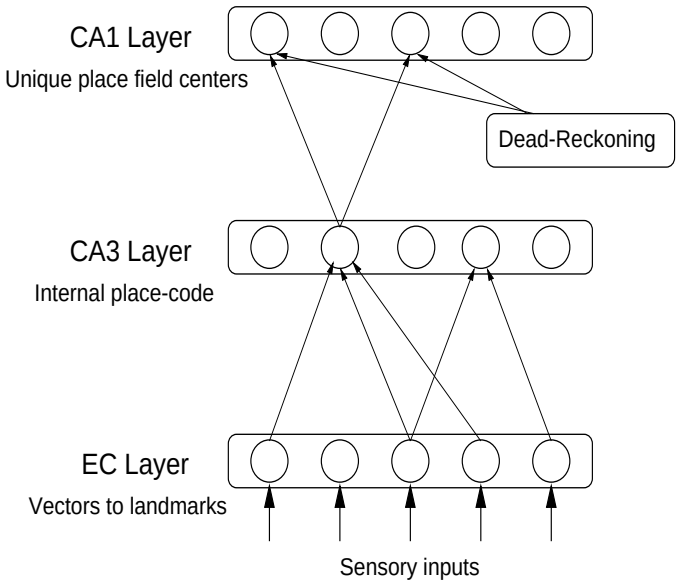

Figure 8.1 Computational model of the hippocampus.

The overall functioning of our model is as follows. During exploration of its environment, the animat detects and recognizes landmarks and estimates their relative positions. This *processed* landmark information is used to *organize* (or create) units in the EC layer in such a way that units respond to specific landmarks appearing at particular positions *relative* to the animal. Thus, EC cells encode and respond to *vectors* to specific landmarks, which is consistent with the observed properties of EC units (Quirk *et al.*, 1992). Since places are characterized by their relationship to the set of landmarks visible from that place and since EC units encode vectors to specific landmarks, concurrent activity of a set of EC units captures the landmark relationships at that place. CA3 place cells are then organized (or created) to capture this EC activity, thereby representing places. The firing of CA3 cells in response to sensory inputs at a given place thus constitutes an *internal place code* for that place. The CA1 cells then associate this internal place code with metric position labels from the dead-reckoning system. Thus, while EC and CA3 cells enable the learning of places using sensory information, the CA1 layer

learns *centers* of these places in terms of dead-reckoning position estimates.

Apart from learning a metric spatial representation, another role for the CA1 layer is suggested by *perceptual aliasing* problems encountered by robots, wherein multiple places in the environment appear sensorily/perceptually identical. Given the limitations on the sensory abilities of most animals and the commonality in the features of many habitats (e.g., vast deserts, thick forests, etc.), it is reasonable to assume that animals too encounter such aliasing problems. The fact that they successfully deal with their environments suggests that they possess some mechanisms for resolving such ambiguities. We hypothesize that perceptually identical places (and hence place codes in the CA3 layer) are disambiguated in the CA1 layer by means of dead-reckoning information. Thus, perceptually aliased places activate the *same* population of cells in the CA3 layer but different populations in the CA1 layer.

It should be noted that our model ascribes slightly different roles to the CA3 and CA1 layers, which leads to differences in the firing of CA3 and CA1 units. However, neurobiological experiments to date have found little difference in the firing properties of CA3 and CA1 cells. In Section 8.7 we provide some explanations to resolve this discrepancy.

Our model also assumes that goals are learned and represented in terms of their metric positions in the animat's dead-reckoning framework. However, this goal representation is believed to reside outside the hippocampus (O'Keefe, 1989; Redish & Touretzky, 1996).

In addition to learning places by appropriately creating EC, CA3, and CA1 units, our model also performs localization. When the animat visits familiar places, incoming sensory inputs excite EC cells which in turn activate a place code in the CA3 layer. As multiple CA1 place codes may respond to this CA3 code (due to perceptual aliasing), the dead-reckoning input is used to determine the CA1 field with place field center closest to the dead-reckoning based position estimate of the animat. The hippocampal system then performs spatial localization by *matching* the *predicted* position of the animat (the dead-reckoning position estimate) with the *observed* position of the place field center (dead-reckoning estimate stored with the activated CA1 place code). Based on this match, the dead-reckoning estimate as well as the place field center can be updated as shown in Figure 8.2.

In our model, place fields are quickly formed through exploration of the environment. The CA1 place cells are driven by both sensory as well as dead-reckoning inputs. Thus, they can be manipulated by changes in the landmark configuration and yet continue to fire in

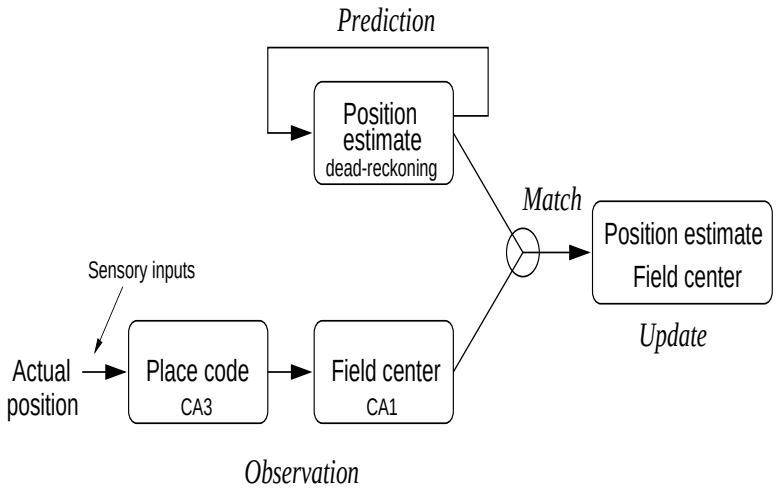

Figure 8.2 A schematic of hippocampal localization.

darkness. In a familiar environment, the animat places high confidence in its dead-reckoning based position estimate. Under these circumstances, even if some landmarks are removed, the high confidence in dead-reckoning overrides the changed sensory activation, allowing the animat to still correctly identify the place. We believe that changes in the EC-CA3 (as well as EC-Dg) synapses soon align the place code with the new sensory input. Upon reintroduction into a familiar environment, the animat initially distrusts its dead-reckoning position estimate. As described above, the animat uses sensory inputs to localize, thereby initializing its dead-reckoning system. In darkness, without the appropriate updates of Figure 8.2, the position estimates accumulate errors and place cell firing in CA1 drifts.

8.2 Need for Probabilistic Localization

Following the locale hypothesis, our model of the hippocampal spatial learning integrates information from two streams: the sensory inputs and the dead-reckoning system. However, it should be noted that information provided by both these streams is uncertain. Sensory systems of animals accommodate considerable errors (for e.g., in the estimation of distance and direction to visible objects, recognition of objects, etc.). Dead-reckoning is also prone to estimation errors and drifts, and the very fact that place cell (and head-direction cell) firings drift in darkness is suggestive of errors in path-integration. In order for the hippocampus to perform

robust spatial localization using these uncertain information sources, it must necessarily be capable of appropriately handling these uncertainties. To borrow a term from the robotics literature, the hippocampus must be capable of *probabilistic localization*. Although several hippocampal models of spatial learning have been proposed, some of them closely related to our own (Redish & Touretzky, 1996), none of them explicitly address the question of handling uncertainty in data. In order to satisfactorily characterize hippocampal spatial learning, we need a probabilistic framework for addressing uncertain information fusion.

Since mobile robots also have to deal with uncertainties in sensing and action, a number of probabilistic localization approaches have been developed in that context. One such localization tool is the *Kalman filter* (Kalman, 1960; Gelb, 1974; Crowley, 1995) (or some extension or generalization of it), which allows the robot to build and maintain a *stochastic spatial map* of its environment (Smith *et al.*, 1990; Moutarlier & Chatila, 1989; Leonard & Durrant-Whyte, 1992), propagate sensory and motion uncertainties, and localize in *stochastically-optimal* ways (Ayache & Faugeras, 1987; Hebert *et al.*, 1995). As we will demonstrate shortly, our model of hippocampal spatial learning and Kalman filter based robot localization are very similar in principle. This encourages us to explore extensions of the Kalman filtering framework to characterize hippocampal spatial learning. But first we digress briefly to look at the use of Kalman filter based approaches in robot localization.

8.2.1 Robot Localization Using a Kalman Filter

The Kalman filter technique for robot localization typically maintains a stochastic map of the robot's environment at each discrete time-step k. This stochastic spatial map, denoted by a state vector $\mathbf{x}_k$, includes an estimate of the robot's current position and possibly the estimated positions of other landmarks in the robot's environment. The *system model* is assumed to be specified and denotes the change in state based on robot motion:

$$\mathbf{x}_k = \Phi_{k-1}\mathbf{x}_{k-1} + \mathbf{u}_{k-1} + \mathbf{v}_{k-1} \tag{8.1}$$

Here, Φ_{k-1} is the transformation of the state based on robot motion (e.g., translation, rotation, etc.), $\mathbf{u}_{k-1}$ is the linear change in state based on the intended robot motion, and $\mathbf{v}_{k-1}$ is a *zero-mean* motion error with covariance matrix $\mathbf{Q}_{k-1}$. Here, $\mathbf{v}_{k-1}$ represents the fact that in real-world environments, the intended motion of the robot ($\mathbf{u}_{k-1}$), is offset by errors like wheel-

slippage, friction, improper tire inflation, etc. This makes the *actual* robot motion different from its *intended* one.

Kalman filtering also requires a *measurement model* to be specified, which captures the relationship between measurements (or observations) and the current state of the robot $\mathbf{x}_k$:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k \tag{8.2}$$

Here, $\mathbf{H}_k$ represents the *sensor model* that maps states into observations. Further, real-world measurements are not perfect and this error, $\mathbf{w}_k$, is modeled as a *zero-mean* noise with covariance matrix $\mathbf{R}_k$.

Given these two models, the Kalman filter stores and updates an *estimate* of the state $\hat{\mathbf{x}}_k$ and its associated covariance matrix $\mathbf{P}_k = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T]$, by making state *predictions* and combining them with *observations*. Suppose the current state estimate is $\hat{\mathbf{x}}_{k-1}^+$ with the covariance matrix $\mathbf{P}_{k-1}^+$ (the '+' superscript indicates *updated* values of $\mathbf{x}_{k-1}$ and $\mathbf{P}_{k-1}$). Based on robot motion, the Kalman filter predicts the new state $\hat{\mathbf{x}}_k^-$ using Equation 8.1. Here the '-' superscript stands for the predicted (and as yet *uncorrected*) values at step k. Since the motion error $\mathbf{v}_{k-1}$ has a zero mean this prediction is given by $\hat{\mathbf{x}}_k^- = \Phi_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{u}_{k-1}$. The corresponding covariance matrix is updated to $\mathbf{P}_k^- = \Phi_{k-1}\mathbf{P}_{k-1}^+\Phi_{k-1}^T + \mathbf{Q}_{k-1}$. Based on this state prediction and using the sensor model $\mathbf{H}$ from Equation 8.2, the system predicts the measurement $\hat{\mathbf{z}}_k = \mathbf{H}_k\hat{\mathbf{x}}_k^-$ using an estimate of zero for the measurement error $\mathbf{w}_k$. This is the sensory input (or observation) the robot *expects to receive* at its new position. Based on actual measurement (or observation) $\mathbf{z}_k$ the Kalman filter updates the state estimate and covariance matrix as follows (refer to (Gelb, 1974) for details of the derivation):

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{8.3}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{8.4}$$

where $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$ is referred to as the *Kalman gain* and $\mathbf{z}_k - \hat{\mathbf{z}}_k$ is called the *innovation*. This Kalman filter formulation is recursive and the process simply repeats as the robot moves. Kalman filtering can thus be described by the schematic shown in Figure 8.3.

It can be shown that the Kalman filter update expressions used above correspond to an estimation process that is *optimal* (minimum variance, maximum likelihood, etc.), provided

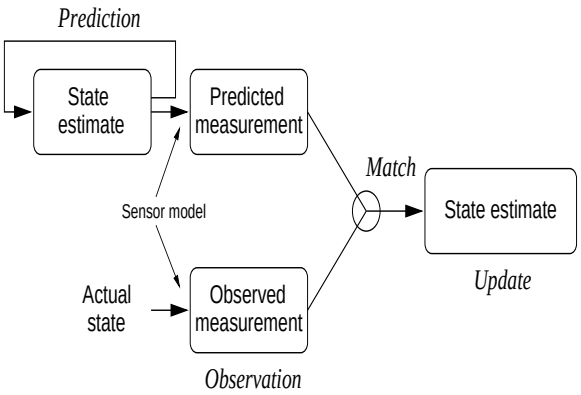

Figure 8.3 A schematic of Kalman filtering.

the system (Φ) and measurement ($\mathbf{H}$) models are linear and the noises $\mathbf{v}$ and $\mathbf{w}$ are *Gaussians* with the following properties (Jazwinski, 1970; Gelb, 1974; Maybeck, 1990).

1. *Zero mean:* $E(\mathbf{v}_i) = 0$ and $E(\mathbf{w}_j) = 0$.

2. *White:* $E(\mathbf{v}_i \mathbf{v}_j^T) = \delta_{ij}\mathbf{Q}_i$ and $E(\mathbf{w}_i \mathbf{w}_j^T) = \delta_{ij}\mathbf{R}_i$, where δ_{ij} is the *Kronecker delta function* ($\delta_{ij} = 1$ iff i=j, and 0 otherwise) and $\mathbf{Q}_i$ is the covariance matrix of the random noise $\mathbf{v}_i$ and $\mathbf{R}_i$ is the covariance matrix of the random noise $\mathbf{w}_i$.

3. *Uncorrelated:* $E(\mathbf{w}_i \mathbf{v}_j) = 0$ for all i, j;

8.3 Kalman Filtering in the Hippocampus

It can be observed from Figures 8.2 and 8.3 that our computational model of hippocampal function and Kalman filter both share the same *predict-observe-match-update* principle. Further, Kalman filter provides a framework for performing stochastically optimal updates even in the presence of prediction and observation errors. Since our goal is to develop a framework for uncertain information fusion in our hippocampal spatial learning model, it is interesting to explore whether hippocampal function could be described in terms of Kalman filtering theory. If so, one can apply Kalman filtering theory to fuse uncertain information in the hippocampus and use it in practical computational realizations. In what follows, we will develop a Kalman filtering framework for our hippocampal localization model.

8.3.1 State Vector Representation

In robot localization, the state vector usually contains the position of the robot and the positions of landmarks in the robot's environment. Since spatial localization in the hippocampus appears to be based on *place recognition*, we have to use a state vector that contains the locations of these places. In our computational model, places are characterized by their centers which are represented in terms of dead-reckoning estimates. Thus, we use a state vector that is composed of the estimated *centers* of places encountered by the animat and represented in the CA1 layer. It should be borne in mind that this notion of center is probabilistic, i.e., it is the expected position of the animat given that the corresponding place has been recognized. Mathematically, $x_i = E(x_{0,k}|i)$, where $x_{0,k}$ is the position of the animat and x_i is the center of place i. To keep the discussion simple we will henceforth assume that place codes in CA3 and CA1 are represented by single units, although in reality ensembles of units are known to code for place (Wilson & McNaughton, 1993). Thus, the state vector at time step k is given by:

$$\mathbf{x}_k = [x_{0,k}, x_1, \ldots, x_n]^T$$

where $x_{0,k}$ denotes the position of the animal, x_i denotes the center of place field i, and n is the number of distinct places that have been visited by the animal. We also assume that these positions (animat position and place field centers) are specified in 2D Cartesian coordinates, i.e., $x_i = (x_{i_x}, x_{i_y})$. We do not consider the *orientation* of the animat (or a place) for two reasons. Firstly, this simplifies the model and makes the computations easier to characterize. Secondly, our computational model currently does not have a mechanism for learning and updating the orientation or head-direction estimates in rodents.

The covariance matrix associated with this state vector, denoted by $\mathbf{P}_k$, is given by:

$$\mathbf{P}_k = \begin{pmatrix} \mathbf{C}_{00} & \mathbf{C}_{01} & .. & \mathbf{C}_{0n} \\ \mathbf{C}_{10} & \mathbf{C}_{11} & .. & \mathbf{C}_{1n} \\ . & . & . & . \\ \mathbf{C}_{n0} & \mathbf{C}_{n1} & .. & \mathbf{C}_{nn} \end{pmatrix}$$

where

$$\mathbf{C}_{ij} = \begin{pmatrix} C_{i_x j_x} & C_{i_y j_x} \\ C_{j_x i_x} & C_{j_y i_y} \end{pmatrix}$$

denotes the covariance between the 2D Cartesian representations of the state elements $x_i = (x_{i_x}, x_{i_y})$ and $x_j = (x_{j_x}, x_{j_y})$.

When a new place is visited, the state vector is augmented by the center of this new place and the state estimate and its covariance matrix are modified accordingly.

8.3.2 The System Model

As described earlier, the system model in Kalman filtering is used to predict the new state vector based on robot motion. Importantly, the state prediction is based on the *intended motion* of the robot, while the *actual motion* of the robot is influenced by a host of environmental factors (e.g., friction, wheel-slippage, etc.). In regular Kalman filtering, errors appear in the motion, not the prediction. In contrast, the prediction of the animat's position in our hippocampal model is provided by *active* dead-reckoning, i.e., the position of the animat is estimated based on the actual motion of the animat (by integrating acceleration and velocity signals). Since dead-reckoning is considered error prone, we are lead to the following modified system models:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{u}_{k-1} && \text{Actual animat motion} \\ \hat{\mathbf{x}}_k^- &= \mathbf{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{u}_{k-1} + \mathbf{v}_{k-1} && \text{Dead-reckoning estimate} \end{aligned}$$

Here, $\mathbf{u}_{k-1}$ is the *actual motion* of the animat (inclusive of its intended motion and the motion errors). The dead-reckoning system produces a position estimate based on the animat's actual motion $\mathbf{u}_{k-1}$ with dead-reckoning errors characterized by a zero-mean white noise $\mathbf{v}_{k-1}$ with covariance matrix $\mathbf{Q}_{k-1}$. As explained earlier, this formulation is different because the error is now in the prediction rather than in the motion.

In our spatial learning model place field centers are learned in global coordinates and do not change with animat motion. Since only the position estimate of the animat changes, we can make the following simplifications to the system model: $\mathbf{u}_k = [u_k, 0, \ldots, 0]^T$ and $\mathbf{v}_k = [v_k, 0, \ldots, 0]^T$. Further, if we assume that the animat only translates in the direction it is facing or turns in the same place, we can take $\Phi_k = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, leading to the following *linear* system models for animat motion and dead-reckoning:

$$\begin{aligned} x_{0,k} &= x_{0,k-1} + u_{k-1} && \text{Actual animat motion} \\ \hat{x}_{0,k}^- &= \hat{x}_{0,k-1}^+ + u_{k-1} + v_{k-1} && \text{Dead-reckoning estimate} \end{aligned}$$

For simplicity, we also assume that variance of the Gaussian error is the same along both the coordinate dimensions of the 2-D representation. Further, errors along the two dimensions are considered to be independent (covariances are zero).

8.3.3 The Measurement Model

The measurement model in Kalman filtering specifies the relationship between the state vector and the measurements the robot is capable of making. This allows the robot to predict and compare observations, and update system state based on the resulting mismatch. Traditionally, Kalman filtering approaches applied to robotics use a *sensor model* $\mathbf{H}$ that allows one to determine the sensory inputs the robot would receive at any given place. However, such sensor models are hard to specify as they implicitly assume that the robot knows the locations of objects in its environment. For this reason, do not want to perform matches in the space of sensor representations.

Since our hippocampal model stores metric positions of place field centers in the CA1 cells, a second possibility is to use sensory inputs to determine the place i_k where the animat is currently located, and treat $\hat{x}_{i_k}$ (the center of this recognized place) as the observation. The prediction is them simply $\hat{x}_{0,k}$ generated by the dead-reckoning system. However, this cannot be done because the observation and prediction are correlated (since $\hat{x}_{i_k}$ is some previous value of $\hat{x}_{0,k}$) and Kalman filtering requires system and measurement errors to be uncorrelated. Other alternatives must be explored.

Another possibility is to choose the measurement model: $z_k = x_{0,k} - x_{i_k} + w_k$ which represents a vector from the center of place i_k where the animat finds itself at time step k to its current position $x_{0,k}$. Although we can predict this measurement based on the estimated values of $\hat{x}_{0,k}$ and $\hat{x}_{i_k}$, we cannot observe z_k since we do not know the true centers of the places (x_{i_k}) or the exact position of the animat in the environment ($x_{0,k}$). Indeed, that is the very reason why we require a spatial learning mechanism.

However, this problem can be circumvented by specifying a measurement function that always observes $z_k = x_{0,k} - x_{i_k} + w_k = 0$, which is equivalent to saying that the measurement model always observes the animat at the center of the corresponding place field. This measurement function constrains the form of the random error to $w_k = x_{i_k} - x_{0,k}$ as shown in Proposition 3. Further, Proposition 4 shows that w_k has zero-mean provided the animat navi-

gates randomly between place fields or moves from one place field center to another. However, it turns out that this error is *autocorrelated* and hence is not a white sequence (Proposition 5). This autocorrelation is difficult to measure, and though we could use extended state vectors to estimate this correlation, leading to augmented Kalman filtering (pp 212, (Jazwinski, 1970)), we choose to ignore it. The justification for this is provided by the navigation behavior of the animat . If the animat moves to random positions in the place fields this autocorrelation will be zero since the current displacement of the animat from the center of the current place field will not be related to its displacement in the previous step (due to the intervening random motion). On the other hand, if the animat moves purposefully from one place field center to the other, the correlation will be very small since the magnitude of the errors (w_k) can be expected to be small. Since it is reasonable to assume that the animal moves randomly during exploration and subsequently from one place field center to the next, we can ignore this autocorrelation term in our computations. Indeed, in our simulations we found this autocorrelation to be negligible.

Thus, if the sensory input at a given place corresponds to a place i_k, we use $\mathbf{H}_k\mathbf{x}_k = x_{0,k} - x_{i_k}$ and $\mathbf{w}_k = w_k$ to obtain the measurement model and predicted measurements as:

$$\begin{aligned} z_k &= x_{0,k} - x_{i_k} + w_k = 0 \quad \text{Observed measurement} \\ \hat{z}_k &= \hat{x}^-_{0,k} - \hat{x}^-_{i_k} \qquad\qquad\quad \text{Predicted measurement} \end{aligned}$$

The covariance matrix $\mathbf{R}_k$ associated with this measurement error is given by:

$$\mathbf{R}_k = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

where $\sigma^2 = R^2\sigma_S^2 - 2\sigma_L^2 \ln(\Gamma_{EC})$, as shown in Proposition 6.

8.3.4 Update Expressions

Given the state vector representation and the system and measurement models described above, we can easily verify that the Kalman filtering requirements (i.e., linear models and zero-mean, white, uncorrelated Gaussian errors) are satisfied. Thus, the Kalman filter update equations of Section 8.2.1 can be directly applied to our computational model. Kalman filtering in the hippocampus then proceeds as follows:

1. Based on the actual animat motion, the dead-reckoning system generates a position estimate at time step k.

$$\hat{x}^-_{0,k} = \hat{x}^+_{0,k-1} + u_{k-1} + v_{k-1}$$

The elements of the covariance matrix associated with the state are also updated as follows:

$$\mathbf{P}^-_k = \mathbf{P}^+_{k-1} + \mathbf{Q}_{k-1}$$

where $\mathbf{Q}_{k-1}$ is the covariance matrix of the dead-reckoning error and is given by:

$$\mathbf{Q}_{k-1} = \begin{pmatrix} \mathbf{q}_{k-1} & 0 & .. & 0 \\ 0 & 0 & .. & 0 \\ . & . & . & . \\ 0 & 0 & .. & 0 \end{pmatrix}$$

The covariance update then reduces to the following:

$$\begin{aligned} \mathbf{C}^-_{00} &= \mathbf{C}^+_{00} + \mathbf{q}_{k-1} \\ \mathbf{C}^-_{ij} &= \mathbf{C}^+_{ij} \qquad\qquad \forall \text{i, j not both 0} \end{aligned}$$

2. Based on its sensory inputs, suppose the animat identifies the place as i_k. Using our model of Kalman filtering in the hippocampus, $z_k = 0$ and $\hat{z}_k = \hat{x}^-_{0,k} - \hat{x}^-_{i_k}$.

3. By matching z_k and $\hat{z}_k$, we update the dead-reckoning position estimate and its variance as follows.

$$\begin{aligned} \hat{x}^+_{0,k} &= \hat{x}^-_{0,k} - (\mathbf{C}^-_{00} - \mathbf{C}^-_{0i_k})(\mathbf{C}^-_{00} - 2\mathbf{C}^-_{0i_k} + \mathbf{C}^-_{i_k i_k} + \mathbf{R}_k)^{-1}(\hat{x}^-_{0,k} - \hat{x}^-_{i_k}) \\ \mathbf{C}^+_{00} &= \mathbf{C}^-_{00} - (\mathbf{C}^-_{00} - \mathbf{C}^-_{0i_k})(\mathbf{C}^-_{00} - 2\mathbf{C}^-_{0i_k} + \mathbf{C}^-_{i_k i_k} + \mathbf{R}_k)^{-1}(\mathbf{C}^-_{00} - \mathbf{C}^-_{0i_k})^T \end{aligned}$$

where $\mathbf{R_k}$ is the covariance matrix of the measurement error.

4. Also, each place field center m, and its associated covariances are also updated as follows:

$$\hat{x}^+_m = \hat{x}^-_m - (\mathbf{C}^-_{m0} - \mathbf{C}^-_{mi_k})(\mathbf{C}^-_{00} - 2\mathbf{C}^-_{0i_k} + \mathbf{C}^-_{i_k i_k} + \mathbf{R}_k)^{-1}(\hat{x}^-_{0,k} - \hat{x}^-_{i_k}) \qquad (8.5)$$

$$\mathbf{C}^+_{mn} = \mathbf{C}^-_{mn} - (\mathbf{C}^-_{m0} - \mathbf{C}^-_{mi_k})(\mathbf{C}^-_{00} - 2\mathbf{C}^-_{0i_k} + \mathbf{C}^-_{i_k i_k} + \mathbf{R}_k)^{-1}(\mathbf{C}^-_{n0} - \mathbf{C}^-_{ni_k})^T \qquad (8.6)$$

As can be observed, based on the match between the predicted and observed measurements, all the place field centers and their covariances are appropriately updated.

8.3.5 Distinguishing Perceptually Similar Places

Often, different locations in the environment produce the same sensory input (perceptual aliasing) and we need mechanisms to handle such cases. In our model, we suggested the possibility that these aliasing problems in CA3 are resolved by CA1 using dead-reckoning information. It appears that we can use another tool from robotics to elegantly make such distinctions, namely, the *Mahalanobis distance* (Ayache & Faugeras, 1987). Mahalanobis distance is a metric that computes the difference between predicted and observed values and normalizes them by their covariance. This distance measure has a χ^2 distribution with q degrees of freedom where q is the rank of the covariance matrix (Ayache & Faugeras, 1987; Crowley, 1995). Since the errors in our system are assumed to be Gaussian and the system equations are linear, we can use the Mahalanobis test to perform matches. This is used as follows. Suppose the animat is currently at a position $x_{0,k}$ and the sensory inputs activate a CA3 place code. Further, let us assume that this CA3 place code is associated with a CA1 code i_k with estimated place field center $\hat{x}^-_{i_k}$. Given the current estimate of the animat's position $\hat{x}^-_{0,k}$, we perform the following test:

$$(\hat{x}^-_{0,k} - \hat{x}^-_{i_k})^T (\mathbf{C}^-_{i_k i_k} + \mathbf{C}^-_{00} - 2\mathbf{C}^-_{0 i_k} + \mathbf{R}_k)^{-1} (\hat{x}^-_{0,k} - \hat{x}^-_{i_k}) < \epsilon \tag{8.7}$$

where $(\mathbf{C}^-_{i_k i_k} + \mathbf{C}^-_{00} - 2\mathbf{C}^-_{0 i_k} + \mathbf{R}_k)$ is the covariance between prediction $(\hat{x}^-_{0,k} - \hat{x}^-_{i_k})$ and observation (0) (Proposition 7) and ϵ is a threshold that is chosen appropriately (Proposition 8).

If condition 8.7 holds, we assert that the current place has indeed been visited before and that the CA1 unit i_k represents the place. However, if this test fails, it implies that the animat is now at a *new* place that perceptually resembles place i_k visited earlier. In this case, we recruit a new place cell in the CA1 layer and include its parameters in the state vector. Thus, our system creates multiple units in the CA1 layer that respond to the same sensory input in the CA3 layer but are tuned to different centers.

8.4 Frame Merging for Incremental Map Learning

The computational model of spatial learning described above allows an *animat* (a robot simulating the behavior of an animal) to learn metric place maps. Once such a place map has been learned, the animat can be removed from the environment, reintroduced at another place, and expected to *localize*, i.e., determine where it is in the environment with regard to

the place map. This situation is referred to as the *kidnapped robot* problem in robotics, where the robot is kidnapped (or removed) from one place in its environment and reintroduced at another (Engelson, 1994). We have shown elsewhere that our spatial learning and localization system allows the kidnapped robot (or animat) to localize reliably (Balakrishnan *et al.*, 1997). When kidnapped and reintroduced in the environment, the animat moves randomly until it recognizes a place that was visited earlier and which is a part of its place map. It then localizes using the Kalman filter based updates described above.

The primary problem with this approach is that though the animat passes through novel places from its point of reintroduction to its eventual localization, it is incapable of *learning* these *new* places. Any extension of the spatial learning system which will allow the robot to learn these new places and integrate them into its prior place map, is thus of considerable interest. Since the animat learns places in a metric coordinate system with origin at the point of introduction into the environment, such an approach must be capable of learning places in two (or more) different coordinate frames and merging them appropriately. In what follows, we develop an extension of the computational model described above that permits the animat to learn separate place maps in different *frames* and to *fuse* them together in consistent ways.

Suppose the animat has learned a place map and has labeled places with metric position estimates derived from its dead-reckoning system. Let us refer to this frame as f_{old}. Remember that the origin of this frame corresponds to the origin of the dead-reckoning system, i.e., the point of introduction of animat in the environment is the origin of the coordinate system used for labeling places. Suppose the animat is now kidnapped and reintroduced at another place. The animat stores away f_{old} in its memory, and begins a new frame f_{new} at this point of reintroduction. It also resets its dead-reckoning estimates to zero, thereby making the point of reintroduction the origin of its new dead-reckoning frame. It now proceeds as before, learning places and creating EC, CA3, and CA1 cells appropriately in f_{new}. At each step it also checks to see if sensory inputs excite CA1 cells residing in f_{old}. If this happens, then the animat is at a place it has seen earlier in the older frame (f_{old}). The spatial learning system then *merges* the two frames f_{old} and f_{new}, labeling places in both the frames in a uniform coordinate system. This frame merge procedure is described below.

Suppose CA1 unit c fires in f_{new} and m fires in f_{old}. The goal is to merge f_{old} into f_{new}. We do this by changing the position labels of all CA1 units in f_{old} to equivalent labels in

f_{new}. Let $\hat{x}_c^{f_{new}}$ denote the estimated center and $x_c^{f_{new}}$ the true center of the animat's current place in the frame f_{new}. Let $\hat{x}_m^{f_{old}}$ and $x_m^{f_{old}}$ denote the corresponding values of the true and estimated place field centers in frame f_{old}. Since $x_c^{f_{new}}$ and $x_m^{f_{old}}$ correspond to the center of the same place field, albeit in different frames, $x_m^{f_{old}} - x_c^{f_{new}}$ denotes the amount by which frame f_{old} has to be transformed to coincide with f_{new}. However, since the true place field centers are not computable or observable, we will have to use estimated field centers to perform the transformation. Let $\Delta x = \hat{x}_m^{f_{old}} - \hat{x}_c^{f_{new}}$. Assuming a metric coordinate representation, we can update the place field centers of units in f_{old} to equivalent ones in f_{new} via the transformation:

$$\hat{x}_i^{f_{new}} = \hat{x}_i^{f_{old}} - \Delta x \qquad \forall i \in f_{old} \tag{8.8}$$

This converts the place field centers represented in f_{old} into equivalent labels in f_{new}. Now we have to update the covariances between units in f_{old} and those in f_{new}. The following two cases lead to appropriate and consistent updates of the covariances.

Case I: If i and j were both units in f_{old}, from the definition of covariance we have:

$$\begin{aligned} \mathbf{C}_{ij}^{f_{new}} &= E(\epsilon_i^{f_{new}}.(\epsilon_j^{f_{new}})^T) \\ &= E((x_i^{f_{new}} - \hat{x}_i^{f_{new}}).(x_j^{f_{new}} - \hat{x}_j^{f_{new}})^T) \end{aligned} \tag{8.9}$$

The frame transformation described in Equation 8.8 can be used to derive the following expressions:

$$\begin{aligned} x_i^{f_{new}} &= x_i^{f_{old}} - (x_m^{f_{old}} - x_c^{f_{new}}) \\ \hat{x}_i^{f_{new}} &= \hat{x}_i^{f_{old}} - (\hat{x}_m^{f_{old}} - \hat{x}_c^{f_{new}}) \\ x_j^{f_{new}} &= x_j^{f_{old}} - (x_m^{f_{old}} - x_c^{f_{new}}) \\ \hat{x}_j^{f_{new}} &= \hat{x}_j^{f_{old}} - (\hat{x}_m^{f_{old}} - \hat{x}_c^{f_{new}}) \end{aligned}$$

These expressions can be combined to produce:

$$x_i^{f_{new}} - \hat{x}_i^{f_{new}} = \epsilon_i^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}} \tag{8.10}$$

$$x_j^{f_{new}} - \hat{x}_j^{f_{new}} = \epsilon_j^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}} \tag{8.11}$$

Using these reductions in Equation 8.9 allows us to simplify the covariance to:

$$\begin{aligned} \mathbf{C}_{ij}^{f_{new}} &= E((\epsilon_i^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}}).(\epsilon_j^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}})^T) \\ \mathbf{C}_{ij}^{f_{new}} &= \mathbf{C}_{ij}^{f_{old}} - \mathbf{C}_{mj}^{f_{old}} - \mathbf{C}_{im}^{f_{old}} + \mathbf{C}_{mm}^{f_{old}} + \mathbf{C}_{cc}^{f_{new}} \end{aligned}$$

Since the errors in the two frames are independent, the covariance between errors in f_{old} and f_{new} are zero, i.e., $\mathbf{C}_{cj}^{f_{new},f_{old}} = \mathbf{C}_{cm}^{f_{new},f_{old}} = \mathbf{C}_{ic}^{f_{old},f_{new}} = \mathbf{C}_{mc}^{f_{old},f_{new}} = 0$. Thus, covariances between all units i and j belonging to f_{old} can be updated using the expression in Equation 8.12.

Case II: If i was a unit in f_{new} and j was a unit in f_{old}

$$\begin{aligned} \mathbf{C}_{ij}^{f_{new}} &= E(\epsilon_i^{f_{new}}.(\epsilon_j^{f_{new}})^T) \\ &= E((x_i^{f_{new}} - \hat{x}_i^{f_{new}}).(x_j^{f_{new}} - \hat{x}_j^{f_{new}})^T) \end{aligned} \tag{8.12}$$

Here the frame transforms are only applied to unit j (since unit i is already in f_{new}). Thus we have:

$$\begin{aligned} x_j^{f_{new}} &= x_j^{f_{old}} - (x_m^{f_{old}} - x_c^{f_{new}}) \\ \hat{x}_j^{f_{new}} &= \hat{x}_j^{f_{old}} - (\hat{x}_m^{f_{old}} - \hat{x}_c^{f_{new}}) \end{aligned}$$

which leads to: $x_j^{f_{new}} - \hat{x}_j^{f_{new}} = \epsilon_j^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}}$. We use this expression in Equation 8.12. Further, as the errors in the two frames are independent, we use $\mathbf{C}_{ij}^{f_{new},f_{old}} = \mathbf{C}_{im}^{f_{new},f_{old}} = 0$ to obtain:

$$\begin{aligned} \mathbf{C}_{ij}^{f_{new}} &= E(\epsilon_i^{f_{new}}.(\epsilon_j^{f_{old}} - \epsilon_m^{f_{old}} + \epsilon_c^{f_{new}})^T) \\ \mathbf{C}_{ij}^{f_{new}} &= \mathbf{C}_{ic}^{f_{new}} \end{aligned}$$

This allows us to establish covariances between units in the two frames. Once these updates have been carried out, frame f_{old} has been effectively merged into f_{new}.

This frame merge procedure not only preserves the estimates of place field centers but also computes and propagates the covariances of these estimates. However, at the moment, this frame merge procedure is blind to *perceptual aliasing*. A kidnapped animat localizes to the first place that sensorily matches a place it has seen before. If multiple places in the environment produce similar sensory inputs, this procedure may lead to incorrect localization. However, in environments free of perceptual aliasing, the frame merging procedure works as desired.

8.5 Learning and Updating Goal Locations

Since our computational model allows the animat to learn places in a metric framework, goals encountered by the animat can also be remembered in terms of their metric positions.

This idea of computing and remembering goal locations from metric place estimates was first developed in (Redish & Touretzky, 1996). In our work, the animats compute goal positions based on their current dead-reckoning estimates and the estimated distance to the goal. Thus, goals are represented in the same coordinate frame as the place field centers.

Although this idea is appealing, a few issues must be addressed before this approach can be used in practice. Since the goal locations are labeled in terms of dead-reckoning estimates and dead-reckoning is error prone, the computed or remembered position of the goal can be erroneous. We thus need mechanisms to explicitly handle such errors and allow the animat to operate reliably in their presence. This is where Kalman filtering based spatial learning provides a much needed inspiration. Remember that place field centers are also labeled using dead-reckoning information and Kalman filtering allows us to maintain reliable estimates. We can adopt a similar mechanism for updating goal location estimates. In order to do so, we need a goal structure that maintains not only the goal position estimate but also its variance. This can be done as follows.

Let G be a goal. When the animat visits the goal location for the first time, say in step j, the position estimate of the goal $\hat{x}_G$ is initialized to the current dead-reckoning estimate of the animat $\hat{x}_{0,j}$. The variance of the goal estimate $\mathbf{C}_{GG}$ is set to the current variance of the animat's position estimate $\mathbf{C}_{00}$. Suppose the animat visits the same goal location again at time step k. The position estimate of the animat $\hat{x}_{0,k}$ may differ from $\hat{x}_G$ owing to dead-reckoning errors in the intervening animat motions. Further, one of these may be more accurate than the other, and the system must then appropriately update the goal position estimates taking their relative accuracy into account. Let us assume that the position $\hat{x}_G$ is updated as shown in Equation 8.13.

$$\hat{x}_G^+ = \alpha.\hat{x}_G^- + (1-\alpha).\hat{x}_{0,k} \tag{8.13}$$

We now need to determine an appropriate value of α. We will derive such a value by choosing an α that *minimizes* the resulting *variance* (Var) of the goal estimate.

$$Var(\hat{x}_G^+) = Var(\alpha.\hat{x}_G^-) + Var((1-\alpha).\hat{x}_{0,k}) \tag{8.14}$$

Here we make a simplifying assumption that $\hat{x}_G^-$ and $\hat{x}_{0,k}$ are independent, although in reality $\hat{x}_G^-$ is some previous value of $\hat{x}_{0,k}$. However, without such an assumption, keeping track of the

covariance between the two quantities is a rather cumbersome process. With this assumption, Equation 8.16 reduces to:

$$Var(\hat{x}_G^+) = \alpha^2 Var(\hat{x}_G^-) + Var((1-\alpha).\hat{x}_{0,k}) \quad (8.15)$$

$$\mathbf{C}_{GG} = \alpha^2.\mathbf{C}_{GG} + (1-\alpha)^2.\mathbf{C}_{00} \quad (8.16)$$

where $\mathbf{C}_{GG}$ is the variance of the goal position estimate while $\mathbf{C}_{00}$ is the variance of the animat's current dead-reckoning estimate. Minimizing the resulting variance, i.e., Equation 8.16, we get:

$$\frac{\partial \mathbf{C}_{GG}}{\partial \alpha} = 0 = 2.\alpha.\mathbf{C}_{GG} - 2.(1-\alpha).\mathbf{C}_{00} \quad (8.17)$$

$$\Leftrightarrow \quad \alpha = \frac{\mathbf{C}_{00}}{\mathbf{C}_{00} + \mathbf{C}_{GG}} \quad (8.18)$$

Thus, the value of α given by Equation 8.18 minimizes the resulting variance (it can be easily verified that $\frac{\partial^2 \mathbf{C}_{GG}}{\partial \alpha^2} > 0$). Using this expression for α in Equation 8.13 leads us to the following update rules for goal position estimate and variance:

$$\hat{x}_G^+ = \frac{\mathbf{C}_{00}}{\mathbf{C}_{00} + \mathbf{C}_{GG}} \hat{x}_G^- + \frac{\mathbf{C}_{GG}}{\mathbf{C}_{00} + \mathbf{C}_{GG}} \hat{x}_{0,k} \quad (8.19)$$

$$\mathbf{C}_{GG} = \frac{\mathbf{C}_{GG}\mathbf{C}_{00}^T}{\mathbf{C}_{00} + \mathbf{C}_{GG}} \quad (8.20)$$

These update expressions are applied each time the animat reaches the goal location. Goal variance $\mathbf{C}_{GG}$ is initialized to ∞. When the animat reaches the goal the first time (say, in step j), the above expressions automatically set $\hat{x}_G^+ = \hat{x_{0,j}}$ and $\mathbf{C}_{GG} = \mathbf{C}_{00}$. Thereupon, every visit to the goal location results in an update of the goal position estimate and its variance, based on the relative uncertainties in the goal and dead-reckoning estimates. This allows the animat to maintain reliable goal position estimates. It should also be pointed out that if goals are encountered in a particular frame f and if at a later point f is merged into another frame, the goal position estimate and its variance must be appropriately transformed into the other frame, as described in Section 8.4.

8.5.1 Representing Multiple Goals

The mechanism described earlier allows the animat to learn and update the metric position of one goal. What would happen if the animat had to learn and represent more than one goal location?

One approach would be to extend our goal representation model to distinguish between goals based either on sensory features pertaining to the different goals or on sensory features visible from the different goal locations. With such an extension, goals locations would simply be treated as special *places*, and the place learning framework described earlier in Section 8.3 can be specialized to learn a *goal map*.

However, if we did not want to distinguish between goals by using sensory features, we could still develop mechanisms for distinguishing between goal positions based on their locations. This is equivalent to the scenario where all goals provide the same sensory information i.e., they are perceptually aliased. Since goals can be thought of as special places, we can use the same mechanism that we used earlier to distinguish between perceptually similar places. As described in Section 8.3.5, we can use the Mahalanobis test to determine if the encountered goal position is same as some remembered one or not.

Let us assume that the navigating animat encounters a goal at step k. Let the current dead-reckoning position estimate of the animat be $\hat{x}_{0,k}$. Suppose the animat has previously visited a goal (G) at the estimated goal position $\hat{x}_G$. The variance of this remembered goal position is given by $\mathbf{C}_{GG}$. In order to test if the goal encountered by the animat is the same as the one visited earlier, the animat simply computes the following test based on the Mahalanobis distance:

$$(\hat{x}_G - \hat{x}_{0,k})^T (\mathbf{C}_{GG} + \mathbf{C}_{00})(\hat{x}_G - \hat{x}_{0,k}) < \epsilon \tag{8.21}$$

where $\mathbf{C}_{GG} + \mathbf{C}_{00}$ is the covariance matrix of this test (see Proposition 9) and ϵ is an appropriately chosen distance (Proposition 8). Here $\hat{x}_G$ is taken to be the prediction of the goal position while $\hat{x}_{0,k}$ is the observed position of the goal.

If this test is satisfied, the goal encountered by the animat is said to be the same as the one visited previously (G). However, if this test fails, the animat is at a new goal location. The animat then learns this new goal location using the algorithm described earlier and adds this to its goal memory. Note that any number of goals can be learned and remembered by the animat.

8.5.2 Navigating to Goals

Once the animat has learned the locations of different goals it can navigate to specific ones when in need. Two processes are involved in the realization of such goal-directed behaviors.

First, from its memory of different goals encountered and represented, the animat must choose one goal location to navigate to. Second, once an appropriate goal has been identified, the animat must move in such a manner as to approach the remembered location of the goal.

It is not clear how animals address these two issues, particularly the former. For instance, assuming all the goals offer that same reinforcement to the animal, how do animals choose the goals to approach? Are goals chosen based on their reliability or their nearness to the animal's current position? Are recently visited goal locations given preference over those visited a while ago?

Our computational specification allows us the liberty of implementing and testing different goal selection hypotheses. In our work, we have used a goal selection scheme that uses a combination of recency of goal encounter, its closeness to the current position of the animat, and a measure of the confidence associated with the goal. This mechanism will be described in Section 10.4.

8.6 Model of Redish and Touretzky

Among the computational models of hippocampus-based spatial learning and navigation that have been proposed, our model bears the closest resemblance to that proposed by Redish and his colleagues (Wan *et al.*, 1994; Redish & Touretzky, 1996). In their model, places are represented in terms of an ensemble of active place cells. Each place cell is tuned to the identities and ego-centric positions of *two* randomly chosen landmarks visible from the current place. The place cells also respond to the *retinal* angle between two visible landmarks (possibly different from the two chosen above) and to the dead-reckoning generated position estimate of the animat. Each of these inputs are Gaussian functions and the overall firing of a place cell is simply a product of these Gaussians. Further, the unit firing is *fuzzy*, in the sense that unknown quantities simply drop out of the product. For instance, if the animat is reintroduced in the environment, it does not have accurate dead-reckoning estimates. In this case, the place cells fire only in response to sensory inputs and the Gaussian function corresponding to the position estimate match simply drops out of the calculations. Importantly, by storing ego-centric positions of landmarks and an angle between two landmarks, their model allows the animat to perform head-direction resets.

As can be observed, this model is closely related to ours since both are based on the locale

hypothesis of O'Keefe and Nadel (1978) and on its implicated substrate in the hippocampus. Thus, both these models develop a functional characterization of the hippocampus wherein dead-reckoning information is combined with sensory inputs to create a metric representation of space. Further, both models assume that sensory inputs are provided to the place learning system in terms of recognized landmark information along with their *vector* position relative to the animat. Finally, both assume that *goals* are represented in terms of their metric positions and goal-directed navigation is simply a matter of localizing in the metric framework and then proceeding to the remembered position of the goal.

However, the two models differ in some significant ways. For instance, unlike their model, we acknowledge that errors exist in the sensory and dead-reckoning input streams and our computational framework explicitly addresses the issue of information fusion from erroneous (or uncertain) sources. Also, by formulating the place learning and localization problem within the framework of Kalman filtering, we have derived *update expressions* that perform *stochastically optimal* updates. In contrast, their model simply learns a place map *without* performing any subsequent updates of places. Importantly, our model includes a novel mechanism for handling perceptual aliasing problems using the CA1 layer. Also, their model includes a mechanism for learning and initializing the head-direction estimates of the animat. Our model is yet to incorporate this feature.

In their simulations of behavioral experiments (Redish & Touretzky, 1996), the *animats* do not navigate. The animats are simply placed at different positions in the environment and learn places based on accurate position estimates provided by the user. Our behavioral simulations on the other hand, have navigating animats (Balakrishnan *et al.*, 1998c). Finally, in their simulations goals are assumed to coincide with the origin of the coordinate framework. Since animals commonly remember multiple goal locations and often plan and execute multi-destination routes (Gallistel & Cramer, 1996), it is not clear how this goal representation will scale up to such tasks. In our model goals are represented in terms of dead-reckoning coordinates and multiple goals can be easily represented, as we have shown in Section 8.5.1. As we detailed in Section 8.4, our model also allows the animat to incrementally learn local spatial maps and integrate them into global ones.

8.7 Discussion

In this chapter we have developed a computational model of the cognitive map hypothesis of hippocampal spatial learning (O'Keefe & Nadel, 1978). According to this hypothesis, the hippocampal system learns places and associates them with dead-reckoning estimates, thereby learning a metric map of the environment. This metric representation allows the animat to navigate to arbitrary goals and locations on a direct trajectory. However, in order for such a representation to exist, the dead-reckoning estimates of the animal must be faithful and reliable, even in the face of perceptual aliasing (when multiple places appear sensorily identical). Further, since the sensory and dead-reckoning input streams can have errors associated with them, it is also imperative that the model be capable of handling such uncertainties.

Based on the parallel between the requirements of hippocampal spatial learning and the Kalman filtering approach for probabilistic robot localization, we have developed a Kalman filter framework for our localization model. Not only does this approach handle uncertain data, but also provides stochastically optimal rules for information updates in the model. The model learns places based on sensory data, and through the Kalman filter process, also learns the centers of different places in terms of dead-reckoning estimates. Using a related tool called the Mahalanobis distance measure, the system is also capable of distinguishing between perceptually similar places and learns and localizes in their presence. The place field centers as well as the current dead-reckoning position of the animat are represented by estimates (with associated covariances) and are updated appropriately when the animat visits or revisits places. With frequent revisits to a known place, the uncertainty associated with that place field center can be seen to decrease and the estimated place field center converges to the true center. The system also handles relocalization (kidnapped robot problem) very easily by setting the variance of the robot's position estimate to ∞ and using the same Kalman filter approach.

We have also developed a computational mechanism for incrementally learning local maps and fusing them into global ones. In addition, we have also proposed a mechanism to represent and update the positions of goal locations in the animat's environment. An extension of this mechanism, based on the Mahalanobis distance measure, allows the animats to differentiate between and learn the locations of multiple goals in the environment. Using these mechanisms and an appropriate navigation strategy, the animat can navigate to specific goals.

Although we have not fully developed it yet, we strongly believe that the hippocampal

formation represents spatial information at many different levels. We believe that the CA3 recurrent collaterals use motion information to associate places, thereby capturing a topological description of the environment. We also suspect that the place cells recently discovered in the subiculum encode ego-centric information, making it possible to initialize and reset head-direction estimates in the postsubicular area. These regions can thus be thought of as encoding a *directional-map* of the environment. Some researchers have begun exploring similar hypotheses (Redish & Touretzky, 1998).

Our model makes several key neurobiological and behavioral predictions. For instance, our model predicts that perceptually identical places in the same environment will excite the same population of cells in the CA3 layer but different populations in the CA1 layer. However, no differences have been observed in the firings of CA3 and CA1 units. How can we explain this inconsistency? It is our opinion that the behavioral experiments performed to date did not allow perceptual aliasing to occur. Even in rodent experiments in mazes of different kinds, the maze arms are perceptually different if we assume that the animals were not disoriented, i.e., they had faithful head-direction estimates. If environments supporting perceptual aliasing can be designed, our hypothesis can be tested through recordings from CA1 and CA3 cells.

In our model of hippocampal spatial function, CA1 firings are maintained by dead-reckoning in darkness. However, CA3 cells have also been found to fire in darkness but they are not influenced by the dead-reckoning system in our model. How can our model account for the firing of CA3 cells in darkness? A few different hypotheses can be forwarded. First, it is possible that the recurrent collaterals encode a topological relationship between places and thus provide a different (second?) dead-reckoning system. Another explanation could be that the CA3 cells associated with CA1 cells at a given place are influenced by *top-down* activations, which causes them to fire in darkness. Finally, it is also possible that dead-reckoning based position information is encoded in cell firings further upstream (e.g., in the EC cells), which is consistent with the reports of Feigenbaum and Rolls (1991), who have found evidence of not only egocentric spatial encodings but also allocentric encodings in the primate hippocampus. These explanations need to be explored in more detail.

Our model also predicts that animals can reduce their computational complexity of localization by navigating either randomly or from one place field center to another (since this allows the Kalman filter based update system to ignore covariances). A related behavioral prediction

is that an exploring animal will search in gradually expanding trajectories, as this allows the propagation of reliable dead-reckoning estimates further and further away from its starting position. These predictions can be verified through appropriately designed experiments.

Assuming that the hippocampus indeed performs computations similar to Kalman filtering, the next question to ask is how these computations might be realized by the hippocampus. Although we cannot detail the precise mechanisms by which they occur, we can definitely suggest some possibilities for the neural basis of such computations. Based on the structural properties of the CA1 layer, O'Keefe (1989) argued that it was capable of performing matrix inversions through an iterative process. Since matrix inversions are required in our model (in the computation of the Kalman gain and the Mahalanobis distance), and we have localized these computations to the CA1 layer, O'Keefe's argument can be directly applied in our context. Further, Buzsaki (1989) proposed a theory of hippocampal learning according to which *sharp waves* observed in resting rats were indicative of *consolidation* of memory traces via successive firings of CA3 populations affected by the CA3 recurrent collaterals. In our model, this might provide the vehicle for the propagation and update of the state estimates and covariances required by Kalman filtering. These predictions have to be verified through systematic computational and behavioral experiments.

In the following chapters we will demonstrate an implementation of this computational model and highlight a number of its properties. We will also use the implementation to provide a computational simulation of some behavioral experiments performed with rodents.

9 EXPERIMENTAL EVALUATION OF THE COMPUTATIONAL MODEL

In the previous chapter we developed a computational specification of the cognitive map hypothesis of hippocampal spatial learning. Importantly, our characterization assumed that the information streams contributing to spatial learning, namely sensory inputs and the dead-reckoning system, were error-prone, and we developed mechanisms to learn and localize reliably in their presence. In addition, our model included a mechanism for detecting and overcoming perceptual aliasing problems. In this chapter we describe an implementation of this computational model and also provide results of some simulation experiments. The results show that our model is capable of learning places and localizing effectively. We also show that the model scales well with increasing dead-reckoning and sensing errors, and produces a compact, multi-resolution metric representation of space.

9.1 Implementation of the Computational Model

In this section we discuss issues related to the implementation of the computational model developed in the previous chapter. In particular, we provide precise descriptions of the algorithms used for place learning and localization.

9.1.1 Sensory Inputs

We assume that our simulated robot is endowed with a set of sensors that provide information pertaining to landmarks in the environment along with their positions relative to the robot. These sensors are assumed to be *virtual* and can be implemented on real robots through a variety of physical devices. Although landmark recognition is error-prone in general, we assume that the landmarks are recognized accurately. However, the range information is

considered imprecise and the position of the landmark relative to the robot is assumed to be corrupted by zero-mean, white Gaussian noise with standard deviation σ_S per unit distance.

9.1.2 Vector Representations in EC Cells

In our model, EC place cells function as *spatial filters* and respond to particular kinds of landmarks at specific positions relative to the animal. Thus, EC cells encode vectors to specific landmarks. We use an EC representation where each cell stores some internal description of the landmark to which it responds, as well as the Cartesian position of the landmark with respect to the animal. This landmark position is considered *allocentric*, i.e., it is not dependent on the direction in which the robot is facing. This stored information is matched against incoming sensory information. While the landmark descriptions are matched directly, the currently observed landmark position is matched with the stored position using a *two-dimensional Gaussian function* with variance σ_L^2. Thus, given that the landmark descriptions match, each EC cell fires over a roughly circular region around the stored position (called the EC field center) with the firing being Gaussian in nature (stronger near the center and gradually decreasing towards the periphery).

The algorithm for determining EC layer firing is given below:

1. Set the EC output to zero, i.e., $\forall$ EC cells i, ECOutput[i]=0.

2. For each landmark L currently sensed at a position (x_L, y_L) relative to the robot, do:

   (a) Determine the activation, ECAct[i], of each EC cell i using the matching procedure described above.

   (b) If none of the cells produce an activation greater than a threshold Γ_{EC}, recruit a new EC cell j. Set its landmark description to the identity of the landmark L and its field center to (x_L, y_L). Set ECOutput[j]=1.0. Exit.

   (c) Else, determine each cell i that fires above the threshold Γ_{EC}.
   Set ECOutput[i] = max {ECOutput[i], ECAct[i]}.

Thus, EC units are created when landmarks are observed in positions not seen before. Also, each EC cell responds to landmarks in an ellipsoidal region about the EC center. This region is circular in our simulations with radius computed in Proposition 1.

9.1.3 Place Codes in CA3

For simplicity, our implementation assumes that the place codes in CA3 are given by single units rather than ensembles of them (although the theory can be easily extended to handle ensemble coding of space). Each CA3 cell is connected to EC cells that are active in a given place. The algorithm used to set up CA3 cells is given below:

1. Determine the output of the EC layer using the algorithm described in Section 9.1.2.

2. Compute the output of each CA3 cell i as follows:
   CA3Output[i] $= \sum_j ECOutput[j] \times w_{ij}$
   where w_{ij} is the weight between EC unit j and CA3 unit i.

3. If none of the CA3 units produce an activation greater than Γ_{CA3}, recruit a new CA3 unit i and connect it to each EC unit j with a weight given by:
   $w_{ij} = \frac{ECOutput[j]}{\sum_k ECOutput[k]}$
   Exit.

4. Else, determine the CA3 unit k with the highest activation and declare it the *winner*.

Thus, a place is said to be recognized if the winning CA3 place unit produces an activation greater than the threshold Γ_{CA3}. If all the CA3 units have an activation below the threshold, the robot is said to be in a new place and a new CA3 unit is created.

9.1.4 Unique Place Recognition in CA1

Each CA1 unit in our model is connected to a CA3 unit and to path-integration information derived from the EC. Note that for the sake of simplicity, we again use single units to denote the place cells of CA1. The algorithm for determining CA1 activation is as follows:

1. If a new CA3 unit was created, recruit a new CA1 unit and connect it to the newly created CA3 unit with a weight of 1. Also, store the x and y coordinates of the current position estimate from dead-reckoning with the new CA1 unit. Exit.

2. Else, for each CA1 unit i connected to the winning CA3 unit, compute the Mahalanobis distance between the current position estimate from dead-reckoning and the position estimate stored at the unit i.

3. Determine the winner, i.e., the CA1 unit j producing the least Mahalanobis distance.

4. If this distance is less than Γ_{CA1} update the position estimate as well as the stored place field centers using the Kalman filter update rules. Exit.

5. Else, create a new CA1 unit and connect it to the winning unit in the CA3 layer with a weight of 1. Store the current dead-reckoning position estimate with the new CA1 unit.

Thus, CA1 units are created when the animal visits a new place or if the Mahalanobis distance of the best matching place happens to be greater than the threshold Γ_{CA1}. This scheme leads to multiple CA1 units connected to the same CA3 unit, one for each of the perceptually aliased regions represented by the CA3 unit.

9.2 Experiments

This section provides details of the simulation environment used and the experiments that were performed to evaluate the spatial learning model developed in this paper.

9.2.1 Details of the Experiment

The simulation environment for our experiments consisted of a largely open room of size 20×20 units, with impenetrable, delimiting walls. The room also contained six identical landmarks, as shown in Figure 9.1. In our experiments the robots only represented landmarks in their spatial maps and not the walls. Consistent with the inputs to the animal hippocampus, we assumed that detection and recognition of landmarks were performed elsewhere and only recognized landmark information was provided to the spatial learning system. Further, in most experiments, the sensor ranges were assumed to be limited, which made some of the landmarks undetectable from certain positions in the room. For instance, with maximum sensing range limited to 10 units, the robot was only capable of detecting landmark 1 or 2 in the shaded regions shown in Figure 9.1. Since all the landmarks were identical, these two regions provided one instance of perceptual aliasing in this environment. It should be noted, however, that increasing the maximum sensing range (e.g., to 30 units) easily removed this aliasing.

The robot was assumed to have a faithful compass (equivalently, the animal was assumed to have a faithful head-direction estimate) and thus knew the *allocentric* bearings (with respect

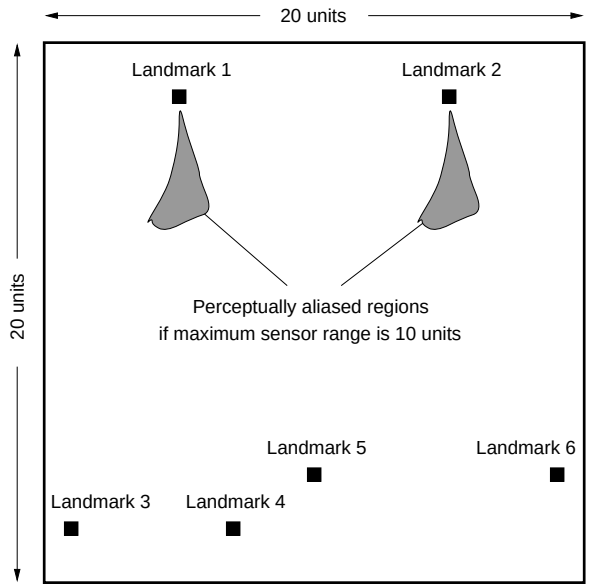

Figure 9.1 The simulation environment with six identical landmarks. The walls are impenetrable. If the maximum sensor range is limited to 10 units, the shaded regions are perceptually aliased, i.e., the robot receives identical sensory inputs in the two regions.

to true north or the origin of the head-direction system) of the sensed landmarks. In addition to recognizing the landmarks and their bearings, the robot sensors also provided estimates of distances to the sensed landmarks. However, this range sensing was assumed to be error-prone, and the error was modeled as a zero-mean Gaussian with standard deviation $\sigma_S = 0.01$ per unit distance. This range error of 1% per unit distance was chosen to be compatible with contemporary sonar-based distance ranging (Everett, 1995).

As described earlier, the sensory information pertaining to the range and allocentric bearings (hence vectors) of detected landmarks were provided as input to the EC layer. The activations of the EC, CA3, and CA1 layers were then computed using the algorithms described earlier. Our simulations used a value of $\sigma_L^2 = 1.0$ as the variance of the distance-matching Gaussians of the EC units. The threshold of the EC units was taken to be $\Gamma_{EC} = 0.6$. As shown in Proposition 1, these choices allowed the EC cells to respond to landmarks within a circular region of radius 1.01 units from the EC field center. The activation threshold of CA3 units were also taken to be $\Gamma_{CA3} = 0.6$. Since CA3 fields were simply *weighted intersections* of EC fields, this choice led to CA3 place fields smaller than or equal in size to the EC fields.

The threshold for the Mahalanobis distance test in the CA1 layer was taken to be 4.61. The rationale for this is provided in Proposition 8.

The robot was introduced into this environment at some position, usually randomly chosen, with a random orientation. The dead-reckoning estimates of the robot were set to (0, 0) signifying that the origin of dead-reckoning coincided with this point of first introduction of the robot in the arena. The robot was then allowed to perform an *a-priori* fixed number of sensing, learning, and action steps. At each such step, the robot obtained sensory inputs as described earlier and executed a movement action. Elsewhere we have shown results of a robot following a circular trajectory (Balakrishnan *et al.*, 1997). In the experiments reported here, the robot executed a random-walk, i.e., it turned in a random direction and moved forward in that direction by $\mathsf{motionStep} = 1$ unit. However, these attempted motions were considered error-prone and the motion errors were modeled as zero-mean Gaussians with $\sigma_M = 0.5$. The robot was assumed to have an *active dead-reckoning* mechanism, one that tracked its *actual* motion rather than its *intended* motion. However, this dead-reckoning was also error-prone with the errors modeled as zero-mean Gaussians with $\sigma_D = 0.1$.

Once the robot had learned a place map it could be removed from the environment (kidnapped) and reintroduced at some other arbitrary place. If the robot had learned a faithful place map, i.e., it had explored the environment well, it must be capable of relocalizing accurately. However, if the environment had perceptually aliased places, the robot would have problems relocalizing. It must then move in a fashion so as to resolve the ambiguity as quickly as possible. Although some techniques had been developed for this problem (e.g., (Dudek *et al.*, 1995; Schuierer, 1997) for polygonal map representations), we chose to let the robot continue its random-walk until it was able to resolve the ambiguity.

The robot relocalized as follows. When it was kidnapped, the robot set the variance of its position estimate to ∞, signaling a complete *distrust* of its dead-reckoning estimate. It also zeroed the covariances between its position estimate and the place field centers that it had learned, thereby accepting its learned map as an accurate source of information. Upon reintroduction in the environment, it simply performed a random-walk. At each step it obtained sensory inputs and applied them to the EC layers of its hippocampal model. Using the algorithms described in Section 9.1, the activations of the EC and CA3 layers were computed. If a single CA3 unit had a above-threshold activation and was associated with a single CA1

unit, the robot was said to be in a perceptually unique place. The robot then localized using the Kalman filter update mechanism, which effectively updated the dead-reckoning estimate of the robot to the center of the place where the robot happened to be at that time. However, if either of these conditions failed, i.e., multiple CA3 units had an above-threshold firing or the CA3 winner was associated with multiple CA1 units, the robot did not localize. It simply continued its random motion.

9.2.2 Spatial Learning In the Presence of Perceptual Aliasing

We will now present results that demonstrate the ability of our model to distinguish between perceptually aliased places and learn reliable spatial maps via exploration.

In this experiment the maximum range of the robot sensors were limited to 10 units. As explained earlier, this creates perceptual aliasing in two regions of the simulation environment as shown in Figure 9.1.

In the exploration phase, the robot was introduced at perceptually aliased region B as shown in Figure 9.2. The picture on the left in Figure 9.2 shows the place field centers in the CA1 layer and the circles around the position estimates represent the 2.5σ boundary, where σ^2 is the variance of the corresponding place field estimate. The rationale for displaying the 2.5σ boundary is discussed in Proposition 10. The actual current position of the robot is shown by "▲" with the orientation of the triangle denoting the robot orientation. Further, the dead-reckoned position estimate of the robot's position is shown by "◇" with the dashed-circle denoting the 2.5σ boundary, with the σ denoting the variance of this position estimate. The picture on the right of Figure 9.2 shows the *place map*, i.e., the firing fields of all the place units created in the CA1 layer. Since this figure shows the state of the system after one motion step, only one CA1 unit was created thus far and the place map contains only one CA1 field. This CA1 unit (and the corresponding CA3 unit) are activated by just one EC unit which responds to the sole landmark visible from the robot's start position, namely landmark L2. Thus, the size and shape of this CA1 field is the same as that of the corresponding CA3 field, which in turn is same as the field of the sole EC unit.

Figure 9.3 shows the result of spatial learning after 10 random moves by the robot. Due to the presence of dead-reckoning errors the difference between actual and estimated positions of the robot starts increasing, as can be observed in the picture on the left. Further, the variance

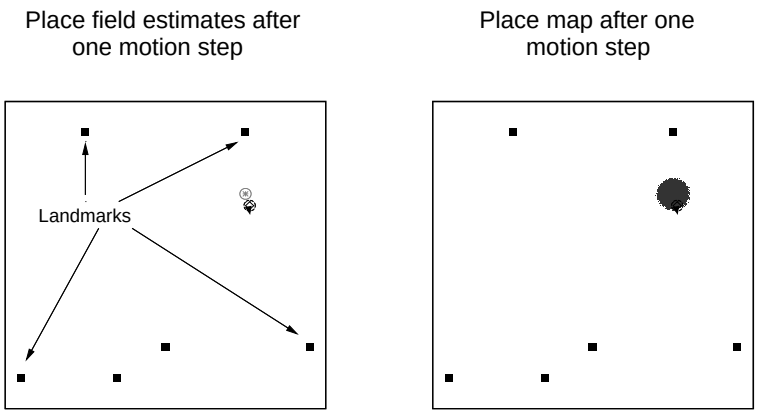

Figure 9.2 State of the system after one step. Place field centers along with their variances (displayed as 2.5σ circles) are shown on the left, while the place map (or CA1 firing field) is shown on the right.

of the robot's position estimate (denoted by the dashed circle around "$\diamond$") and the variances of successively created CA1 units (gray circles) can be seen to increase. This is because the Kalman filtering mechanism reduces variances of the estimates only when the robot revisits familiar places, and it has not done that in the course of its current trajectory. The place map shown in Figure 9.3 also shows that the firing fields of units can vastly differ in shape and size. This is because CA3 (and hence CA1) fields are conjunctions of the firing fields of EC units. Thus if multiple EC units are active at any given place, the corresponding CA3 (and hence CA1) field will be an *intersection* of these EC firing fields and as a result may be much smaller than individual EC fields.

Figure 9.4 shows the state of the system after 100 steps of exploration. As can be readily observed, there is a significant drift between the true position of the robot and its dead-reckoned position estimate. Further, the variance of the position estimate is also quite large. This is largely because the random-walking robot has not revisited places it witnessed early in its trajectory and Kalman filter update mechanisms are unable to kick in. An important thing to note is that the robot is now in region A, which is perceptually identical to the region where the robot originally started it motions (region B). However, the Mahalanobis test ensures that this perceptually familiar place is classified as a new (and different) location than the robot's starting position. Thus, the learning system functions correctly even in the presence of

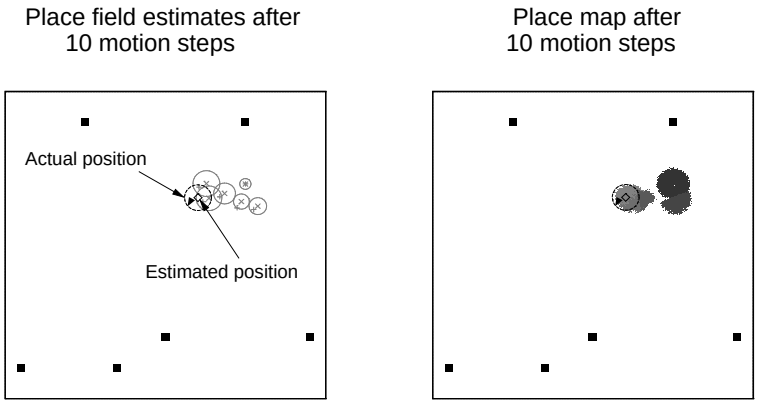

Figure 9.3 State of the system after 10 steps of exploration. The variance of the place field estimates steadily increase without revisits to places visited earlier. Place map shows firing fields of different shapes and sizes.

perceptual aliasing and can easily distinguish between such aliased places based on its current *confidence* (measure in terms of its variance) in its position estimate.

Specifically, CA3 unit 0 was created by the system at its starting position to respond to the sensory inputs available there, and CA1 unit 0 was created to represent the place in terms of dead-reckoning coordinates. As can be observed from Figure 9.5, CA3 unit 0 also responds to the sensory inputs at the robot's current position in the alternate perceptually aliased region. However, this does not cause any problems because the Mahalanobis test fails during the comparison of the place field center of CA1 unit 0 and the robot's current dead-reckoning estimates in region A, leading to the creation of a new CA1 unit numbered 61. It is clear from Figure 9.5 that perceptual ambiguities in the sensory space, reflected in the firing of CA3 unit 0, are resolved through the use of dead-reckoning information in the CA1 layer. In these firing fields, lighter shades correspond to regions of stronger firing.

Figure 9.6 shows the state of the system after 200 exploration steps. It can be observed that the drift between estimated and true robot position has increased even further. What is more important is that the system also guesses this and indicates it by the corresponding increase in the variance of this estimate (large dashed-circle). However, the random-walking robot seems to be headed towards its start region where it has comparatively more accurate

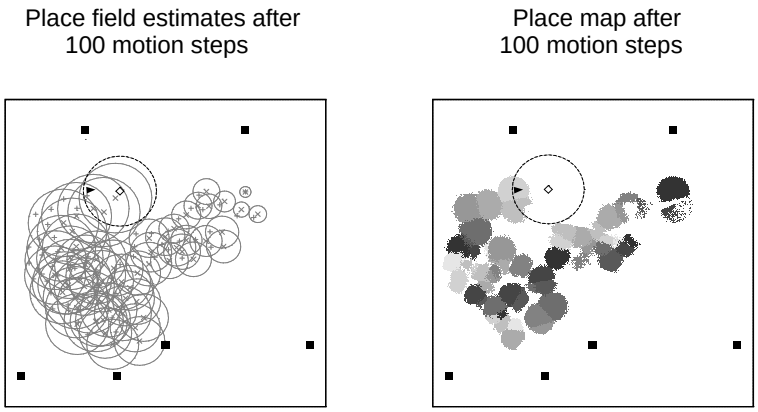

Figure 9.4 State of the system after 100 exploration steps. The robot is now at a place that is perceptually aliased to its starting place. However, the Mahalanobis test correctly classifies this as a new place. Some of the CA1 cells have extremely small firing fields.

place field estimates and if it revisits those places it can perform significant corrections of its position estimate.

Figure 9.7 shows the state of the system at the end of its training (or exploration) phase i.e., the completion of 250 motion steps. It can be seen that the robot indeed revisits places created early in its exploration trajectory. At such locations, Kalman filter updates lead to corrections and updates of the position estimate of the robot and the place field centers (as well as their associated variances and covariances). Thus not only do the variance circles shown in the left picture in Figure 9.7 move (with changes in place field centers and position estimate) but they also shrink (decrease in variance). Notice the consequently accurate dead-reckoning estimate of the robot. With frequent visits to very familiar places, the robot can indeed update its place map to arbitrary levels of accuracy.

The trajectory followed by the random-walking robot during this exploration phase is shown in Figure 9.8. Here, "•" denotes the starting position of the robot and "○" denotes the final robot position.

Thus our model allows the robot to learn places and their metric positions, distinguish between perceptually similar regions, and learn faithful place maps in their presence. Further, the model allows the robot to effectively correct its estimates during revisits to familiar places.

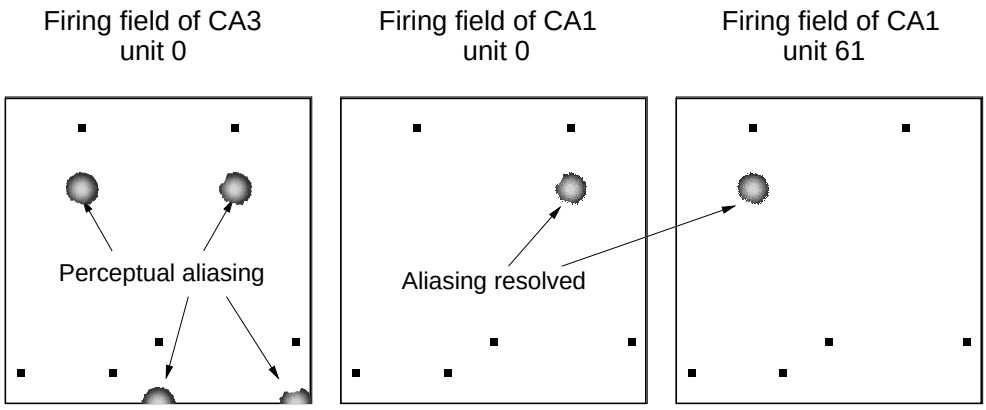

Figure 9.5 Firing fields of CA3 and CA1 units. CA3 unit 0 responds to perceptually aliased places. This ambiguity is resolved by the Mahalanobis distance in the CA1 layer with CA1 units 0 and 61 responding to the two aliased places.

9.2.3 Relocalization in the Presence of Perceptual Aliasing

We demonstrated the ability of our spatial learning and localization model in learning faithful metric place maps even in the face of perceptual aliasing. We will now show the ability of our model to *localize* when introduced into previously learned environments.

In our experiment, we let the robot learn a place map as explained in the previous section. After 250 steps of exploration and map-building, we kidnapped the robot from its current position and introduced it in the perceptually aliased region A. The variance of the position estimate of the robot is set to ∞ to signal a complete distrust of its dead-reckoning position estimate. As explained earlier, in order to localize, the robot moves randomly until a perceptually *unique* place is found. A place is unique if only one CA3 unit fires above its threshold and is associated with only one CA1 unit. The robot then uses the Kalman filtering mechanism to relocalize at that place.

Figure 9.9 shows the state of the robot after one and five steps of relocalizing. During this period the robot does not localize because it detects perceptual aliasing (via above-threshold firing of multiple CA3 units or by the presence of multiple CA1 units connected to the CA3 winner. It thus keeps moving randomly.

After 10 steps the robot finally reaches a place south-west of landmark L1. This place

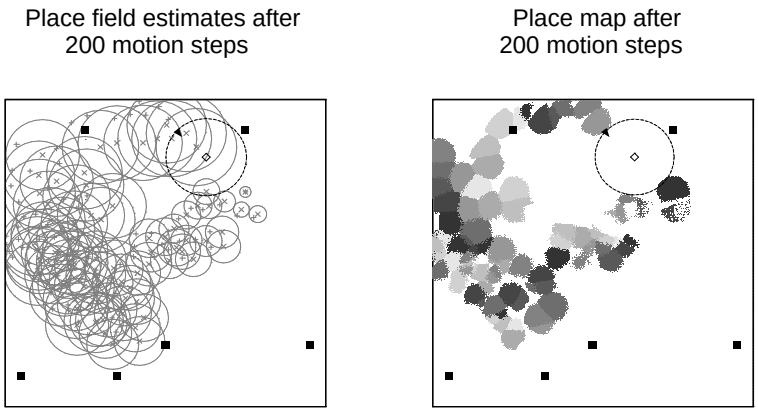

Figure 9.6 After 200 exploration steps the error between the actual and estimated positions of the robot is quite alarming. However, the robot cannot correct this error without revisiting places for which it as accurate estimates.

is perceptually unique and the robot relocalizes as shown in Figure 9.10. The Kalman filter mechanism not only updates the current position estimate of the robot to an approximately correct value, but also reduces the variance of its position estimate from ∞ to a much smaller one. Once relocalized, the robot can move around and further improve its position estimate and variance by visiting other places in its map, as shown on the right in Figure 9.10.

Although our current scheme allows the robot to relocalize correctly in the presence of perceptual aliasing, this approach will not always work. For instance, if the robot encounters only *one* of the perceptually aliased regions during its exploration but is introduced into the *other* after being kidnapped, it will localize incorrectly. Improving relocalization algorithms to work in such scenarios is an area of active research (Dudek *et al.*, 1995; Schuierer, 1997).

9.2.4 Empirical Evaluation of the Spatial Learning Model

We also performed a number of experiments to empirically evaluate and validate our model of spatial learning. The primary attributes under study were the usefulness or need for spatial learning, the scalability of the system (particularly from the point of view of handling increased sensing and dead-reckoning errors), the effect of these parameters on the numbers of EC, CA3, and CA1 units required, and the representation of space learned by this system. These

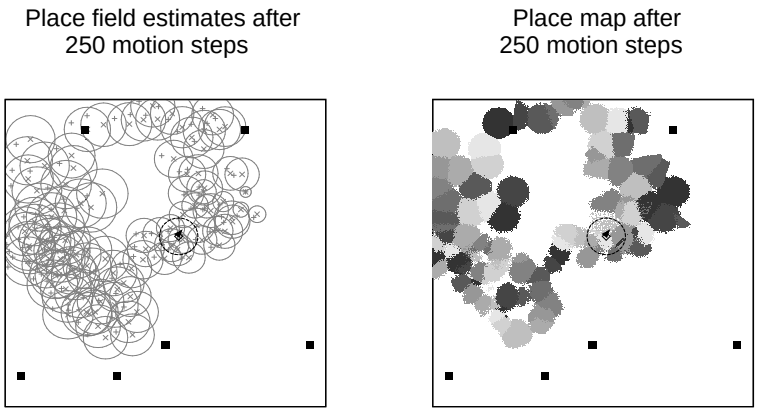

Figure 9.7 System state at the end of exploration. The robot finally revisits familiar places and significantly corrects its position estimate error.

experiments were performed by choosing 100 random exploration trajectories for the robot. The robot was then allowed to execute 250 exploration steps along each of these trajectories under the different parameter values constituting the experiment and the results were recorded. By choosing 100 fixed trajectories for the experiments, we basically guaranteed that the path taken by the robot and the sensory inputs it received during its motion were identical for each of the experiments. Thus, any differences in the results would then necessarily be a result of the particular parameter values used. These results are presented in the following sections.

9.2.4.1 Usefulness and Need for Spatial Learning

Our model performs spatial learning and localization. It not only learns places in terms of dead-reckoning position estimates but also updates and corrects the robot's position estimate using the Kalman filtering mechanism. Is this necessary? Can the robot be expected to perform well without such a learning and localization mechanism?

Figure 9.11 shows the performance of the robot with and without the spatial learning mechanism. In this case, the error in the final position of the robot, i.e., the Euclidean distance between the final true position of the robot and its estimated position, was the measure of performance. As can be observed, without spatial learning and localization, the distance error increases almost linearly with increase in the standard deviation of the Gaussian dead-reckoning

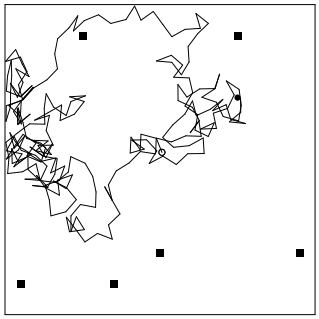

Figure 9.8 Trajectory followed by the robot during the exploration phase.

error. For instance, with dead-reckoning error with variance 1, the error without learning is approximately 20 units while the system with learning has a net error of approximately 5 units. This error is large considering the fact that the simulation environment was of size 20× 20 units. Further, it can be readily observed that the two curves are increasingly divergent.

Similar results are obtained if the sensory range of the robot is set to 30 units, i.e., even if the robot is capable of observing all the landmarks from all positions in the room. However, with increased sensing range, the errors in the final robot position are slightly lesser than earlier, as shown in Figure 9.12. This is directly attributable to the sensor range, since with a range of 30 units there is no perceptual aliasing in the environment. Since all the places are sensorily unique, this allows dead-reckoning drifts to be corrected. However, if perceptual aliasing exists, large dead-reckoning variances lead to the Mahalanobis test being satisfied for places that are significantly distant but sensorily similar. This leads to wrong updates of the robot's position estimate and contributes to the final position error.

Given the fact dead-reckoning using contemporary, affordable devices is quite erroneous, it is clear that the use of our spatial learning and localization mechanism is of immense value. Although the resulting position errors cannot be eliminated completely they can be significantly reduced by updating the robot's position estimates based on the Kalman filtering mechanism.

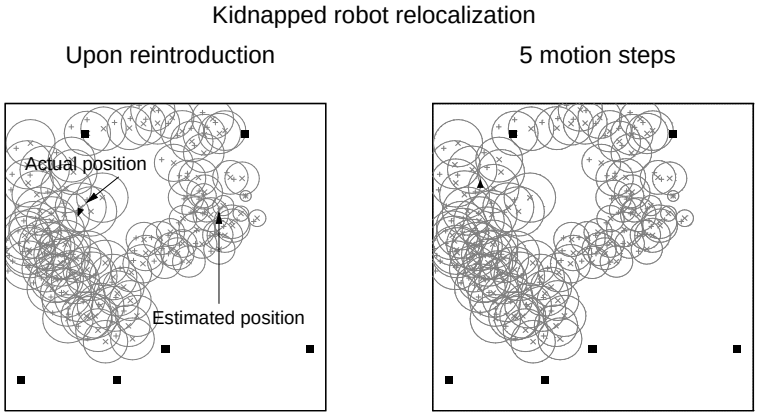

Figure 9.9 The kidnapped robot does not relocalize because it detects perceptual aliasing. It simply moves randomly looking for a sensorily unique place at which to localize.

9.2.4.2 Importance of Updating Place Field Centers

A critical contribution of Hippocampal Kalman filtering is its ability to update not only the dead-reckoned position estimate of the robot but also the place field centers of the learned map. The advantage of this feature can be observed in Figure 9.13, which shows the distance error in the final robot position as a function of the dead-reckoning error, with and without the update of place field centers. Updates can be seen to lead to lower errors. For instance, with $\sigma_D = 0.6$, updates lead to an average distance error of 3 units, while without updates the average error is approximately 4 units. This difference is significant considering the size of the room which is only $20\times$ 20 units.

Thus, the ability to update not only the robot's position estimate but also the estimated positions of the different places, is an important advantage offered by our computational model. Although other hippocampal models in the literature support metric spatial learning (Redish & Touretzky, 1996), they do not have mechanisms to update place field centers.

9.2.4.3 Effect of Range Sensing Errors on the Numbers of Units

In this experiment, we explored the effect of increased sensing error on the numbers of EC, CA3, and CA1 units required in our model. As can be observed in Figure 9.14, the *mean*

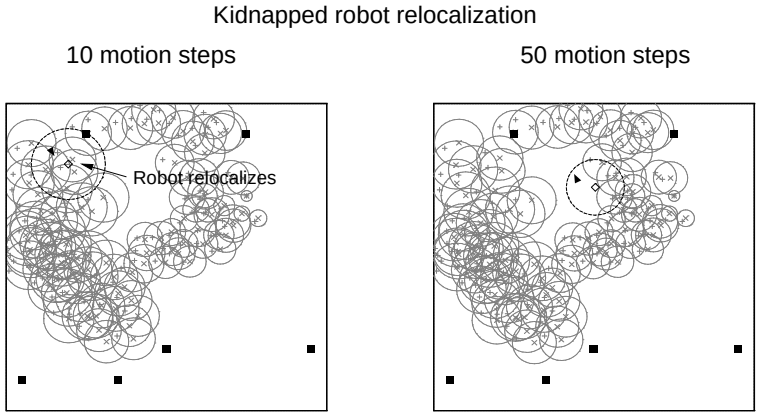

Figure 9.10 After 10 random steps the robot reaches a sensorily unique place. The Kalman filtering mechanism in our model relocalizes the robot to this place, updating not only its position estimate but also its variance. The localized robot can then move to other places on the map and further improve its position estimate as shown in the picture on the right.

number of EC units required (averaged over the 100 trajectories), can be seen to grow almost linearly with the standard deviation of the Gaussian range sensing error. Since the robots are following the same set of 100 trajectories in each case, the increased number of EC units required is a direct result of the increased range sensing error. With large range errors the robot gets (possibly) radically different vectors to the same landmark from the same position on each sensing step. Owing to the limited size of the EC fields, these different vectors must be represented by new EC units. For instance, with deviation of 0.2 the system only requires approximately 150 EC units, while with a deviation of 0.8 it requires 250 EC units.

It can also be observed that the numbers of CA3 and CA1 units increase very slowly above a certain level of error. This is because the sensory inputs in the EC layer are still able to activate the same place code in the CA3 layer, and consequently in the CA1 layer. Thus, sensory uncertainty is absorbed by the EC layer, and does not affect the scalability of the CA3 and CA1 layers. It can also be seen that there is a difference in the numbers of CA3 and CA1 units. This is because of the sensory range, which was limited to 10 units in this experiment. As we have mentioned earlier, this causes perceptual aliasing in the environment. At such aliased locations, our model creates multiple CA1 units that respond to the same CA3 unit.

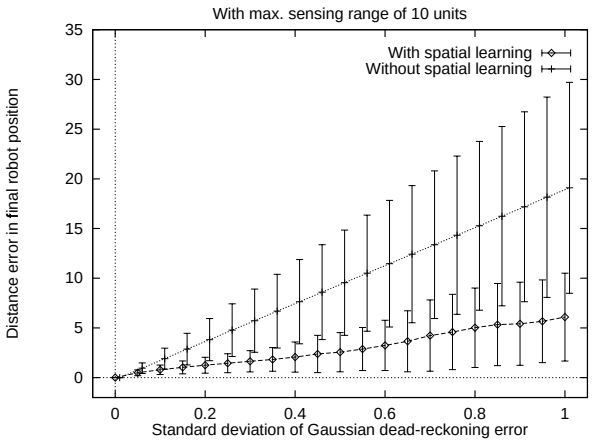

Figure 9.11 Comparison of final robot position error with and without spatial learning ability. Here the robot sensors have a range of 10 units. It is plainly obvious that without the spatial learning and update mechanism, the dead-reckoning estimates drift significantly away from the true robot position. The plot shows the mean over the 100 trajectories along with the standard deviation.

Figure 9.14 also shows another interesting phenomenon. For range errors less than $\sigma_S = 0.4$, the number of EC units in the model is less than (or marginally greater than) the number of CA3 and CA1 units, while for larger errors EC units are significantly more in number. This is because with such small errors the fields of EC units created at a given place (each responding to one observed landmark) coincide almost perfectly. Consequently, the CA3 unit place field, which is an intersection of the EC fields, is as large as the individual EC field. Thus, fewer CA3 (and hence CA1) units are required to cover the same region of the environment. In fact, with very small range error values (e.g., $\sigma_S \leq 0.05$) the number of CA3/CA1 units are fewer than the number of EC units largely due to this efficiency of space coverage. It should also be clarified that with sensor range restricted to 10 units, the number of CA1 units required by our model is always greater than the number of CA3 units since many of the 100 trajectories pass through the perceptually aliased regions. Thus, the apparent equality of CA3 and CA1 numbers for low values of σ_S in Figure 9.14 are simply an artifact of the scale chosen for representing that axis.

When the range of the robot sensors is increased to 30 units, it can be observed that

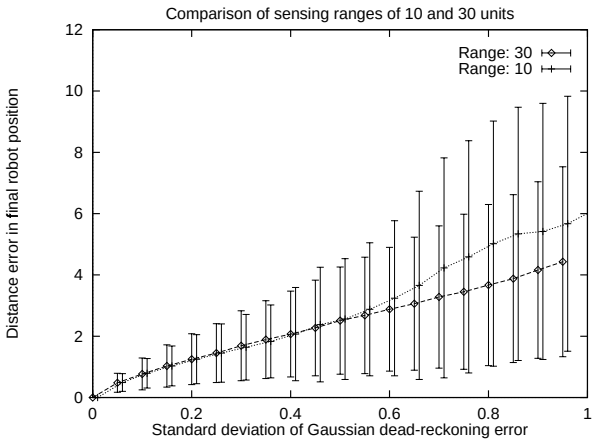

Figure 9.12 Error in the final robot position with sensor ranges restricted to 10 and 30 units. With a range of 30 units, there is no perceptual aliasing in the environment, which leads to better localization. With a range of 10 units perceptual aliasing interferes with faithful map learning, thereby conceding larger position errors.

increasing standard deviation of the range sensing error leads to a near-linear increase in the number of EC units recruited, as shown in Figure 9.15. Further, since there is no perceptual aliasing in this environment, the numbers of CA3 and CA1 units required are *exactly the same*. Also, the increasing range error is effectively absorbed by the EC layer, with little increase in the numbers of CA3 and CA1 units beyond a certain point.

Thus, the size of the spatial map learned by our model (denoted by CA1 fields), scales quite well with increase in range sensing error.

9.2.4.4 Efficiency of Spatial Coding

Our spatial learning and localization system automatically develops a multi-resolution representation of space with frequently visited places (or thoroughly explored regions) being represented at a finer resolution than regions where the robot spent a few passing moments. Since finer resolution helps in more accurate localization while coarser resolution saves valuable memory space, this tradeoff, automatically achieved through simple experience, is a valuable point in favor of our spatial learning model. This multi-resolution representation arises as a result of range-sensing errors and their consequent role in the creation of new EC units, the algorithm

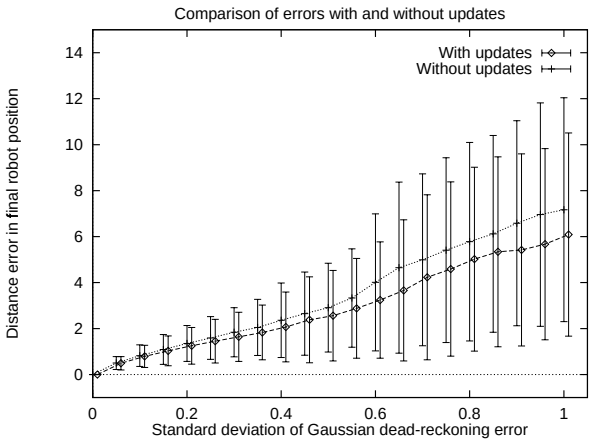

Figure 9.13 Hippocampal Kalman filtering updates place field centers in addition to the robot's position estimate, leading to lower localization error.

for determining the activations of CA3 units, the role of the Mahalanobis distance in the creation of CA1 units, and importantly, the variance updates performed by Kalman filtering. For instance, Figure 9.16 shows the firing fields of four CA1 units. As can be observed, the fields differ in size with the larger firing fields corresponding to CA1 units created in rarely visited areas and smaller fields in regions quite frequently visited by the robot during its exploration run.

Kalman filter updates provide one way in which this multi-resolution representation of space can arise. Suppose the robot is thoroughly exploring a small region of the environment. It first creates a certain number of place cells for this region. Now, as it keeps revisiting the places, Kalman filtering updates the estimates of the field centers and reduces their variances. As the variances decrease, the Mahalanobis distance measure increases until it fails at some point. Even though the robot is in a familiar place the system now creates another CA1 unit for this region, thereby distributing the firing field of the original place between two place cells, and so on.

It should be noted that this change in resolution is an *emergent property* of our system and arises automatically via the experience of the robot in the environment.

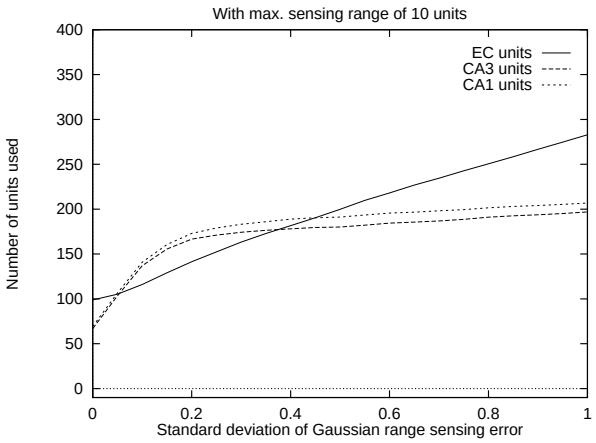

Figure 9.14 Mean numbers of EC, CA3, and CA1 units created by the spatial learning system, given sensor range of 10 units. The numbers of CA3 and CA1 units scale up well with increase in range error. However, the number of CA1 units is more than CA3 units because the system creates more CA1 units to resolve perceptual ambiguities.

9.3 Related Approaches for Robot Localization

Since localization plays a critical role in robot navigation, it is hardly surprising that numerous techniques have been developed for robot spatial learning and localization. A number of these approaches appear closely related to the localization system described in this paper. In the following sections we will briefly discuss and compare some examples belonging to each of these different approaches.

9.3.1 Approaches Related to Kalman Filtering

Since robots perceive aspects of their environments through their sensors, any attempt at world-modeling must be preceded by sensory information fusion (also called sensor fusion). Smith, Self, and Cheeseman (1990) argued strongly for the use of Bayesian estimation theory in robot spatial representation (called the *stochastic map*), and showed that an *optimal* information fusion approach was equivalent to a simple form of Kalman filtering (Kalman, 1960). However, the linearity restrictions imposed by Kalman filtering (see Section 8.2.1) were unrealistic in real-world situations, and the *Extended Kalman filter* framework was developed

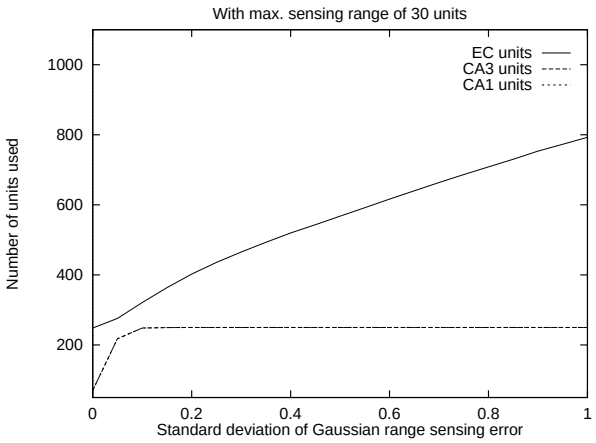

Figure 9.15 Increasing range error affects the numbers of EC units required but not the number of places represented (indicated by the number of CA3 and CA1 units). Since there is no perceptual aliasing with sensor range of 30 units, the number of CA3 units equal the number of CA1 units.

for building 3D maps of the environment (Ayache & Faugeras, 1987). The idea here was to use a Taylor-series expansion of the non-linear function and truncate it at the first derivative, thereby obtaining a linear *approximation* of the original function. Moutarlier and Chatila (1989) developed an alternate stochastic information fusion framework for map-building and showed that it reduces to Kalman filtering as a special case. A good discussion of Kalman filtering approaches for robot localization can be found in (Crowley, 1995), while the necessity and difficulties associated with maintaining correlations of the state variables of a stochastic map are detailed in (Hebert *et al.*, 1995).

Kalman filtering approaches for robot localization require a *sensor model of the environment* in the measurement function. This sensor model provides the sensory inputs (or measurement) that the robot would receive when in any given position. Conventional approaches use a measurement function that specifies the *relative positions* of landmarks with respect to the robot. In such cases, the landmarks are represented in a *robot-centered frame* and the landmark positions are updated based on robot motion. These updated positions serve as predictions while the observed relative positions of the landmarks serve as observations, making Kalman filtering possible (Smith *et al.*, 1990; Ayache & Faugeras, 1987; Leonard & Durrant-Whyte,

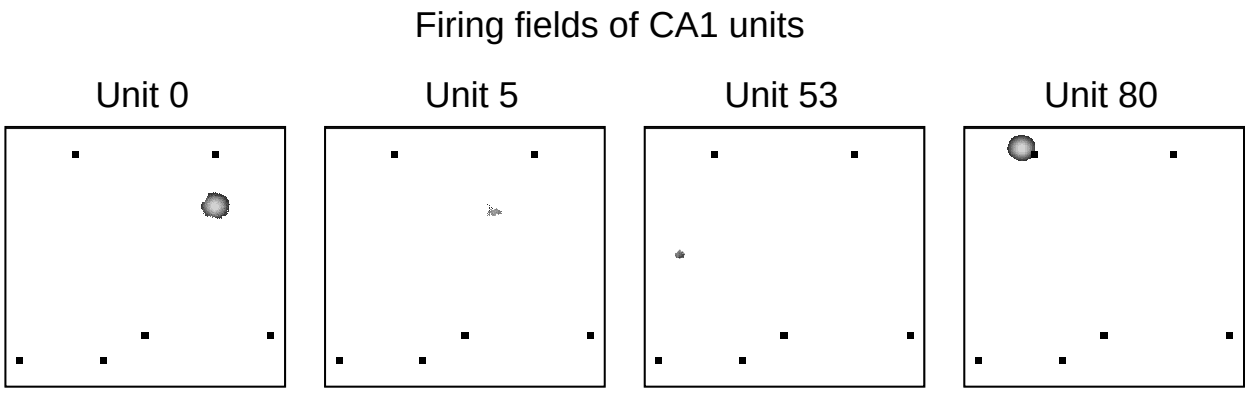

Figure 9.16 Multi-resolution representation of space. Here, frequently visited or thoroughly explored regions of the environment are represented by more CA1 units with smaller firing fields, while cursorily explored regions are represented by fewer CA1 units with larger firing fields. The firing fields of a sample of CA1 units from different regions are shown.

1992; Moutarlier & Chatila, 1989; Crowley, 1995),

However, if the environment contains multiple identical landmarks and sensor ranges are limited, this measurement function leads to matching problems (i.e., which state vector element should be matched against the sensed landmark?) and to consequent localization problems in the presence of perceptual aliasing. These problems also arise when a kidnapped robot is asked to relocalize. In contrast, the model described and implemented in this paper performs *place-based* localization. Since places are more distinct and distinguishable than relative landmark positions, it is easier to relocalize with a place-based system. Some researchers have recently begun exploring place-based extensions of Kalman filtering approaches (Crowley, 1995; Hebert *et al.*, 1995).

Another key distinction of our model is that it does not require a sensor model, apart from the simple neural system for recognizing places. By labeling places with dead-reckoning inputs, the system develops a kind of *inverse sensor model* that produces a place code (and a position estimate) in response to sensory input. Thus, instead of matching in *sensory space* our localization system performs matches in *position space*.

9.3.2 Approaches Based on Cognitive Mapping Theories

A number of cognitive mapping theories of spatial representation have been proposed that espouse organization of spatial information in multiple forms and levels (Kaplan, 1973a; Kaplan, 1973b; Kuipers, 1978). For instance, Kuipers proposed a three level representation of space called the *spatial semantic hierarchy* (SSH) where the lowest level consists of control rules that define *distinctive places* as some property of sensory inputs. The next higher level creates a topological representation by linking distinctive places by *travel edges* and the highest level in the hierarchy contains a *geometric representation* of the robot's sensory environment (Kuipers & Byun, 1991). This cognitive spatial learning architecture was implemented on a simulated robot NX (Kuipers & Byun, 1991) and later extended to physical robots (Kuipers *et al.*, 1993).

Kortenkamp's RPLAN is an implementation of the PLAN (Prototypes, Location and Associative Networks) model of human cognitive mapping (Chown *et al.*, 1995). In his implementation, sonar-based detection of *gateways* and vision-based detection of *scenes* are combined using a Bayesian network to provide robust definitions of place. The system then uses these place definitions to construct routes, networks of routes, and global spatial representations involving many such networks (Kortenkamp, 1993).

Such multi-level spatial representations are also found in Qualnav (Levitt & Lawton, 1990). This simulated robot represents regions of space called *viewframes* at the lowest level. Viewframes are composed of relative angles and estimated distances to landmarks visible from the current place. At the next higher level, pairs of landmarks are used to define a virtual boundary called the *landmark pair boundary* (LPB). These LPBs lead to a topological division of space called *orientation regions*. Together, these two divisions of space allow for specific localization of the robot using viewframes within more general orientation regions.

The models described here provide implementations of cognitive mapping theories while our model attempts a computational characterization of a particular brain region (the hippocampus). The models differ in this regard. However, it is interesting to observe that the spatial representation hierarchies proposed in these cognitive theories closely resemble the different forms of information purported to exist in the hippocampus (Section 7.3).

9.3.3 Approaches Based on Neurobiological Models

Some neurobiological models of spatial learning have also been implemented on mobile robots. Mataric (1992) implemented a topological *place graph* loosely based on the place unit model of McNaughton and Nadel (1990). Using her model, the robot explored the environment and built a topological place graph. The nodes in the graph represented places and the links encoded the robot motions required to move between places. Places in the model were associated with *individual landmarks* like walls, corridors, etc. Further, place units in the graph were associated with a set of instructions that were executed whenever the corresponding unit was activated (the robot reached the corresponding place). Once the map was built, goal-directed navigation was performed by *spreading activation* backwards from the goal node and following the path to the *strongest activated* neighbor of the current node.

Bachelder and Waxman (1994) also implemented a spatial learning system on a mobile robot called MAVIN (mobile adaptive visual navigator). This robot wanders around the environment recognizing places based on the configuration of landmarks visible from that location. Landmarks are recognized using a Seibert-Waxman neural 3D object recognition system (Seibert & Waxman, 1992) which is trained to recognize the landmarks *prior* to the exploration by the robot. As the robot moves, an associative neural network uses a Hebbian rule to learn the movements that lead from one place to the other.

Recce and Harris (1996) recently implemented a robot navigation system based on interactions of neocortical and hippocampal theories. In their model, the hippocampus functions as an *auto-associative* memory for matching *ego-centric* maps which are constructed in the neocortex. The hippocampus stores snapshots of this ego-centric map and the activity of head-direction cells are used to determine the best map rotation to match the snapshots stored in the hippocampus.

The approaches described above are implementations of computational accounts of hippocampal spatial learning. However, all of them deal with a topological representation of space unlike the metric ones created in our model. Further, even though the models were implemented on real robots, none of the models had any explicit mechanisms for handling sensory and motion errors. In contrast, our model explicitly fuses uncertain information in a stochastically optimal manner.

9.4 Discussion

This chapter presents an implementation of the computational model of hippocampal spatial learning developed in the previous chapter. In addition to an algorithmic description of the implementation, we also presented details and results of various simulation experiments. The results demonstrate the ability of our model to learn places based on sensory data. Further, the Mahalanobis distance test used in the system allows the robot to distinguish between and learn perceptually similar places, and localize in their presence. Importantly, our model updates not only the current dead-reckoning position estimates but also the estimated centers of the place fields, leading to lower localization errors. With frequent revisits to familiar places, the uncertainty associated with place field centers automatically decrease, signaling increased accuracy of the place field estimates. The system also handles relocalization (kidnapped robot problem) very easily by setting the variance of the robot's position estimate to ∞ and using the same Kalman filtering approach to localize at a perceptually unique place.

The results of our experiments demonstrate the ability of the model to scale well to increasing sensing and dead-reckoning errors. Further, the model allows the robot to learn a multi-resolution representation of space that automatically balances accuracy and cost of spatial representation. This is achieved by assigning many place cells with small firing fields to frequently visited regions, and using a few cells with large fields, in infrequently visited areas.

This computational model makes some interesting contributions to robot localization. Primarily, it provides a place-based extension of Kalman filtering for robot localization, by allowing the robot to learn places as conjunctions of landmark positions and estimating the centers of these places in terms of dead-reckoning coordinates. The spatial map learned using this model represents space at multiple resolutions. Thus, the model offers advantages of both metric as well as relational representations of space, as discussed in Section 6.3. In addition, it uses an inverse-sensor model to perform matches in position space rather than sensory space. Thus, it does not require a sensor-model which is often difficult to specify in conventional Kalman filtering (see Section 8.2.1). Finally, unlike pure sensor-based systems, the model allows the robot to disambiguate between perceptually similar places, and unlike pure dead-reckoning based systems (Yamauchi & Beer, 1996), it can *relocalize* after being kidnapped.

Although we have not shown any results to that effect in this chapter, goal-directed navigation can be easily realized in our system. The next chapter presents some examples of

goal-directed navigation behaviors generated by animats using our computational model.

10 SIMULATION OF BEHAVIORAL EXPERIMENTS

In the previous chapter we presented an implementation of our computational model of hippocampal spatial learning. We also highlighted a number of properties of our model using a simulated robot task. As the model was inspired by, and based on, neuroscientific and behavioral data from animals, we complete the discussion of our model by presenting computational simulations of behavioral experiments performed with rodents. In this chapter, we use our computational model to simulate and explain two behavioral experiments considered rather important in the rodent spatial learning literature - the gerbil experiments of Collett et al. (1986) and the Morris water-maze (1981).

10.1 Gerbil Experiments of Collett et al.

Collett et al. (1986) performed a series of experiments with female Mongolian gerbils (*Meriones unguiculatus*) and characterized many aspects of their spatial learning and search behaviors. These experiments were conducted on the floor of an approximately circular arena (diameter 3.5 m) placed in a light-tight, black painted room. Sunflower seeds were placed in specific geometric relationships to a set of conspicuously visible landmarks placed on the floor of the arena and the gerbils were trained to find these seeds. However, the floor of the arena was also covered with black painted granite chips, which prevented the gerbils from spotting the seed until they were very close to it. A single bright light (150W) was hung over the center of the arena such that the arena walls were in shadow. This was done to ensure that the gerbils only used the landmark arrays for spatial learning rather than the arena walls. The researchers also translated the entire landmark array, along with the relative location of the seed, to different regions of the arena to ensure that the location of the seed was associated with the landmark array and not with other cues in the room.

Each animal was given between 6 and 12 individual training trials everyday, 6 days a week.

The researchers observed that over roughly 150 such trials (i.e., one month of training), the animals learned to directly approach the seed when released into the environment (Collett *et al.*, 1986). Once the gerbils were trained to satisfaction, they were subjected to testing trials intermingled with training ones in a ratio of 1:6. In the test trials, the gerbils were released into the arena which contained landmarks but no seed. Further, the landmarks themselves were either left in the training configuration or manipulated in controlled ways (e.g., change in the number, size, or arrangement of landmarks). Using a video camera connected to a computer, the paths followed by the searching gerbils were recorded. This was later used to compute and display not only the trajectories of the different animals, but also a measure of the amount of time spent by the animals in different regions of the arena (referred to as a *search histogram*).

Based on a variety of experiments and controlled manipulations Collett et al., reached a number of conclusions. Since the gerbils demonstrated an ability to complete a planned trajectory in complete darkness, the researchers concluded that the computation of the distances and directions (*vectors*) to landmarks must be in the same metric system as animal motion, i.e., possibly in terms of *actual* Euclidean distances. Further, they suggested that the gerbils remember goal locations in terms of vectors to different landmarks visible from the goal. Given such a scheme, the researchers suggested that when the gerbils were introduced at another position in the arena, they computed current vectors to the different landmarks and subtracted the corresponding stored vectors from the goal to the same landmarks, thereby obtaining direct trajectories from the current location to the goal.

Using a number of experimental manipulations they also addressed the issue of how vectors to different landmarks were used in spatial learning. They concluded that gerbils treat each landmark independently when planning a path to the goal. Based on current vectors to each such landmark and stored vectors to the same landmark from the goal location, they hypothesized that the gerbils computed *independent trajectories* to the goal, one for each landmark. If some of these trajectories happened to conflict, i.e., led to radically different goal positions, the researchers suggested that the gerbils followed coincident trajectories that led to the same goal and ignored outriders. Although this *vector voting hypothesis* (Redish & Touretzky, 1996) explained some elements of goal-directed behavior, it failed to explain the ability of gerbils to *relocalize* or *reorient* themselves when the training configuration of landmarks was changed during test trials. The gerbils thus appear to possess some means for also learning something

about the geometrical relations between the landmarks in the environment and show themselves capable of extracting these spatial relations of landmarks from their representations (Collett *et al.*, 1986).

10.2 Simulation of Experiments of Collett et al.

We have reproduced a number of experiments of (Collett *et al.*, 1986), using the computational model of spatial learning developed in Chapter 8. In this section we present results of these experiments.

10.2.1 Simulation Details

In our simulations we used a circular arena of radius 10 units. The walls of the arena were assumed to be impenetrable and devoid of any distinguishing sensory stimuli. This is in keeping with the original experiment in which the walls were in complete darkness and presumably not visible to the animal. The landmarks, on the other hand, were assumed to be visible to the animat from all points within the arena. The animats could not only recognize landmarks, but also estimate landmark positions relative to themselves. However, this estimate was assumed to be corrupted by a zero-mean Gaussian *range sensing error* with standard deviation $\sigma_S = 0.01$ units per unit distance. Sensory inputs obtained in this fashion were used to generate the activations of the EC layer and the place firings of the CA3 and CA1 layers, using the algorithms described in Section 9.1. The animat motions were also considered error-prone, with *motion errors* modeled as zero-mean Gaussians with $\sigma_M = 0.5$. The animats were assumed to possess fairly accurate dead-reckoning mechanisms to track these motions, with the *dead-reckoning errors* being modeled as zero-mean Gaussians with $\sigma_D = 0.05$ units.

Sunflower seeds (goals) were modeled as points in the arena and the animat was said to have consumed a seed (reached the goal) if it approached within 0.33 units of it. It was also assumed that the goal was visible to the animats from a distance of 1.5 units and once spotted, the animats could run directly to the goal.

The experiments of Collett et al., were simulated by first setting up the arena with the landmark(s) in the appropriate positions. The animat was then introduced into the arena at a random position and allowed to perform 500 steps of *sensing, processing*, and *moving*. In this mode the animats learned places by creating EC, CA3, and CA1 units in appropriate ways,

and updating the position estimates using the Kalman filtering mechanism described earlier. If the animat happened to spot the goal (seed) during these sessions, it was made to approach and consume it. This constituted one *training trial*. Once a trial was complete, the animat was removed from the environment and reintroduced at another random position for the next trial. Each animat was subjected to five such training trials. In each trial the animat learned places in a new *frame* and merged frames when appropriate, as detailed in Section 8.4. The firing threshold of CA3 units (CA3Threshold), which signals place recognition based on sensory inputs, was set to 0.75 during training.

Once training was complete, the animats were subjected to ten *testing trials* in which the landmarks in the arena were manipulated in specific ways and importantly, the goal was absent. During these tests the animat was released at predetermined positions in the arena with its dead-reckoning variance set to ∞. Spatial learning ability was turned off in these animats and they were only capable of localizing. The animats had a maximum of 150 time steps within which to localize by visiting a familiar place. Unlocalized animats were removed from the environment, with that testing trial being dubbed a failure, and the process continued with the next testing trial. During testing, CA3Threshold was lowered to 0.25 to enable the animats to localize even if the landmark arrangements had been changed substantially. Since the animats were prevented from relocalizing to frames in which they never visited the goal location, a localized animat always had some memory of where the goal was located. It was then allowed a maximum of 300 time steps within which to navigate to the remembered position of the goal.

As the goals were absent during testing, the animats searched in the region of the remembered goal location. If the animat reached a circular region of radius 0.5 units around the predicted goal location, it was allowed to spend 25 time steps randomly searching for the goal in that area. After this, the variance of the position estimate of the animat was once again set to ∞ and the animat was allowed to *re-localize*. Often, this feature enabled the animat to correct its position estimates, if it had wrongly localized earlier. This had interesting behavioral consequences as will be explained later.

For the training as well as testing trials, the trajectories followed by the animats were recorded. The circular arena was decomposed into cells of size 0.33×0.33 and a count of the amount of time spent by the animats in each cell was kept. These statistics for training

and testing were computed for *five different animats*. The cell with the largest value (total time spent in that cell by the five animats) was used to normalize the values in the other cells, and was plotted in the form of a search histogram. Thus, larger, darker cells in the histogram indicated that the animats spent more time in that region of the arena compared to the regions corresponding to the smaller, lighter ones.

10.2.2 Experiments and Results

We now present results from the simulations the behavioral experiments of Collett et al., using the computational model of spatial learning and localization described earlier.

10.2.2.1 One Landmark Experiment

In this experiment Collett et al. placed the seed at a constant distance (50 cm) and orientation (South) from a single landmark and trained gerbils to reliably approach the goal position. They found that well-trained gerbils ran directly to the seed when introduced into the environment. Further, in testing trials the gerbils were found to concentrate their search efforts at the expected location of the seed even though the seed was absent (Figure 1 in (Collett *et al.*, 1986)). In our simulation of this experiment, the goal location was 4 units to the south of a single landmark, as shown in Figure 10.1 (Left). In these figures, and the ones that follow, landmarks are represented by large filled squares, goal locations by filled circles, and the search histograms by considerably smaller filled squares. As explained earlier, the sizes of the filled squares in the search histograms were directly proportional to the amount of time spent by the animats in that region of the environment.

Figure 10.1 (Middle) shows the search trajectories followed by the animats during the test trials in this one landmark experiment. The corresponding search histogram is shown in Figure 10.1 (Right). It may be observed from the search histogram that the animats spend most of their time searching in regions near the expected position of the goal. This search histogram compares rather well with the observations of (Collett *et al.*, 1986).

10.2.2.2 Two Landmark Experiments

In the next set of experiments, Collett et al. trained gerbils to locate a sunflower seed placed to the south of a line connecting two identical landmarks. The seed was equidistant

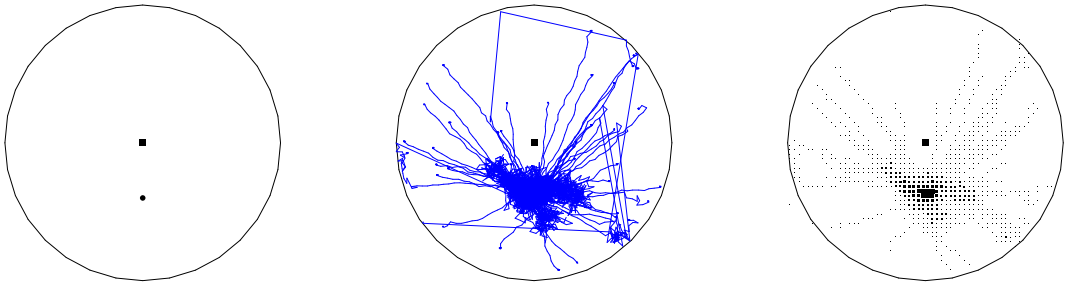

Figure 10.1 One landmark experiment. Left: Training environment with goal location in a fixed relationship to a single landmark. Middle: Search trajectories of the animats during testing. Right: Search histograms of the animats.

from the two landmarks. In our simulations, the goal was positioned 4 units to the south of the line connecting two landmarks placed 4 units apart, as shown in Figure 10.2 (Left). Figure 10.2 (Right) shows the search effort of the animats in test trials when the goal was removed. As can be observed, the animats concentrate their search to a region close to the position of the goal in the training trials. This figure compares well with Figure 7b in (Collett *et al.*, 1986).

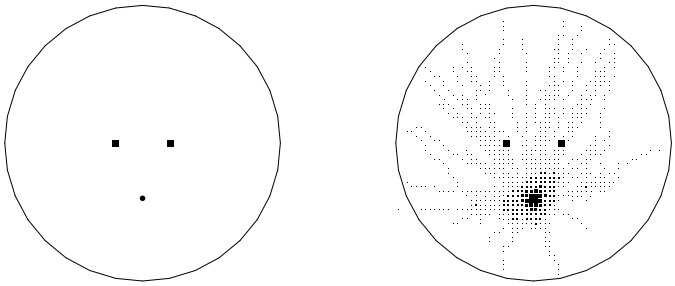

Figure 10.2 Two landmark experiments. Left: Training configuration with two landmarks and the goal equidistant from the two. Right: Search histogram of the animats during tests with goal removed.

When trained on the two landmark task and tested them with one landmark removed, Collett et al. found that the gerbils searched on both sides of the sole landmark, apparently matching it to either the left or the right landmark of the original configuration (Figure 7c in (Collett *et al.*, 1986)). Our animats demonstrated a similar behavior as seen by their search

distribution in Figure 10.3 (Left).

Also, when the gerbils were trained with two landmarks and tested with the distance between the landmarks doubled, Collett et al. found that the gerbils searched *predominantly* at the two interior locations, each at the correct distance and orientation from one of the landmarks (Figure 7d in (Collett *et al.*, 1986)). Results from similar experiments with our animats are shown in Figure 10.3 (Right). It can be observed that our animats too search predominantly at the interior locations. We also observed that some animats searched at multiple locations, i.e., when allowed to relocalize after 25 steps of futile searching, the animats chose a completely different location to search in.

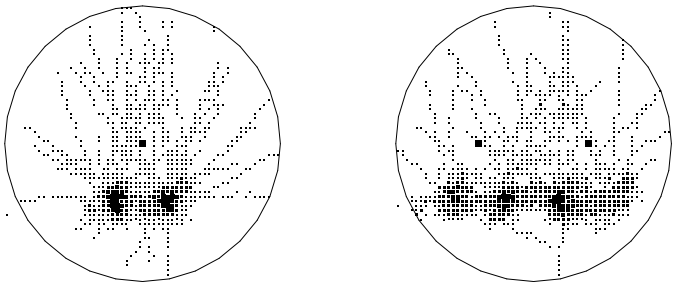

Figure 10.3 Two landmark experiments. Left: Search distributions when trained with two landmarks but tested with one of them removed. Right: Search histograms when tested with the distance between the original landmarks doubled.

10.2.2.3 Three Landmark Experiments

In another set of experiments, three identical landmarks were arranged to form the vertices of an equilateral triangle with the goal located at the centroid of the triangle, as shown in Figure 10.4 (Left). When trained in this environment, our animats produced search histograms concentrated reliably at the correct position of the goal, i.e., at the centroid of the triangle as shown in Figure 10.4 (Middle). This compares favorably with Figure 6b in (Collett *et al.*, 1986).

Collett et al. also trained the gerbils on the three landmark task and tested them in environments with one or two of the landmarks removed. With one landmark removed they found that the gerbils searched at a location at the correct distance and orientation from

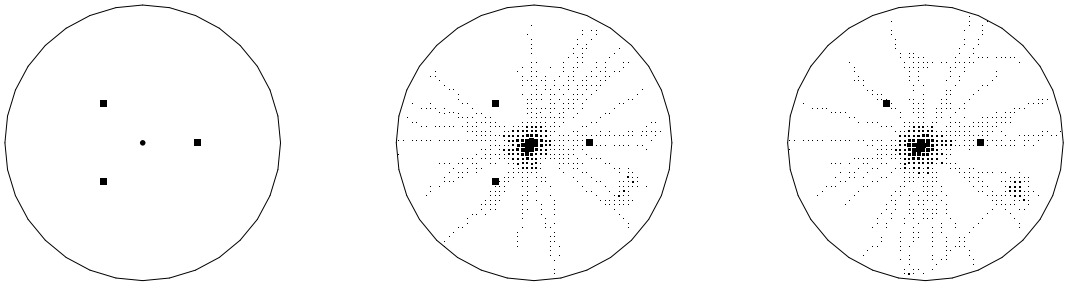

Figure 10.4 Three landmark experiments. Left: Training configuration for the three landmark experiments. Middle: Search distribution of the animats when tested in the three landmark environment. Right: Search distribution when tested with one of the three landmarks removed.

the two remaining landmarks (Figure 6c in (Collett *et al.*, 1986)). It can be observed from Figure 10.4 (Right), that our animats demonstrate largely similar search behaviors.

With two of the three landmarks removed during testing, Collett et al. found that the gerbils distributed their search time between three sites, one for each of the three possible matches of the sole landmark (Figure 6d in their report). This can be compared directly with our simulation results in Figure 10.5 (Left). Similarly, when gerbils were trained on the three landmark task but tested with one landmark distance doubled, they were found to search at a goal location at the correct distance and bearing from the two unmoved landmarks (Figure 8 in (Collett *et al.*, 1986)). The result of this experiment with our animats is shown in Figure 10.5 (Middle).

When gerbils were trained on the three landmark task, but tested in an environment with an additional landmark placed so as to create another equilateral triangle with a different orientation, Collett et al. found that the gerbils reliably searched at the goal location within the correctly oriented triangle. Our simulation of this experiment produced similar results as shown in Figure 10.5 (Right).

10.2.3 Discussion

We have used our computational model of hippocampal spatial learning to simulate the behavioral experiments of Collett et al. (1986). The goal of the experiment was to test whether

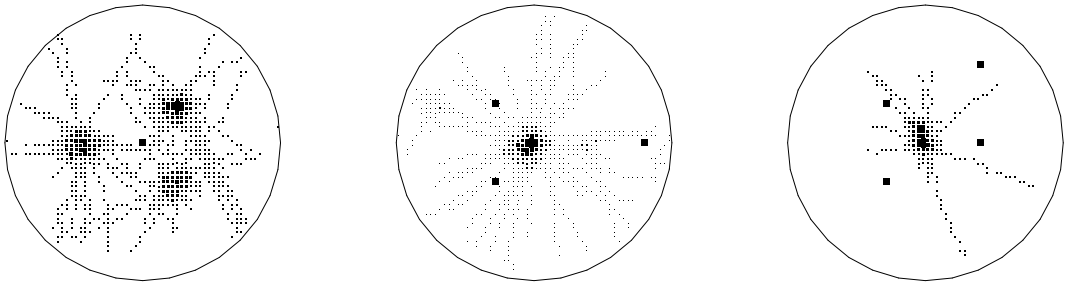

Figure 10.5 Three landmark experiments. Left: Search distributions when trained with three landmarks but tested with only one landmark present. Middle: Search distributions when one landmark distance was doubled during testing. Right: Search distribution when an extra landmark was added during testing, thereby creating an extra triangle.

our model of hippocampal spatial learning and localization could explain their behavioral results with gerbils. We simulated a number of their experiments and the search histograms generated by our animats were very similar to those produced by the gerbils in their experiments. This is particularly interesting because our animats did not remember goal locations in terms of *independent vectors* to individual landmarks, as hypothesized by Collett et al. Nor did the animats navigate by performing vector subtractions. Instead, our spatial learning system allowed the animats to learn sensory descriptions of *places* in terms of vectors to individual landmarks, and used self-generated dead-reckoning position estimates to learn the metric positions of the places and goals.

With such a scheme, the animats simply localized using sensory cues from the landmarks, and then navigated directly to the remembered metric position of the goal (with respect to the localized position). It thus appears that the results of Collett et al., can be explained using a metric spatial representation and navigation hypothesis. This leads to an important observation. Notice that the search histograms produced by our animats are greatly influenced by the dead-reckoning and sensing errors in our model. For instance, if there were no errors in the system, spatial learning and localization would be perfect and goal locations would be accurately and unambiguously predicted, leading to histograms with a *single* peak and little *spread*. This would be unlike the ones Collett et al., observed with their gerbils. Given the fact

that the search histograms of animats with sensing and dead-reckoning errors better match the search distributions of the gerbils, one might strongly suspect that similar errors must also exist in animal spatial learning systems.

10.2.3.1 Related Work

To the best of our knowledge the only other computational simulation of the experiments of (Collett *et al.*, 1986), is the work of Redish and Touretzky (1996). Their model was described earlier in Section 8.6. In their simulation of the Collett et al. experiments, the animat was placed at random positions in the arena and was *given* its position relative to the goal (which was assumed to coincide with the origin). The animat then created place cells using a combination of this position estimate and sensory inputs from the visible landmarks as described earlier. *Ego-centric* angles between landmarks were also encoded in the place cells, which allowed the animat to also initialize its head-direction if it happened to be disoriented. In test trials the animats were introduced at a random position and allowed to localize, i.e., the animats performed head-direction and position estimate resets. Once the animat had localized, it could *predict* the goal location, which was simply the origin of the coordinate frame with respect to its current localized position. This process was repeated a number times and a histogram of predicted goal positions was computed (Redish & Touretzky, 1996).

Although the results obtained by Redish and Touretzky were quite similar to the observations of Collett et al., a few relevant issues must be clarified. First, the position estimates used to label places were explicitly provided to the animats. Additionally, these estimates were considered error-free. Similarly, the sensory inputs pertaining to the landmarks and their vector positions were also assumed to be accurate. It is not clear how this model will scale to situations where these streams are erroneous, as would be the case with contemporary robots. Second, goal locations in their model corresponded to the origin of the position estimate framework. It is unclear how the animats would represent multiple goals locations if they had to. Third, the animats used in their simulations did not explicitly move. Consequently, the histograms reported in (Redish & Touretzky, 1996) correspond to *predictions* of the goal position rather than the time spent by the animat in different regions of the environment. Thus, a dark histogram cell far from the goal in their model implies that the animat has a completely wrong estimate of the goal position and hence a completely wrong localization. In contrast,

a similar cell in the histograms of Collett et al., simply indicates that the animal spent some time in that region, either localizing or while navigating to the goal. It does not necessarily imply that the animal's localization or its prediction of the goal position is wrong. Thus there are qualitative differences between the histograms in these two reports.

These issues do not arise in our experiments because our model explicitly addresses information fusion from uncertain sources, provides mechanisms to learn and represent multiple goals, and importantly, utilizes navigating animats.

10.2.3.2 Future Work

We would like to extend our spatial learning system in two ways, to allow us to simulate the other experiments of Collett at al. The first extension involves a mechanism for learning and representing the *significance* of landmarks and using this to influence spatial learning. For instance, in some of their experiments Collett et al. found that the gerbils appeared to give more importance to visually discriminative landmarks. In other experiments, they found that the gerbils appeared to trust landmarks nearer to the goal than those farther away. It is rather easy to incorporate these notions in our computational model. For instance, the strength of firing of EC units could be made proportional to some quantity denoting the *significance* of the corresponding landmark (e.g., visually discriminative elements, nearness of the landmark from the current location, etc.). With such a set up, spatial learning and localization of the animat would be influenced by the significance of the landmarks, with the more significant ones exerting more control over the localization process. We can manipulate this significance parameter in controlled ways to study their effect on behavior, and determine ways in which animals might uses these features in spatial learning.

We would also like to extend our model to incorporate a head-direction reset system. As mentioned earlier, such a mechanism has already been developed by (Redish & Touretzky, 1996). With such a system the animat can also initialize its orientation, in addition to just localizing to a place. Our current belief is that the head-direction reset system resides in the subicular and postsubicular regions of the hippocampus, with the place cells in the subiculum being associated with orientation information in the postsubicular head-direction cells. We are currently exploring this hypothesis.

10.2.3.3 Suggestions and Predictions

Collett et al. performed a number of experiments which suggested that gerbils attach differing significance to the individual landmarks in a configuration. In some of their experiments they found that landmarks *closer* to the goal location had a more profound effect on the search behaviors of the gerbils than the ones farther away. However, in those experiments the landmarks used were quite different from each other. Thus, it is hard to conclude with confidence that the gerbils were using the nearness of as the measure of significance to focus their search behaviors, rather than other physical aspects of the landmarks. We feel that a number of behavioral experiments can be designed to address this issue to satisfaction.

Experiment 1: The gerbils could be trained in an environment with two identical landmarks, as in the two landmark task of (Collett *et al.*, 1986). However, instead of placing the sunflower seed equidistant from the two landmarks, it could be placed closer to one (and consequently farther away from the other). Once trained in this environment, the gerbils could be tested with the landmark distance doubled. If the search distribution contains two peaks of roughly *equal* sizes, it may be inferred that the gerbils do not use distance information to assign priorities (or significance) to the landmarks. However, if the search distribution displays unequal peaks, with the larger peak closer to a landmark, we may have reason to suspect that the distance of landmarks from a given place affects their significance in learning a representation of that place. We could also train gerbils on the two landmark task described above and test them with only one landmark. Again, if the search distribution happens to be *skewed*, with the larger peak closer to the landmark than farther away, one may confirm that distance information is being used in the representation of places.

Experiment 2: Collett et al. performed an experiment using three *different* landmarks and showed that the gerbils appear to give priority to landmarks closer to the goal. However, as the landmarks were all different, some of these conclusions are a little suspect. If this experiment were to be repeated with all three landmarks identical and similar manipulations were performed, the results would help lay to rest a number of issues concerning prioritization of landmarks. For instance, if the search behavior of the gerbils remains same, one can assert with considerable certainty that the nearness of landmarks exerts a greater influence on spatial learning and representation. If the new results turn out to be different, then we might be able to conclude that the gerbils used some other measure of significance in the original experiment.

Experiment 3: One can also attempt to characterize the *tradeoff* between landmark prominence and closeness. For instance, we can use a two landmark task, with one prominent and one dull landmark. The sunflower seed could be placed closer to the dull landmark and farther from the prominent one. Once trained on this task, the search behaviors of the gerbils in test environments with the landmark distances doubled would provide extremely interesting insights into the relative importance accorded to physical attributes and closeness of landmarks. In particular, if the search distributions are concentrated at the correct distance and direction from the dull landmark, we may suspect that closeness of landmarks overrides sensory differences. If the opposite scenario is observed, i.e., the search distributions are concentrated in a region at the correct distance and direction from the prominent landmark, we may conclude that sensory characteristics are given more importance in spatial learning.

These experiments (and their extensions) can be used to identify the roles played by landmark distance and appearance, in the representation of space in gerbils.

10.3 The Morris Water-Maze

Morris (1981) experimented with male hooded rats of the Lister strain in an attempt to demonstrate that these rats in particular (and rats in general) were capable of rapidly learning to locate an object (or goal) that they could not see, hear, or smell. An additional object of his experiments was to show that the rats could learn the locations of these objects relative to *distal* room cues, and that they did not require *local* cues for this purpose. In order to achieve these objectives, he designed a special apparatus now known as the *water-maze*.

The water-maze consisted of a large circular pool of diameter 1.3 m and height 0.6 m, which was made of hardboard lined with fiberglass. This pool was filled with water up to a height of 0.4 m above the base. Milk was mixed with the water roughly in the ratio of 1:264, rendering the water opaque for all practical purposes. The entire apparatus was placed in the center of a room measuring 3 by 4 m. The room had a door on one wall with windows on the opposite side. The other two walls were lined with shelves and a cupboard respectively. A video camera was mounted above the center of the pool and the picture was relayed to recording equipment in an adjacent room.

Morris used two platforms of diameter 0.11 m that could be placed at different locations in the pool. One platform was white in color and 0.39 m high, while the other was painted

black and was 0.41 m tall. When placed in the pool, the black platform thus protruded 0.01 m above the water level while the white platform remained 0.01 m below the water surface. Since milk was added to the water in the pool, the white platform was essentially invisible under the water. Rats were then introduced into the pool at the periphery and their movements were observed. In particular, the behavior of the rats in moving towards the location of the platform (visible or invisible) was noted.

All the animals being tested were first *pre-trained*, wherein each rat was placed in the pool and allowed to swim for a period of 180 sec. During this phase the pool did not contain either of the platforms. This pre-training was done on two consecutive days and the regular experiments started on day 3 with each rat participating in eight trials. The rats were divided into four groups of eight members and each group was used for a different experiment. The experiments were characterized by the platform used (white or black) and its position across trials (fixed or random). In each of the trials, the platform could be placed at one of four locations designated NW, NE, SE, and SW. The animal was introduced into the pool facing the wall, at one of four positions N, S, E, or W along the periphery. Although this sequence of starting positions were randomly chosen, it was ensured that each individual rat started exactly twice from each position over the course of its eight trials every day. Once the rat was in the pool, a timer was started and the time taken by the animal to escape to the platform, called *escape latency*, was recorded.

The first experiment designated, Cue + Place, used the black platform. Further, the platform was always at the same location (NW, NE, SE, or SW) across trials for a *given* rat, but in possibly different locations for different rats. The second experiment was exactly like the first except that the white platform was used instead. This was designated the Place experiment. The third experiment consisted of rats trained using the black platform. However, in this case the platform was not at a fixed location; rather, it was placed in one of the positions NW, NE, SE, or SW, but in an unpredictable sequence. This was referred to as the Cue-only experiment. Finally, the Place-Random experiment was similar to the Cue-only experiment except in the use of the white platform.

Following training on day 3, the rats were subjected to a further 8 trials on day 4. On day 5, a further four training trials were given followed by one of two *test procedures*. For these tests, the rats in each of the groups were divided into two subgroups of 4 members each. One

subgroup was then subjected to *Test A* while the other subgroup participated in *Test B*.

In Test A the platform used in the training trials of the experiment, was removed and the animals were introduced into the pool. Their search behavior was observed and the amount of time spent by the rats in each of the four quadrants was calculated. After 60 sec of swimming, the rats were removed from the pool.

Test B on the other hand, was a test of the ability of the animals to adapt their learning to a new situation. Rats trained with fixed platform positions (Cue + Place and Place) were tested with the platform now placed in the quadrant diagonally opposite the one used in training. Similarly, rats trained with random platform positions (Cue-only and Place-Random) were now tested in the water-maze with the platform position held fixed across trials. The escape latencies during this test were recorded and the trajectories followed by the rats were observed.

Based on these experiments, Morris found that the rats in Cue-only, Cue + Place, and Place experiments quickly learned to approach the escape platform directly and there was no sign that the animals treated the invisible platform any different from the visible one. However the rats in the Place-Random experiment showed considerably poorer performance and learned the task more slowly than the other groups. Importantly, groups Cue-only and Cue + Place did not differ (in a statistical sense) in their ability to escape to the visible platform. However, their escape latencies were slightly smaller than those of the rats in the Place experiment (Morris, 1981).

Results of Test A indicated that the animals in the Place experiment show a strong bias for their training quadrant and spend considerable amount of time searching in that quadrant. This effect was also present, albeit much weaker, in animals of group Cue + Place. The animals of groups Cue-only and Place-Random did not appear to possess such biases and spent roughly equal amounts of time in each of the four quadrants.

In Test B, the animals of group Place were found to swim directly to the training location of the platform. After spending considerable time searching for the platform, the animals embarked on an exploratory behavior, eventually discovering the platform in the diagonally opposite quadrant. Thereafter, learning of the new platform location proceeded rather swiftly. Animals in the Cue + Place experiment also showed *some* place-bound swimming even though the platform was visibly at a different location. With the platform now in a fixed position, animals in the Place-Random experiment showed a marked decrease in escape latency, demon-

strating that they could adapt to the changed circumstances and benefit from the predictable (fixed) position of the platform.

In summary, the rats demonstrated the ability to find hidden objects by learning their positions in a familiar space. They also showed themselves capable of using only distal cues in learning these object locations. An added observation was their ability to quickly adapt to experimental changes, particularly in responding to changes in goal positions.

10.4 Simulation of the Morris Water-Maze

We have used the computational model of spatial learning and localization developed earlier, to simulate the Morris water-maze experiments.

10.4.1 Simulation Details

In our simulations, we used a circular arena of radius 3.75 units inside a square room measuring 20 by 20 units. Consistent with the ratio of pool and platform sizes in Morris' experiments, we chose the radius of our simulated platform to be 0.65 units. It was assumed that the platform was not completely invisible and that the animat could see it from a distance of 0.325 units (we felt this was a reasonable assumption since one could not guarantee that the white platform was completely invisible to the rats). Four environmental cues were assumed to be present, one along each wall of the simulated room to account for the window, door, shelf, and cupboard in the original experiment of Morris.

The sensing, motion, and dead-reckoning errors were same as in Section 10.2.1. We also assumed that rats swam slower than their normal walking speeds, and hence the size of motionStep was set to 0.4 units rather than 1. The animats used the spatial learning mechanism to create EC, CA3, and CA1 cells , thereby learning local spatial maps. Where appropriate, these local maps were merged into global ones using the mechanisms described earlier. If the animat reached the escape platform, it updated its goal position estimates using the expressions described in Section 8.5. The animats could also represent multiple goals, as described in Section 8.5.1.

The animats were introduced at specific locations in the arena, consistent with the experiments of Morris. The behaviors of the animats were observed and the time taken by the

animats to escape to the platform was determined. It was assumed that the animats accurately remembered their past experiences (i.e., the spatial maps remained uncorrupted) across trials.

When introduced into the environment, the animats first localized themselves using sensory cues. Once localized, the animats navigated to remembered position of the escape platform (or the goal). Since the animats could possible remember *multiple* goal locations (e.g., in Place-Random experiment), they needed a mechanism to determine the goal to visit first. We implemented a heuristic goal selection strategy that chose the most *recently* visited goal location and made the animat navigate to it. If the goal was not found at this location, the animats then selected alternate goals with probabilities proportional to a *measure of confidence* associated with the goals and their distance from the current position of the animat. We used a mechanism in which the animat chose to approach the nearest goal with probability 0.5 and the most *confident/reliable* goal with the remaining probability.

At each goal location, the animat spent 15 time steps randomly searching for the platform, before choosing an alternate goal location to navigate to. If none of the goal locations searched by the animat contained the escape platform, the animat simply changed to a random-search behavior.

10.4.2 Experiments and Results

In line with the original experiments of Morris, we allowed our simulated animats four pre-training trials in which they randomly explored the environment for 500 steps. During this stage, our spatial learning system allowed the animat to acquire a spatial map of the environment. Groups of eight animats were then used in four experiments corresponding to Cue-only, Cue + Place, Place, and Place-Random experiments of Morris. For each of the experiments, the mean and standard deviation of the escape latencies of the animats, are shown in Figure 10.6.

As can be observed, the animats in the Place and Place-Random experiments take a much longer time to find the escape platform because it is invisible. The cue experiments, on the other hand, lead to very small escape latencies. The primary reason for this observation is that our animats had a built-in mechanism to directly approach visible goal locations. Animals, on the other hand, may not have such direct approach behaviors and may approach the platform rather cautiously in the early trials This explains the slight discrepancy between our results

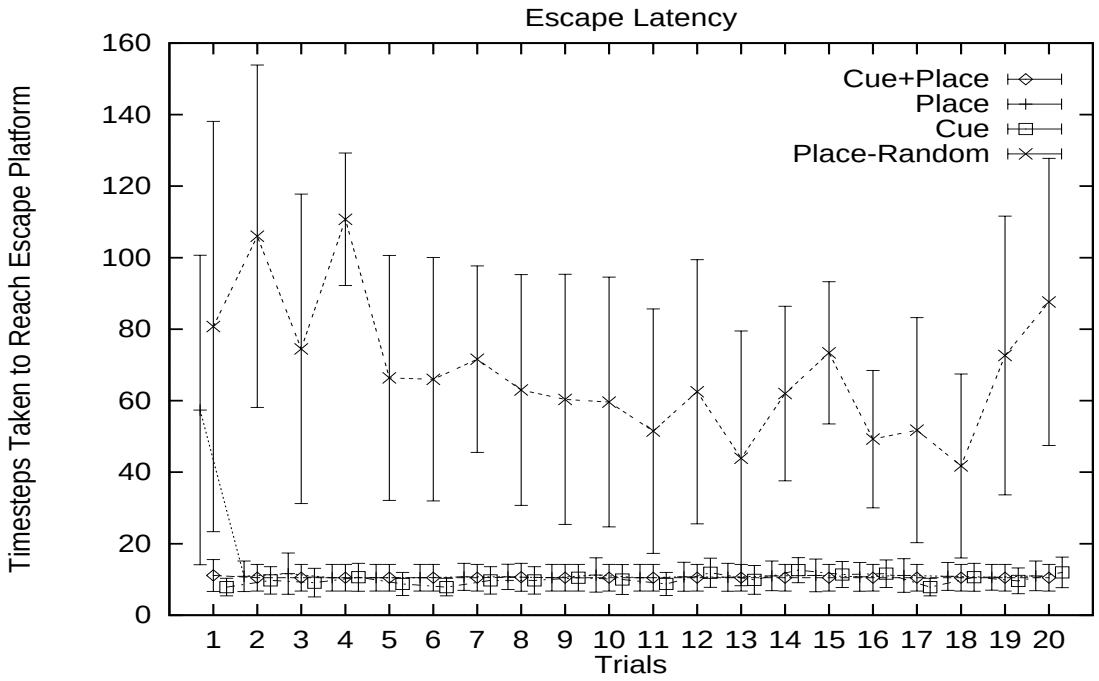

Figure 10.6 Escape latency in the Morris water-maze over training trials 1-20. Latency is measured in terms of the number of motion steps required to reach the goal location.

shown in Figure 10.6 and Figure 3 in Morris (1981), where the latencies in the first trial for animals in groups Cue-only and Cue + Place were fractionally more than in the succeeding trials. Figure 10.6 also indicates the ability of the rats in the Place experiment to quickly learn to approach the goal location directly. In fact, beginning in trial 2, the animats can be seen to directly approach the hidden goal. As observed by Morris, our animats too perform rather poorly in the Place-Random experiment owing to the unpredictability of the goal locations. In fact, our animats conduct a systematic search for the goal location using the goal selection algorithm described earlier.

Figure 10.7 shows the paths taken over trials 17 through 20 of the animat in each group that is just worse than the median performer of that group. Small gray circles in this figure indicate the location of the corresponding escape platform (visible or invisible) and the lines shown in black correspond to the escape trajectories followed by the animats. This figure compares with Fig 5 in Morris (1981).

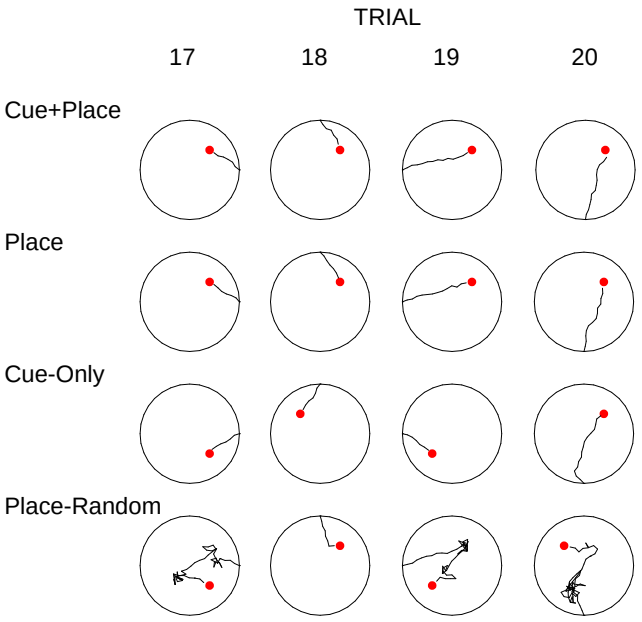

Figure 10.7 Escape paths taken over trials 17 through 20 by animats just worse than the median performer in each group. The gray circles indicate the position of the escape platform.

It can be observed from Figure 10.7 that in the experiments with the visible platform (Cue only and Cue+Place), the animats directly approach the escape platform. However, these navigation trajectories are not perfectly straight because we have added a slight drift to the animat motion in each step. This works as follows. When the animat moves forward by motionStep, it does not always move in the direction it is facing. We add a small Gaussian error to the direction of the animat, causing its intended trajectory to drift by a small amount. It is our contention that such errors are reasonable and lead to more realistic navigation trajectories. In fact, the trajectories of the rats in the Morris water-maze are quite similar to those produced by our animats.

The animats in the Place experiments can also be observed to directly approach the escape platform even though the platform is hidden in these experiments. This is a direct result of the ability of these animats to learn metric spatial maps, represent goal locations in this

map, and navigate to their remembered positions after re-localizing in a new trial. Since the platforms are not only hidden, but also in unpredictable locations, animats have difficulty in navigating directly to the platform in the Place-Random experiments. It may be observed that in trial 17 in Figure 10.7, our goal selection algorithm makes the Place-Random animat search unsuccessfully in two previous goal locations before finding the goal. In trial 18, the goal selection algorithm makes the animat navigate first to the most recent goal location, i.e., the location of the platform in trial 17. While on this trajectory, the animat fortuitously finds the escape platform. However, these chance events are rather rare and on an average, our goal selection mechanism makes the animats search in two locations before finding the goal, as observed in trials 19 and 20 in Figure 10.7.

Figure 10.8 shows the paths taken during the first test trial by the same four animats whose paths were shown in Figure 10.7. It can be observed that when subjected to Test A (where the platforms are absent), the animat belonging to group Cue + Place, shows a slight bias towards searching more in the NE quadrant, which was the quadrant where it found the platform in its training trials. However, once it searches the quadrant and does not find the platform, it begins a random exploration of the surrounding region.

The animat from group Place, when subjected to Test B (where the goal location is changed from the training trials), spends considerable time in the NE quadrant (where it found the invisible platform in the training trials). However, after fruitlessly searching this quadrant, it initiates a random exploration and by chance, finds the platform in the diagonally opposite quadrant, as seen in Figure 10.8.

Test B has little effect on the animat of the Cue-only category. As these animats have learned to directly approach the goal, no matter where it is, the animat in this test directly approaches the visible platform, as observed in Figure 10.8. Using our goal selection mechanism, the animat from the Place-Random group engages in a systematic search for the goal location. As can be observed (and expected) it spends roughly equal amounts of time searching for the platform in each of the four quadrants. This is expected because in these experiments the animats encounter goals in one of four different positions in each trial. As each of these positions is in a different quadrant, the animat remembers four goal locations, one in each quadrant.

Results of the performance of the animats on Test A (where the platform is absent) are

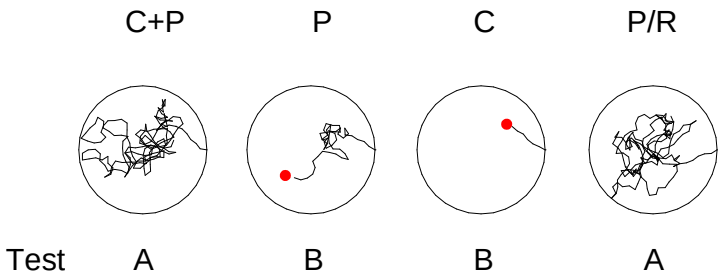

Figure 10.8 Trajectories followed by the animats shown in Figure 10.7, on their first test trial. Two of the animats are performing Test A while the other two are participating in Test B.

summarized in Figure 10.9. It can be observed that Place and Cue + Place experiments indicate a strong spatial bias towards the training quadrant (TR), i.e., these animats spend maximum time searching in the quadrant in which the goal was present during training. While the former observation is consistent with the results of Morris, the latter is a surprise. However, this is a direct result of our implementation of the Cue+Place experiment, where we allowed our animats to use their spatial learning mechanisms to learn a place map even though the platform (or goal) was visible. There is a possibility that in the presence of reliable goal descriptions (e.g., visible platform), place learning may not be necessary and the animals may simply be using guidance-taxon behaviors to directly approach the visible goal. This hypothesis regarding differences in place learning in the presence or absence of reliable cues, remains to be studied via appropriate computational modeling efforts.

It can also be observed from Figure 10.9 that Place-Random and Cue-only animats spend roughly equal amounts of time searching the four quadrants. This happens because the animats use the goal selection algorithm to systematically search the possible goal locations. Since the animats have encountered goals in each of the four quadrants during training, this systematic search procedure makes them search in the four quadrants.

When our animats were subjected to Test B, we observed behaviors rather similar to those

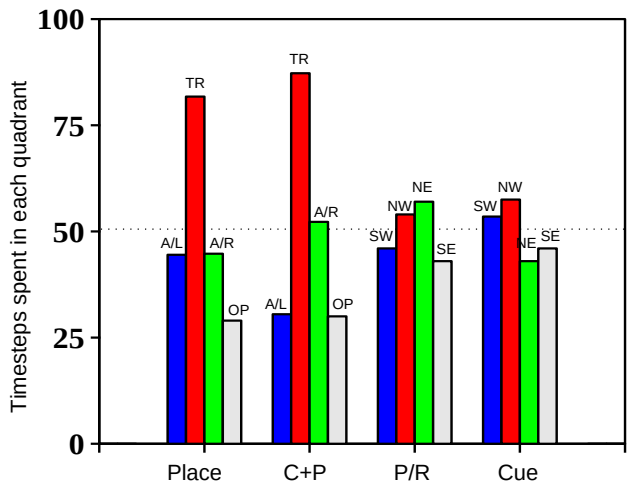

Figure 10.9 Performance on Test A. Histogram shows the duration of time spent by the animats in each quadrant. Here TR is the training quadrant, A/L and A/R are the adjacent quadrants to the left and right respectively, and OP is the opposite quadrant for groups Cue + Place and Place. For the other groups quadrants are labeled NW, NE, SE, and SW.

observed by Morris. Remember that in Test B, Place and Cue+Place rats were tested in environments with the platform in the quadrant diagonally opposite the quadrant in which the platform was placed during training. Although this change did not affect our Cue+Place animats significantly, the animats in the Place experiment spent considerable time searching in their training quadrant before embarking on a random exploration of the arena. However, once the platform was found in its new location, these animats quickly updated their goal representation and could directly approach this new goal location in subsequent trials.

Test B for the Place-Random and Cue-only rats involved keeping the platform position constant across the testing trials. Although this experiment did not alter the behaviors of the Cue-only group, the animats in the Place-Random group were able to decrease their escape latencies considerably. This happened because the goal location now became predictable (as it was kept constant across trials), with the result that the Place-Random animats could directly approach the goal as their counterparts in the Place experiment had done earlier.

10.4.3 Discussion

We have used the computational model of spatial learning developed in Chapter 8 to simulate the behavioral experiments of (Morris, 1981). It may be observed that our results are rather similar to those in the experiments of Morris. Thus, a metric spatial representation hypothesis, as underlies our hippocampal model, can be used to explain the water-maze behaviors of rats.

In our opinion, the most important contribution of our work was in the development of a mechanism to represent multiple goal locations, which was needed in the Place-Random and Cue-only experiments. In order to simulate the Place-Random experiments of Morris, we also had to address another important issue, namely, how do animals choose the goal to approach if they happen to remember multiple ones? In answer to this question, we developed a goal selection mechanism that chose the most recently visited goal to approach first. If the animat did not find the goal at this location, the next goal to visit was chosen based on a combination of the *confidence* associated with each goal location (count of the number of times the goal was there when the animat navigated to it in the past) and its *distance* from the current animat position.

Our results using this mechanism closely parallel the behaviors observed by Morris, which causes us to wonder whether similar goal selection mechanisms might be employed by the rats in the water-maze. In fact, our computational framework can be easily used to implement and test different hypotheses of goal selection, which can help further our understanding of goal selection processes in navigating animals.

10.4.3.1 Related Work

A number of researchers have developed computational models to simulate the behavioral experiments of Morris (1981). Here we will describe a few that also appear to be inspired by hippocampal data.

Blum and Abbott developed a model of hippocampal spatial learning, which was described in Section 7.3.1. In their model, locations were described by place cell activity and the animat could learn paths through the environment by changing the synaptic connections between different place fields. In their simulation of the Morris water-maze, the animats initially moved randomly through their environments. However, the LTP mechanism used by the animats

changed the synaptic weights between place fields such that current activations encoded place locations further ahead on the path. The animat then navigated towards the position coded by the place cell activity at any given place. This scheme led to the learning of routes that the animat could follow to reach the goal. However, the researchers only simulated the Place experiment of Morris and did not provide any accounts of the other experiments.

Sharp et al. (1996) simulated the Place and Place-Random experiments of Morris using a model developed by Brown and Sharp (1995). This model was described in Section 7.3.1. In their model, recently active synapses between the head-direction and nucleus accumbens cells were strengthened when the animat encountered a goal location. This led to the association of motor responses with appropriate sensory stimuli, such that sequences of motor actions, triggered by sensory stimuli, would lead the animat to the goal location. Although their results closely match those of (Morris, 1981), is not clear how multiple goals were incorporated in the system (for the Place-Random experiment) or how the animat navigated (did it use a deterministic strategy or a non-deterministic mechanism to choose the action to perform).

Recently, Redish and Touretzky (1998) have provided a comprehensive account of rodent spatial learning and memory by suggesting that the hippocampus participates in both locale as well as *route-based* learning of space. While the locale-based learning component is rather similar to the one in their earlier work (Redish & Touretzky, 1996), the route-based learning component is a new addition. This extension uses the CA3 recurrent collaterals to store specific paths (routes) traversed by the animat, and plays an important role in transferring these routes to *long-term storage* via *slow cortical learning*. Using this theory, they have not only explained the performance of the rats in the Morris water-maze, but also provided accounts of *anterograde* and *retrograde* amnesia caused by hippocampal lesions.

10.4.3.2 Suggestions and Predictions

Although we have successfully simulated all the four experiments of Morris (1981), there are still a number of issues that must be explored and explained, including mechanisms for goal selection in rats. In the following we suggest a few experiments that can be performed in the Morris water-maze and how they might add to our knowledge of spatial learning processes in animals.

Experiment 1: The experimental setup of Morris can be easily used to verify *latent learning* in these rats. This can be done by repeating the experiments without *pre-training* the rats. If the escape latencies of rats in the early trials of the new experiment are *significantly* (in a statistical sense) larger than those observed by Morris, we can conclude that rats are indeed capable of latent learning. In this case we can conclude that pre-training allowed the rats to learn aspects of their spatial environment in the absence of goals (no platform).

Experiment 2: What would happen if both kinds of platforms (visible and hidden) were present in each of the trials? What would happen if the rat was introduced into the environment at a position such that the visible platform was farther away than the hidden one? Would the rat choose to approach the visible one or the hidden one? This extension of the Morris water-maze might shed more light on the processes involved in selecting goals to approach. In particular, this will help characterize the relative importance of goal visibility and goal distance. For instance, if the rats reliably approach the visible platform, we can conclude that goal visibility overrides nearness. However, if the opposite scenario is observed, we can conclude that rats choose to approach closer goals even though they may be hidden.

Experiment 3: We can also perform a number of experiments to identify the nature of goal representation in rodents. For instance, the platform was always at the same place in the Place and Cue+Place experiments. In the first trial of Test B, the position of this platform was changed to the diagonally opposite quadrant, but was kept fixed on subsequent trials. What would happen if the platform was removed in the second trial of Test B? Would the animats in the Place experiment first navigate to the *new* position of the platform observed in trial 1 of the test or would they still visit the location of the platform in the training trials? Further, if the animats do not find the platform at the position they search first, do they navigate *directly* to the alternate goal position? Results from these experiments might help us identify whether rodents have some built-in mechanism to *systematically* search multiple goal locations.

10.5 Discussion

In this chapter we have used our computational model of hippocampal spatial learning to simulate and explain a number of behavioral experiments with rodents. In particular, we have successfully reproduced a number of experiments of Collett et al. (1986) and Morris (1981).

We have presented simulation results for the one, two, and three landmark experiments of

Collett et al. In each case, we have shown the search histograms generated by our animats. These histograms compare extremely well to the observations of (Collett *et al.*, 1986). In addition to the simulation results, we have also identified a number of behavioral experiments that can be performed to answer a number of questions regarding the processing of landmark information. These experiments attempt to identify the roles played by physical characteristics of landmarks and their distance from a place, in the learning and representation of that place.

We have also simulated the experiments and tests of Morris (1981). To the best of our knowledge, we are the first to present simulations of all four tasks. In addition, we have suggested a number experiments that could be performed to characterize the representation of goal locations and their role in rodent navigation. In particular, these studies can help identify the strategies followed by rodents when choosing between multiple goals. For instance, we might be able to identify whether rodents first visit goal locations that recently provided a reward or whether they visit goal locations that have proven to be stable and reliable based on past experience.

Although these behavioral simulations look promising and have helped us identify a number of interesting research directions to pursue, we must point out the following caveat. We must bear in mind that each animal is an *individual* in its own right and different animals might have differing capabilities for spatial information acquisition and navigation. Thus, observed differences in the behaviors of the animals arise either due to differences in the physical/mental abilities of the animals or differences in their experiences in the environment. In contrast, all our animats are *identical* in terms of their abilities, i.e., they all have the same spatial learning and localization algorithm. The only difference observed in our simulated animats comes from differences in their experiences (e.g., different starting points, different environments, etc.). Thus, results from such computational simulations are *biased* and extreme care should be taken while making direct comparisons between simulation results and animal behaviors.

11 CONCLUSION

The biological processes of evolution and learning confer animals (or biological agents) with designs, traits, and behaviors of immense adaptive value. Evolution operates over generations at the population level and is an excellent (possibly unmatched) process that governs much of the physical design of the animals and their in-born or innate tendencies and early behaviors. Learning operates during the life-span of an individual and is largely responsible for shaping its behaviors based on its experiences in the environment into which it is born. Together, these processes lead to animals that are well adapted to the environments they inhabit. Drawing inspiration from the biological processes at work in the design of these biological agents, this dissertation has explored artificial analogs of such processes in the design of artificial agents and robots.

In particular, we used an evolutionary approach to design neural network architectures for specific robot behaviors. We also employed evolution to design robot sensory systems. We then considered one form of learning in animals, namely the ability to learn, recognize, and localize in space and developed a computational characterization of the hippocampal formation, which is known to play a significant role in spatial learning. We also showed the implications of this model in robot localization. Our key developments and observations are summarized below.

1. We developed reasonably precise notions of properties of genetic representations that characterize the genetic encoding as well as the decoding process. This aids the user in identifying and choosing genetic representations that are well-suited to the problem on hand, and that allow the evolutionary system to discover solutions efficiently and effectively.

2. We used these properties to choose a genetic representation to evolve neurocontrollers for a box-pushing robot task. We presented a number of arguments, along with empirical results, to show the difficulty involved in designing good box-pushing behaviors. Unlike

other researchers (Teller, 1994), we also *analyzed* the evolved neural structures and determined precisely *how* our agents achieved their high fitnesses. This led to important insights into the kinds of behaviors that would *intuitively* work well given the constraints of the robot task (Balakrishnan & Honavar, 1996a).

3. By manipulating the *coding of the outputs* of the neurocontrollers we also demonstrated that the behaviors of successful agents did not change in principle. We additionally showed that evolution discovered radically different neural structures, each producing the same robot behavior. We used this to argue that the evolved behaviors were truly characteristic of the properties and constraints of the environment (Balakrishnan & Honavar, 1996a).

4. We also relaxed and modified the environmental constraints in controlled ways and showed that evolution produced high fitness designs that exploited these changes (Balakrishnan & Honavar, 1996a; Balakrishnan & Honavar, 1996c). For instance, when we provided a simple form of feedback to robots engaged in futile pushing, radically different behaviors emerged that *exploited* the feedback mechanism to confer the robots with high fitnesses (Balakrishnan & Honavar, 1996a).

5. We showed that artificial evolution could be used in the design of robot sensory systems by allowing it to choose the number, placement, and ranges of the sensors (Balakrishnan & Honavar, 1996b). We discovered that having more sensors was potentially detrimental to the fitness of the robot, possibly due to conflicts or confusion caused by excess sensory information.

6. When the sensors were assumed to be noisy (as is the case with many contemporary robot sensors (Everett, 1995)) we found that evolution of sensory systems led to *robust, fault-tolerant designs* that involved duplication of sensory resources along critical dimensions (Balakrishnan & Honavar, 1996c). For instance, most of the evolved robots made use of two or more sensors placed to sense the key cell immediately ahead of the robot. In these designs even if one sensor failed, the others would provide reliable sensor readings.

7. By mapping the different components of the agent into the common currency of *power consumption*, we also evolved neurocontrollers that were optimized across multiple dimen-

sions (e.g., performance, numbers of sensors, units, links, etc.). In these cases, evolution produced high fitness agents with a few sensors in key positions and neurocontrollers with no hidden units (Balakrishnan & Honavar, 1999).

8. We also demonstrated the increased complexity in the designs of the energy-constrained robots when they had access to a spatial learning mechanism to learn, remember, and navigate to power sources within the environment. With such abilities built-in, evolution discovered designs that employed multiple sensors and hidden units, and that appeared to have better box-pushing behaviors.

9. With this motivation, we developed a computational model of spatial learning and localization in rodents. The model was inspired by the *locale system* hypothesis of hippocampal spatial learning (O'Keefe & Nadel, 1978), and assumed that animals learned place maps grounded in *metric* information derived from animal dead-reckoning.

10. Our model differed from other metric models (e.g., (Wan *et al.*, 1994; Redish & Touretzky, 1996), in *explicitly* handling uncertainty in the information sources (e.g., sensory inputs, dead-reckoning, etc.). To the best of our knowledge, none of the other hippocampal models of spatial learning have addressed this issue of information fusion and localization from uncertain sources (Balakrishnan *et al.*, 1997).

11. We drew a parallel between the posited hippocampal function and probabilistic localization approaches used in contemporary robotics (e.g., Kalman filter). Based on this parallel, we developed a Kalman filtering framework for hippocampal spatial localization with update expressions that could be shown to be *stochastically optimal* (Balakrishnan *et al.*, 1997).

12. We extended the framework to support incremental map learning and learning of goal locations. This made it possible to learn local metric maps and fuse them into global ones in a consistent manner. We also developed a framework for learning, representing, and navigating to multiple goal locations in the environment (Balakrishnan *et al.*, 1998a).

13. We used the model to simulate a number of behavioral experiments, primarily the gerbil experiments of (Collett *et al.*, 1986) and the water-maze task of (Morris, 1981). In either

case, the behaviors demonstrated by our animats were largely similar to those observed with rodents (Balakrishnan *et al.*, 1998b; Balakrishnan *et al.*, 1998a).

14. Based on our simulation results, we also suggested a number of further experiments that could be performed by cognitive scientists and animal behaviorists, to improve our understanding of these processes in animals.

15. Our characterization of hippocampal spatial learning also generated many testable predictions for neuroscientists. For instance, we suggested that the CA1 layer of the hippocampus was involved in distinguishing between perceptually similar places, a hypothesis that could be confirmed via neuroscientific experiments.

16. We also contributed to the literature on robot navigation by providing a *place-based* extension of Kalman filtering used in robotics. We also provided a mechanism for distinguishing between perceptually similar places in the environment (also known as *perceptual aliasing* in robotics), by using the *Mahalanobis distance* and the robot's dead-reckoning estimates (Balakrishnan *et al.*, 1997).

11.1 Directions for Future Work

This dissertation research has opened up many other interesting avenues of research. In this section, we describe some of these directions that appear promising and suggest ways in which one may fruitfully pursue them.

11.1.1 Adaptation of Behavior

In addition to evolution and spatial learning, we can also explore the use of other forms of learning phenomena like *classical* and *operant conditioning* (Mackintosh, 1983; Levine, 1991), in altering the genetically-programmed behaviors of the agents. Thus, evolution might provide the robot with a set of innate behaviors that the robot might alter based on its experiences in the environment in which it is currently operating (Colombetti & Dorigo, 1992). For example, *reinforcement learning* is a paradigm of machine learning that is inspired by such *conditioning* mechanisms in animals (Barto & Sutton, 1981; Sutton & Barto, 1998). Owing to its ability to operate in feedback-impoverished environments, reinforcement learning has found applica-

tion in robot learning (Connell & Mahadevan, 1993b). We are exploring the use of similar mechanisms in our agents to allow them to adapt to the specific environments they encounter.

11.1.2 Decision-Theoretic Mechanisms for Behavior Selection

Decision-making in animals has been the subject of considerable research (Grier & Burk, 1992; McFarland, 1993; McFarland & Bosser, 1993). Some researchers believe that animal decision-making can be characterized by utility-theoretic tradeoffs (von Neumann & Morgenstern, 1944; Keeney & Raiffa, 1976) and have suggested that animals make decisions by maximizing *expected utilities* of different action choices (McFarland, 1993; McFarland & Bosser, 1993). These *utility functions* signify the *usefulness* of specific actions in given contexts. The problem then is to determine the right utility functions that will allow the animals to engage in *rational* decision-making (von Neumann & Morgenstern, 1944). It has been suggested that these utility functions are pre-programmed by evolution and possibly adapted by the experiences of the animal (McFarland & Bosser, 1993). We are exploring the use of such utility-based decision making mechanisms in our autonomous agents.

A related research direction is in the evolution of reward systems for *self-guided learning*. Operant conditioning or reinforcement learning paradigms require reward/penalty feedback from the environment in order to learn or adapt behaviors. However, there are many animal behaviors that appear to be self-supervised. We can thus think of a built-in *reward system* that automatically rewards or penalizes specific behaviors or actions performed by the robot. Operant conditioning or reinforcement learning paradigms can then be used to adapt the behaviors based on these rewards and penalties. If such reward systems are subjected to an evolutionary design process, it is possible that some innate properties of the environment and its constraints will be captured by the reward system. This will allow the robots to quickly learn effective behaviors without the need for any explicit feedback from the environment itself. This research direction is extremely promising and has many potential applications.

11.1.3 Extensions of the Spatial Learning Model

The hippocampal spatial learning model developed in the course of this dissertation research can be extended along several crucial dimensions. The current model does not include any mechanism for adapting place codes. For instance, places are defined in the CA3 layer based

on the firing of EC layer cells. Remember that the EC layer cells encode vectors to specific landmarks. If some landmarks move or are removed from the environment, our spatial learning model cannot adapt to the changed sensory conditions. We have developed a simple extension (that uses a variation of the *competitive Hebbian learning* employed in (Sharp, 1991)), to adapt the representation of place in the CA3 layer based on the sensory inputs available in the EC layer. This extension allows the spatial learning system to cope with dynamic environments, in which objects move. It is our hope that such a mechanism will allow the system to learn places based on *stable* sensory cues and ignore unstable or dynamic ones (Biegler & Morris, 1996; Bennett, 1996).

Our current implementation of frame merging fuses local place maps if it finds a place that is sensorily common to the two maps. However, this will not work if multiple places in the environment are sensorily identical (perceptual aliasing). In such cases it is quite possible that the two maps will be incorrectly fused at the wrong place. Since we have argued that perceptual aliasing may be quite prevalent in the environments of animals, improving the frame merging procedure to work in perceptually aliased environments is an important research direction to pursue.

11.1.4 Testing Predictions of the Spatial Learning Model

As we have outlined in Chapters 8 and 10, our model of spatial learning makes a number of behavioral and neurobiological predictions. For instance, our model suggests that animals can increase their accuracy of localization by frequently returning to the origin of their excursions. Further, our model suggests that exploring animals will search in slowly expanding trajectories from their place of safety (or nest) as this allows more precise position estimates to be maintained and propagated. These predictions can be verified via controlled experiments in appropriately designed environments.

We have also suggested several extensions of the behavioral tasks of Collett et al. (1986) and Morris (1981). These were outlined in Sections 10.2.3.3 and 10.4.3.2. Much insight can be gained from the study of these experiments.

Our spatial learning model ascribes specific roles to the different hippocampal regions. Our primary assertion is that the CA1 layer helps in the resolution of perceptual ambiguities through the use of dead-reckoning information. We have also suggested that the CA3 recur-

rent collaterals play a role in the propagation and update of covariances required in Kalman filtering. These suggestions have to be verified (or refuted) through appropriate neuroscientific experiments.

11.2 Overall Contributions of this Dissertation

This dissertation contributes to science and technology of designing intelligent, autonomous agents (robots and softbots) by using artificial analogs of biologically inspired mechanisms like evolution and learning. The use of evolution to design agent architectures and innate behaviors, and the subsequent role of learning in the fine-tuning of the behaviors, offers an attractive approach for the design of agent applications in a variety of real-world scenarios.

The primary advantage of evolution, in this context, is its ability to design agents, using little domain-specific knowledge. Often, these evolutionary approaches exploit the properties and constraints of the environment in novel ways, and produce designs that are unique. Evolutionary design can thus be profitably used in applications where little domain-specific knowledge exists or where the intricacies and subtleties of the problem are hard to specify.

Learning mechanisms, on the other hand, make it easy and practical to design agents for a variety of specific applications. For instance, agents can be designed in *smaller* versions of their intended operating environments. They can then be expected to use their learning mechanisms to *scale* up to their actual environments. As an example, consider an *indoor mail delivery robot* that can be designed in a small building with a few rooms and later made operational in larger office buildings.

These mechanisms also permit robots to be designed in simulated environments and later transferred onto actual robots operating in real-world, physical environments. Such a design strategy will be of use, for instance, in the design of a space explorer like the Mars sojourner, where the robot can be designed in a simulation of the Martian environment and later *adapt* to the actual Martian surface. Importantly, learning mechanisms allow the agents to be *customized* to the needs and constraints of specific users (Mitchell *et al.*, 1994; Nwana, 1996; Maes, 1997). This is of import in the design of software agents for applications such as news filtering. In these situations, a *general* news filtering agent, equipped with appropriate learning abilities, can be designed, which can then tailor itself to the whims and fantasies of specific users.

By choosing to explore the design of artificial agents using biologically inspired structures

and processes, this dissertation also contributes to modeling efforts in computational neuroscience and computational biology. Our computational model of hippocampal spatial learning and localization is a direct neuro-cognitive modeling effort. In addition, our framework can be easily used to test *other* computational aspects of animal behavior (e.g., paradigms of classical and operant conditioning; models of visual input processing, learning and recognition, etc.). Our evolutionary framework can be used to study the evolution of such learning and behavior mechanisms in animals, and possibly isolate the evolutionary constraints and pressures that led to the formation of these structures and processes in the biological agents.

APPENDIX A CRITICAL ASSUMPTIONS OF THE HIPPOCAMPAL COMPUTATIONAL MODEL

Some key assumptions have to be made for our computational model to be realizable. These assumptions are clarified in this section.

1. **Recognized landmark information**

   Our model assumes that the EC inputs consist of recognized landmarks along with information pertaining to their allocentric position relative to the robot. This was required in our model because of its analogy to the hippocampal formation, which is known to derive highly processed inputs from other associational areas of the brain. Since object and scene recognition are still big problems in contemporary robotics, is this approach only a simulation toy that is unimplementable on real robots? We suggest not. It might be observed that the functioning of our localization system depends on the notion of a place but does not depend on the exact mechanisms for defining a place representation. Thus, any prudent choice of a place representation would work. For e.g., a *local occupancy-grid* representation of a place can be used in our model, making it implementable on a mobile robot with sonar sensors. Such representations have been successfully used by other researchers (Langley & Pfleger, 1995; Recce & Harris, 1996; Yamauchi & Langley, 1997).

2. **Reliance on dead-reckoning**

   The model discussed in this paper assumes that the robot has a reliable dead-reckoning system with known error models. Dead-reckoning in contemporary robots is usually performed through odometric techniques that use optical and magnetic wheel encoders. Since the odometric approaches derive their navigational parameters from wheel rotation, they are subject to problems arising from slippage, tread wear, and/or improper tire inflation (Balakrishna & Ghosal, 1995; Everett, 1995). However, Doppler and inertial navigation systems can often be used to considerably reduce these sources of errors

(Everett, 1995). *Doppler navigation* systems operate on the principle of the Doppler shift in frequency, observed when radiated energy reflects off a surface that is moving with respect to the emitter. *Inertial navigation* systems rely on a set of mechanisms that continuously sense minute accelerations in each of the three directional axes. These are then integrated over time to yield velocity and position. Such systems, although capable of producing reasonably precise dead-reckoning, currently have limited applications owing to their prohibitive cost (Everett, 1995). However, with advances in technology, it is conceivable that accurate and affordable dead-reckoning devices will soon become available, making our approach quite practical.

3. **Maintaining correlations**

   With probabilistic localization approaches like the Kalman filter, it is imperative that the correlations between the state variables be maintained correctly. Estimating and maintaining these correlations is often difficult owing to the approximation errors stemming from linearizing the (usually) non-linear system and measurement functions, biases on the robot position, and the computational complexity associated with updating state vectors containing a large number of elements (Hebert *et al.*, 1995). However, these correlations and variances cannot be neglected since doing so will lead to inconsistencies on the uncertainties (Hebert *et al.*, 1995). In our model, we ignore the autocorrelation of the measurement noise. However, as we argued before, if the animal navigates randomly or purposefully between place field centers, this correlation term will be negligible.

APPENDIX B SOME USEFUL PROPOSITIONS FOR HIPPOCAMPAL KALMAN FILTERING

Proposition 1 (Size of the EC fields) *In two dimensional space the EC field, or the region over which it responds, is roughly* circular *with a radius of* $r \leq \sqrt{-2\sigma_L^2 \ln(\Gamma_{EC})}$.

If the variance of the EC cell distance matching Gaussians is given by σ_L^2 and the EC cell is said to recognize a landmark if the cell activation is greater than the threshold of Γ_{EC}, then the range over which the EC cell with center (c_x, c_y) responds is given by:

$$\exp^{-\frac{(x-c_x)^2+(y-c_y)^2}{2\sigma_L^2}} \geq \Gamma_{EC} \Longleftrightarrow (x-c_x)^2 + (y-c_y)^2 \leq -2\sigma_L^2 \ln(\Gamma_{EC}) \Longleftrightarrow r^2 \leq -2\sigma_L^2 \ln(\Gamma_{EC})$$

Thus, with $\sigma_L^2 = 1$ and $\Gamma_{EC} = 0.6$, we obtain $r \leq 1.01$, i.e., the EC unit responds to a landmark within a circular region of radius 1.01 from its center.

Proposition 2 (Variance of an EC cell) *The variance associated with an EC unit is given by* $\sigma^2 = R^2\sigma_S^2 - 2\sigma_L^2 \ln(\Gamma_{EC})$

The variance associated with an EC unit firing has two components. The first component is a measure of the uncertainty associated with the EC unit center, while the second component is the imprecision tolerated by the EC firing field. Suppose the sensory range error is characterized by a zero-mean, white Gaussian with standard deviation σ_S (per unit distance), and the maximum sensory range is R. When a new EC unit is recruited and its center is set up, the imprecision (or uncertainty) associated with the center is characterized by the sensing uncertainty and has a variance of $R^2\sigma_S^2$. From Proposition 1, the position uncertainty within the EC field is given by $r^2 \leq -2\sigma_L^2 \ln(\Gamma_{EC})$. Since these two sources of uncertainty are independent, the net variance associated with an EC cell is given by: $\sigma^2 = R^2\sigma_S^2 - 2\sigma_L^2 \ln(\Gamma_{EC})$. Note that this is the variance along each coordinate axis of the Cartesian system.

Proposition 3 (Form of the measurement error) *The measurement error is given by* $w_k = x_{i_k} - x_{0,k}$.

In our model the measurement function is assumed to represent the distance between the current position of the animal ($x_{0,k}$) and the center of the place field that it is currently in (x_{i_k}), corrupted by some random measurement noise. Thus

$$z_k = x_{0,k} - x_{i_k} + w_k$$

Since the true centers of the place fields are not known and the true position of the animal is also unknown (and is the very subject of this localization process), z_k cannot be measured. In our model we assume that the animal always measures 0, i.e., if the animal is in place i_k the system assumes that is must be at the center of that place. This constrains the form of the random noise to:

$$z_k = x_{0,k} - x_{i_k} + w_k = 0 \implies w_k = x_{i_k} - x_{0,k}$$

Thus, Kalman filtering in our model uses observed measurement of 0 and predicted measurement of $\hat{x}_{0,k} - \hat{x}_{i_k}$ to perform the required updates.

Proposition 4 (Measurement error has zero mean) $E(w_k) = 0$

In order to use Kalman filtering, we have to show that our measurement noise has zero mean. As shown in Proposition 3, the measurement noise is given by $w_k = x_{i_k} - x_{0,k}$. Thus:

$$E(w_k) = \int \sum_{\text{places } p} (x_p - x_{0,k}) \Pr(x_{0,k}/p) \Pr(p/k) dx_{0,k}$$

$$\iff E(w_k) = \sum_{\text{places } p} \underbrace{\int (x_p - x_{0,k}) \Pr(x_{0,k}/p) dx_{0,k}}_{0} \Pr(p/k) \iff E(w_k) = 0$$

Notice that the noise is zero-mean only if there is no prior bias on the position of the animal in the place field, i.e., each position in the place field is either equiprobable or the animal navigates from one place field center to another.

Proposition 5 (Measurement error is autocorrelated) *The measurement error* w_k *is autocorrelated, i.e.,* $E(w_k w_{k-1}^T) \neq 0$.

At time step $k-1$, suppose the robot was at the position $x_{0,k-1}$. Suppose the sensor inputs determined that the robot was in place i_{k-1} with place field center $x_{i_{k-1}}$. Then:

$$w_{k-1} = x_{i_{k-1}} - x_{0,k-1} \tag{B.1}$$

Since the robot moves by u_{k-1} (and assuming linear robot transformations), its position at step k is given by:

$$x_{0,k} = x_{0,k-1} + u_{k-1} \tag{B.2}$$

Suppose this position corresponds to place i_k with center x_{i_k}, then:

$$w_k = x_{i_k} - x_{0,k} \tag{B.3}$$

Using equations B.2 and B.1 in B.3, we get:

$$w_k = x_{i_k} - x_{i_{k-1}} - u_{k-1} + w_{k-1} \tag{B.4}$$

As can be observed, w_k depends on w_{k-1}, and hence is an autocorrelated sequence.

Proposition 6 (Variance of the measurement error) *The variance of the measurement error is given by*

$$\mathbf{R}_k = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

where $\sigma^2 = R^2\sigma_S^2 - 2\sigma_L^2 \ln(\Gamma_{EC})$.

Since the measurement error is given by $w_k = x_{i_k} - x_{0,k}$, i.e., the deviation of the current animat position from the CA3 field center, the variance of this error is the variance associated with CA3 fields. Now, CA3 fields are weighted intersections of EC fields. Thus, in the worst case (i.e., CA3 and EC fields have the same size) the variance of CA3 fields equal the variance of EC fields. Since the variance of EC cells is given by $\sigma^2 = R^2\sigma_S^2 - 2\sigma_L^2 \ln(\Gamma_{EC})$ along each coordinate axis of the Cartesian system (Proposition 2), and the covariance between the axes are assumed to be zeros, the variance of the measurement error becomes:

$$\mathbf{R}_k = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

Proposition 7 (Covariance in the Mahalanobis test) *The covariance matrix of the Mahalanobis distance test used in Section 8.3.5 is given by* $(\mathbf{C}_{i_k i_k} + \mathbf{C}_{00} - 2\mathbf{C}_{0i_k} + \mathbf{R}_k)$

Suppose we are considering CA1 unit i_k with true place field center x_{i_k} and estimated center $\hat{x}_{i_k}$. Since predictions in our model are given by $\hat{z}_k = \hat{x}_{0,k} - \hat{x}_{i_k}$ while the observations are given by $z_k = x_{0,k} - x_{i_k} + w_k$, we can determine the covariance between the predicted and observed measurements of the Mahalanobis test for CA1 unit i_k.

$$
\begin{aligned}
& E((\hat{z}_k - z_k)(\hat{z}_k - z_k)^T) \\
\iff \quad & E((\hat{x}_{0,k} - \hat{x}_{i_k} - x_{0,k} + x_{i_k} - w_k)(\hat{x}_{0,k} - \hat{x}_{i_k} - x_{0,k} + x_{i_k} - w_k)^T) \\
\iff \quad & E((\epsilon_{i_k} - \epsilon_{0,k} - w_k)(\epsilon_{i_k} - \epsilon_{0,k} - w_k)^T) \\
\iff \quad & \mathbf{C}_{i_k i_k} + \mathbf{C}_{00} - 2\mathbf{C}_{0i_k} + \mathbf{R}_k
\end{aligned}
$$

where $\epsilon_{i_k} = x_{i_k} - \hat{x}_{i_k}$ and $\epsilon_{0,k} = x_{0,k} - \hat{x}_{0,k}$. We also assume that the dead-reckoning and measurement noises are independent, i.e. $E(\epsilon_{i_k} w_k^T) = E(\epsilon_{0,k} w_k^T) = E(w_k \epsilon_{i_k}^T) = E(w_k \epsilon_{0,k}^T) = 0$.

Proposition 8 (Chi-square (χ^2) distribution and the Mahalanobis test) *The distance threshold for the Mahalanobis test is 4.61*

The probability density function of a χ^2 distribution is an asymmetric curve with a long right-hand tail. Given the number of degrees of freedom, we can determine the value of the distribution (χ^2_α) such that the area under the curve to the right of it is α (Johnson & Bhattacharyya, 1996). Since the Mahalanobis distance has a chi-square distribution, we can choose a value of the distance such that the area to the right of it is, say, 10%. Since the covariance matrix of the Mahalanobis test shown in proposition 7 has a rank of 2, we determine the value of the chi-square distribution with 2 degrees of freedom such that the area to the right of it is 10%. This value is 4.61 (Johnson & Bhattacharyya, 1996).

Thus, in our model, if we compute the Mahalanobis distance and declare that the match is correct if this distance happens to be less than 4.61, then we can be 90% confident of *not* rejecting actually correct matches.

Proposition 9 (Covariance in the Mahalanobis test for goal updates) *The covariance matrix of the Mahalanobis distance test used in updating goal position estimates is given by* $\mathbf{C}_{GG} + \mathbf{C}_{00}$.

Since $\hat{x}_G$ serves as a prediction of the goal position and $\hat{x}_{0,k}$ constitutes an observation, the covariance between the predicted and observed goal positions is given by

$$\begin{aligned} & E((\hat{x}_G - \hat{x}_{0,k})(\hat{x}_G - \hat{x}_{0,k})^T) \\ \iff \quad & E((\hat{x}_G - x_G + x_G - \hat{x}_{0,k} + x_{0,k} - x_{0,k})(\hat{x}_G - x_G + x_G - \hat{x}_{0,k} + x_{0,k} - x_{0,k})) \\ \iff \quad & E((-\epsilon_G + \epsilon_{0,k} + x_G - x_{0,k})(-\epsilon_G + \epsilon_{0,k} + x_G - x_{0,k})) \\ \iff \quad & \mathbf{C}_{GG} + \mathbf{C}_{00} \end{aligned}$$

assuming $\mathbf{C}_{0G}, \mathbf{C}_{G0}$ and other covariances are all zero.

Proposition 10 (The 2.5σ boundary) *The 2.5σ boundary around an estimated parameter is expected to include the true value with probability greater than 84%*

According to Chebyshev's inequality, if μ and σ are the mean and standard deviation of a random variable X, then for any positive constant k:

$$\Pr(|X - \mu| < k\sigma) \geq 1 - \frac{1}{k^2} \tag{B.5}$$

Thus, if x_i is the estimated parameter with current estimate $\hat{x}_i$ and variance σ, using $k = 2.5$ in Chebyshev's theorem yields:

$$\Pr(|x_i - \hat{x}_i| < 2.5\sigma) \geq 0.84 \tag{B.6}$$

Thus, the 2.5σ boundary is expected to include the true value with probability greater than 84%.

BIBLIOGRAPHY

Ackley, David H., & Littman, Michael L. (1991). Interactions between Learning and Evolution. *Pages 487–509 of: Proceedings of the Second International Conference on Artificial Life.*

Albus, J. (1981). *Brains, Behavior, and Robotics.* Peterborough, NH: McGraw Hill.

Anand, D., & Zmood, R. (1995). *Introduction to Control Systems.* Oxford: Butterworth-Heinemann.

Anderson, E. (1983). *Animals as Navigators.* New York, NY: Van Nostrand Reinhold.

Arbib, M. (1966). Simple Self-Reproducing Universal Automata. *Information and Control*, **9**, 177–189.

Ayache, N., & Faugeras, O. (1987). Maintaining Representation of the Environment of a Mobile Robot. *In: Proceedings of the International Symposium on Robotics Research.*

Bachelder, I., & Waxman, A. (1994). Mobile Robot Visual Mapping and Localization: A View-Based Neurocomputational Architecture that Emulates Hippocampal Place Learning. *Neural Networks*, **7**, 1083–1099.

Bäck, T., Rudolph, G., & Schwefel, H-P. (1993). Evolutionary Programming and Evolution Strategies: Similarities and Differences. *In: Proceedings of the Second Annual Conference on Evolutionary Programming.*

Balakrishna, R., & Ghosal, A. (1995). Modeling of Slip for Wheeled Mobile Robots. *IEEE Transactions on Robotics and Automation*, **11**(1), 126–132.

Balakrishnan, K., & Honavar, V. (1995a). *Evolutionary Design of Neural Architectures — A Preliminary Taxonomy and Guide to Literature.* Tech. rept. CS TR 95-01. Department of Computer Science, Iowa State University, Ames, IA.

Balakrishnan, K., & Honavar, V. (1995b). Properties of Genetic Representations of Neural Architectures. *In: Proceedings of the World Congress on Neural Networks.*

Balakrishnan, K., & Honavar, V. (1996a). Analysis of Neurocontrollers Designed by Simulated Evolution. *In: Proceedings of IEEE International Conference on Neural Networks ICNN'96.*

Balakrishnan, K., & Honavar, V. (1996b). On Sensor Evolution in Robotics. *In: Proceedings of Genetic Programming Conference – GP-96.*

Balakrishnan, K., & Honavar, V. (1996c). Some Experiments in the Evolutionary Synthesis of Robotic Neurocontrollers. *Pages 1035–1040 of: Proceedings of the World Congress on Neural Networks.*

Balakrishnan, K., & Honavar, V. (1998). Intelligent Diagnosis Systems. *International Journal of Intelligent Systems*, **8**(3/4). (In press).

Balakrishnan, K., & Honavar, V. (1999). Some Experiments in Evolutionary Neuro-robotics. *In:* Patel, M., Honavar, V., & Balakrishnan, K. (eds), *Evolutionary Synthesis of Neural Systems.* Cambridge, MA: MIT Press. (To appear).

Balakrishnan, K., Bousquet, O., & Honavar, V. (1997). *Spatial Learning and Localization in Animals: A Computational Model and its Implications for Mobile Robots.* Tech. rept. CS TR 97-20. Department of Computer Science, Iowa State University, Ames, IA. (To appear in Adaptive Behavior).

Balakrishnan, K., Bhatt, R., & Honavar, V. (1998a). A Computational Model of Rodent Spatial Learning and Some Behavioral Experiments. *In: Proceedings of the Twentieth Annual Meeting of the Cognitive Science Society.*

Balakrishnan, K., Bhatt, R., & Honavar, V. (1998b). Spatial Learning and Localization in Animals: A Computational Model and Behavioral Experiments. *Pages 112–119 of: Proceedings of the Second European Conference on Cognitive Modelling.*

Balakrishnan, K., Bhatt, R., & Honavar, V. (1998c). *Spatial Learning in the Rodent Hippocampus: A Computational Model and Some Behavioral Results.* (In preparation).

Barnes, C., McNaughton, B., Mizumori, S., Leonard, B., & Lei-Huey, L. (1990). Comparison of Spatial and Temporal Characteristics of Neuronal Activity in Sequential Stages of Hippocampal Processing. *Progress in Brain Research*, **83**, 287–299.

Barto, A., & Sutton, R. (1981). Associative Search Network: A Reinforcement Learning Associative Memory. *Biological Cybernetics*, **40**, 201–211.

Bennett, A. (1993a). Remembering Landmarks. *Nature*, **364**, 293–294.

Bennett, A. (1993b). Spatial Memory in a Food Storing Corvid. I. Near Tall Landmarks are Primarily Used. *Journal of Comparative Physiology A*, **173**, 193–207.

Bennett, A. (1996). Do Animals have Cognitive Maps. *The Journal of Experimental Biology*, **199**(1), 219–224.

Biegler, R., & Morris, R. (1996). Landmark Stability: Studies Exploring Whether the Perceived Stability of the Environment Influences Spatial Representation. *The Journal of Experimental Biology*, **199**(1), 187–193.

Blair, H., & Sharp, P. (1995). Anticipatory Firing of Anterior Thalamic Head Direction Cells: Evidence for a Thalamocortical Circuit that Computes Head Direction in the Rat. *Journal of Neuroscience*, **15**, 6260–6270.

Blum, K., & Abbott, L. (1996). A Model of Spatial Map Formation in the Hippocampus of the Rat. *Neural Computation*, **8**, 85–93.

Bousquet, O., Balakrishnan, K., & Honavar, V. (1998). Is the Hippocampal Formation a Kalman Filter? *In: Proceedings of the Pacific Symposium on Biocomputing.*

Bradshaw, J. (ed). (1997). *Software Agents.* Cambridge, MA: MIT Press.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* Cambridge, MA: MIT Press.

Brooks, R. (1985). Visual Map Making for a Mobile Robot. *In: Proceedings of the IEEE Conference on Robotics and Automation.*

Brown, M., & Sharp, P. (1995). Simulation of Spatial Learning in the Morris Water Maze by a Neural Network Model of the Hippocampal Formation and Nucleus Accumbens. *Hippocampus*, **5**, 189–197.

Burgess, N., & O'Keefe, J. (1996). Neuronal computations underlying the firing of place cells and their role in navigation. *Hippocampus*, **6**, 749–762.

Burgess, N., Recce, M., & O'Keefe, J. (1994). A Model of Hippocampal Function. *Neural Networks*, **7**(6/7), 1065–1081.

Burgess, N., Recce, M., & O'Keefe, J. (1995). Hippocampus – Spatial Models. *Pages 468–472 of:* Arbib, M. (ed), *The Handbook of Brain Theory and Neural Networks.* Cambridge, MA: MIT Press.

Buzsaki, G. (1989). Two-State Model of Memory Trace Formation: A Role for "Noisy" Brain States. *Neuroscience*, **31**, 551–570.

Cecconi, F., Menczer, F., & Belew, R. (1995). Maturation and Evolution of Imitative Learning in Artificial Organisms. *Adaptive Behavior*, **4**(1), 179–198.

Chalmers, David J. (1990). The Evolution of Learning: An Experiment in Genetic Connectionism. *Pages 81–90 of: Proceedings of the 1990 Connectionist Models Summer School.*

Chapuis, A., & Droz, E. (1956). *The Jaquet-Droz Mechanical Puppets.* Neuchatel, Switzerland: Neuchatel Historical Museum.

Chen, L., Lin, L-H., Green, E., Barnes, C., & McNaughton, B. (1994b). Head-Direction Cells in the Rat Posterior Cortex. I. Anatomical Distribution and Behavioral Modulation. *Experimental Brain Research*, **101**, 8–23.

Chen, L., Lin, L-H., Barnes, C., & McNaughton, B. (1994a). Head-Direction Cells in the Rat Posterior Cortex. II. Contributions of Visual and Ideothetic Information to the Directional Firing. *Experimental Brain Research*, **101**, 24–34.

Chown, E., Kaplan, S., & Kortenkamp, D. (1995). Prototypes, Location and Associative Networks (PLAN): Towards a Unified Theory of Cognitive Mapping. *The Journal of Cognitive Science*, **19**(1).

Churchland, P., & Sejnowski, T. (1992). *The Computational Brain.* Cambridge, MA: MIT Press.

Cliff, D., Husbands, P., & Harvey, I. (1993a). Analysis of Evolved Sensory-Motor Controllers. *In: Second European Conference on Artificial Life.*

Cliff, D., Harvey, I., & Husbands, P. (1993b). Explorations in Evolutionary Robotics. *Adaptive Behavior*, **2**, 73–110.

Cohen, M., & Drabkin, I. (1948). *Source Book in Greek Science.* New York, NY: McGraw Hill.

Cohen, N., & Eichenbaum, H. (1993). *Memory, Amnesia, and the Hippocampal System.* Cambridge, MA: MIT Press.

Collett, T., Cartwright, B., & Smith, B. (1986). Landmark Learning and Visuo-Spatial Memories in Gerbils. *Journal of Neurophysiology A*, **158**, 835–851.

Collins, R., & Jefferson, D. (1990). An Artificial Neural Network Representation for Artificial Organisms. *Pages 259–263 of: Proceedings of the Conference on Parallel Problem Solving from Nature.*

Collins, R., & Jefferson, D. (1991). AntFarm: Towards Simulated Evolution. *In: Proceedings of the Second International Conference on Artificial Life.*

Colombetti, M., & Dorigo, M. (1992). Learning to Control an Autonomous Robot by Distributed Genetic Algorithms. *In: From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.*

Connell, J., & Mahadevan, S. (1993a). Rapid Task Learning for Real Robots. *Chap. 5, pages 105–140 of:* Connell, J., & Mahadevan, S. (eds), *Robot Learning.* Boston, MA: Kluwer Academic.

Connell, J., & Mahadevan, S. (eds). (1993b). *Robot Learning.* Boston, MA: Kluwer Academic.

Cox, I., & Wilfong, G. (eds). (1990). *Autonomous Robot Vehicles.* New York, NY: Springer-Verlag.

Crowley, J. (1995). Mathematical Foundations of Navigation and Perception for an Autonomous Mobile Robot. *Pages 9–51 of: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Culbertson, J. (1957). Robots and Automata: A Brief History. *Computers and Automation*, **6**(3/4), 32–37/28–35.

Culbertson, J. (1963). *The Minds of Robots: Sense Data, Memory Images, and Behavior in Conscious Automata.* Urbana-Champaign, IL: University of Illinois Press.

Dasgupta, Dipankar, & McGregor, Douglas. (1992). Designing Application-Specific Neural Networks using the Structured Genetic Algorithm. *Pages 87–96 of: Proceedings of the International Conference on Combinations of Genetic Algorithms and Neural Networks.*

Dayhoff, J. (1990). *Neural Network Architectures: An Introduction.* New York: Van Nostrand Reinhold.

Dean, T., Allen, J., & Aloimonos, Y. (1995). *Artificial Intelligence - Theory and Practice.* Redwood City, CA: Benjamin Cummings.

Douglas, R. (1972). Pavlovian Conditioning and the Brain. *Pages 529–549 of:* Boakes, R., & Halliday, M. (eds), *Inhibition and Learning.* London: Academic Press.

Douglas, R., & Pribram, K. (1966). Learning and Limbic Lesions. *Neuropsychologia*, **4**, 197–220.

Dudek, G., Romanik, K., & Whitesides, S. (1995). Localizing a Robot with Minimum Travel. *Pages 437–446 of: Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms.*

Durkin, J. (1994). *Expert Systems – Design and Development.* New York, NY: Macmillan.

Elfes, A. (1989). *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* Ph.D. thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA.

Elfes, A. (1992). Dynamic Control of Robot Perception Using Multi-Property Inference Grids. *In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation.*

Elfes, A. (1995). Robot Navigation: Integrating Perception, Environmental Constraints and Task Execution Within a Probabilistic Framework. *Pages 93–130 of: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Engelson, S. (1994). *Passive Map Learning and Visual Place Recognition.* Ph.D. thesis, New Haven, CT, Department of Computer Science, Yale University.

Etienne, A. (1985). The Control of Short-Distance Homing in the Golden Hamster. *Pages 233–251 of:* Ellen, P., & Thinus-Blanc, C. (eds), *Cognitive Processes and Spatial Orientation in Animals and Man.* Boston, MA: Martinus Nijhoff.

Etienne, A. (1992). Navigation of a Small Mammal by Dead Reckoning and Local Cues. *Current Directions in Psychological Science*, **1**(2), 48–52.

Etienne, A., Teroni, V., Hurni, C., & Portenier, V. (1990). The Effect of A Single Light Cue on Homing Behaviour of the Golden Hamster. *Animal Behavior*, **39**, 17–41.

Etienne, A., Maurer, R., & Seguinot, V. (1996). Path Integration in Mammals and its Interaction with Visual Landmarks. *The Journal of Experimental Biology*, **199**(1), 201–209.

Everett, H. (1995). *Sensors for Mobile Robots: Theory and Application.* Wellesley, MA: A. K. Peters Ltd.

Feigenbaum, J., & Rolls, E. (1991). Allocentric and Egocentric Spatial Information Processing in the Hippcampal Formation of the Behaving Primate. *Psychobiology*, **19**, 21–40.

Fenton, A., & Bures, J. (1994). Interhippocampal Transfer of Place Navigation Monocularly Acquired by Rats During Unilateral Functional Ablation of the Dorsal Hippocampus and Visual Cortex with Lidocaine. *Neuroscience*, **58**, 481–491.

Floreano, D., & Mondada, F. (1994). Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. *Pages 421–430 of: From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.*

Fogel, D. (1994). Asymptotic Convergence Properties of Genetic Algorithms and Evolutionary Programming: Analysis and Experiments. *Cybernetics and Systems: An International Journal*, **25**, 389–407.

Fonseca, C., & Fleming, P. (1995). An Overview of Evolutionary Algorithms in Multi-Objective Optimization. *Evolutionary Computation*, **3**(1), 1–16.

Foster, T., Castro, C., & McNaughton, B. (1989). Spatial Selectivity of Rat Hippocampal Neurons: Dependence on Preparedness for movement. *Science*, **244**, 1580–1582.

Freund, J. (1992). *Mathematical Statistics.* Englewood Cliffs, NJ: Prentice Hall.

Gallant, S. (1993). *Neural Network Learning and Expert Systems.* Cambridge, MA: MIT Press.

Gallistel, C. (1990). *The Organization of Learning.* Cambridge, MA: MIT Press.

Gallistel, C., & Cramer, A. (1996). Computations on Metric Maps in Mammals: Getting Oriented and Choosing a Multi-Destination Route. *The Journal of Experimental Biology*, **199**(1), 211–217.

Gelb, A. (1974). *Applied Optimal Estimation.* Cambridge, MA: MIT Press.

Ginsberg, M. (1993). *Essentials of Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison Wesley.

Grier, J., & Burk, T. (1992). *Biology of Animal Behavior.* 2 edn. New York, NY: Mosley-Year Book.

Gruau, Frederic. (1994). Genetic Micro Programming of Neural Networks. *In:* Kinnear, Kim (ed), *Advances in Genetic Programming.* Cambridge, MA: MIT Press.

Harvey, I., Husbands, P., & Cliff, D. (1992). Issues in Evolutionary Robotics. *In: From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.*

Harvey, I., Husbands, P., & Cliff, D. (1994). Seeing the Light: Artificial Evolution, Real Vision. *In: From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.*

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks.* Cambridge, MA: MIT Press.

Haykin, S. (1994). *Neural Networks.* New York, NY: Macmillan.

Hebb, D. (1949). *The Organization of Behavior.* New York, NY: John Wiley.

Hebert, P., Betge-Brezetz, S., & Chatila, R. (1995). Probabilistic Map Learning: Necessity and Difficulties. *Pages 307–321 of: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the Theory of Neural Computation.* Redwood City, CA: Addison Wesley.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press.

Honavar, V. (1990). *Generative Learning Structures and Processes for Generalized Connectionist Networks.* Ph.D. thesis, Department of Computer Science, University of Wisconsin, Madison, WI.

Honavar, V. (1994). Toward Learning Systems That Integrate Different Strategies and Representations. *Pages 561–580 of:* Honavar, V., & Uhr, L. (eds), *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* San Diego, CA: Academic Press.

Honavar, V., & Uhr, L. (1993). Generative Learning Structures and Processes for Generalized Connectionist Networks. *Information Sciences*, **70**, 75–108.

Horn, J., & Nafpliotis, N. (1993). *Multiobjetive Optimization Using the Niched Pareto Genetic Algorithm.* IlliGAL Technical Report 93005. University of Illinois, Urbana-Champaign, IL.

Hull, C. (1934a). The Concept of Habit-Family Hierarchy and Maze Learning. I. *Psychological Review*, **41**, 33–54.

Hull, C. (1934b). The Concept of Habit-Family Hierarchy and Maze Learning. II. *Psychological Review*, **41**, 134–152.

Hull, C. (1943). *Principles of Behavior.* New York, NY: Appleton-Century-Crofts.

Hull, C. (1951). *Essentials of Behavior.* Westport, CT: Greenwood Press.

Hull, C. (1952). *A Behavior System.* New Haven, CT: Yale University Press.

Hummel, R. (1995). Uncertainty Reasoning in Object Recognition by Image Processing. *Pages 131–145 of: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Husbands, P., Harvey, I., & Cliff, D. (1993). Analysing Recurrent Dynamical Networks Evolved for Robot Control. *In: Proceedings of the 3rd IEE International Conference on Artificial Neural Networks.*

Hwang, Y., & Ahuja, N. (1992). Gross Motion Planning – A Survey. *ACM Computing Surveys*, **24**(3), 219–291.

Jakobi, N., Husbands, P., & Harvey, I. (1995). Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics. *In: Proceedings of the Third European Conference on Artificial Life.*

James, W. (1890). *The Principles of Psychology.* New York, NY: Holt, Rinehart and Winston.

Jarrard, L. (1993). On the Role of the Hippocampus in Learning and Memory in the Rat. *Behavioral Neural Biology*, **3**, 279–287.

Jazwinski, A. (1970). *Stochastic Processes and Filtering Theory.* New York, NY: Academic Press.

Jensen, O., & Lisman, J. (1996). Hippocampal CA3 Region Predicts Memory Sequences: Accounting for the Phase Precession of Place Cells. *Learning and Memory*, **3**, 279–287.

Johnson, R., & Bhattacharyya, G. (1996). *Statistics: Principles and Methods.* New York, NY: John Wiley.

Juedes, D., & Balakrishnan, K. (1996). Generalized Neural Networks, Computational Differentiation, and Evolution. *In:* Berz, M., Bischof, C., Corliss, G., & Griewank, A. (eds), *Computational Differentiation: Applications, Techniques, and Tools.* Philadelphia, PA: SIAM Press.

Jung, M., & McNaughton, B. (1993). Spatial Selectivity of Unit Activity in the Hippocampal Granular Layer. *Hippocampus*, **3**, 165–182.

Kalman, R. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME*, **60**.

Kaplan, S. (1973a). Cognitive Maps, Human Needs, and the Designed Environment. *In:* Preiser, W. (ed), *Environmental Design Research, Vol 1.* New York, NY: Dowden, Hutchinson, and Ross.

Kaplan, S. (1973b). Cognitive Maps in Perception and Thought. *In:* Downs, M., & Stea, D. (eds), *Image and Environment.* New York, NY: Aldine Publishing Company.

Keeney, R., & Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York, NY: John Wiley.

Keith, J., & McVety, K. (1988). Latent Place Learning in a Novel Environment and the Influences of Prior Training in Rats. *Psychobiology*, **16**(2), 146–151.

Kimble, D. (1968). Hippocampus and Internal Inhibition. *Psychological Bulletin*, **70**, 285–295.

Knierim, J., Kudrimoti, H., & McNaughton, B. (1995). Hippocampal Place Fields, the Internal Compass, and the Learning of Landmark Stability. *Journal of Neuroscience*, **15**, 1648–1659.

Koenig, S., Goodwin, R., & Simmons, R. (1995). Robot Navigation with Markov Models: A Framework for Path Planning and Learning with Limited Computational Resources. *Pages 322–337 of: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Kolodner, J. (1993). *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann.

Kortenkamp, D. (1993). *Cognitive Maps for Mobile Robots: A Representation for Mapping and Navigation.* Ph.D. thesis, Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI.

Kortenkamp, D., & Weymouth, T. (1994). Topological Mapping for Mobile Robots Using a Combination of Sonar and Vision Sensing. *In: AAAI-94: Proceedings of the American Association of Artificial Intelligence.*

Koza, J. (1991). Genetic Evolution and Co-Evolution of Computer Programs. *In: Proceedings of the Second International Conference on Artificial Life.*

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press.

Kubie, J., & Ranck, J. (1983). Sensory-Behavioral Correlates in Individual Hippocampus Neurons in Three Situations: Space and Context. *Pages 433–447 of:* Seifert, W. (ed), *Neurobiology of the Hippocampus.* London: Academic Press.

Kuipers, B. (1978). Modeling Spatial Knowledge. *Cognitive Science*, **2**, 129–153.

Kuipers, B., & Byun, Y-T. (1991). A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Robotics and Autonomous Systems*, **8**.

Kuipers, B., Froom, R., Lee, W., & Pierce, D. (1993). The Semantic Hierarchy in Robot Learning. *Chap. 6, pages 141–170 of:* Connell, J., & Mahadevan, S. (eds), *Robot Learning.* Boston, MA: Kluwer Academic.

Kung, S. Y. (1993). *Digital Neural Networks.* New York, NY: Prentice Hall.

Langley, P. (1995). *Elements of Machine Learning.* San Mateo, CA: Morgan Kauffman.

Langley, P., & Pfleger, K. (1995). Acquisition of Place Knowledge Through Case-Based Learning. *In: Proceedings of the International Conference on Machine Learning.*

Latombe, J-C. (1991). *Robot Motion Planning.* New York, NY: Kluwer Academic.

Leonard, J., & Durrant-Whyte, H. (1992). Dynamic Map Building for and Autonomous Mobile Robot. *International Journal of Robotics Research*, **11**(4).

Levine, D. (1991). *Introduction to Neural and Cognitive Modeling.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Levitt, T., & Lawton, D. (1990). Qualitative Navigation for Mobile Robots. *Artificial Intelligence*, **44**(3), 305–360.

Levy, W. (1989). A Computational Approach to Hippocampal Function. *The Psychology of Learning and Motivation*, **23**, 243–305.

Lewis, F., Abdallah, C., & Dawson, D. (eds). (1993). *Control of Robot Manipulators.* New York, NY: Macmillan.

Lewis, H., & Papadimitriou, C. (1981). *Elements of the Theory of Computation.* Englewood Cliffs, NJ: Prentice Hall.

Lewis, M., Fagg, A., & Sodium, A. (1992). Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot. *In: Proceedings of the IEEE International Conference on Robotics and Automation.*

Li, M., & Vitanyi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications.* New York, NY: Springer-Verlag.

Linsker, R. (1990). Perceptual Neural Organization: Some Approaches Based on Network Models and Information Theory. *Annual Review of Neurosciences*, **13**, 257–281.

Lund, H., & Parisi, D. (1995). Preadaptations in Populations of Neural Networks Evolving in a Changing Environment. *Artificial Life*, **2**(2), 179–198.

Lund, H., Hallam, J., & Lee, W-P. (1997). Evolving Robot Morphology. *In: Proceedings of IEEE Fourth International Conference on Evolutionary Computation.*

Mackintosh, N. (1983). *Conditioning and Associative Learning.* New York, NY: Clarendon.

Maes, P. (1997). Agents That Reduce Work and Information Overload. *In:* Bradshaw, J. (ed), *Software Agents.* Cambridge, MA: MIT Press.

Marr, D. (1971). Simple Memory: A Theory for Archicortex. *Philosophical Transactions of the Royal Society of London*, **176**, 23–81.

Mataric, M. (1992). Integration of Representation into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, **8**(3).

Maybeck, P. (1990). The Kalman Filter: An Introduction to Concepts. *In:* Cox, I.J., & Wilfong, G. T. (eds), *Autonomous Robot Vehicle.* New York, NY: Springer-Verlag.

McCurdy, E. (1948). *The Mind of Leonardo da Vinci.* New York, NY: Dodd, Mead, and Co.

McFarland, D. (1993). *Animal Behavior.* Essex, England: Longman Scientific and Technical.

McFarland, D., & Bosser, T. (1993). *Intelligent Behavior in Animals and Robots.* Cambridge, MA: MIT Press.

McNaughton, B., & Nadel, L. (1989). Hebb-Marr Networks and the Neurobiological Representation of Action in Space. *Pages 1–63 of:* Gluck, M., & Rumelhart, D. (eds), *Neuroscience and Connectionist Theory.* Hillsdale, NJ: Lawrence Earlbaum Associates.

McNaughton, B., & Nadel, L. (1990). Hebb-Marr Networks and the Neurobiological Representation of Action in Space. *Pages 1–63 of:* Gluck, M., & Rumelhart, D. (eds), *Neuroscience and Connectionist Theory.* Hillsdale, NJ: Lawrence Earlbaum Associates.

McNaughton, B., Leonard, B., & Chen, L. (1989). Cortical-hippocampal Interactions and Cognitive Mapping: A Hypothesis Based on Reintegration of the Parietal and Inferotemporal Pathways for Visual Processing. *Psychobiology*, **17**(3), 230–235.

McNaughton, B., Knierim, J., & Wilson, M. (1995). Vector Encoding and the Vestibular Foundations of Spatial Cognition: A Neurophysiological and Computational Hypothesis. *Chap. 37 of:* Gazzaniga, M. (ed), *The Cognitive Neurosciences.* Cambridge, MA: MIT Press.

McNaughton, B., Barnes, C., Gerrard, J., Gothard, K., Jung, M., Knierim, J., Kudrimoti, H., Qin, Y., Skaggs, W., Suster, M., & Weaver, K. (1996). Deciphering the Hippocampal Polyglot: the Hippocampus as a Path-Integration System. *The Journal of Experimental Biology*, **199**(1), 173–185.

Menczer, F., & Belew, R. (1994). Evolving Sensors in Environments of Controlled Complexity. *In: Proceedings of the Fourth International Conference on Artificial Life.*

Menczer, F., & Belew, R. (1995). Latent Energy Environments. *In:* Belew, R., & Mitchell, M. (eds), *Adapting Individuals in Evolving Populations.* Reading, MA: Addison Wesley.

Miglino, O., Nafasi, K., & Taylor, C. (1994). Selection for Wandering Behavior in a Small Robot. *Artificial Life*, **2**(1), 101–116.

Miglino, O., Lund, H., & Nolfi, S. (1995). Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, **2**(4), 417–434.

Miller, G.F., Todd, P.M., & Hegde, S.U. (1989). Designing Neural Networks Using Genetic Algorithms. *Pages 379–384 of: Proceedings of the Third International Conference on Genetic Algorithms.*

Minai, A., & Levy, W. (1993). The Dynamics of Sparse Random Networks. *Biological Cybernetics*, **70**, 177–187.

Minsky, M. (1967). *Computation: Finite and Infinite Machines.* Englewood Cliffs, NJ: Prentice Hall.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

Mitchell, T. (1997). *Machine Learning.* New York, NY: McGraw Hill.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience With a Learning Personal Assistant. *Communications of the ACM*, **37**(7).

Mittelstaedt, H., & Mittelstaedt, M. (1982). Homing by Path Integration in a Mammal. *Pages 290–297 of:* Papi, F., & Wallraff, H. (eds), *Avian Navigation*, vol. 67. Berlin: Springer-Verlag.

Mittelstaedt, M., & Mittelstaedt, H. (1980). Homing by Path Integration in a Mammal. *Naturwissenschaften*, **67**, 566–567.

Mizumori, S., & Williams, J. (1993). Directionally Selective Mnemonic Properties of Neurons in the Lateral Dorsal Nucleus of the Thalamus of Rats. *Journal of Neuroscience*, **13**, 4015–4028.

Mondada, F., Franzi, E., & Ienne, P. (1993). Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms. *In: Proceedings of the Third International Symposium on Experimental Robotics.*

Moravec, H., & Elfes, A. (1985). High Resolution Maps from Wide Angle Sonar. *Pages 116–121 of: Proceedings of the 1985 IEEE International Conference on Robotics and Automation.*

Morris, R. (1981). Spatial Localization Does Not Require the Presence of Local Cues. *Learning and Motivation*, **12**, 239–261.

Morris, R., Garrud, P., Rawlins, J., & O'Keefe, J. (1982). Place Navigation Impaired in Rats with Hippocampal Lesions. *Nature*, **297**, 681–683.

Moutarlier, P., & Chatila, R. (1989). Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modeling. *In: The Fifth International Symposium on Robotics Research.*

Muller, R., & Kubie, J. (1987). The Effects of Changes in the Environment on the Spatial Firing of Hippocampal Complex-Spike Cells. *Journal of Neuroscience*, **7**, 1951–1968.

Muller, R., Kubie, J., & Ranck, J. (1987). Spatial Firing Patterns of Hippocampus Complex-Spike Cells in a Fixed Environment. *Journal of Neuroscience*, **7**, 1935–1950.

Muller, R., Kubie, J., & Saypoff, R. (1991). The Hippocampus as a Cognitive Graph. *Hippocampus*, **1**(3), 243–246.

Nadel, L. (1996). The Hippocampus: Selective Functions in Memory. *Pages 305–317 of:* Ono, T., McNaughton, B., Molotchnikoff, S., Rolls, E., & Nishijo, H. (eds), *Perception, Memory, and Emotion: Frontiers in Neuroscience.* New York, NY: Pergamon Press.

Nehmzow, U. (1995). Animal and Robot Navigation. *Robotics and Autonomous Systems*, **15**, 71–81.

Nolfi, S., & Parisi, D. (1995). *Evolving Non-Trivial Behaviors on Real Robots: An Autonomous Agent that Picks up Objects.* Tech. rept. TR 95-03. Institute of Psychology, C.N.R, Rome, Italy.

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994a). How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics. *In: Proceedings of the Fourth International Conference on Artificial Life.*

Nolfi, S., Elman, J., & Parisi, D. (1994b). Learning and Evolution in Neural Networks. *Adaptive Behavior*, **3**(1), 5–28.

Nwana, H. (1996). Software Agents: An Overview. *Knowledge Engineering Review*, **11**(3).

O'Keefe, J. (1976). Place Units in the Hippocampus of the Freely Moving Rat. *Experimental Neurology*, **51**, 78–109.

O'Keefe, J. (1989). Computations the Hippocampus Might Perform. *Pages 225–284 of:* Nadel, L., Cooper, L., Culicover, P., & Harnish, R. (eds), *Neural Connections, Mental Computation.* Cambridge, MA: MIT Press.

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, UK: Clarendon.

O'Keefe, J., & Speakman, A. (1987). Single Unit Activity in the Rat Hippocampus During a Spatial Memory Task. *Experimental Brain Research*, **68**, 1–27.

Omidvar, O., & van der Smagt, P. (eds). (1997). *Neural Systems for Robotics.* San Diego, CA: Academic Press.

Pagac, D., Nebot, E., & Durrant-Whyte, H. (1995). An Evidential Approach to Probabilistic Map-Building. *In: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics.*

Parekh, R. (1998). *Constructive Learning: Inducing Grammars and Neural Networks.* Ph.D. thesis, Department of Computer Science, Iowa State University, Ames, IA.

Prepscius, C., & Levy, W. (1994). Sequence Prediction and Cognitive Mapping by a Biologically Plausible Neural Network. *Pages 164–169 of: Proceedings of the World Congress on Neural Networks.*

Prescott, T. (1995). Spatial Representations for Navigation in Animats. *Adaptive Behavior*, **4**(2), 85–123.

Prescott, T., & Mayhew, J. (1992). Building Long-Range Cognitive Maps Using Local Landmarks. *Pages 233–242 of: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.*

Quirk, G., Muller, R., & Kubie, J. (1990). The Firing of Hippocampal Place Cells in the Dark Depends on the Rat's Recent Experience. *Journal of Neuroscience*, **10**, 2008–2017.

Quirk, G., Muller, R., Kubie, J., & Ranck, J. (1992). The Positional Firing Properties of Medial Entorhinal Neurons: Description and Comparison with Hippocampal Place Cells. *Journal of Neuroscience*, **12**(5), 1945–1963.

Ranck, J. (1984). Head Direction Cells in the Deep Cell Layer of Dorsal Presubiculum in Freely Moving Rats. *Society for Neuroscience Abstracts*, **10**, 599.

Recce, M., & Harris, K. (1996). Memory for Places: A Navigational Model in Support of Marr's Theory of Hippocampal Function. *Hippocampus*, **6**, 735–748.

Redish, A., & Touretzky, D. (1996). Navigating with Landmarks: Computing Goal Locations from Place Codes. *In:* Ikeuchi, K., & Veloso, M. (eds), *Symbolic Visual Learning.* New York, NY: Oxford University Press.

Redish, A., & Touretzky, D. (1998). The Role of the Hippocampus in Solving the Morris Water Maze. *Neural Computation*, **10**, 73–111.

Reynolds, C. (1994a). Evolution of Corridor Following Behavior in a Noisy World. *In: From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.*

Reynolds, C. (1994b). Evolution of Obstacle Avoidance Behavior: Using Noise to Promote Robust Solutions. *In:* Kinnear, K. (ed), *Advances in Genetic Programming.* Cambridge, MA: MIT Press.

Rich, E., & Knight, K. (1991). *Aritificial Intelligence.* New York, NY: McGraw Hill.

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* New York, NY: Cambridge University Press.

Rolls, E. (1989a). Functions of Neuronal Networks in the Hippocampus and Neocortex in Memory. *Pages 240–265 of:* Byrne, J., & Berry, W. (eds), *Neural Models of Plasticity: Theoretical and Empirical Approaches.* New York, NY: Academic Press.

Rolls, E. (1989b). The Representation and Storage of Information in Neuronal Networks in the Primate Cerebral Cortex and Hippocampus. *Pages 125–159 of:* Durbin, R., Miall, C., & Mitchison, G. (eds), *The Computing Neuron.* Wokingham, UK: Addison Wesley.

Rolls, E. (1990). Functions of the Primate Hippocampus in Spatial Processing and Memory. *Pages 339–362 of:* Kesner, R., & Olton, D. (eds), *Neurobiology of Comparative Cognition.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Rolls, E. (1996). The Representation of Space in the Primate Hippocampus and Episodic Memory. *Pages 375–400 of:* Ono, T., McNaughton, B., Molotchnikoff, S., Rolls, E., & Nishijo, H. (eds), *Perception, Memory, and Emotion: Frontiers in Neuroscience.* New York, NY: Pergamon Press.

Rudy, J., & Sutherland, R. (1989). The Hippocampal Formation is Necessary for Rats to Learn and Remember Configural Discriminations. *Behavioral Brain Research*, **34**, 97–109.

Rumelhart, D. E., & McClelland, J. L. (eds). (1986). *Parallel Distributed Processing, Vol I-II.* Cambridge, MA: MIT Press.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence - A Modern Approach.* Englewood Cliffs, NJ: Prentice Hall.

Salomon, R. (1991). Improved Convergence Rate of Back-Propagation with Dynamic Adaptation of the Learning Rate. *Pages 269–273 of: Proceedings of the First International Conference on Parallel Problem Solving from Nature.*

Sarton, G. (1936). *The Study of the History of Science.* Cambridge, MA: Harvard University Press.

Schmajuk, N., & Thieme, A. (1992). Purposive Behavior and Cognitive Mapping: A Neural Network Model. *Biological Cybernetics*, **67**, 165–174.

Scholkopf, B., & Mallot, H. (1995). View-Based Cognitive Mapping and Path Planning. *Adaptive Behavior*, **3**(3), 311–348.

Schone, H. (1984). *Spatial Orientation: The Spatial Control of Behavior in Animals and Man.* Princeton, NJ: Princeton University Press.

Schuierer, S. (1997). Efficient Robot Self-Localization in Simple Polygons. *Pages 20–22 of: Proceedings of the 13th European Workshop on Computational Geometry.*

Schwefel, H-P. (ed). (1981). *Numerical Optimization of Computer Models.* Chichester, UK: John Wiley.

Schwefel, H-P. (1987). Collective Phenomena in Evolutionary Systems. *Pages 1025–1033 of: Proceedings of 31st Annual Meeting of the International Society for General System Research.*

Seibert, M., & Waxman, A. (1992). Adaptive 3D Object Recognition from Multiple Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**, 107–124.

Shannon, C. (1952). Presentation of a Maze-Solving Machine. *In: Transactions of the 8th Cybernetics Conference.*

Sharp, P. (1991). Computer Simulations of Hippocampal Place Cells. *Psychobiology*, **19**, 103–115.

Sharp, P. (1996). Multiple Spatial/Behavioral Correlates for Cells in the Rat Postsubiculum: Multiple Regression Analysis and Comparison to Other Hippocampal Areas. *Cerebral Cortex*, **6**, 238–259.

Sharp, P., & Green, C. (1994). Spatial Correlates of Firing Patterns of Single Cells in the Subiculum of the Freely Moving Rat. *Journal of Neuroscience*, **14**, 2339–2356.

Sharp, P., Kubie, J., & Muller, R. (1990). Firing Properties of Hippocampal Neurons in a Visually Symmetrical Environment: Contributions of Multiple Sensory Cues and Mnemonic Properties. *Journal of Neuroscience*, **10**, 2339–2356.

Sharp, P., Blair, H., & Brown, M. (1996). Neural Network Modeling of the Hippocampal Formation Spatial Signals and Their Possible Role in Navigation: A Modular Approach. *Hippocampus*, **6**, 720–734.

Simon, H. A. (1983). Why should machines learn? *In:* Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (eds), *Machine Learning – An Artificial Intelligence Approach.* Palo Alto, CA: Tioga.

Skaggs, W., & McNaughton, B. (1996). Replay of Neuronal Firing Sequence in Rat Hippocampus During Sleep Following Spatial Experience. *Science*, **271**, 1870–1873.

Smith, R., Self, M., & Cheeseman, P. (1990). Estimating Uncertain Spatial Relationships in Robotics. *In:* Cox, I., & Wilfong, G. (eds), *Autonomous Robot Vehicles.* New York, NY: Springer-Verlag.

Smythies, J. (1966). *Brain Mechanisms and Behavior.* New York, NY: Academic Press.

Squire, L. (1986). Mechanisms of Memory. *Science*, **232**, 1612–1619.

Squire, L., Shimamura, A., & Amaral, D. (1989). Memory and the Hippocampus. *Pages 208–239 of:* Byrne, J., & Berry, W. (eds), *Neural Models of Plasticity: Theoretical and Empirical Approaches.* New York, NY: Academic Press.

Steels, L., & Brooks, R. (eds). (1995). *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Sutherland, R., Whishaw, I., & Kolb, B. (1982). A Behavioural Analysis of Spatial Localization Following Electrolytic, Kainate- or Colchicine-Induced Damage to the Hippocampal Formation in the Rat. *Behavioural Brain Research*, **7**, 133–153.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. *In: Proceedings of the Third International Conference on Genetic Algorithms.*

Tanimoto, S. L. (1995). *Elements of Artificial Intelligence Using Common Lisp.* New York, NY: Computer Science Press.

Taube, J. (1995). Head-Direction Cells Recorded in the Anterior Thalamic Nuclei of Freely Moving Rats. *Journal of Neuroscience*, **15**, 70–86.

Taube, J., Muller, R., & Ranck, J. (1990a). Head Direction Cells Recorded from the Postsubiculum in Freely Moving Rats: I. Description and Quantitative Analysis. *Journal of Neuroscience*, **10**, 420–435.

Taube, J., Muller, R., & Ranck, J. (1990b). Head Direction Cells Recorded from the Postsubiculum in Freely Moving Rats: II. Effects of Environmental Manipulations. *Journal of Neuroscience*, **10**, 436–447.

Teller, A. (1994). The Evolution of Mental Models. *Pages 199–219 of:* Kinnear, Kim (ed), *Advances in Genetic Programming.* Cambridge, MA: MIT Press.

Thorndike, E. (1898). Animal Intelligence: An Experimental Study of the Associative Processes in Animals. *Psychological Monographs*, **2,4**.

Thrun, S. (1996). *A Bayesian Approach to Landmark Discovery and Active Perception in Mobile Robot Navigation.* Tech. rept. CMU-CS-96-122. Carnegie Mellon University, Pittsburgh, PA.

Tolman, E. (1932). *Purposive Behavior in Animals and Men.* New York, NY: Irvington Publishers.

Tolman, E. (1948). Cognitive Maps in Rats and Men. *Psychological Review*, **55**, 189–208.

Tolman, E., & Honzik, C. (1930). "Insight" in Rats. *University of California Publications in Psychology*, **4**(14), 215–232.

Tolman, E., Ritchie, B., & Kalish, D. (1946). Studies in Spatial Learning. I. Orientation and the Short-Cut. *Journal of Experimental Psychology*, **36**, 13–24.

Traub, R., & Miles, R. (1991). *Neuronal Networks of the Hippocampus.* New York, NY: Cambridge University Press.

Treves, A., & Rolls, E. (1991). What Determines the Capacity of Autoassociative Memories in the Brain? *Network*, **2**, 371–397.

Treves, A., & Rolls, E. (1992). Computational Constraints Suggest the Need for Two Distinct Input Systems to the Hippocampal CA3 Network. *Hippocampus*, **2**, 189–199.

Treves, A., & Rolls, E. (1994). A Computational Analysis of the Role of the Hippocampus in Memory. *Hippocampus*, **4**, 374–391.

Trullier, O., Wiener, S., Berthoz, A., & Meyer, J-A. (1997). Biologically-based Artificial Navigation Systems: Review and Prospects. *Progress in Neurobiology*, **51**, 483–544.

Tsuji, S., & Li, S. (1993). Memorizing and Representing Route Scenes. *In: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.*

Uhr, L., & Honavar, V. (1994). Introduction. *In:* Honavar, V., & Uhr, L. (eds), *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* San Diego, CA: Academic Press.

Valavanis, K., & Saridis, G. (1992). *Intelligent Robotic Systems: Theory, Design, and Applications.* New York: Kluwer Academic.

Vinogradova, O. (1975). Functional Organization of the Limbic System in the Process of Registration of Information: Facts and Hypothesis. *Pages 3–69 of:* Isaacson, R., & Pribram, K. (eds), *The Hippocampus.* New York, NY: Plenum Press.

von Neumann, J. (1956). Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. *Pages 43–98 of:* Shannon, C., & McCarthy, J. (eds), *Automata Studies.* Princeton, NJ: Princeton University Press.

von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior.* Princeton, NJ: Princeton University Press.

Walker, J. (1995). *Evolution of Simple Virtual Robots Using Genetic Algorithms.* M.Phil. thesis, Department of Mechanical Engineering, Iowa State University, Ames, IA.

Walter, W. (1950). An Imitation of Life. *Scientific American*, **182**(5), 42–45.

Walter, W. (1951). A Machine That Learns. *Scientific American*, **185**(2), 60–63.

Wan, H., Touretzky, D., & Redish, A. (1994). Towards a Computational Theory of Rat Navigation. *Pages 11–19 of: Proceedings of the 1993 Connectionist Models Summer School.*

Wilson, M., & McNaughton, B. (1993). Dynamics of the Hippocampal Ensemble Code for Space. *Science*, **261**, 1055–1058.

Winston, P. (1992). *Artificial Intelligence.* New York, NY: Addison Wesley.

Wooldridge, M., & Jennings, N. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, **10**(2), 115–152.

Worden, R. (1992). Navigation by Fragment Fitting: A Theory of Hippocampal Function. *Hippocampus*, **2**(2), 165–188.

Yamauchi, B., & Beer, R. (1994). Integrating Reactive, Sequential, and Learning Behavior Using Dynamical Neural Networks. *In: From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.*

Yamauchi, B., & Beer, R. (1996). Spatial Learning for Navigation in Dynamic Environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part B, Special Issue on Robot Learning*, **26**.

Yamauchi, B., & Langley, P. (1997). Place recognition in Dynamic Environments. *Journal of Robotic Systems, Special Issue on Mobile Robots*, **14**(2), 107–120.

Yang, J., & Honavar, V. (1998). Feature Subset Selection Using a Genetic Algorithm. *In:* Motoda, & Liu (eds), *Feature Extraction, Construction and Selection - A Data Mining Perspective.* Boston, MA: Kluwer Academic.

Yang, J., Parekh, R., & Honavar, V. (1998). DistAl: An Inter-Pattern Distance-Based Constructive Learning Algorithm. *Intelligent Data Analysis.* (To appear).

Zalzala, A., & Morris, A. (eds). (1996). *Neural Networks for Robotic Control: Theory and Applications.* New York, NY: Ellis Horwood.

Zipser, D. (1986). Biologically Plausible Models of Place Recognition and Place Location. *Pages 432–470 of:* Rumelhart, D., McClelland, J., & the PDP Research Group (eds), *Parallel Distributed Processing: Explorations in the Micro-Structure of Cognition, Volume 2.* Cambridge, MA: MIT Press.