ABSTRACT

Title of Document: THE INFLUENCE OF COLLECTIVE WORKING MEMORY STRATEGIES ON AGENT TEAMS

Ransom Kershaw Winder, Doctor of Philosophy, 2007

Directed By: Professor James A. Reggia
Department of Computer Science

Past self-organizing models of collectively moving "particles" (simulated bird flocks, fish schools, etc.) typically have been based on purely reflexive agents that have no significant memory of past movements or environmental obstacles. These agent collectives usually operate in abstract environments, but as these domains take on a greater realism, the collective requires behaviors use not only presently observed stimuli but also remembered information. It is hypothesized that the addition of a limited working memory of the environment, distributed among the collective's individuals can improve efficiency in performing tasks. This is first approached in a more traditional particle system in an abstract environment. Then it is explored for a single agent, and finally a team of agents, operating in a simulated 3-dimensional environment of greater realism. In the abstract environment, a limited distributed working memory produced a significant improvement in travel between locations, in some cases improving performance over time, while in others surprisingly achieving an immediate benefit from the influence of memory. When strategies for accumulating and manipulating memory were subsequently explored for a more realistic single agent in the 3-dimensional environment, if the agent kept a local or a cumulative working memory, its performance improved on different tasks, both when navigating nearby obstacles and, in the case of cumulative memory, when covering previously traversed terrain. When investigating a team of these agents engaged in a pursuit scenario, it was determined that a communicating and coordinating team still benefited from a working memory of the environment distributed among the agents, even with limited memory capacity. This demonstrates that a limited distributed working memory in a multi-agent system improves performance on tasks in domains of increasing complexity. This is true even though individual agents know only a fraction of the collective's entire memory, using this partial memory and interactions with others in the team to perform tasks. These results may prove useful in improving existing methodologies for control of collective movements for robotic teams, computer graphics, particle swarm optimization, and computer games, and in interpreting future experimental research on group movements in biological populations.

THE INFLUENCE OF COLLECTIVE WORKING MEMORY STRATEGIES ON AGENT TEAMS

By

Ransom Kershaw Winder

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Acknowledgements

I would like to express my gratitude to my advisor, Dr. James Reggia, both for his guidance over many years through changing projects and for his extraordinary commitment to helping me find funding for my education and research. I would not have come so far without him, and I am very fortunate to have had the opportunity to work under his advisement.

I also thank each of the members of my defense committee for their valuable observations and insights on my dissertation work. I appreciate the time and effort they have spent to gain an understanding of what I did and its significance.

For years of support and love, I profusely thank my family for their patience waiting for me to finish this degree. Also for her years of care and tutelage, I give my deepest thanks to Dr. Michelle Hugue, who has been there for me as a mentor and friend through both my undergraduate and graduate careers.

I am also particularly grateful to my father and mother for their invaluable help in proofreading this massive document. Their clever eyes caught many poor turns of phrase and helped me make them far more eloquent and comprehensible.

Finally, I give my special thanks to Timur Chabuk for his assistance in developing the agent's 3-dimensional urban playground in this work. His expertise was invaluable in allowing me to design the agents, environments, and experiments that make up much of these results.

Table of Contents

List of Tables

List of Figures

Chapter 1:

Introduction

In recent years, there has been increasing interest in and research involving computational models of the self-organizing collective movements made by groups of locally interacting "particles." Examples of these systems include models of bird flocks [Reynolds, 1987; Reynolds, 1999], fish schools [Huth, 1992; Tu, 1994; Reynolds, 2006], social insect swarms [Bonabeau, 1999], and self-assembling molecules [Edwards, 1998]. All of these systems, referred to here as particle systems, can be viewed as consisting of very simple but numerous autonomous entities, usually taking biological inspiration from the observed behavior of birds, fish, ants, molecules, etc. The collective movements of particles/agents in space are governed primarily by local forces exerted on them by other nearby particles/agents or objects in the environment. Methodologically-related approaches, such as particle swarms [Kennedy, 2001; Clerc, 2002] and bacterial chemotaxis algorithms [Muller, 2002], have generalized this idea to abstract, n-dimensional cognitive spaces. These systems have proven to be very effective in past and recent applications, not only for modeling theories about group movements in biological populations, but also as methodologies in computer graphics [Ilmonen 2006; Reynolds, 2006], numerical optimization [Kennedy, 2001; Pan, 2006], and robotic control [Hodgins, 1994; McCook, 2007].

Even though each individual particle in these systems is effectively mindless and its movements are completely determined in a reflexive manner, the population or collective of the interacting particles has displayed remarkably interesting behaviors. These populations are said to exhibit self-organization because their global behavior emerges from the numerous concurrent local interactions between particles. The collection of particles as a whole has the tendency to act like a "superorganism" that moves through space, progressing like a single entity of loosely connected parts.

Past self-organizing collective movement systems have typically been based on very simple and purely reflexive particles having no significant individual intelligence. Few previous studies have directly developed and studied how these models can be extended so that the particle collective can autonomously pursue goals and solve problems [Rodriguez, 2004], although there is related robotics work [Jones, 2003]. Such a distributed team of semi-autonomous entities could ultimately have substantial practical value in controlling vehicle or robotic actions. For example, extrapolation of recent robotics advancements [Ayers, 2002] suggests a future in which teams of mobile semi-autonomous physical machines will play an increasing role in search and rescue operations [Nourbakhsh, 2005], and reconnaissance or exploration in remote or hostile environments [Elfes, 2006]. Further, the general principles that may be uncovered by studying systems with intelligent particles may prove useful in the specific application areas others have previously addressed, such as combinatorial optimization, and computer graphics or games. They may also be interesting and of practical use to biologists who are increasingly interested in understanding self-organization in nature [Camazine, 2001].

Much of the work in past collective movement systems has taken place in more abstract simulation environments that do not easily translate into more real-world environments. As the environments, and the agents' interaction with them, assume greater realism, the need for agents that act with a greater autonomy than reflexive behavior becomes more pronounced. In particular, agents require not only knowledge about their environment and others in their collective, but also the ability to process the more complex data they receive. Such agents also require the ability to remember information regarding their environment and collective (as opposed to solely reactive behavior based on present conditions). The satisfaction of both requirements will likely prove crucial for the future success of these models.

For instance, take the scenario of a collection of agents attempting to find a stationary location in an environment. Reactive agents might have no problem with this behavior. If a “force” exerts a pull on an agent toward the goal location when the agent discovers the location, then the changes in that agent’s movement act as an indirect communication to the other agents to move in that direction and approach the goal as well. However, when the environment includes obstacles that obstruct the view of the goal or the movement of other agents, or the goal is mobile and evasive, or the agent’s view radius is limited and its visual input difficult to interpret, then a greater agent complexity is required to allow the collective to navigate the environment more efficiently. Among the crucial elements required is the ability for the agents to maintain an individual working memory of experienced environmental information. The term working memory means a low capacity structure where information is temporarily stored and manipulated.

1.1 Goal and Specific Aims

The central goal of this research is to explore the effects and implications of adding a limited working memory to each particle/agent in a multi-agent system where agents are mostly reactive and operating to collectively perform some task. More specifically, the issue is how the performance of an agent team is affected when each agent has only part of the relevant information (typically a small part), and more complete information is only available collectively, being distributed across the memories of the population of agents. In this context, the specific objectives of this research were:

1. *Extend basic particle collective methods so that individual particles are capable of retaining a limited working memory of the environmental features and obstacles that they encounter.* This was intended to allow the exploration of the benefits and effects of enhancing these simple agents with individual and independent information, and to examine how their collective memory influenced task performance.
2. *Develop a more powerful agent operating with only limited visual input in a more realistic simulated 3-dimensional urban environment that, while still influenced by goal locations and observed environmental features, is capable of performing visual processing, understanding simple commands, and accumulating memories of the experienced environment.* This was intended to explore the benefits and effects of a working memory, both local and cumulative, when granted to an agent that is trained to interpret visual information in the scene and incoming spoken input, allowing it to learn information about its environment either temporarily or permanently.
3. *Build a multi-agent system consisting of agents described in the previous specific aim that retain features of particle systems plus have the ability to*

communicate and coordinate movement while retaining independent, limited memories. This was intended to examine the effects of memory and the different memory strategies in the more realistic environment of a team of agents operating in the pursuit domain with a common goal.

1.2 Overview

The rest of this dissertation is organized as follows. Chapter 2 discusses previous work that is significantly relevant to the topics examined in this dissertation. It is intended as background material.

Chapter 3 describes a model of a simple self-organizing multi-agent system where agent capabilities are extended to include a very limited partial memory of their environment. This chapter examines through experimental scenarios the benefits of this limited distributed memory, and provides a discussion of the results. This work is in a particle system where agents are required to make repeated tours through several simulated "landscapes" featuring different terrain types that slow their progress. A limited distributed memory is added to each particle, giving it the ability to only retain a small number of locations in the environment that are occupied by features capable of slowing its progress. This list is continuously updated, with some small probability, allowing the agent to retain memories from a small number of locations all over the environment. This extension allows the movement of the individual agents to be determined by both their natural reactive behaviors and by partial individual memories of previously encountered environmental features. In this way, the agents exhibit an internal state that differentiates one agent from another, and the agents are not merely reacting to present external stimuli. Additionally, this information is transmitted through the collective indirectly as other agents adjust their movements based on the actions of an agent operating with memory relevant to a specific situation, thereby allowing the opportunity for an increasing benefit over time for the touring agents as their collective learns the environment's relevant features distributed among its individuals.

While these initial results were interesting, the lack of realism of the environment raises the issue of whether they are relevant to more complex, practical situations. As a prelude to examining this issue, Chapter 4 presents an outline of a more complex agent operating in a more challenging environment and examines different strategies for memory manipulation in order to allow the agent to guide itself efficiently through the environment. Here the "forces" mediating collective behavior are supplemented or replaced by a discrete controller that is capable of maintaining the agent goals and knowledge of the environment, and the incoming information (a sequence of images seen by the agent as it moves) requires some interpretation by the agent. A hybrid hierarchical structure involving self-organizing maps, neural networks, and a high-level controller are used to achieve this behavior. Agents act on incoming and remembered information, the latter proving crucial to its success. Such memory further distances the agent from being reactive. Information can travel from its high-level control to its outputs, either in changing the agent's movement or in producing an output statement. Responsible for interpreting the visual stimuli, the agent can translate a sequence of images into an internal representation, or map, of its environment that will allow it to navigate the obstacles quickly. The agent can therefore use a local memory strategy (where it only remembers its nearby surroundings) or a cumulative memory strategy (where it can potentially remember the entire experienced environment). The basic behaviors of this agent are tested in the simulated urban environment for several different commands, issued by an outside source. These commands include instructions to go to a location or a region, find stationary objects in the environment when the agent does not know their exact location, patrol and map specific urban districts, and evade hostile objects by staying out of their line of sight. Several scenarios are examined in accord with the different memory strategies, and the results are presented and discussed.

Chapter 5 extends the results of Chapter 4 to a multi-agent collective in the urban environment and presents and examines the benefits of memory in conjunction with other features. There are some scenarios that can only be addressed by a multi-agent system. One classic example that makes use of all the behaviors the agent control mechanism can perform is the pursuit scenario, where a collection of agents are attempting to surround and thereby capture a moving target. Agents can be told to patrol an area and locate and pursue the target, while the target is an agent separate from the collective and is assigned the behavior of evading its hunters. In this scenario, control is modified to create a hybrid architecture that operates while using particle system forces in conjunction with a control that makes decisions based on command priority, environmental stimuli, and the memory strategies. It also allows for new methods of coordination because the multi-agent system allows for a greater complexity of individual agent behavior than the traditional particle system. The agent control mechanism is not a global phenomenon though. It is localized within each agent, meaning an agent is responsible for its own information processing, its own executive control based on the commands, data, and its memory, and its own actions in the environment, even if those actions are influenced by the behavior of other agents.

The pursuit scenario provides a challenge that can demonstrate the different benefits granted by the memory manipulation strategies in conjunction with agent coordination and communication. Furthermore, this scenario enables the exploration of the potential benefit of even a limited memory in improving the efficiency of this more complex system, as it would in the much simpler particle system operating in an abstract environment.

Finally, Chapter 6 is a general discussion of the dissertation’s findings and includes contributions, limitations, and topics for future work.

Chapter 2:

Background

The multidisciplinary nature of my research makes a variety of past work relevant. This chapter begins with an overview of multi-agent systems as they are studied in the field of artificial life. Following this is an examination of the related field of particle systems, with an emphasis on collective movement behavior. The third background section is an overview of past models of cognitive architectures and pertinent biologically-inspired cognitive methods in neural networks. Finally, I give a brief overview of working memory as relevant to the previously described fields.

2.1 Artificial Life Multi-agent Systems

The field of artificial life incorporates a broad range of studies that utilize computer simulation or robotics and biochemicals, to examine life processes, behaviors, and evolution. The goal here is to recreate phenomena in these domains that are observable in living systems [Bedau, 2003]. Among the wide range of topics that can be tied to the description of artificial life are self-replication [Smith, 2003; Griffith, 2005], self-assembly [Sahin, 2002; Klavins, 2002], and evolutionary computation [Banzhaf, 1998; Spector, 2003], while techniques used to pursue computational artificial life range from cellular automata [Sipper, 1995; Dorin, 1998] to neural networks [Nolfi, 1997]. Though related to artificial intelligence, the difference in these approaches is that artificial life tends to have a bottom-up approach to the emergence of its behavior from the base elements while artificial intelligence tends to work top-down, where the system overview is first defined and then refined in greater detail.

For computer science efforts into the study of artificial life, multi-agent systems are often used to recreate these complex behaviors. Multi-agent systems are composed of several interacting agents. While autonomous, these agents are also interdependent, allowing them to be capable of collective behaviors that a single agent either has difficulty achieving or cannot achieve. While multi-agent systems are used outside artificial life, within that field there has been past research using such systems for a wide variety of topics, including models of ecology [Bousquet, 1999], the behavior of social insect colonies [Drogoul, 1992], and collective robotics [Baldassarre, 2003]. There have also been efforts for many years at using these systems for the generation of maps [Singh, 1993; Howard, 2006a].

Of particular interest to this dissertation is the pursuit domain, which is sometimes also referred to as predator-prey systems. Predator-prey simulations can be defined in many different manners. For instance, the competition between populations of predators and prey (often foxes and rabbits, respectively) can be simulated with the Lotka-Volterra system of differential equations. Yet the domain is also quite suited to explore the benefits of a multi-agent system where predators (hunters or chasers) are attempting to capture prey (targets or chasees). In these models, the predators and prey can be explicitly modeled as individual agents, either cooperating or competing.

In the original work where the predator-prey pursuit scenario was introduced [Benda, 1986], the scenario was established with four predator agents chasing a single, randomly moving, prey agent on a grid world. The goal of the predators was to capture the prey by surrounding it on all four sides. While the predators could occupy the same cell in the environment, they were restricted to moving only one grid cell either vertically or horizontally at each time step. The target in this system was also restricted to orthogonal movement.

The pursuit domain has many variations, and no theoretically-established optimal solution. Some scenario dependent variations include using a hexagonal grid, allowing or disallowing predators to occupy the same cell, extending the movement range to include diagonal movements for the either one or both of the competing groups, and changing the prey's movement behavior. The domain's environment can also be extended to accommodate multiple predators and multiple prey agents [Alcazar, 2004].

The pursuit scenario has been used as a problem in many previous studies, where the purpose is usually to discover efficient cooperation strategies among the predators and successful predator configurations. Some solutions have been simple, such as one that involved adding attractive forces between the predators and their prey while adding repulsive forces between the predators themselves [Korf, 1992]. More complex strategies have been explored in other studies that employed finite state automata [Lenzitti, 2005], reinforcement learning [Zhao, 2005], genetic programming [Haynes, 1995; Luke, 1996], and co-evolution of predators and prey [Haynes, 1996].

The results of this latter study found that prey exhibiting a simple linear movement strategy (that is, moving continually in a straight line) did especially well, better than random—and even evolved—behavior. This was because the linear movement avoided locality of movement, to which random prey behavior succumbs and evolved predators can exploit more easily.

These past environments are limited in that they all take place in very abstract universes, usually on a grid or lattice rather than a continuous space. When this is the case, there is an easy metric for determining a capture of the prey by the predators. Many of these environments feature only the predators and prey and no other obstacles to hinder their movement. Most significantly, these environments either assume or explicitly state that their agents have an overhead view of the environment, giving them complete knowledge of the area within a specified radius surrounding the agents. In the 2005 Lenzitti paper, this is cleverly described as an overhead convex mirror which presents undistorted images to the agents on the ground below it. While this works in a controlled robotics environment [Hicks, 1999; Bonarini, 2000], it is more difficult to see this extended to a real-world scenario. The pursuit scenario presented in this dissertation faces a more complex domain, where the agents receive information about their 3-dimensional environment as a sequence of 2-dimensional, limited-view images as they move about. This has been explored before in the case of a lone predator interpreting an environment, such as a fish looking for prey of a particular color in a simulated underwater world [Terzopoulos, 1997]. In the following work, this type of vision is placed in an eye-level view of the environment, as opposed to an overhead view, and each agent is required to extract the environmental information from these images.

2.2 Swarm Intelligence and Collective Motion

The discipline of swarm intelligence can be viewed as an important part of the field of artificial life, taking inspiration from biology, in particular social animals. The key facet of swarm intelligence is the decentralized nature of the behavior, where self-organization arises from local interactions where the global picture is unavailable to the individual participants [Kennedy, 2001]. This is also often called emergent behavior. Unlike other decentralized techniques, such as neural networks and cellular automata, both of which can be included under the banner of artificial life, the "swarm," or the collective of the individual particles, is moving through space, either discrete or continuous and typically 2 or 3-dimensional, but sometimes with higher dimensionality. Applications for this discipline range from computer graphics [Reynolds, 1987; Reynolds, 2007] to optimization techniques such as particle swarm optimization [Carlisle, 2000; Pan, 2006] and ant colony optimization [Dorigo, 1997; Di Caro, 1998].

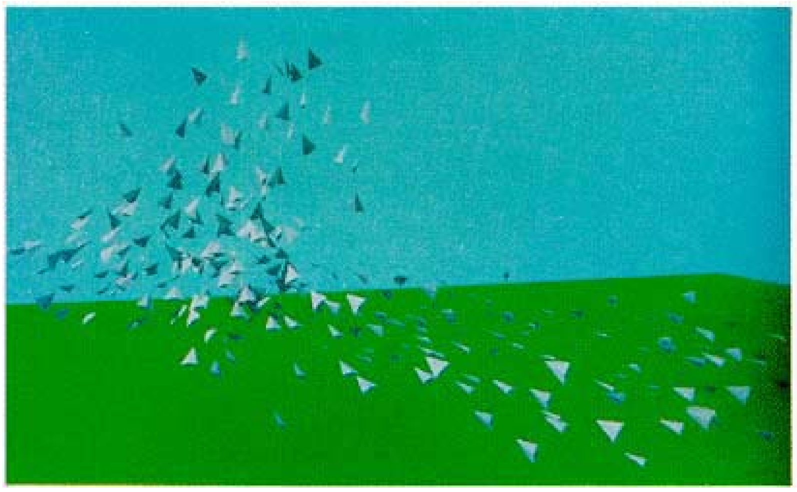


Figure 2.1. Sample environment of "boids" flocking in three-dimensional space. This image is from [Reynolds, 1987].

Of particular interest to this research is past work in collective movement. The key properties of these collective movement models are typically that the actions of a particle are dependent on the states or actions of other particles and that the behavior of the entire system is governed by these interactions [Mataric, 1995]. Reynolds demonstrated some of the earliest work in this area, developing "bird-oid" particles, or boids, as an elaboration on particle systems. These boids exhibited a collective motion determined by their own position relative to their neighbors' positions and their neighbors' angles, the goal being to simulate the behavior of flocking. Figure 2.1 is an example collective of these flocking particles in a 3-dimensional, computer-animated environment from Reynolds's original paper [Reynolds, 1987].

Behavior in this original model was governed by three principal local influences between the agents. These behaviors in order of decreasing precedence are:

1. *Avoidance*: the influence to avoid collisions with other nearby boids.
2. *Matching Velocity* (or *Alignment*): the influence to match the average velocity of other nearby boids.
3. *Centering* (or *Cohesion*): the influence to stay close to other nearby boids.

When combined, these forces produced flock-like behaviors that looked realistic. There was no centralized control governing the swarm behavior, and the organic movement emerged from these local interactions of the particles.

This work has since been extended in many areas. Robotics has found it useful, and behaviors other than flocking have been defined that can emerge from these sorts of local interactions between the robots. These include such behaviors as following, homing, dispersing, foraging, [Mataric, 1995] and shepherding [Bayazit, 2002]. An extension has also been made to the basic model where the system features "leaders," i.e. particles with information regarding the location of food sources or what route to take. It was found that only a small proportion of leaders was needed to influence group movement, even though there was no explicit communication of this information [Couzin, 2005]. Another scenario looked at simulated heterogeneous robot systems, where flocking agents included attackers, defenders, and those attacked or defended [McCook, 2007].

Other studies have explored the possibility of even greater control acting from a top-down direction on the collective motion systems. Often created as finite state automata, these controls allow different states of behavior for each particle depending on the situation of its current state, its position, the position and behavior of known neighboring particles, and the environment [Reynolds, 2000; Rodriguez, 2005]. One study has even shown that these collective motion teams, when provided with a goal-driven control mechanism, can be extended to act as problem solvers, while still retaining their locally-based interactions. In fact, the cooperative behavior was shown to provide the agents with an advantage in engaging the problem domains, demonstrating the possibility of these agents being extended to more than models of biological collective movement [Rodriguez, 2005; Lapizco-Encinas, 2005].

Yet most collective motion scenarios are limited in the following ways. While the particles' local interactions give rise to interesting and useful behaviors, they are reacting to presently observable stimuli or forces pulling them to locations in the environment. They are not typically designed to learn the features of their environment, even in a limited fashion, and instead move reactively to their immediate surroundings rather than adjusting their movement for obstacles not yet encountered. The addition of even a limited ability to learn the environment can provide the system with a richer behavior in a more complex environment with only simple modifications that preserve the essential properties of the collective movement. It is this latter concept that guides the research in this dissertation.

2.3 Biologically-inspired Cognitive Architectures and Neural Networks

In recent years, there has been growing interest in developing cognitive architectures that are inspired by the neurobiological basis of cognition. This has included work on large-scale modular neural networks, which consist of parts that carry out individual sub-tasks of the network's global task. This approach differs from many traditional artificial neural networks in one crucial way: as in the biological brain, the neurons are sparsely connected when compared to full connectivity and are arranged in a clustered, hierarchical set of modules. The network's modules can be either one of its substructures or its learning subprocedures, separately identifiable while adding to the global task. While the goal of these networks is not necessarily to simulate biological computation, modular neural networks may allow greater imitation of human thinking and handle complex problems traditional approaches cannot tackle [Auda, 1999; Caelli, 1999]. Traditional artificial neural networks that have several layers, such as multiple hidden layers, are typically not modular because they are generally fully connected. A large-scale modular neural network differs from other modular neural networks only in the increased number of modules involved and in the increased complexity, though not density, of its connectivity. In many cases, these networks are intended to be high-level models of brain regions.

There were initial forays into this field as early as 1987, including a modular neural network model of the vertebrate retina used to study the effects of lateral connectivity and a proposal that neural networks could be used as an architecture for multi-modular space systems [reviewed in Auda, 1999]. These two examples demonstrate the early applicability of modular neural networks both for reverse engineering of a biological system and for providing an engineering model meant to handle computational problems.

A wide range of previous models of biology have been tackled by using modular neural networks. For example, in one case, a modular neural network was used to explore why separate visual systems processed “what” and “where” information about visual stimuli [Rueckl, 1989]. A different modular model of dorsal and ventral paths of the visual cortex attempted to match both functional magnetic resonance imaging and electrophysiological data results [Corchs, 2002]. A third model looked at visual word recognition when the visual field is split between two hemisphere modules and was demonstrated to capture a greater range of reading behavior than previous models [Shillcock, 2000].

The modularity of these kinds of networks tries to mimic biological neural networks in ways that traditional artificial neural networks do not. Among the biologically interesting features that these models capture are functional specialization, cooperation, competition, and extendibility [Auda, 1999]. Regarding functional specialization, distinct modules are able to process different attributes of the input, much as shape and position are handled by different regions in the human visual system. Regarding cooperation and competition, communication between the different modules potentially allows for more complex behavior from the network. Regarding extendibility, because of the relatively sparse connectivity, new modules can be added without affecting the behavior of present modules.

While these were designed to be models of biological systems, large-scale modular neural networks have also been employed in solving problems in different domains. Systems have been designed to tackle the problems of character recognition [Iwata, 1991], facial detection and recognition [El-Bakry, 2000; Melin, 2005], and syllable and word recognition [Chen, 1998; Lee, 1998]. Many of these tasks involve interpretation of very complicated or noisy data sets. There is less of an emphasis in these models in staying true to the structure of biological neural networks, and the concern is to solve real-world problems. This is in contrast to the models of biology where the goal is to engineer complexity to make observations and predictions regarding biological systems.

Many methods have been developed for training neural networks [Rumelhart, 1985; Kaelbling, 1995; Pearlmutter, 1995; Baldi, 1995; Girosi, 1995; Haykin, 1999; Gerstner, 2002; Serpen, 2002; Tang, 2003; Bosman, 2004]. In supervised learning (learning with a teacher), error signals (difference between correct and actual output) are used to adjust network weights to be closer to the correct response. The ultimate goal is for the network to emulate the teacher and eventually converge to a point where its responses would always match the teacher's. Essentially, this means the error is a performance function of the parameters of the network, and the goal is to find the minimum over the error hypersurface. In reinforcement learning, the network is given no exact correct answer for any of its decisions, but instead a judgment (reinforcement signal) on how good its responses are provided by a critic rather than a teacher [Sutton, 1998]. In unsupervised learning, or self-organized learning, there is not even a critic. Instead, the network is optimized to a task-independent measure of its representation of the environment, essentially encoding its features. Examples of this kind of learning are Hebbian and competitive learning [Haykin, 1999; Andrecut, 2002].

An artificial neural network subtype of key interest to this work are self-organizing maps [Kohonen, 1990; Kangas, 1990; Kohonen, 2002]. These are neural networks that are trained with unsupervised methods and are commonly used for reducing high-dimensional data into low-dimensional spaces, often for visualization purposes, as the topological features of the data are frequently preserved. Typically, these are feedforward networks of a single layer where the network nodes are tuned to the input patterns, and either one node or a local group of nodes are activated by an input. Every input connects to every neuron, and each neuron is associated with a weight vector of the same dimensionality as the input. When trained on an input, the neuron with the closest weight vector is considered the winner. Its vector, along with that of its neighbors, moves closer to the input vector. An example of a two-dimensional, self-organizing map is in Figure 2.2.

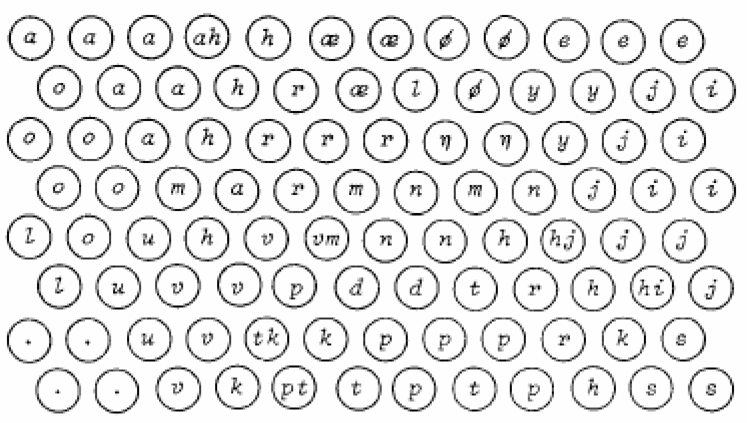


Figure 2.2. An example self-organizing map of phonemes common to Finnish speech based on features generated by an inner ear model performing frequency analysis. This image is from [Kohonen, 1990].

By removing the limitation that there is only one winner in the map and allowing for multiple islands of simultaneous activity, the resulting network is closer to what is observed in cortical systems [Schulz, 2004]. Some have also been extended to allow the self-organizing map to regulate its size [Alahakoon, 2000]. Additionally, while traditional computational self-organizing maps feature one-shot training, i.e. they are trained on a single input, they can be further extended to accommodate a sequence of data and recurrent connections. A model relevant to this work demonstrated this for multi-winner maps [Weems, 2004]. The goal here was to follow the Wernicke-Lichtheim-Geschwind model and develop a computational model that reproduced human performance in the tasks of picture naming and word repetition. The visual input came in the form of small images, while auditory input for the individual words came in the form of a sequence of phoneme features. Either of these stimuli produced the simple response of a sequence of motor features, representing either a statement of the word associated with the image or the repetition of the heard word.

This model used both self-organizing maps and neural networks trained with resilient error-backpropagation. The earlier areas of processing used multi-winner self-organizing maps to produce unique representations for each stimulus. For the audio stimuli, the self-organizing map was recurrent and dependent not only on the next phoneme in the sequence, but the previous state as well. Once these patterns were developed, the final pattern in the self-organizing map was used as the input to networks trained via resilient error-backpropagation with recurrent connections to produce the sequence of phonemes defined by their motor features. This was found to successfully name and repeat the different stimuli, and it was used to perform lesion studies to compare the model to various human aphasic syndromes.

While successful, this past work was limited in scope because it dealt with phoneme sequences comprising only single words. If the system was able to encode single words, it could be extended to create representations of full sentences. Additionally, while the repetition and naming made for a good measure of success, it did not take the system to its full potential. When used in a larger hybrid system, its capability of creating simplified representations of a wide variety of stimuli could be used to interact with a simulated world with incoming audio and visual information. Instead of just repeating the results, such a network could be trained to produce new results based both on the external stimuli being processed and an internal controller capable of state changes and remembering past stimuli.

2.4 Working Memory

Also relevant to this dissertation is the past study of memory. The topic of memory encompasses a wide range of systems and functions. A hierarchical depiction of these functions is presented in Figure 2.3. Memory is often separated by dichotomies, where short-term memory is distinguished from long-term memory. Long-term memory can then be broken down into categories called implicit and explicit, and explicit memory can be split into categories called semantic and episodic [Mishkin, 1997]. The split between short-term and long-term is the most significant. The former operates on the order of seconds, while the latter is integrated with existing knowledge and can remain available for use up to years later [Baddeley, 1997].

Working memory is a term that is often synonymous with short-term memory, the distinguishing feature usually being that working memory also implies the information is being manipulated by cognitive processes. It is a necessary system for complex tasks such as learning, reasoning, and comprehension, with evidence of fractionation into separate supporting systems [Baddeley, 1998]. There are theories supporting different components of working memory that handle storage of information and executive behavior [Baddeley, 1992]. Other theories contend that the executive control of working memory involves the maintenance of a task's goals and rules [Miller, 2001]. Working memory is also distinguished from other types of memory in that it has a much more limited capacity, which is particularly noticeable when rehearsal, semantic grouping, and access to supplemental long-term memory is reduced [Cowan, 2000]. Working memory is also believed to have a tendency to disappear quickly, on the order of seconds, whether due to decay or interference. In essence, it has low capacity, and patterns persist for a brief time.

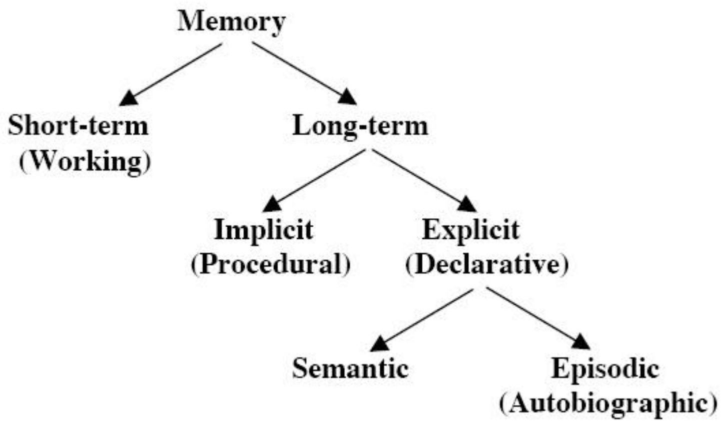


Figure 2.3. Hierarchical representation of the range of memory functions in human cognition. This is a typical representation of how modern neurospsychologists view the organization of human memory. From [Reggia, 2006].

Working memory has been a topic of great interest in the computational modeling community, and a wide variety of models have been produced examining its features. Often the interest is in how working memory generates its persistent activity. Hypotheses here range from using activity that circulates in feed-forward loops called “synfire chains”—where there is no direct feedback between neuronal groups—to activity being maintained by membrane currents allowing cellular bistability. Most models, however, approach the topic through cell assemblies featuring strong recurrent excitatory connections [Durstewitz, 2000].

In one case, a computational model tried to limit its complexity, while targeting the working memory system without the effects of rehearsal, chunking, and episodic long-term memory. Item representations in this model were connected with self-excitation, which drives the retention of information, and lateral inhibition of their neighboring representations, which was used to induce the working memory capacity limitations. The model was found to make predictions that behavioral experiments confirmed [Haarmann, 2001].

Another model focused on the executive control system of working memory. Aiming for biological realism, it modeled the prefrontal cortex which maintained the sensory information being processed and the basal ganglia which used gating mechanisms to modulate the patterns that should be kept active in the working memory. This behavior is learned by the model with an actor-critic method and was found to compare well to other neural network learning methods. Unlike the previous model, there was no need for inhibition here to limit capacity [O’Reilly, 2006].

In addition to these models of working memory, I have contributed work to this field outside the scope of this dissertation. I designed a training method that determined unique interregional weight strengths for a model of the brain's visual ventral pathway. The task here involved a working memory manipulation of stimuli presented, processed, and encoded in the earlier regions of the model. In the working memory, a circuit described in [Tagamets, 2000], patterns were maintained during a delay and then compared to successive stimuli to determine matches. The goal of the learning method was to preserve this behavior, while discovering the magnitude of the interregional connection strengths that would allow its simulated fMRI signals to match those observed in behavioral experiments on both healthy and schizophrenic patients [Winder, 2007].

I developed other models of working memory outside this dissertation that more fully explored the retention ability and capacity of working memory, but without the need for inhibitory connections or gating mechanisms to limit capacity. The first of these used an energy minimization network featuring Hebbian learning. This network also experienced decay on the weight matrix being constructed during learning, where older connections had a tendency to diminish as newer stimuli were added. Unlike typical Hebbian neural networks, the decay caused dependence on the order of presentation of the stimuli and when recall was tested, a strong recency effect was observed as occurs in behavioral performance results [Weems, 2007]. I have also developed an oscillatory neural network that experiences a similar Hebbian training featuring decay. In a typical oscillatory network, the network state is no longer able to achieve a minimum due to the constant changes to the thresholds based on the network performance [Horn, 1991].

Memory has also proven to be useful in artificial intelligence methods outside of just computational modeling of biological systems. For example, in one algorithm, the addition of limited memory led to the ability to learn policies to solve problems that a similar algorithm without this capacity could not [Chong, 2006].

In the following dissertation, working memory is of interest because it is the problem-specific information that agents acquire during problem solving. While working memory is not being employed here to fit a computational model of a biological system, the goal will be to keep memory as simple as possible, but still have the system retain a significant improvement in tasks by accumulating and accessing it, even when it has a limited capacity and it remains distributed among the individual members of a collective.

Chapter 3:

Improving Self-Organized Collective Movement with Distributed Partial Memories

3.1 Motivation

The goal of this chapter is to make a significant extension of current multi-agent models of self-organizing collective movements (particle systems, a type of swarm intelligence) to more "intelligent" particles by examining the effects of giving the entities used in such systems a limited memory of past events. Such particles with memories will be referred to in the following as *agents* to emphasize that, although they continue primarily to move passively/reactively in response to local external influences as in conventional particle systems, they may also alter their individual movements due to an internal state that differs from particle to particle.

The agents move in environments with various types of terrains, containing regions that impede their forward movement. The memory of the collective is truly distributed, with each individual recalling only a tiny part of any environmental obstacles. For example, a single obstacle occupying a substantial portion of space is remembered as a pattern of memories spread across the agent team and not by any individual agent, a situation that is reminiscent of social knowledge in a population [Hutchins, 1995; Wegner, 1995]. Since these agents (or particles) still move collectively, the memories of each will influence not only its own movement, but also the movements of others. The basic questions being asked are: When agents (or particles) are repeatedly moving through various terrains as they proceed to target locations, does being able to recall and avoid previously observed environmental impediments to movement lead to significant improvements in the times needed to achieve their goals? If so, how do different strategies of using a very limited memory influence the efficiency of their movements? It is hypothesized that teams of agents that keep a simple record, or limited-capacity "working memory," of the positions of obstacles in the terrain they pass and who use it to influence their future movement will be able to perform better (i.e., arrive at sequential target goals faster) than particles who do not.

3.2 Terrain and Tasks

Agents having self-organized collective movements exist in a bounded two-dimensional artificial world (see Figure 3.1). A sequence of goals (target destinations) scattered about the terrain is given to the agents and their task is to move collectively and sequentially from goal to goal in the shortest time possible. An implicit coordinate grid (not used by the agents) is conceptually superimposed on the otherwise continuous artificial world. While the agents move in a continuous, real-valued space, for convenience, mountain and swamp regions of the terrain that slow or hinder the agents are taken to occupy one or more contiguous unit cells of the coordinate grid. Generally, agents would only want to move across open terrain (i.e. regions that are not swamp or mountain), where they can reach their maximum velocity.

3.3 Agents

As in past particle systems, each agent j in the model is effectively represented by its real-valued coordinates $\mathbf{x}_j$ and its velocity vector $\mathbf{v}_j$ in the two-dimensional space. Figure 3.2 shows the basic organization of an agent and its relationship to the environment. The velocity vector for each agent is applied at every time step to update its coordinates. The velocity vector is in turn updated at every time step by various external influences. These influences, which are often competing, include acceleration towards the center location of neighboring agents, away from swamp or mountain cells within view, etc., as introduced in [Reynolds, 1987; Reynolds, 1999]. There is also an acceleration directed toward the current goal, similar to the homing implemented in [Heppner, 1990].

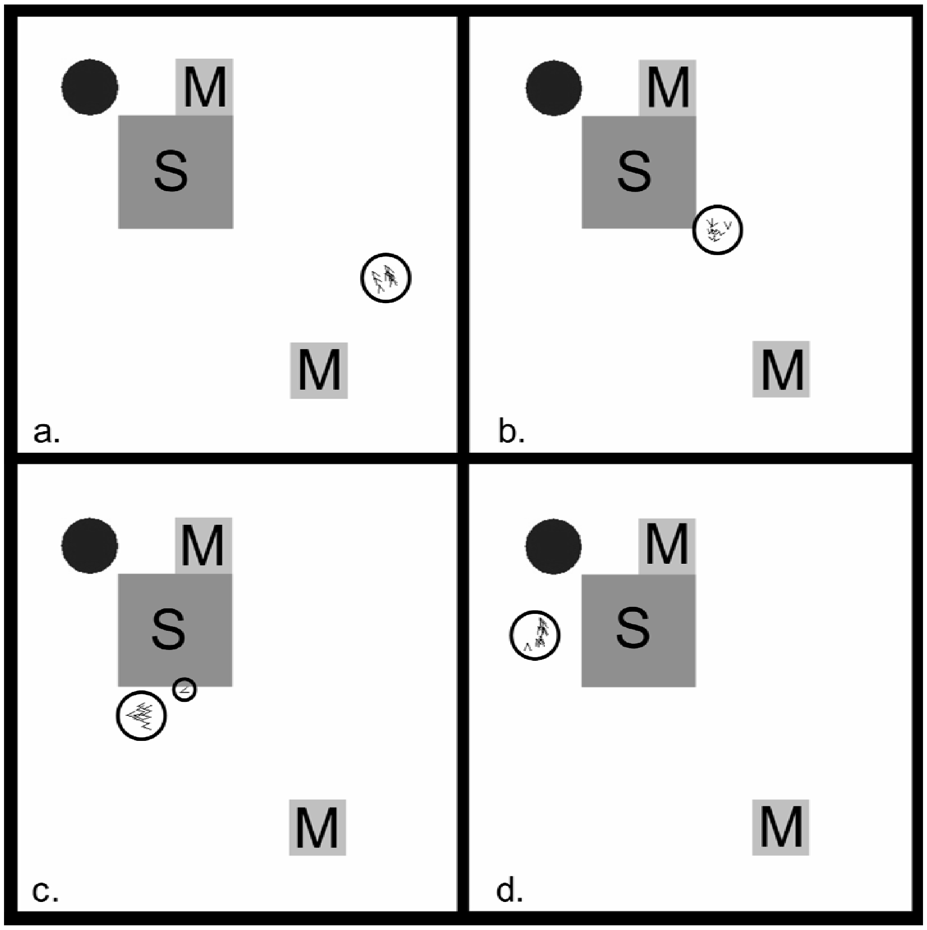


Figure 3.1. A simple 8x8 grid example environment. The ten collectively-moving agents shown here are the clusters of small arrows indicating agent location and direction. The arrow shows the direction of their average velocity. Their current destination, or goal, is the solid black circle. Shaded cells labeled “M” and “S” are mountain and swamp, respectively. Blank areas are open areas. **a.** Agents move across the open cells heading toward the goal. **b,c.** Having detected obstacles (a swamp and a mountain), the agents move through the area below the swamp. **d.** Having bypassed the swamp, the agents have a clear path to their goal.

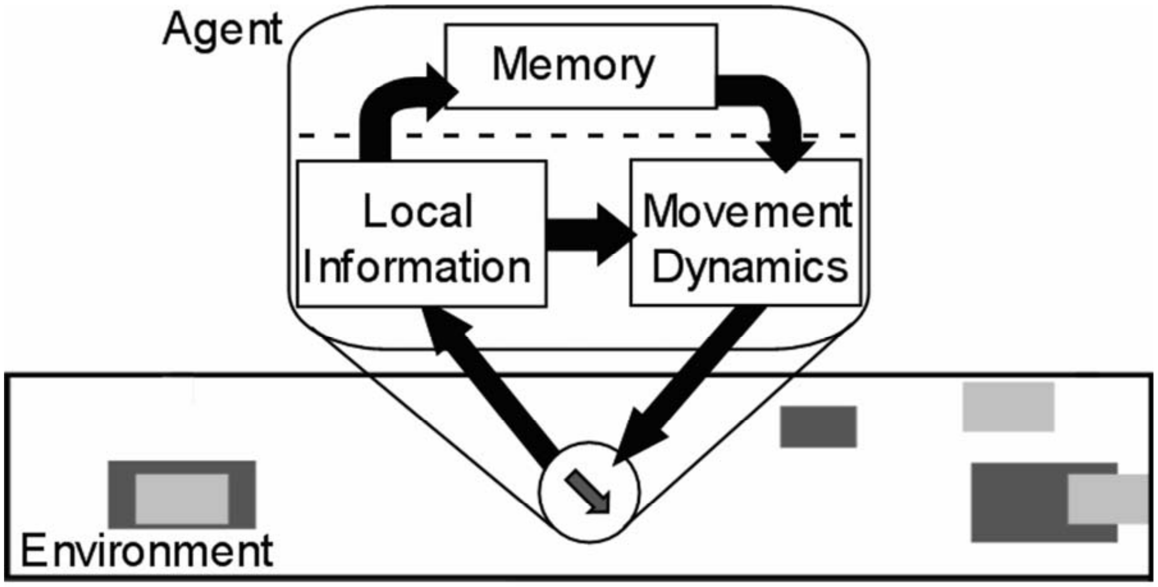


Figure 3.2. Representation of an agent and its interactions with the environment. The agent, illustrated by a small arrow in the environment that indicates its location and velocity, is aware of only its immediate surroundings (circle surrounding the arrow). As in past particle system models, local information within the radius of this circle determines the dynamics (i.e., acceleration) of the agent. What is new here is the inclusion of a memory of previously encountered local information (shown above the dashes in the upper "exploded view" of the agent), which is also used to influence movement dynamics.

Figure 3.3 presents the algorithm used to update an agent's velocity and position at every time step. Agents are all given the location and sequence of the goals in advance and, for the current goal, repeatedly compute adjustments to various velocity components (Part 1 of algorithm). The influence due to the current goal, $\mathbf{v_g}$, is stronger when the agent is close to the goal and weaker when further away. Vector $\mathbf{v_a}$, for collision avoidance, causes an agent to accelerate in the opposite direction of any neighbor that is considered too close. Vector $\mathbf{v_{mv}}$ causes an agent to accelerate to match the average velocity of its visible neighbors. Vector $\mathbf{v_c}$ causes an agent to accelerate toward the center of mass of its visible neighbors. These four velocity components are combined as a linear weighted sum to compute the new, resultant $\mathbf{v_j}$. Initial weights are chosen so that the agents tend to remain close to one another (weight values are given in Table 3.1). Additionally, k_t is 0 if cell is open, 0.05 if cell is swamp, 0.2 if cell is mountain, 0.203 if cell is map boundary, and -0.17 if cell is current goal. k_s is 0.5 if cell is swamp and about 0.03 if cell is mountain. The strongest weight is given to centering, and the weakest weight to collision avoidance. This is in contrast to some earlier models (e.g. [Reynolds, 1987]) where collision avoidance is given the strongest precedence. The effect is that the collectively moving particles rarely break apart, and all tend to arrive at each goal at about the same time.

```
For agent j with current position $\mathbf{x_j}$ and velocity $\mathbf{v_j}$
    $M = |\{\text{agents within radius } r_{TC} \text{ of agent } j\}|$          % other agents too close to j
    $N = |\{\text{agents within radius } r_V \text{ of agent } j\}|$            % agents in view of j ($r_V > r_{TC}$)
    $\mathbf{g}$ = coordinates of current goal cell
1)  Compute Influences
  •   $\mathbf{v_g} = (\mathbf{g} - \mathbf{x_j}) / |\mathbf{g} - \mathbf{x_j}|$                    % influence of goal destination
  •   $\mathbf{v_a} = k'' M^{-1} \sum_{i=1\ldots M} (\mathbf{x_j} - \mathbf{x_i})$             % collision avoidance
  •   $\mathbf{v_{mv}} = k' N^{-1} \sum_{i=1\ldots N} \mathbf{v_i}$                   % match velocity of neighbors
  •   $\mathbf{v_c} = k (N^{-1} \sum_{i=1\ldots N} \mathbf{x_i} - \mathbf{x_j})$              % center of mass of neighbors
  •   $\mathbf{v_j} = \mathbf{v_j} + w_g \mathbf{v_g} + w_c \mathbf{v_c} + w_{mv} \mathbf{v_{mv}} + w_a \mathbf{v_a}$   % aggregate velocity for agent j
2)  For all cells containing obstacles visible to agent j
  •   $\mathbf{u_{view}}$ = unit vector from agent j towards closest edge of cell   % direction of adjacent cell
  •   $\mathbf{v_j} = \mathbf{v_j} - k_t \mathbf{u_{view}}$                          % veloc. considering adjacent cell
    For all obstacles in agent j's memory and within radius $r_{mem}$     % $r_{mem} \geq r_V$
  •   $\mathbf{u_{mem}}$ = unit vector from agent j towards obstacle cell's center   % direction of obstacle from agent
  •   $\mathbf{v_j} = \mathbf{v_j} - d_{mem} k_t \mathbf{u_{mem}}$                     % veloc. considering obstacle
3)  Update Agent Memory (see text)
4)  Adjust Velocity
  •   if $|\mathbf{v_j}| > v_{max}$, then $\mathbf{v_j} = \mathbf{v_j} * v_{max} / |\mathbf{v_j}|$             % ensure velocity stays below $v_{max}$
  •   $newv_{max} = v_{max} * k_s * 2 / (1 + 1 / e^{|\mathbf{v_j}| - .5 * V_{max}})$        % slow if in a swamp/mountain cell
  •   $\mathbf{v_j} = \mathbf{v_j} * newv_{max} / |\mathbf{v_j}|$
5)  Update Position
  •   $\mathbf{x_j} = \mathbf{x_j} + \mathbf{v_j}$                                 % update position
```

Figure 3.3. This algorithm updates an agent's velocity vector and coordinates at each time step. Table 3.1 lists the values of the constant parameters. Vertical bars | … | indicate the size of enclosed entity.

Having computed a velocity update, the agent then considers local obstacles in neighboring cells within its vision radius (Part 2 of Figure 3.3). If any neighboring cell is a swamp, mountain, or boundary of the map, the agent accelerates away from that cell. It computes the unit vector $\mathbf{u}_{\mathbf{view}}$ indicating the direction of the nearest edge of the offending cell, and adjusts $\mathbf{v}_\mathbf{j}$ to move more in the opposite direction.

Table 3.1. Table of the default parameters for the model.

k	1	r_{TC}	0.27
k'	1	r_V	1
k''	1	v_{max}	0.25
c_{mem}	0.01	w_g	0.1
d_{mem}	0.3	w_c	0.76
n_{mem}	10	w_{mv}	0.4
r_{mem}	3	w_a	0.04

In addition, and unlike past particle systems, each agent has a limited-capacity memory storing previously encountered swamp and mountain cells that is updated as the agent moves about the world (the upper part of Figure 3.2). This memory typically contains a small number (e.g. 10) of remembered obstacle cells, depending on a predetermined maximum. Each individual remembered obstacle cell location represents only a single non-empty grid cell, defined by that cell's coordinates, the obstacle's type (mountain or swamp), and a timestamp equal to the time step when that obstacle cell was added to the agent's memory. A "memory" is thus typically only a small part of a swamp/mountain, with different agents remembering different cells, so the full representation of an obstacle is truly distributed across the agent collective. No single agent learns a substantial part of a complete obstacle. Each remembered obstacle cell's influence is similar to the influence from obstacles in view. It differs in that the radius from the agent in which remembered obstacles (swamp, mountain) affect movement is generally much larger than the vision radius, and the strength of the effect on the agent of the remembered obstacle need not be the same as the strength of the effect of obstacles in view. Newly encountered obstacles are stored in memory in Part 3 of the algorithm (details given below).

The agent then adjusts its velocity so that the magnitude of $\mathbf{v_j}$ does not exceed v_{max}, the maximum value. If the agent is in a swamp or a mountain cell, the maximum velocity is reduced and the agent's velocity is then rescaled to this new maximum. In Part 5 of the algorithm, the agent's position is updated based on the final $\mathbf{v_j}$ value.

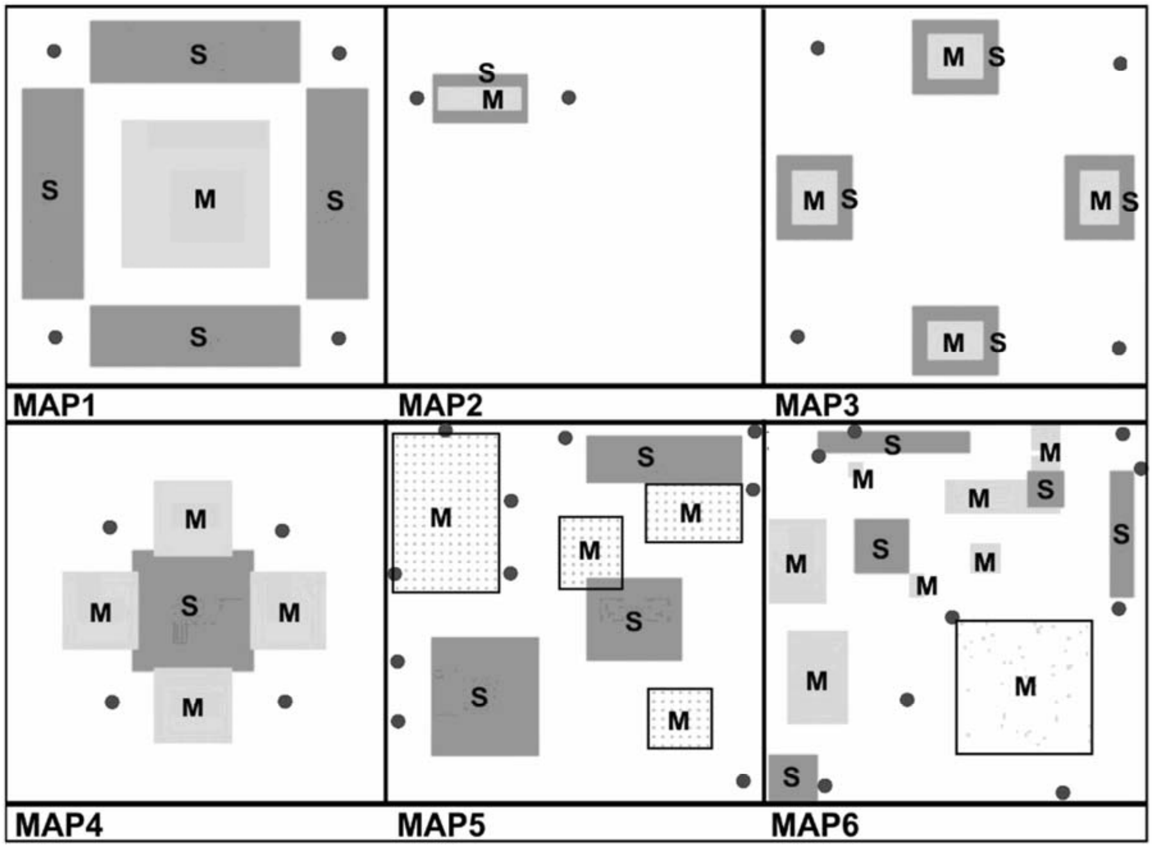


Figure 3.4. Maps of the 128x128 terrains used in the simulations. White areas are open regions, areas labeled "S" are swamps, areas labeled "M" are mountains, and circles are the locations of goals on the map. As a size reference region, M in Map 2 is 28 x 8 cells in size. In Maps 5 and 6, the areas outlined in black contain single cell mountains. These are regularly spaced in Map 5 and scattered in Map 6.

3.4 Experimental Methods

To test the advantage (or lack thereof) of agents with memory, agents with memory are put into situations where memory is potentially useful and compared to equivalent “control” agents without memory (particles). Specifically, the simulations take agents on multiple tours through a sequence of goal destinations, so that they can learn the terrain and benefit from their memory record.

Figure 3.4 shows maps of the six terrains used in the simulations, all of size 128x128 cells. In Map 1, agents are directed to cycle through four goal locations counterclockwise. Ideally, they would be expected to move in the open terrain outside or between the swamp and mountain areas. In Map 2, agents alternate between two goals near a single obstacle. This is the simplest test of agents’ ability to avoid rough terrain. Map 3 combines the counterclockwise movement in Map 1 with four obstacles similar to the one in Map 2. Map 4 tests agent ability to avoid mountain cells separating the goals. Ideally, in this map, the agents moving counterclockwise from goal to goal would be most efficient if they came to move in just the open terrain. In Maps 5 and 6, the challenge is no longer just to acquire simple tours about obstacles because the goals are scattered across the map randomly, with thin open paths between mountain cells allowing more complex movements.

In each experimental simulation, ten agents begin with random initial positions and velocity vectors within the same grid cell so they are all initially within one another’s vision radius. The number of agents was chosen to be ten so there would be a sufficient number of agents to exhibit collective behavior without excessive computational costs. Larger numbers of agents were tested unsystematically in a few cases and produced qualitatively similar results except that the processing time was greater. The time (number of required time steps) it takes the ten agents to complete each consecutive tour made during a simulation is recorded. Agents can become trapped in a region of the map, so any simulation has an upper time limit of 60,000 time steps. Though it would be possible for a simulation to terminate after 60,000 time steps, it is also possible that a simulation could never terminate. Therefore, because the tasks set for the agents can be completed in far fewer time steps than 60,000, this number was made a simulation's upper limit. A goal destination is considered to be reached when 90% of the agents reach it. At this time, the next target goal becomes the new "current goal" for the agents.

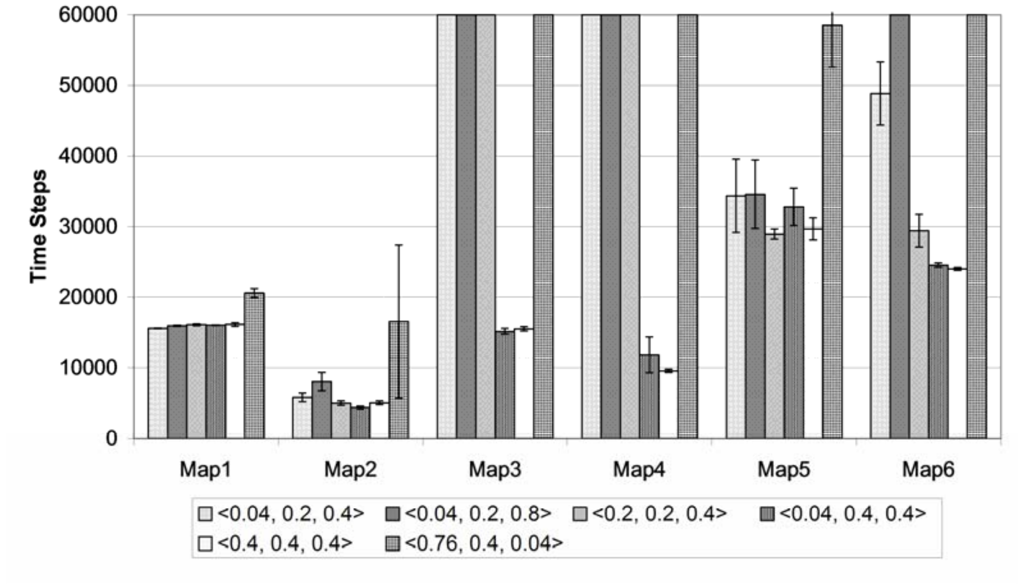
Different experiments compared agents both with and without memory to test the hypothesis that agents with even a limited memory of terrain will find goals more quickly than if they merely know the goal locations. Four memory-related parameters were varied in these experiments:

1) *Memory size*: the number of individual obstacle locations an agent can retain before old obstacles must be removed to make room for new ones (n_{mem}, e.g. 10).
2) *Recording probability*: probability of adding an encountered obstacle to an agent's memory per time step (c_{mem}, e.g. 0.01).
3) *Memory dampening*: strength of influence of remembered obstacles relative to seen ones (d_{mem} in Part 2 of algorithm, e.g. 0.3).
4) *Memory radius*: Number of cells away that a remembered swamp or mountain can be from an agent and still influence the agent's movement (r_{mem} in Part 2 of algorithm, e.g. 3.0).

Upon determining the combination of velocity weights that gave the lowest number of required time steps without memory for each map, experiments were run with memory added where each parameter varied over a range of values. The memory size was tested over a range of 5 to 100. The recording probability was tested over a range of 0.005 to 0.25. That range may seem low, but note that given a probability of 1, agents will constantly be adding memories of terrain cells nearby and forgetting them almost as soon as they move out of view. The memory dampening was tested over a range of 0.05 to 2. The memory radius was tested over a range 1 to 30. However, the goal in this study was not to find optimal memory parameters. Thus, for the experiments in this work, values within these ranges were selected that gave reasonable performance (see Results section).

When an agent's memory becomes full, older remembered obstacles in the memory must be eliminated to make room for new ones. Several strategies for eliminating obstacles from memory were tested: remove the oldest, remove the newest, remove randomly, and remove the i^{th} newest obstacle with probability $(½)^i$. Random removal is the default.

When an agent has entered a large area of swamp cells, it would be expected to slow down and potentially accumulate many memories of the interior cells of this area. It would thus appear to be more useful for agents just to remember the "edges" of such a region (i.e., mountain or swamp cells closer to open terrain) and not the cells deeper within the region. To see if this was true, a naive strategy was compared with a strategy where agents are limited to adding to their memory only when they have open terrain within their vision radius. The result is that only cells on or near the exteriors of large obstacles could be added to an agent's memory. This latter strategy works much better and is the default.



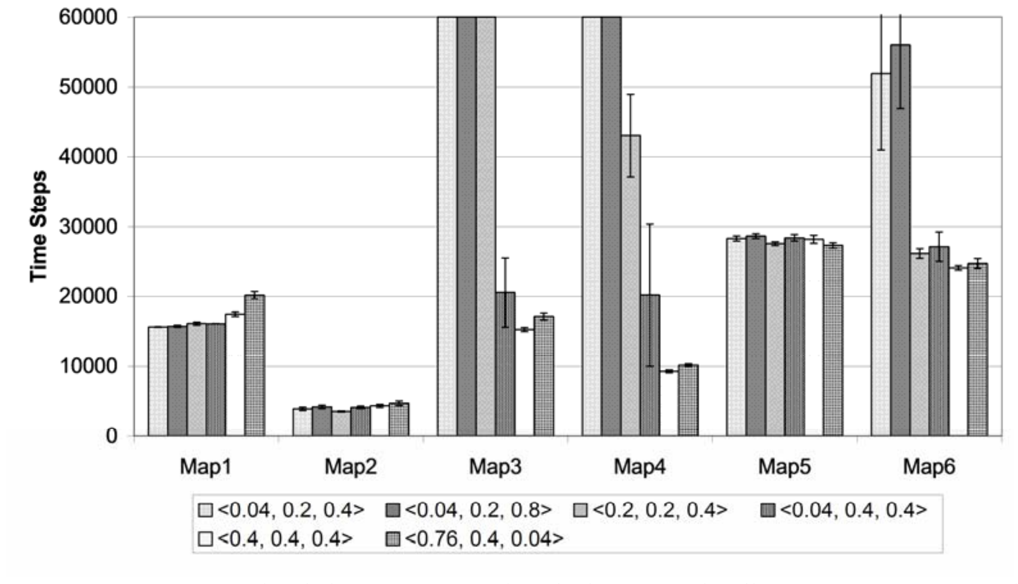


Figure 3.5. a. Mean required time steps of simulations on six different test maps, without memory implemented, for different velocity weights <w_c, w_{mv}, w_a >. Error bars here and in all following figures are standard deviations over 20 runs. Simulations with a mean of 60,000 time steps have no standard deviations because they always hit the maximum. **b.** Mean required time steps of simulations on six different test maps, with memory implemented, for different velocity weights <w_c, w_{mv}, w_a >. The memory parameters used were $d_{mem} = 0.3$, $r_{mem} = 3$, $n_{mem} = 10$, $c_{mem} = .01$.

Agents in open terrain have a maximum speed of 0.25, recalling that the cell size is of unit length. In swamp cells, the maximum velocity drops by a factor of 0.5, cutting an agent's speed in half. In mountain cells, the maximum velocity drops by a factor of roughly 0.03, bringing an agent to a speed so slow that it is ineffective. The parameters listed in the algorithm (see Appendix A) were specifically chosen to give good results when there is no memory implemented, so the memory-less agents would be competitive with agents having memory.

3.5 Results

This section first examines how a number of weight variations and parameter variations influenced agent behavior, and then presents results showing that, with suitable parameter values, agent collectives with memories can substantially outperform those without memories.

3.5.1 Weights Governing Collective Movements

The first simulations examined the effects of varying the weights $<w_c,w_{mv},w_a>$ that govern organizing collective movements, while holding all other parameters constant at their default values. These weights governed how the agents moved collectively. These consist of weights for moving to center of neighbors (w_c), matching velocity of neighbors (w_{mv}), and avoiding collisions (w_a), as seen in Figure 3.3. A wide range of weights were examined, and representative examples are reported here. Figure 3.5a shows the mean required time steps for simulations using the six terrains where agents cannot generate memories, while Figure 3.5b shows the same results with memory implemented using the example memory parameter values listed in Section 3.4. Each of the results shown is an average taken over twenty runs with ten agents on each map. Successful simulations were defined as those that terminate (i.e., agents reach all goals in the correct sequence) before the maximum 60,000 allowed time steps. Simulations requiring more time were classified as unsuccessful, and 60,000 was taken as their required time steps in computing the mean time steps in Figure 3.5. Lower mean required time steps are considered superior to higher mean required time steps.

It can be seen that some cases in Figures 3.5a and 3.5b have no successful runs. In these cases, both the medians and the means are 60,000. In some cases, there were both successful and unsuccessful runs. In those cases, the medians are especially informative. In Figure 3.5a, those cases are in Map 2 at column <0.76,0.4,0.04> with a median of 5,871, in Map 5 at column <0.76,0.4,0.04> with a median of 60,000, and in Map 6 at column <0.04,0.2,0.4> with a median of 48,452. In Figure 3.5b, those cases are in Map 4 at columns <0.2,0.2,0.4> and <0.04,0.4,0.4> with medians of 42,501 and 9,510, respectively, and in Map 6 at columns <0.04,0.2,0.4>, <0.04,0.2,0.8>, and <0.04,0.4,0.4> with medians of 60,000, 60,000 and 25,271, respectively.

By comparing entries in Figure 3.5a to corresponding entries in Figure 3.5b, it is evident that there was a general tendency for agents with memory to do better or about the same as those without memory. Sometimes having memory led to a substantial improvement. Simulations where avoidance was the dominant influence and where centering influence had the least effect (e.g., <0.04,0.2,0.4>, <0.04,0.2,0.8>, and <0.2,0.2,0.4>) were largely unsuccessful on Maps 3 and 4, regardless of whether or not memory was implemented. In contrast, simulations where avoidance was equal in influence to other factors (e.g., <0.04,0.4,0.4> and <0.4,0.4,0.4>) usually did well on all maps regardless of whether or not memory was implemented. When centering was dominant and avoidance had the least influence (e.g., <0.76,0.4,0.04>), simulations without memory were usually unsuccessful for Maps 3 through 6, but dramatically improved when memory was added.

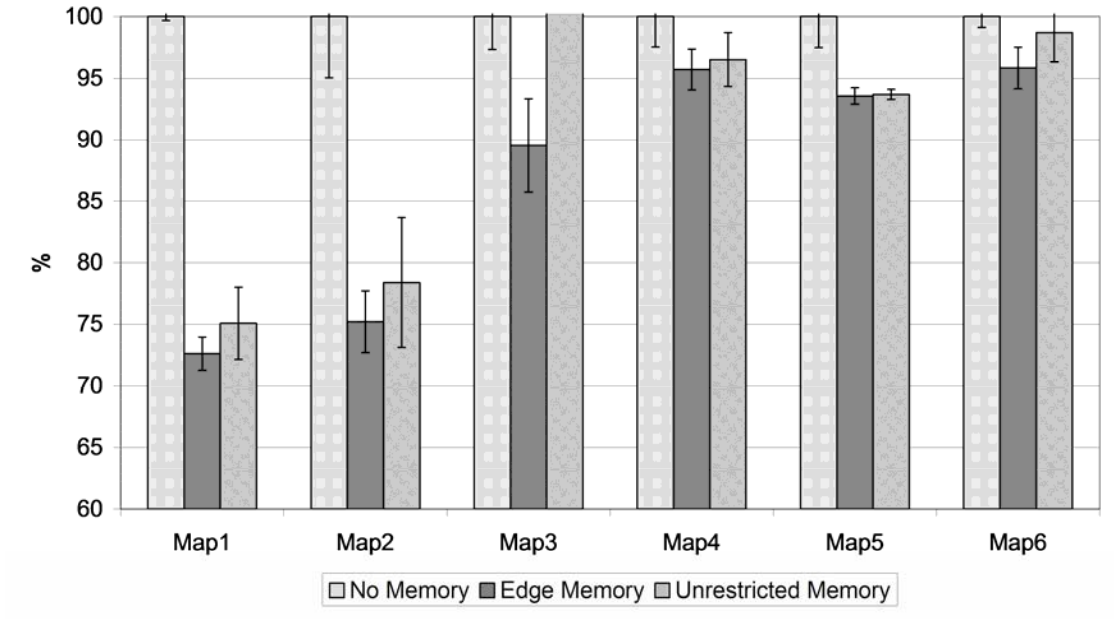


Figure 3.6. Mean required time steps of simulations where agents only update their memory when open terrain is in their view radius versus simulations where the memory update is unrestricted. Results are computed as a percentage of the mean required time steps of simulations where memory is not implemented. The chosen weights for the self-organizing collective movement on each map gave the best observed results without memory. The chosen memory parameters gave the best observed results with memory implemented. These weights and parameters are
for Map1, $w_c = 0.04$, $w_{mv} = 0.2$, $w_a = 0.4$, $d_{mem} = 0.75$, $r_{mem} = 2$, $n_{mem} = 30$, $c_{mem} = .25$;
for Map2, $w_c = 0.04$, $w_{mv} = 0.4$, $w_a = 0.4$, $d_{mem} = 0.5$, $r_{mem} = 5$, $n_{mem} = 30$, $c_{mem} = .02$;
for Map3, $w_c = 0.04$, $w_{mv} = 0.4$, $w_a = 0.4$, $d_{mem} = 0.5$, $r_{mem} = 20$, $n_{mem} = 40$, $c_{mem} = .01$;
for Map4, $w_c = 0.4$, $w_{mv} = 0.4$, $w_a = 0.4$, $d_{mem} = 0.5$, $r_{mem} = 2$, $n_{mem} = 10$, $c_{mem} = .05$;
for Map5, $w_c = 0.2$, $w_{mv} = 0.2$, $w_a = 0.4$, $d_{mem} = 0.3$, $r_{mem} = 3$, $n_{mem} = 30$, $c_{mem} = .03$; and
for Map6, $w_c = 0.4$, $w_{mv} = 0.4$, $w_a = 0.4$, $d_{mem} = 1$, $r_{mem} = 1$, $n_{mem} = 30$, $c_{mem} = .01$;
Other parameters use the default values in the Appendix. The Map 3 simulation with unrestricted memory had one successful run that took 386% of the time of the average run without memory.

3.5.2 Strategic Selection of Memories

A strategy of allowing each agent to update its memory only when there was open terrain within its local view radius was compared to a naive strategy of allowing an agent to update its memory at any point. As Figure 3.6 shows, limiting the memory update to this situation did at least marginally better in all cases. This was seen most dramatically with Map 3 where only one simulation had a successful run using the naive memory strategy, and this run took nearly the maximum allowed number of time steps. With Maps 4 and 5, the limited update strategy had only a small advantage.

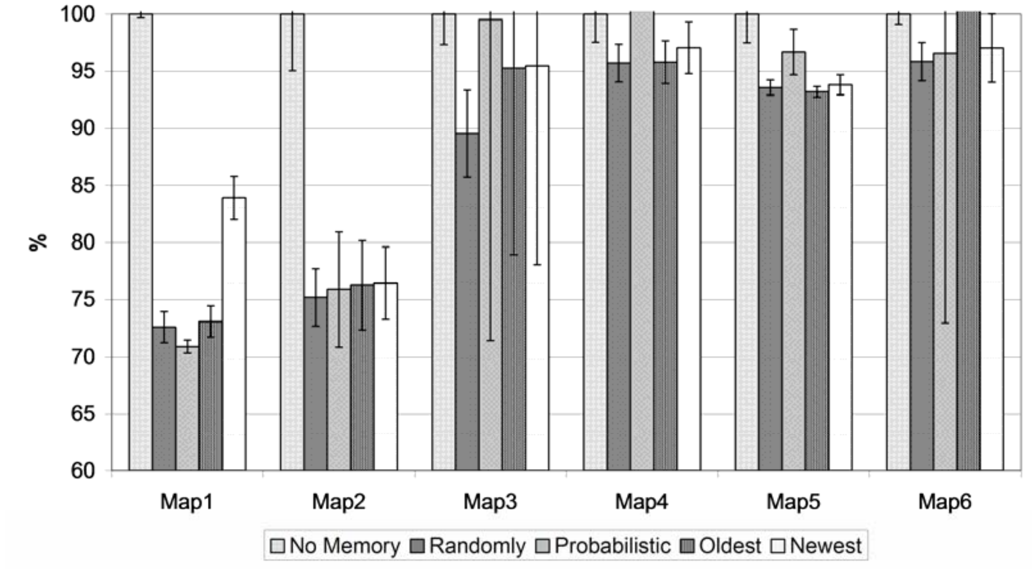


Figure 3.7. Mean required time steps of the various strategies for memory management as a percentage of the mean required time steps for each map without memory. The weights for the self-organizing collective movement are ones that gave the best observed results without memory on each map. The memory parameters chosen gave the best observed results with memory implemented and are described in the caption of Figure 3.6.

3.5.3 Memory Management

The four methods tested for selecting a memory to delete when needed were: (1) randomly, (2) the ith newest memory with probability $(\frac{1}{2})^i$, (3) oldest, and (4) newest.

Figure 3.7 shows the results of using these different methods on the different maps, based on the best observed values of weights for the self-organizing collective movement and memory parameters. While the probabilistic random removal did better than the other methods on Map 1, it performs substantially worse than the simulations without memory on Maps 4 and 6. Overall, the random removal strategy appeared very effective; no other method consistently improved on it. The “remove oldest” and “remove newest” strategies did not show any substantial improvement for any single case over random removal.

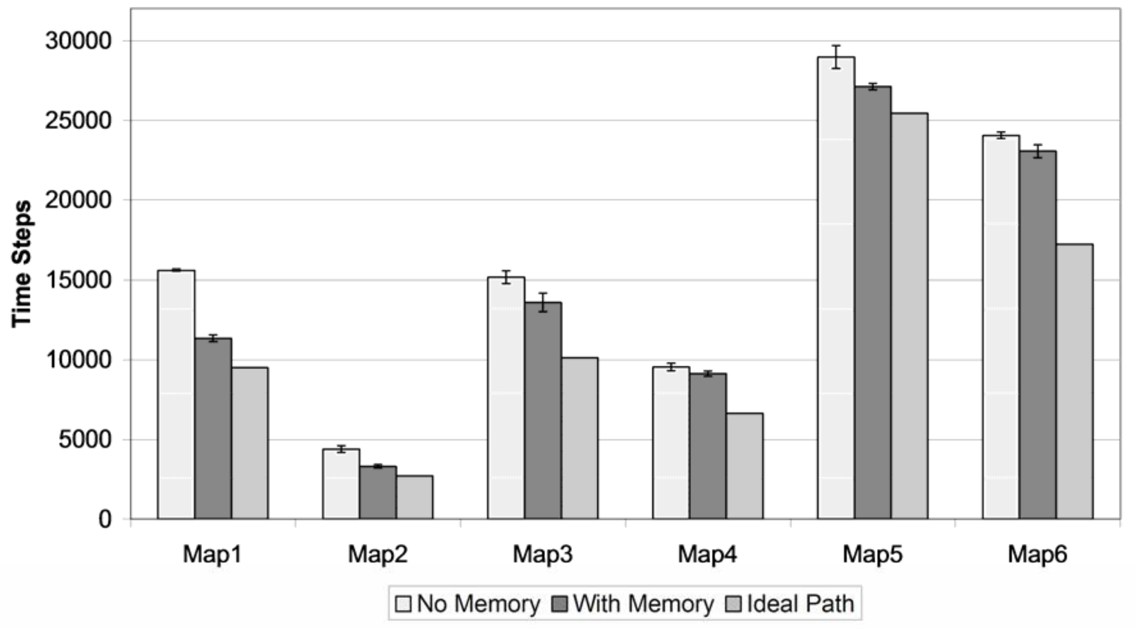


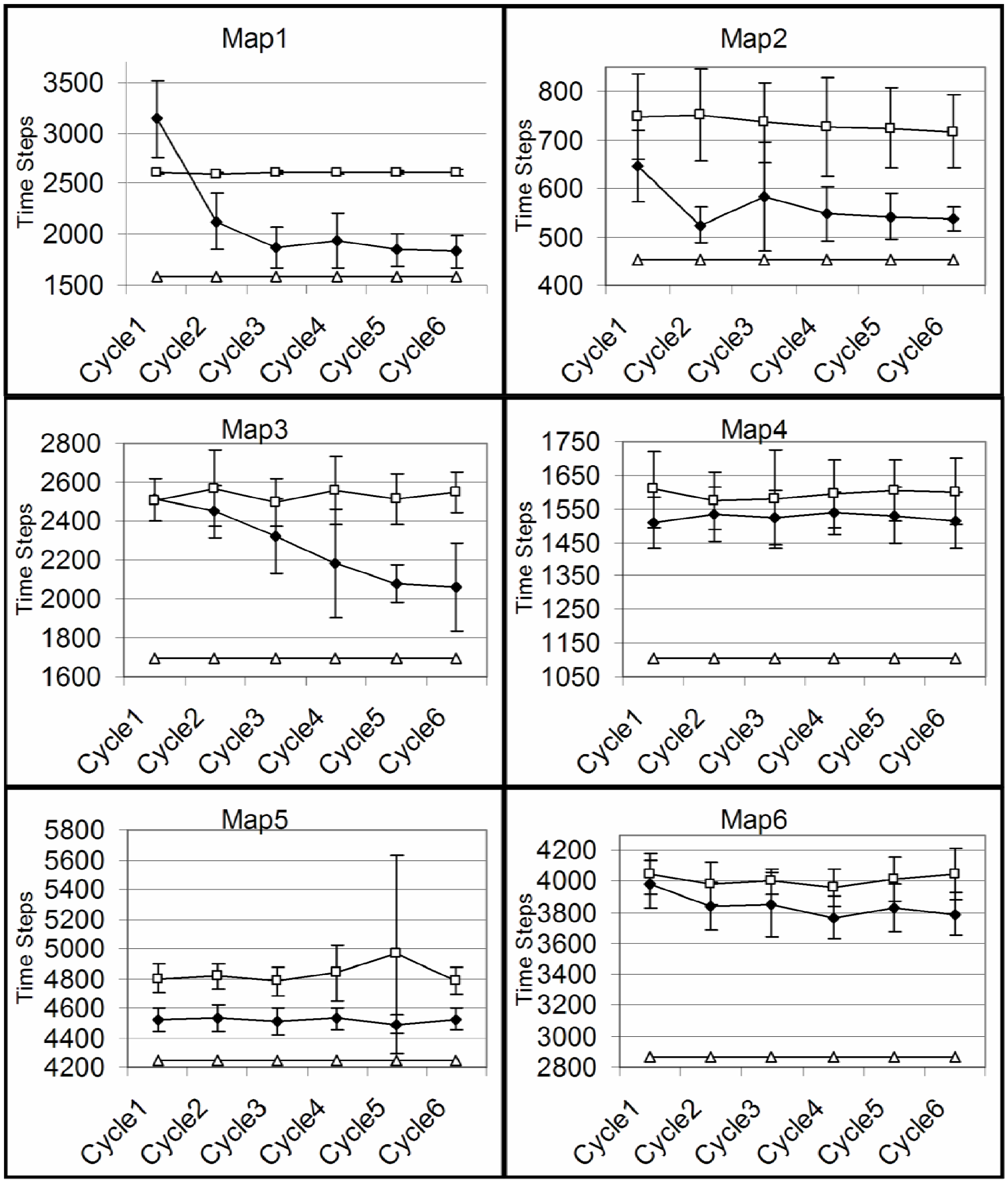
Figure 3.8. The best observed mean required time steps for simulations without memory, the mean required time steps for simulations with memory and the same weights, and the required time steps of the ideal path. The weights and parameters chosen are described in the caption of Figure 3.6.

3.5.4 Best Observed Results

In order to judge whether or not having a memory reduced the time needed for agents to find their goals in the different terrains, the best results observed in the simulations *without* memory for each individual terrain (those with a perfect success rate and the lowest mean required time steps) were found first. Thus, with Map 3, for example, w_c = 0.04, w_{mv} = 0.4, and w_a = 0.4 were used (see Figure 3.5a). These results were compared to the best observed required time steps with memory implemented under otherwise identical conditions (i.e., the same weights and other parameters). It was hypothesized that using a memory would progressively improve the performance of each simulation over time, so the agents were run through six tours through their goals.

For comparative purposes, the required time steps of an "ideal path" at maximum velocity through the map were also computed. An ideal path is a path the agents could take that would give them the best possible required time steps, i.e. the shortest path avoiding obstacles. This path would be extremely difficult for a simulation to approach because, for example, agents would be required to move along the edges of obstacle cells in many cases. This condition would be contrary to the agents' programmed aversion to such cells. Also, on the first tour of the goal destinations, the agents (remembering no past obstacles initially) were not expected to do better than agents without memories. Knowing the duration of an ideal path provided a metric with which to judge how much improvement was theoretically possible.

Figure 3.8 displays the three following results: (1) the mean required time steps for each map of the best observed simulation without memory, (2) the matching simulation using the same weights and other parameters as the former but implementing memory with random replacement when memory becomes full, and (3) the theoretically ideal path. For every map, using memory was effective in reducing the required time steps to achieve all goals.



◆ = With Memory, □ = No Memory, △ = Ideal Path

Figure 3.9. Change in the mean time steps of 20 simulations for individual maps over the course of six consecutive cycles. Both simulations with and without memories acquired between cycles are plotted (◆ and □, respectively), as well as the ideal time steps for each cycle. The weights and parameters are the same as in Figure 3.6, except the memory parameters for Map 1, which are now: $d_{mem} = 2$, $r_{mem} = 3$, $n_{mem} = 30$, $c_{mem} = .01$

3.5.5 Improvement over Time

It was hypothesized that most performance improvement due to remembered obstacles would occur following the first cycle that agents made through their goal destinations, with perhaps lesser improvements following subsequent cycles. To test this, the time taken with each terrain for agents to complete each consecutive tour was recorded. The self-organizing collective movement weights chosen were the ones found to be optimal in the previous tests for simulations *without* memory. The memory parameters chosen were not ones that gave the highest overall improvement, but gave the highest improvement for the last two tours because it was the later tours that reflected the ability of the memory to improve the performance over the long term. In every case except Map 1, these turned out to be the parameters used in Figure 3.6.

The results, shown in Figure 3.9, indicate that the simulations without memory did not radically improve or worsen over time, which was expected since the cycles were identical in terms of obstacles encountered and the agents' lack of foreknowledge of the obstacles. The only exception is Map 5, which has an unusual spike in runtime on cycle 5, a reflection of the map's complex structure, in which small aberrations in the initial state and collective movements could easily cause agents to be delayed in local areas.

While the results for simulations varied, in all cases, the addition of a limited memory improved performance. Maps 1, 2, 3, and 6 all support the theory that agents with memory will perform progressively better in later cycles, but sometimes (e.g., Map 3) the improvements were more gradual than expected. The simulations began the first one or two cycles with mean time steps close to those of the simulations without memory—and exceeding it in the case of Map 1—but the mean time steps of subsequent cycles dropped closer and closer to the ideal time steps. However, in Maps 4 and 5, there seemed to be no improvement over time, even though memory was significantly beneficial, even during the first cycle.

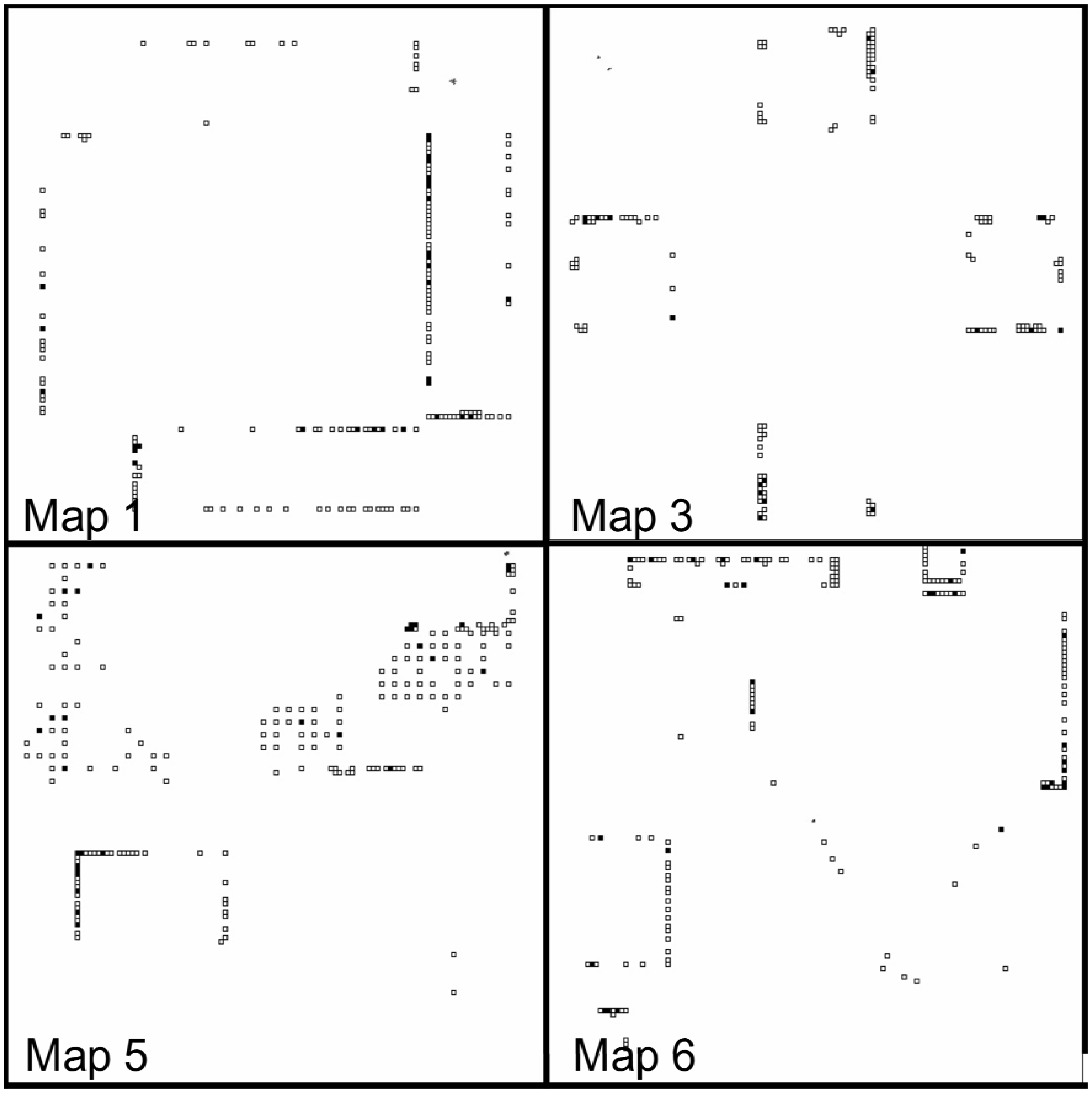


Figure 3.10. Maps as seen by the agent collective at the beginning of the sixth tour. Compare these memory maps to the actual terrain maps shown in Figure 3.4. The squares about the terrain are cells currently in at least one agent's memory. Hollow squares are in any agent's memory, while filled squares are in a single randomly-chosen agent's memory. Taken by itself, that single agent's memory does not outline the terrain, but when viewed in terms of the collective's distributed memory, the outlines of obstacle regions often become clearly defined. The weights and parameters used in these examples are the same as those in Figure 3.6.

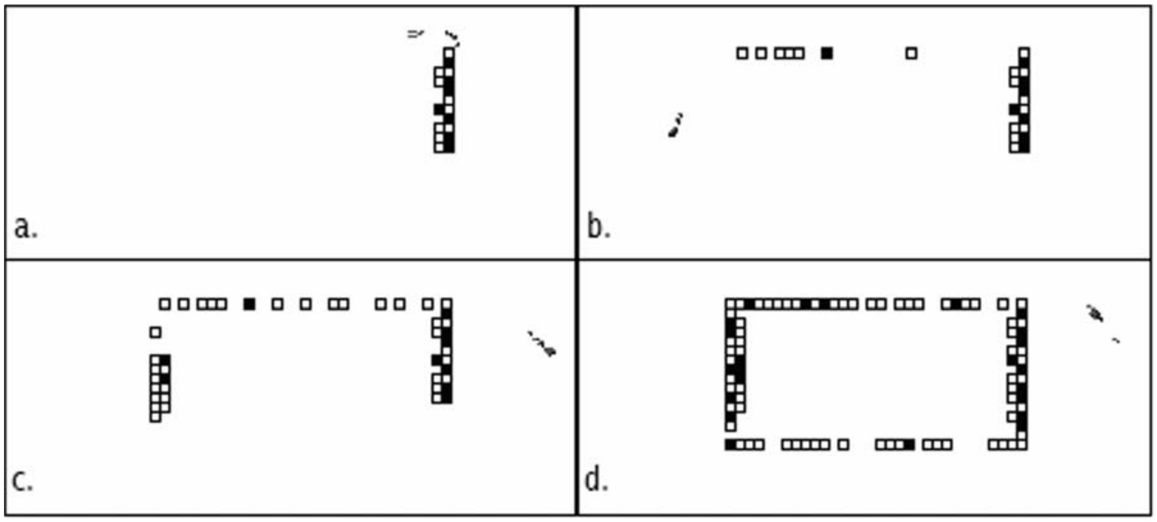


Figure 3.11. An example simulation run on the terrain Map 2 from the agent collective's perspective. Hollow squares are obstacle cells remembered by at least one agent. Filled squares are obstacle cells remembered by a single randomly-chosen agent. **a.** At the beginning of the simulation, agents have just encountered the obstacle for the first time. **b.** Agents have completed half of their first tour. **c.** Agents have completed their first tour. **d.** Agents have completed five tours through the terrain.

The unexpected result that having a memory improved performance on the *first* cycle was initially puzzling. How could memory be improving agent performance during the first tour of the goals, before encountering remembered obstacles again on another tour? Inspection of agent behaviors during the first tour of simulations provided an explanation for this phenomenon. A remembered obstacle has an immediate effect on the movement of the agent collective, as follows. Sometimes the memory-less agent collective would become "trapped" or substantially delayed, such as when the agents were unable to find their way around a certain arrangement of obstacles. When this occurred to agents with memories, the remembered obstacles would influence the collection of agents to move away from the obstacles even when they were not in direct view, and thus increased the speed at which the agents circumvented the objects and reached their goal. Thus, memory often exerted *two* distinct beneficial effects on agent behaviors: (1) an immediate avoidance of repeated visits to a just observed, interfering obstacle (Maps 4, 5), and (2) a delayed, long-term avoidance of repeat encounters with previously seen obstacles (Maps 1, 3). In some cases, both effects were observed (Maps 2, 6). Notice that Maps 4 and 5, which displayed no improvement over time, also had the highest probability of learning a new memory per turn. They acquired memories so quickly that the memories only had immediate effects and were replaced before the agents could use them again and gain incremental improvement. Therefore, their performance did not improve over time.

3.5.6 Memory Distribution

The distribution of agent memories throughout the terrain over the course of a simulation depends on the memory parameters chosen, but certain features remain consistent among all runs. Due to the rule that agents only add memories when open terrain is in view, only cells at a maximum of unit distance from open terrain will be added to memory. Though the ten agents moving through the terrain encounter the same obstacles typically, they often encounter different parts of the obstacles and thus have very different individual memories. An individual agent will only know scattered obstacle cells, but as a whole, their memories tended to form an outline of the edges of the obstacles they encountered. Examples of this are shown for Maps 1, 3, 5, and 6 in Figure 3.10. The distributed collective memory becomes an entity far more complex than what any individual agent is capable of forming. Figure 3.11 shows the development of the collective memory over time for Map 2. By the end of the simulation, the obstacle in this simple case becomes clearly defined.

3.6 Discussion

There are three main conclusions to emerge from the computational experiments described in this chapter. First, the results support the hypothesis that adding even a limited memory to collectively moving agents in particle systems can substantially improve their performance in repetitively seeking goal destinations scattered throughout an environment containing obstacles. This occurred with all six terrains used. Second, this improvement in collective movements was due to two phenomena: (1) the expected avoidance during later tours of obstacles seen in earlier tours, and (2) an unexpected ability that, even during the first tour, the agent collective could escape more quickly from substantial delays encountered with some obstacle configurations. Third, of the various memory replacement strategies tried, none outperformed simply randomly replacing arbitrary remembered obstacles when memory was full and a new obstacle was to be remembered.

A question that remains is how to know what weights and parameters to choose for a particular map. Simply adding memory was no guarantee that performance would improve. Examining the memory parameters in the best simulations leads to the following observations. A dampening, or weighting, of the effects of recalled obstacles by 0.5 relative to viewed objects generally worked well, assuming that the radius of obstacle influences was larger than the direct visual radius, in enabling agents to avoid obstacles well before seeing them. A good value for the radius of obstacle influences varies depending on properties of the terrain. For terrains with smaller obstacles and small open paths that must be traversed, a radius of 4 or less generally works well. For terrains with large, solitary or well-spaced objects, a radius of 5 generally works well. Keeping the chance of storing a new obstacle in memory low (under 0.05) generally works well as it tends to preserve older memories for recall on later cycles. A maximum number of memories in the range of 30 to 40 generally worked best, but smaller sizes could also work well.

It is concluded that adding individual "rote learning" of obstacles encountered by collectively moving agents (particle systems) can significantly improve their efficiency in both avoiding obstacles and in limiting the delays the obstacles cause, even when first encountered. This information may prove useful in complementing recent advances in control methods in particle systems [Rodriguez, 2004], in improving existing methodologies for control of collective movements in computer graphics, robotic team control, computer games, particle swarm optimization, and other computational applications, and in interpreting the results of future experimental research on group movements in biological populations.

Chapter 4:

Memory Management Strategies for an Agent in a Simulated World of Greater Realism

4.1 Motivation

In the previous chapter, particles that were extended to have a limited memory were examined as simple self-organizing collective entities. The addition of working memory allowed their natural reflexive movements, which were governed by basic flocking rules of acceleration, to be influenced by information outside their immediate senses. The environment and agents were both simple, raising the question of whether the principles employed and the questions asked can be extended to agents of greater complexity operating in more challenging and realistic environments. To answer such a question, we need to develop a more realistic agent that builds upon the basic reactive movement behaviors used in particle systems.

With their ability to pursue goals and perform tasks without intervention from an outside controller, autonomous agents have been of particular interest in robotics. These agents have many uses, and potential uses, ranging from building maps using sonar [Pagac, 1998; Konolige, 2004] to being parts of multi-agent exploration teams on other planets [Atherton, 2006]. A wide variety of environmental processing or control methods has been used for these autonomous systems, including Markov localization [Wang, 2006], reinforcement learning [Tan, 2002; Taniguchi, 2005], fuzzy logic [Tan, 2002], and artificial neural networks [Best, 2004]. A few past autonomous robots have used potential fields of attractive and repulsive forces to guide agents [Vail, 2003; Kurihara, 2005], yet environmental information is either usually already provided to the agent or is only used while observed, rather than constructed by the agent over time in order to facilitate movement in an environment of unknown layout.

In this chapter, I develop an agent with the capability of accumulating and utilizing memory in a more realistic environment than the previous chapter. This agent retains the basic particle system approach, but now also does much more. In the next chapter, the same questions about the impact of working memory considered in Chapter 3 will be asked of a team of such agents operating in a simulated 3-dimensional environment representing a city. The agent will receive information about the urban environment from a sequence of eye-level images, and subsequently must interpret the image sequences it sees as it moves through the city. The agent will also be required to obey simple commands, as opposed to reacting solely on reflex and attraction to final goal locations. Taking further inspiration from biology, neural networks will be involved in achieving these behaviors for the agent, particularly in interpreting visual stimuli and incoming commands. Because of the added complexity this agent entails, this chapter focuses on creating and testing a single agent in the city environment as an initial step.

While no existing machine intelligence has proved capable of matching general-purpose human cognition, the pursuit of the creation of such an intelligence has not abated since the earliest inception of artificial intelligence. Part of the problem is that the disparate methods in AI, while capable of simulating human cognition in certain regards, are unable at the current state of the art to serve general purpose functions. Neural network methods, while successful in the areas such as pattern recognition and data visualization, are difficult to extend into other areas of cognition. Similarly, more symbolic methods of AI, while applicable to inference and knowledge representation, have their own limitations. At present, it seems that a hybrid architecture would be a reasonable first step toward an optimal and attainable general purpose machine intelligence. This manner of architecture combines a variety of AI methods and allows the different AI methods that have been implemented effectively in the past to be applied to the areas where they achieve the most success,

The following is a skeleton model of an agent operating in an urban environment, and represents only a small step toward such a hybrid architecture. The agent is trained to interact with environmental stimuli with neural networks, both in consuming and producing information. Within the agent's executive control, a simple finite state automaton is used as a placeholder for ultimately more complex and intelligent behavior based on methods of symbolic artificial intelligence. Memory is present in all agents when it comes to commands issued or certain reflex behaviors (like fleeing a dangerous area), but variations of the agent considered in the following will have different memory strategies open to them when it comes to knowledge of the environment.

The present scenario for the agent is as an operative moving about an urban environment, potentially containing both "friendlies" and "hostiles," taking simple orders from a person in the form of spoken commands, reacting to visual stimuli, and communicating with "natural language" the relevant information discovered in its environment.

The following questions are asked: Now that the agent must interpret visual input and a new set of movement dynamics needs to be defined, does the baseline reactive particle-like movement control create difficulties for the agent? Is it feasible to use simple neural networks to convert the images the agent "sees" as it moves into a 2-

dimensional map of relevant parts of the city? When the agent is repeatedly touring the city heading to different target locations, does being able to recall and avoid previously observed environmental impediments to movement lead to significant improvements in the times needed to achieve its goals? If so, how do different strategies of using working memory influence the efficiency of their movements? Specifically, how does using only a local memory versus a cumulative memory covering a wider area affect results? It is hypothesized that an agent that converts the image data of the 3-dimensional scene into a 2-dimensional chart, or map, of the area and makes use of this chart will have an easier time with movement. It is hypothesized that an agent keeping a map of the positions of obstacles in the terrain they pass and who use it to influence their future movement will be able to perform better (i.e. complete commands faster) than an agent who does not. At the same time, it is hypothesized that agents who are limited to only a local memory will experience more success than those with no memory whatsoever. Finally, it is hypothesized that an agent with local memory will be comparable, and perhaps more successful, than an agent using cumulative memory in cases where the agent is not covering the same terrain over and over.

4.2 Structure of the Environment

The agent exists in a bounded two-dimensional artificial world. The organization of this urban world is illustrated in Figure 4.1. It consists of a network of streets between buildings and surrounded on three sides by open, grassy areas and on the other by a body of water. An implicit coordinate grid is superimposed on the otherwise continuous artificial world. While the agent moves in a continuous, real-valued space, for convenience, regions of the terrain that hinder the agents are taken to occupy one or more contiguous unit cells of the coordinate grid. Generally, agents would only want to move across open or street terrain (i.e. places not occupied by buildings or water). In addition to the agent and terrain features, the environment contains static objects that exist at points in the real-valued space. Agents do not know about this grid, and do not have access to the map depicted in Figure 4.1. An agent only sees a temporal sequence of 512 by 512 pixels from which it must generate the influences that guide its movements.

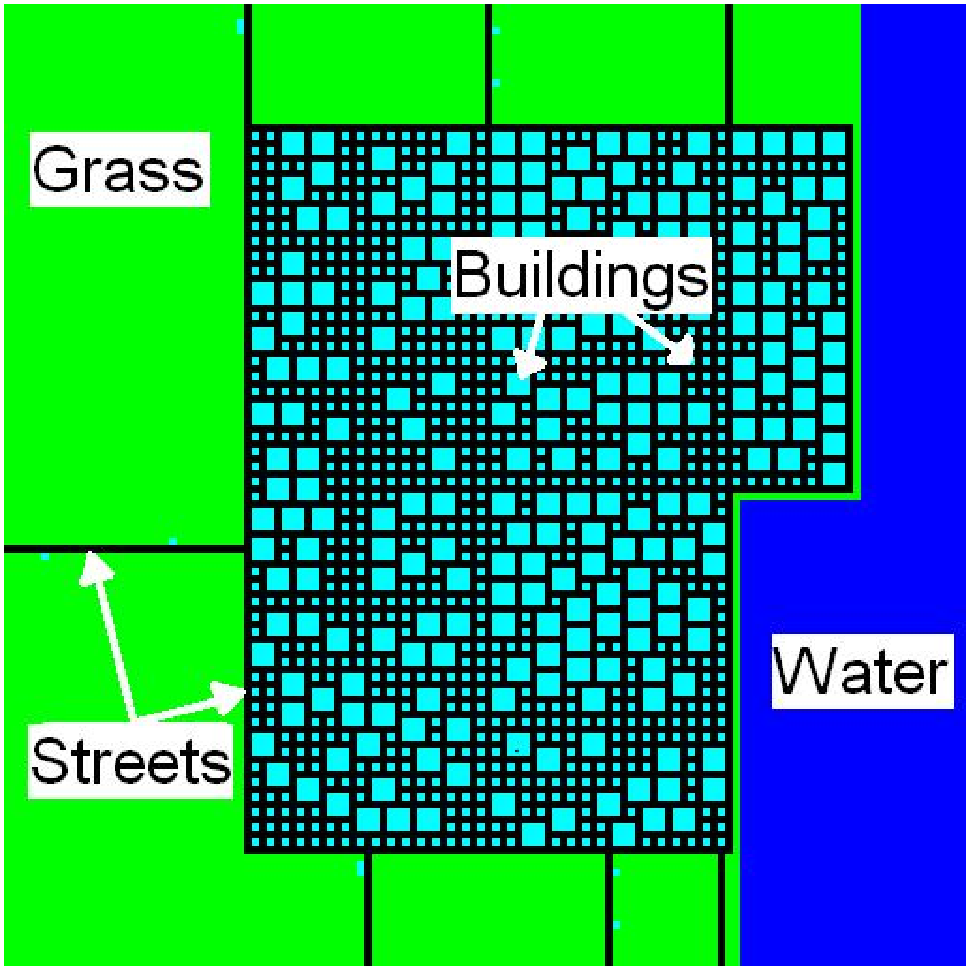


Figure 4.1. Overhead representation of an example city with the different types of terrain labeled. In the center is a collection of buildings (light-colored) separated by streets (dark-colored) which is taken to represent the city. Surrounding the city on the north, west, and south is a wide grassy area. Some streets extend into this area, representing paths outside the city. On the east is a large body of water.

4.2.1 Terrain Types, Buildings, and Districts.

Each cell of the grid superimposed on the real-valued space contains a different type of terrain, much as was done in the previous chapter. The countryside outside the city consists of grassy fields or bodies of water. However, the environment where the agent mostly operates is the city. Here cells treated as streets separate cells defined as buildings. How the environment appears to the agent is described in Section 4.3.2.

The city is divided into different districts, which are distinguished by differing probabilities of certain building types appearing there. For instance, the industrial district has a high percentage of factories, while a significant percentage of the downtown district will be occupied by office buildings. The nine districts implemented in the sample city are westside, southside, dock, industrial, historic, downtown, uptown, slum, and market.

A city is generated by a pseudorandom algorithm that places buildings of certain types in specified fields that are one of the nine types of districts. Buildings either occupy 1x1 cells or 3x3 cells, and cells between buildings are always occupied by streets. Surrounding terrain is placed using the same method of specifying an x-y range of that terrain, but with uniformity.

4.2.2 Environmental Objects

Objects do not occupy entire cells, but instead are centered at a real-valued point in the environment space. Each object also has an associated 3-dimensional shape that determines the object’s size. While not visible on the overhead map in Figure 4.1, these objects are visible at the level of the agent. In the sample scenario, objects are taken to be people of various kinds, vehicles (cars, trucks, etc.) and other large objects (e.g. dumpsters).

4.3 Agent in Environment

4.3.1 Nature of Time, Space, and Movement

Time is modeled simply as a counter that increases as the simulation progresses. During each discrete time step, the agent is able to process the visual scene before it, process any auditory input that is incoming, adjust its memory of the environment and received commands, make decisions based on stimuli and memory, update its angle of orientation, speak if necessary, and move incrementally if not blocked.

The currently implemented agent in the environment is capable of either walking or flying depending on the preconditions set at runtime. This allows for two different tasks and perspectives and changes the interaction with the environment. While flying, agents do not necessarily have to avoid buildings, but can fly over them. While walking, agents cannot pass through buildings or over bodies of water and will experience slowdown when walking through forests. For comparability with the results in the previous chapter, the agent is assumed to be walking at ground level for the rest of the dissertation.

4.3.2 3D Environment

The walking agent operates along the continuous 2-dimensional surface of streets, grass, etc., but because certain terrain features in this environment have a height, the environment can be viewed as 3-dimensional. In fact, the agent sees a temporal sequence of 2-dimensional images (512 by 512 pixels) of the 3-dimensional world, limited to a field of view determined by the angle and direction of its camera. Figures 4.2 and 4.3 are examples of the raw images processed by the agent. This eye level view of the agent must be generated from this environment, so we are able to synthesize what the agent would see given its position and orientation.

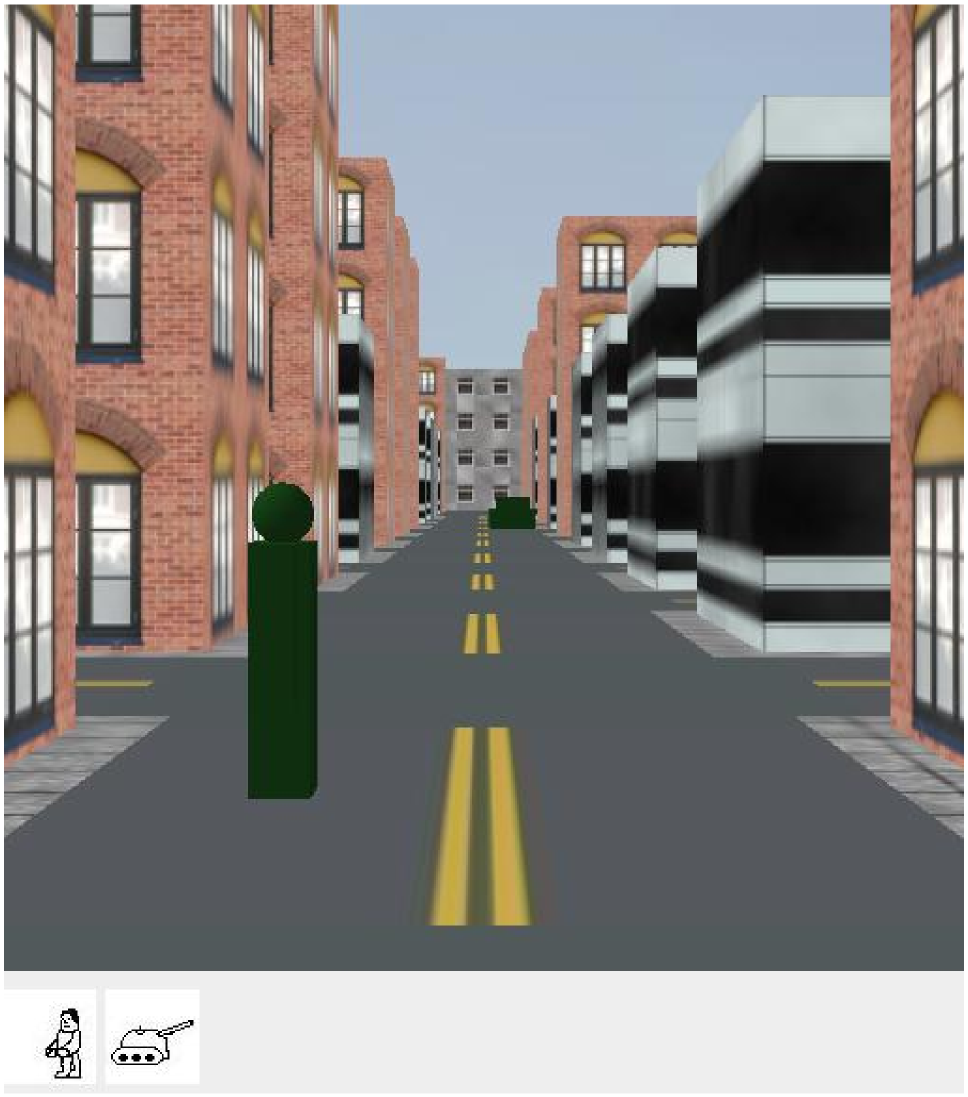


Figure 4.2. 2-dimensional, 512 by 512 pixel image of the agent view's of the city. The agent is looking down a street, with buildings and intersecting streets along the side. This shot contains two objects, a man and, in the distance down the street, a tank. The thumbnails under the 3D view image represent the objects that the agent has segmented from the scene (how this is done is described in the text).

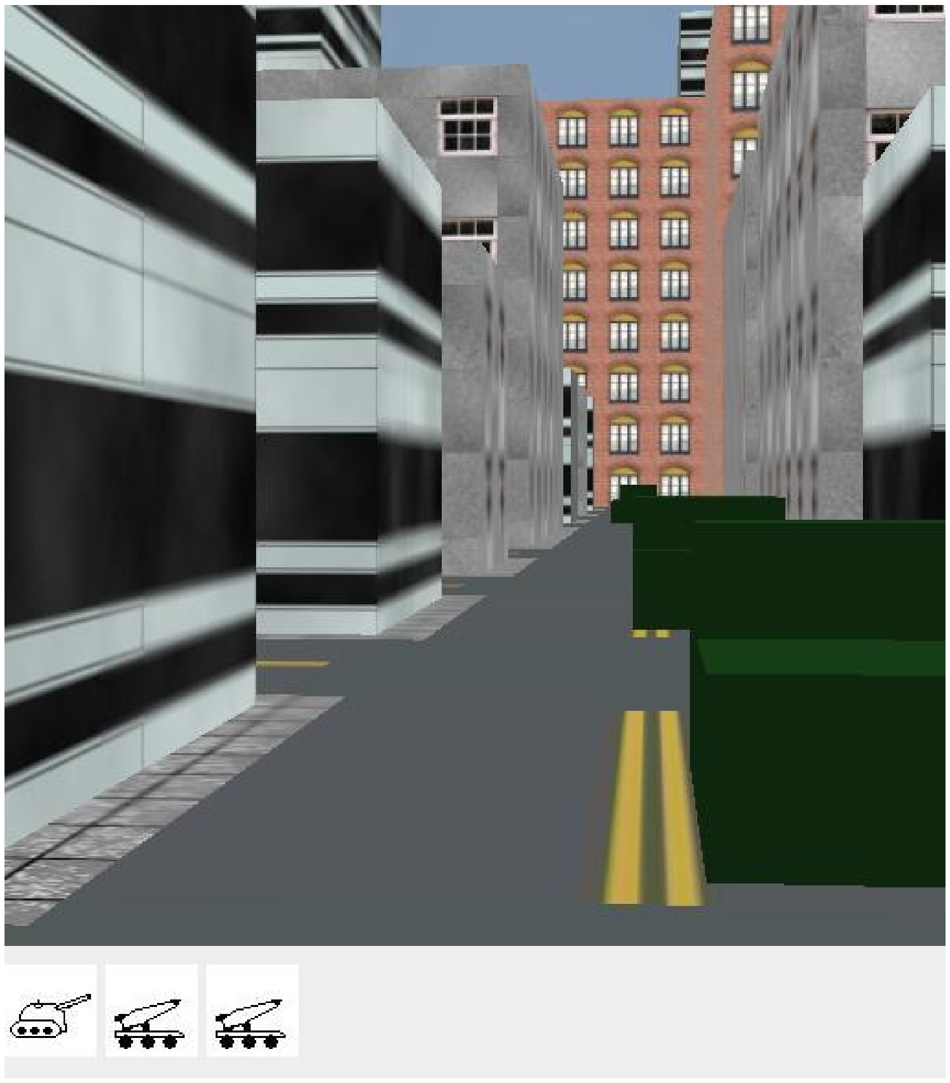


Figure 4.3. Another 2-dimensional, 512 by 512 pixel image of the agent's perspective. As in Figure 4.2, this image is part of a temporal stream of images the agent is required to process. In this shot, the three objects are a tank and two missile launchers. Once again, the thumbnails below represent the segmented objects.

The 3D environment is constructed in Open-GL. Using the agent's real-valued x and y position on the map and its angle of orientation, a camera ("eye") is assumed present that generates a 512 by 512 pixel view of the environment. This is done at every time step, as the agent moves and/or changes its orientation, creating a steady and fluid stream of images of what the agent sees as it explores the environment. The view angle is 30 degrees, giving the agent a very limited range of vision at any given time step.

Different textures were used for the grass, streets, and water. Three different textures are used for the various building types. It is important to note that this use of these artificial textures and colors makes the image processing required for the agent much easier. Image segmentation is an open problem, and the processing here is not an innovation of my work, but rather something that needed to be kept simple and manageable to allow the exploration of the dissertation topic of working memory in this environment.

4.4 Structure of Agent Processing of Input and Output

The internal network structure of the agent consumes environmental input via visual and auditory processors, makes decisions based on this information, its present state, and memories, and changes its movement behavior and makes appropriate comments for the situation (responding to commands issued previously).

This structure and its connections to sensory input and output are depicted in Figure 4.4 at a high level. Visual input (the 3D image described above and close-up images of objects and buildings) and auditory input (a sequence of phonemes) are processed with separate neural networks modeled after those in [Weems, 2006]. Movement is updated as changes to the agent's angle of orientation in the 2-dimensional ground plane. Simulated speech is generated from the command memories via a neural network into a sequence of phonemes. This happens at the appearance of relevant visual stimuli or under certain circumstances of the agent position in the environment.

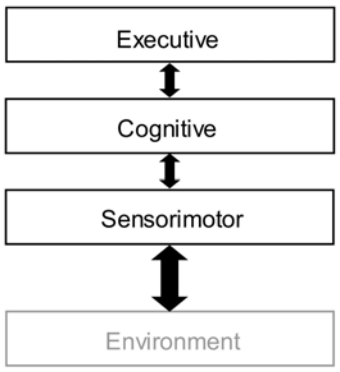


Figure 4.4. High-level depiction of interaction between various internal systems (shown in black) and the environment (shown in gray). The agent's sensorimotor systems connect it to the environment, allowing it to process visual and auditory information as well as speak and adjust its movement. The cognitive network encodes incoming visual and sequential verbal information and also constructs speech output sequences. The executive control governs agent behavior based on current and remembered stimuli and internal rules of movement and speech.

4.4.1 Visual Input: "Where" and "What" Pathways

In the human brain, vision is processed in two pathways, the dorsal and the ventral [Rueckl, 1989]. The dorsal is more responsible for recognition of the location of objects in the visual field and is considered a "where" pathway, while the ventral handles the identification of the particular objects in a scene, making it the "what" pathway. Inspired by this, these two pathways are separate in the agent, thereby simplifying the task of recognizing objects and knowing their location.

The "where" pathway takes a 2-dimensional image of the 3-dimensional scene depicted in Figures 3.1 and 3.2 as its input. Features of interest in the scene are certain terrain features and objects. This visual field is available to the agent, so that it can avoid obstacles that are in its path and too close, specifically buildings and terrain that are either impassable or difficult.

The visual field is processed in several ways, but the remembered commands determine which will influence the agent's movement. If there are no commands for the agent to patrol or go to a specific location and if there is no evasion command that is actively being followed (i.e. the agent has seen recently an object it was told to evade), then the agent operates using reflexive movement. Otherwise, the agent navigates by using waypoints it places in its understanding of the environment.

Reflexive movement is based on analysis of the environment using seven different 1-dimensional self-organizing maps, each associated with a different patch of the environmental image available to the agent. These patches are spaced out along the bottom of the image, since that represents the part of the environment closest to the agent. The average color of each of these patches is determined. Then, from this average color, the agent makes a judgment that the patch holds either a street, grass, a building, water, or an object in the environment. Each judgment is in turn regarded as "the winner" of its self-organizing map. This list of "winners" is sent to the executive control, so it can determine the degree of the turn angle necessary to make in order to avoid the obstacles. Figure 4.5 shows the structure for a single navigation network.

These self-organizing maps, where each node in a map is initialized to a random weight vector of length 3, learn on 2000 random 2-dimensional 512 by 512 pixel images of the 3-dimensional environment. When presented with one of these images, the average color in the patch associated with each map is computed. The winner in the associated map is the node with the closest weight vector to the average color (the values of the vector being degree of red, green, and blue). The winning node and its neighbors are all adjusted by some amount towards the average color of the patch. The number of neighbors that change with the winner drops over the course of the learning (beginning with 3 neighbors in both directions, and dropping by 1 every 500 updates), and the rate at which the self-organizing maps adjust themselves drops by 5% every 50 updates.

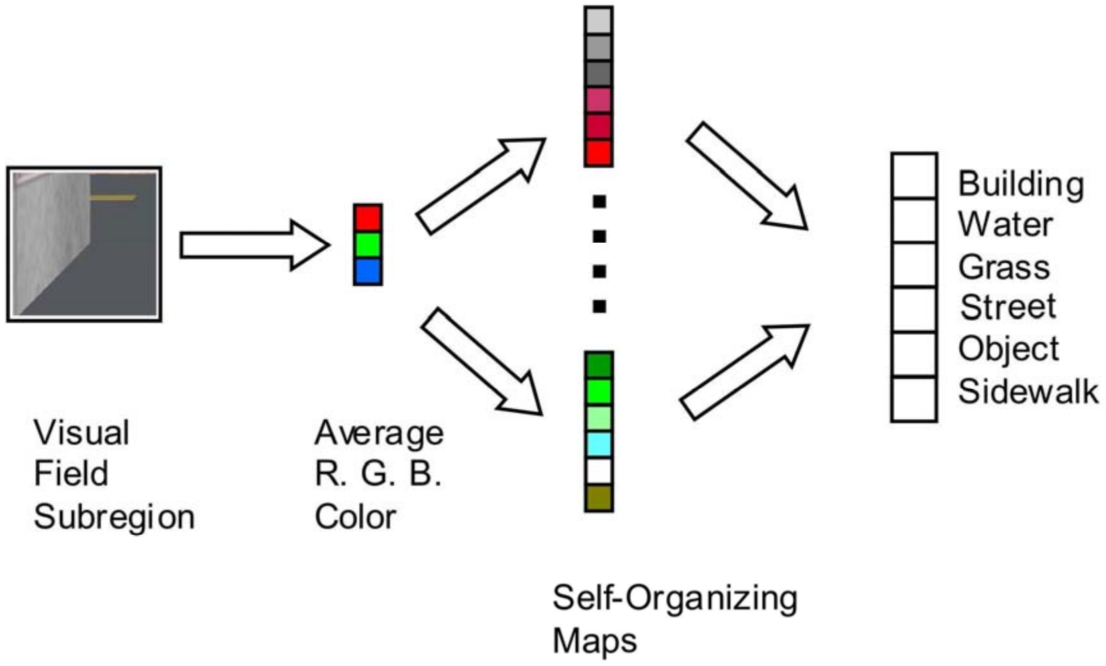


Figure 4.5. Representation of one of the navigation networks. The average color of a subregion of the visual field is calculated. From this, the closest cell in the self-organizing map is taken to be the winner. This winner then turns on one of the decision units, indicating whether the subregion depicts mostly a street, water, grass, a building, a sidewalk, or an object.

After this process, the map is fixed, and the networks are trained to recognize what the patches represent in the environment. These features can be streets, buildings, grass, water, objects, or sidewalks. For another 2000 random images of the agent's perspective, the networks are trained as follows. Every pixel in the patch is associated with proper environmental feature it represents. The environmental feature with the most number of pixels in a patch is associated with the corresponding winner in the self-organizing map. If two different features ever become associated with the same winner, the first one retains its association. These cases usually happen where there is ambiguity, for instance where a building and road are both present in a patch.

Movement based on waypoints overrides the reflexive navigation in the cases where the agent is following a command to patrol or go to a specific district. This in turn is overridden by commands to evade when the object to evade is seen, an event which also uses a movement based on waypoints. Overall, the effect of the waypoints is that they guide the agent in the appropriate direction in the environment, either toward a goal state or away from a dangerous location. The positioning of the waypoints in the agent's understanding of the environment in order to achieve the desired course is described below, but the operation of the agent based on the waypoint is simple. The waypoint exerts an attractive force on the agent. Thus the agent moves toward the waypoint, unless it is physically blocked by an object or building. However, the agent chooses waypoint locations such that a direct path to it features no obstructions. Should a waypoint be aberrantly placed and the agent be stuck at an obstruction for a few time steps, the agent will recognize that its waypoint is placed in error and must be deleted and set elsewhere.

The agent uses similar information from the scene as the reflexive navigation, but instead of seven self-organizing maps of the closest visible locations, the agent processes the entire ground out to the horizon, dividing it into square cells of 32 by 32 pixels. As above, self-organizing maps determine the nature of each of these cells and classify them either as passable, impassable, or an object. The agent will only place waypoints in locations it believes passable.

The "what" pathway's input is generated from the 3D scene by using a simple object segmentation method. The grid that does this covers the entire 3D view, once again each cell being equivalent to 32 by 32 pixels in the scene. The average color of each of these cells is then processed by a self-organizing map, similar to the one used in navigation. A different self-organizing map is used for each row of the grid. From these mappings of average color, the agent is trained to recognize which winners correspond to objects (i.e. have a majority of pixels that belong to objects) and which do not.

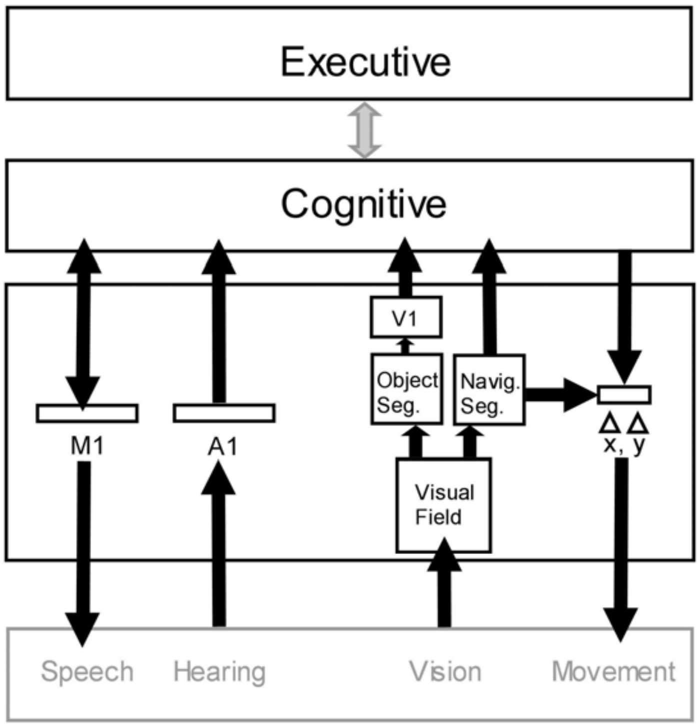


Figure 4.6. Expanded view of the agent's sensorimotor system. Here the agent acquires input information from external sources, either a sequence of phonemes (sequentially presented in A1) or visual information of a 3-dimensional scene in the environment. Visual information is segmented into a sequence of the individual objects in the environment and into a field that aids in reflex actions of the agent, such as avoiding a building or certain terrain when too close. These two visual paths roughly correspond to the "what" and "where" pathways in the brain, respectively. Here, the agent also produces motor information on movement through the environment and sequences of spoken phonemes (sequentially output to M1).

After training, while the agent is operating and searching for objects in the environment, for every time step, the agent uses this grid to determine which 32 by 32 plots of pixels in the 3D view possibly contain objects. All contiguous cells are taken to be part of the same object. For each contiguous set of cells, pixels are randomly queried up to twenty times, since there are potentially some pixels that will not correspond to objects. A successful query of an object pixel will return a 50 by 50 pixel thumbnail image of the corresponding object (e.g. a car). For each object, these thumbnails are presented in succession to a region named V1 to inform the agent of what specific objects it is seeing. These images represent what the stimuli actually looks like, and the agent is required to identify them.

Table 4.1. Regional names in the neural networks and what they represent.

V1:	Primary Visual Cortex
IT:	Inferotemporal Cortex
AG:	Anteriror Gyrus
WA1:	Wernicke's Area
WA2:	Wernicke's Area
BA:	Broca's Area
A1:	Primary Auditory Cortex
M1:	Primary Motor Cortex

The neural network structure here, while not intended to model the human cortex, does take some inspiration from relevant brain regions, including suggestive names (see Table 4.1). Information travels from the input layer, named V1, via connections to a region named IT. These connections undergo unsupervised learning (generating a multi-winner self-organizing map) on the possible visual stimuli that will be encountered, and thus IT generates a unique representation for each image. Information then passes to a hidden layer named AGv, and finally an output layer, where the final features of the item (whether or not it is an enemy, whether or not it is friendly, etc.) are generated. The IT→AGv→Output connections are trained using resilient error-backpropagation.

In addition to recognizing objects, the agent is required to estimate the location in the environment from the image, information that will be needed for certain scenarios. This is done by querying pixels along the base of the object, which are places where the object is close to the ground. Essentially, since the agent has a fixed height and a fixed angle of view in the z-dimension, it knows the distance of any row of pixels in the snapshot. Using this information and the distance from the central vertical line of the snapshot, the agent is able to estimate the location in the environment.

When the agent wants to make this estimate, it begins by choosing a pixel in the camera image (at coordinates x and y) that represents a part of an object close to the ground (done by choosing a pixel from one of the lowest cells of the contiguous object in the segmented image). The distance outward in a straight line for each horizontal row of pixels to the horizon line is known. When a pixel is chosen, the distance straight ahead (dependent on y) is called d_h. The camera dimension (512 in this case) is called c. The k is a constant depending on the screen size, and it is 955.6 for 512 by 512. The agent is at coordinates a_x, a_y and has angle α. The following equations show how the agent estimates the object's coordinates in the environment from this information.

$$d_{lr} = \frac{\left(c/2 - (x)\right)d_h}{k}$$

$$\phi = \tan^{-1}\left(\frac{d_{lr}}{d_h}\right)$$

$$d = \frac{d_h}{\cos(\phi)}$$

$$o_x = a_x + \cos(\alpha + \phi)d$$
$$o_y = a_y + \sin(\alpha + \phi)d$$

The equations use the information about the environmental distance of the pixel's row from the agent to compute d_{lr}, which is the distance either to the left (positive) or right (negative). This is used to compute the angle of that position in the environment from the agent, $\emptyset$. The cosine of this angle is the adjacent (d_h) over the hypotenuse (d), so d is computed next. Using this computed angle and distance, along with the orientation and position of the agent, the object's location is estimated. When completed, coordinates o_x, o_y contain the approximate location represented by the pixel, assuming the pixel is on or close to the ground. Figure 4.7 shows an idealized example of the information taken from the picture (x and y) and the environment (d_h).

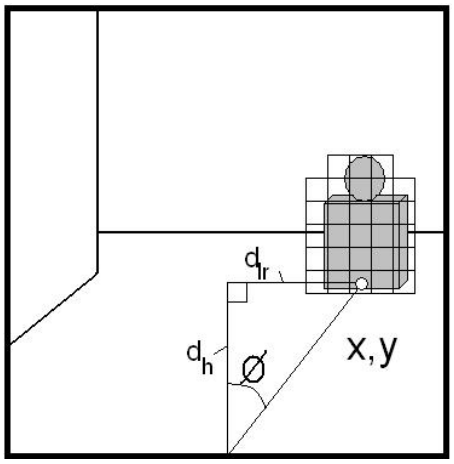


Figure 4.7. Depiction of how object's locations are approximated from image information and knowledge of distances of the pixels in a straight line from the agent ahead through the environment. Boxes around the object represent 32 x 32 patches recognized as containing an object.

4.4.2 Auditory Input

While vision is processed at every time step, auditory stimuli only come once in a while. These take the form of a command from an outside person instructing the agent to perform a certain behavior. They are received as a sequence of phonemes, defined by certain auditory features. They consist of multiple words, separated by /stop/ sounds. A command sentence is taken to have ended when there are two /stop/ sounds in succession.

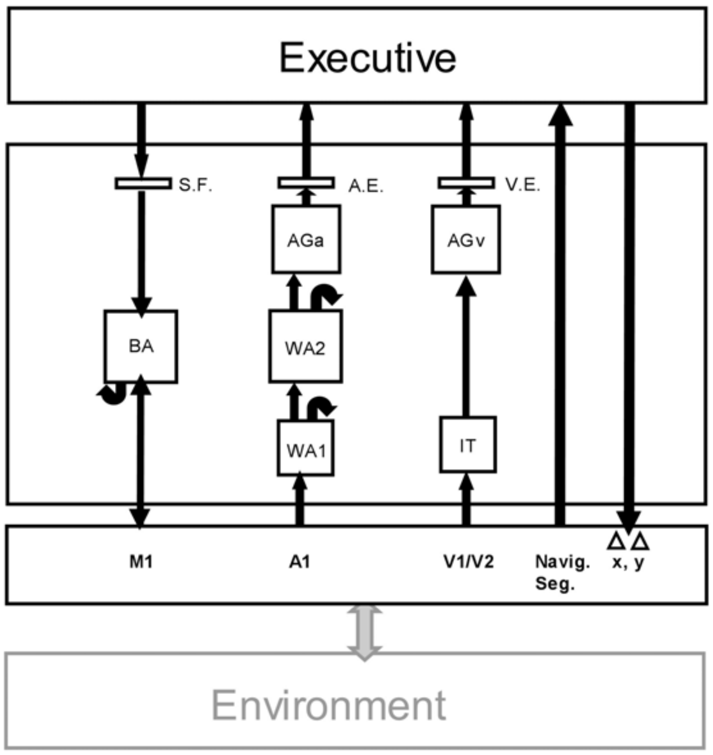


Figure 4.8. Expanded view of the agent's cognitive network. Object images are processed, encoded, and converted into features here. Sequences of phoneme input are also processed and encoded, first word by word (in WA1) and then into multiple word representations (in WA2). This representation is then converted into features of the sentence as well. Both visual and auditory features are passed to the executive control to influence agent decisions. In the opposite direction, phonemes are also produced from a set of concepts the agent has decided to state and sent through BA to M1 to be spoken. S.F., A.E., and V.E. stand for speech features, verbal encoding, and visual encoding, respectively.

The values of input region A1 change throughout the sequence of phonemes and stops constituting a command, moving from one phoneme to the next. It connects to a region called WA1, which produces a representation of a single word based on the present phoneme and the phonemes that preceded it. This learns with all the possible commands as input, creating a multi-winner self-organizing map. When a /stop/ sound is encountered, information is then passed to a region called WA2, which produces a representation of the entire command sentence. Connections into WA2 are also trained using a multi-winner self-organizing map.

This representation then acts as input into a hidden layer called AGa and an output layer encoding the significant features of the command ("was I told to find something?", "was I told to patrol an area?", etc.). As with the visual system, the WA2→AGa→Output connections are trained with resilient error-backpropagation. Figure 4.8 depicts this pathway, as well as the visual pathway.

4.4.3 The Executive Control

This control mechanism operates under a set of simple rules depending on what commands have been issued, what it sees at present, what it remembers seeing, and its direction and velocity. At each time step, it performs the series of actions and state changes described below. In addition to memories and current stimuli, the agent also maintains an awareness of the general layout of the city in terms of its districts. That is to say, the agent knows where the different districts are located, and which district it is presently in, if it is in the city.

First, the control mechanism reacts to the "where" pathway, reflexively turning in response to whatever obstacles are visible. Obstacles include objects, buildings, and water. If each of the judgments of what is in front of the agent are obstacles, then the agent makes a 30 degree turn to the left, but does not move forward, a response that prevents it from crashing headlong into the obstacle. It is consistent in the decision to make a left turn. This means that it does not continually get stuck in a loop around a set of obstacles where at one point it will turn left and then change its mind in a sequence of turns and decide to turn right, gaining no improvement. If some of the judgments are areas free of obstacles, then the agent adjusts its angle of movement to an average of the locations that are not obstructed, to a maximum of 15 degrees either left or right.

The agent then checks its command memories (including commands that are newly added) for any conditions that are met. The command verbs include “go…”, “patrol…”, “find…”, and “evade.” When told to go to a location, either a district or a terrain type, it will follow this command until it reaches that area. When that happens, it abandons that command. When told to patrol a location, the agent does essentially the same thing, except that it will then maintain the memory of the command and randomly move about the specified district or terrain type. In either of these commands (“go” and “patrol”), the agent uses waypoints, which override the navigation behavior. When told to find an object or a certain condition of objects (e.g. tanks without infantry), the agent will announce the object it sees whenever the conditions are met in its visual field. When told to evade an object and the object is visible, the agent will announce the object it sees, update its orientation angle or determine its path away from the offending object, and remember the object’s location for a period of time. When active, the evasive behavior overrides any other movement command. How the particular behaviors are determined is described in Section 4.5.

The agent then will update its orientation angle to avoid the closest remembered object it was instructed to evade if the object is too close. If there are no such memories, then the agent’s orientation angle remains unaffected. In this list of obstacle memories, old memories are removed if the capacity is full or, more commonly, after a time limit.

Finally, the agent’s orientation angle is adjusted so that it will not take the agent outside the allowed bounds of the simulation.

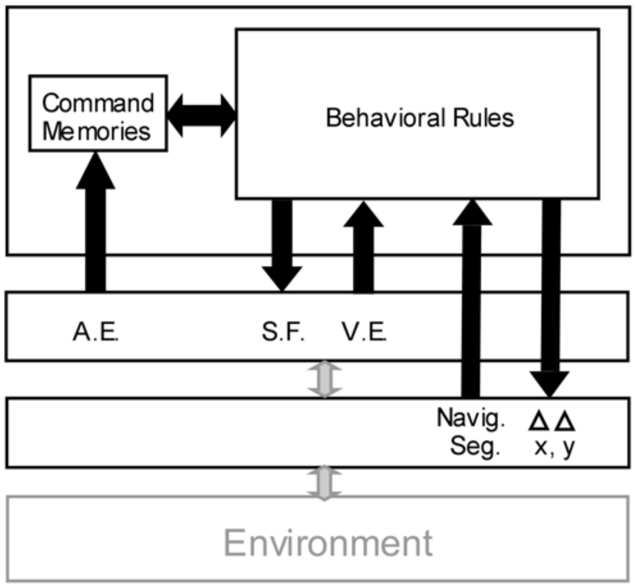


Figure 4.9. Expanded view of the agent’s executive control. The command memories are a list of all the auditory stimuli passed into the executive control via the cognitive network that have not become obsolete yet. Depending on these memories, along with current visual stimuli and obstacles present in the reflex vision, the behavioral rules determine both the movement of the agent, any spoken output the agent would make, and also which command memories are obsolete, or accomplished, and may be removed.

4.4.4 Movement Adjustment

Once the agent has finished updating its states according to its rules, it then moves and/or changes its angle of orientation. Its decision is based on what obstacles immediately impede it and what commands it has been issued. If each of the self-organizing map networks that govern navigation return an obstacle (meaning the agent’s field of view is completely blocked), then the agent will only make a turn and will not update its x and y position coordinates, so as to avoid running into an obstacle. If only some of these self-organizing map networks return an obstacle, then the agent will update its x and y position coordinates and make the appropriate turn. However, when agents are following commands, they place waypoints in their environment to guide their movement. The placement of these waypoints is based on vision and memory. At any point when the agent makes any updates to its internal state or interaction with the environment, it has access to the following information:

1) Agent's position in space
2) Agent's velocity vector
3) Buildings and terrain features it sees presently
4) Objects in the environment it sees presently
5) Buildings, terrain features or objects it remembers seeing
6) Locations of city districts (e.g. downtown district, industrial district, etc.)

The agent may also have access to a working memory of the local area or the accumulated memories over the entire environment. These will be discussed below.

4.4.5 Spoken Output

When the agent is required to speak, it produces a vector of relevant features that define the desired output statement. These features represent the specific concept that needs to be stated. At present, since the agent is simply observing the environment, the concepts are very simple, either an object or a collection of objects or a building type.

This vector acts as input to a region called BA, which in turn connects to an output region and has recurrent connections to itself. This is because the output is not static, but instead is a sequence of phonemes and /stop/s (much like the auditory input, though described by production features rather than auditory features). BA's self-connections allow it to remember its previous state and therefore generate novel output as new phonemes are produced. Additionally, the output region connects back to BA. Individual words are separated by /stop/ phonemes. Production of phonemes continues until there are two /stop/ designators in succession, signifying the end of the multiple word phrase. The Input$\rightarrow$BA$\rightarrow$Output pathway is trained using resilient error-backpropagation, and the self-connections in BA also experience this training.

4.4.6 Details of Neural Network Architecture

This section gives the details of how the neural networks of the agent are trained on the environment. First, there is a description of the structure of the different modules, or regions, of the network as well as a description of the features incoming commands and viewed objects are trained on. Following this is a description of the dynamics of the neural networks for each of the following paths: visual object input, verbal input, and spoken output.

4.4.6.1 High-Level Description of Modular Structure

The following is a bulleted description of the different neural network regions of the agent. Figures 4.6 and 4.8 display the connections between the following modules in the Cognitive and Sensorimotor systems.

Verbal Input Pathway

- A1: 34 phoneme features (e.g. liquid?, voiced?, etc.)

- WA1: multi-winner self-organizing map (accumulates input from A1 and self-state until /stop/ phoneme received).
- WA2: multi-winner self-organizing map (accumulates input from WA1 and self-state until two /stop/ phonemes appear in succession in A1).
- AGa: Hidden units trained with resilient error-backpropagation in WA2→AGa→Audio Encoding
- Audio Encoding: 34 on/off features of sentence input including the following categories
    - *Verb*: Go, Go to, Patrol, Find, Evade
    - *Direction*: North, South, East, West
    - *Location*: City, Market District, Uptown District, Downtown District, Slum District, Westside District, Southside District, Industrial District, Dock District, Historic District
    - *Adjective*: Our, Enemy
    - *Object*: Tanks, Infantry, Missile Launchers, Man, Woman, Car, Truck, Van, Bus, Dumpster
    - *Absence of Object*: Without Tanks, Without Infantry
    - *Modifier*: With Stingers

Visual Object Input Pathway

- V1: 50 x 50 pixel image
- IT: multi-winner self-organizing map
- AGv: Hidden units trained with resilient error-backpropagation in IT→AGv→Visual Encoding

- Visual Encoding: 16 on/off features of picture input including the following categories:
    - *Alignment*: Ours, Enemy, Neutral
    - *People*: People?
        - *Type*: Infantry, Man, Woman
        - *Modifier*: With Stingers
    - *Vehicle*: Vehicle?
        - *Type*: Tank, Car, Truck, Van, Bus
    - *Other*: Missile Launcher, Dumpster

Spoken Output Pathway

- Speech Features: 28 on/off speech features including the following categories:
    - *Action*: Location Arrival, Found Object, Evading Object
    - *Locations*: City, Market District, Uptown District, Downtown District, Slum District, Westside District, Southside District, Industrial District, Dock District, Historic District
    - *Adjective*: Our, Enemy
    - *Object*: Tanks, Infantry, Missile Launchers, Man, Woman, Car, Truck, Van, Bus, Dumpster
    - *Absence of Object*: Without Tanks, Without Infantry
    - *Modifier*: With Stingers
- BA: Hidden units trained in the path Speech Features$\rightarrow$BA$\rightarrow$M1 with resilient error-backpropagation. It receives feedback from M1 and self-state information
- M1: 20 output motor features for producing the desired phonemes.

4.4.6.2 Network Dynamics

The following is a description of the dynamics of the agent's neural networks and a description of how each pathway learns. Figures 4.5, 4.6, and 4.8 serve as guides for the neural network structure and paths.

Visual Object Input: Architecture and Activity Dynamics

V1/V2 [50x50] acts as the input region to the visual pathway of neural networks that interpret objects. A 50x50-pixel .jpg image is loaded into the region before the activation dynamics begin and the image is processed.

IT [15x20 neural elements] receives activity from V1/V2, fully connected. IT nodes follow the activation rule:

$$sum_j = \alpha_1 \sum (in_i \cdot w_{ji})$$

where *sum* is the total input to node *j* of IT, *in* is the activation of a node of V1/V2, w_{ji} is the weight between nodes *j* and *i*, and gain parameter α_1 is 2.4.

The IT area forms a multi-winner self-organizing map. A winner within each region of connectivity (a 9x9 neighborhood) retains its value, and other "non-winner" nodes are assigned values as follows:

$$a_j = a_{winner} \cdot \gamma^{|\text{distance to neaerest winner}|}, \gamma = 0.2773 \big/ (1 + e^{(\text{epoch}/500 - 0.1334)/0.796})$$

The value a_{winner} is the closest winner within a node's connectivity range, and the gamma function represents the drop-off function that was dependent on the training epoch during training, but once training is complete becomes fixed to the following value:

$$\gamma = 0.2773 \big/ (1 + e^{0.8367})$$

The AG [15x20] receives activity from IT in this path, fully connected. AG nodes follow the activation rule:

$$a_k = \frac{1}{(1 + e^{-g(\sum(0.2 \cdot a_j \cdot w_{kj}) + bias_k)})}$$

where a_j is the activation of node j in IT, $bias_k$ is the bias value for AG node a_k, and logistic function gain g is 4.

The VE (visual encoding) region [1x16] receives activity from AG, fully connected. Similar to the AG, VE nodes follow the activation rule:

$$a_m = \frac{1}{(1 + e^{-g(\sum(0.2 \cdot a_k \cdot w_{mk}) + bias_m)})}$$

where a_k is the activation of node k in AG, $bias_m$ is the bias value for VE node a_m, and logistic function gain g is 4.

Visual Object Input: Learning Phase I

The learning rule for IT afferent connections is:

$$\Delta w_{ji} = \mu \cdot a_j \cdot in_i$$

where $\mu = .5$ and the other variables are as defined above. All incoming weight vectors to neural elements in IT are normalized after each learning step.

Visual Object Input: Learning Phase II

All connection weights throughout the path other than those learned during Phase I are randomly initialized between 0 and 1.

Learning from IT via AG to VE occurs through RPROP (resilient error-backpropagation) with delta-values as follows:

$$\delta_m = a_m \cdot (1 - a_m) \cdot (a_m^{target} - a_m)$$
$$\delta_k = a_k \cdot (1 - a_k) \cdot \sum(\delta_m \cdot w_{mk})$$

The measure of its correctness is the proximity of an element in VE to its target after information has propagated through the entire network.

Verbal Input: Architecture and Activity Dynamics

A1 [1x34] acts as the input region to the verbal input processing pathway of neural networks. Each of the 34 input nodes represents a feature of a phoneme, either present or absent. These phoneme values change over the course of a single input, representing a string of incoming phonemes and /stop/ values (where there are no phoneme features present). When two /stop/ values occur in a row, the input stream has terminated. This amounts to inputs consisting of several words.

WA1 [15x20] receives activity from A1, fully connected. WA1 nodes follow the activation rule:

$$sum_j = \alpha_1 \sum(in_i \cdot w_{ji}) + \alpha_2 \sum(a_h^{stored} \cdot w_{jh})$$

where *sum* is the total input to node *j* of WA1, *in* is the activation of a node of A1, w_{ji} is the weight between nodes *j* and *i*, w_{jh} is the weight between nodes *i* and *h* in WA1, and gain parameters α_1 and α_2 are both 0.8. The stored *a* is the previous value held in that node.

As with the IT area, WA1 forms a multi-winner self-organizing map. A winner within each region of connectivity (a 9x9 neighborhood) retains its value, and other "non-winner" nodes are assigned values as follows:

$$a_j = a_{winner} \cdot \gamma^{|\text{distance to neaerest winner}|}, \gamma = 0.2773/(1 + e^{(\text{epoch}/1000 - 0.1334)/0.796})$$

The value a_{winner} is the closest winner within a node's connectivity range, and the gamma function represents the drop-off function that was dependent on the training epoch during training, but once training is complete becomes fixed to the following value:

$$\gamma = 0.2773 \big/ (1 + e^{-0.4196})$$

If the input is a /stop/ (i.e. an input where all features are 0), then WA2 will then be updated based on the values in WA1.

WA2 [25x30] receives activity from WA1, fully connected. WA2 nodes follow the activation rule:

$$a_k = \alpha_1 \sum (a_j \cdot w_{kj}) + \alpha_2 \sum (a_h^{stored} \cdot w_{kh})$$

where a_k and a_h are nodes of WA2, a_j is a node of WA1, w_{kj} is the weight between a_k and a_j, w_{kh} is the weight between a_k and a_h, and gain parameters α_1 and α_2 are both 0.8. The stored a is the previous value held in that node.

As with WA1, WA2 forms a multi-winner self-organizing map. Its rules and settings are the same as WA1's. WA2 will pass on its data to the AG when two /stop/ phonemes have occurred in succession.

The AG receives activity from WA2 in this path, fully connected. AG nodes follow the activation rule:

$$a_m = \frac{1}{(1 + e^{-g(\sum (0.2 \cdot a_k \cdot w_{mk}) + bias_m)})}$$

where a_k is the activation of node k in WA2, $bias_m$ is the bias value for AG node a_m, and logistic function gain g is 4.

The AE (audio encoding) region [1x34] receives activity from AG, fully connected. Similar to the AG, AE nodes follow the activation rule:

$$a_n = \frac{1}{(1+e^{-g(\sum(0.2\cdot a_m \cdot w_{nm})+bias_n)})}$$

where a_m is the activation of node m in AG, $bias_n$ is the bias value for AE node a_n, and logistic function gain g is 4.

Verbal Input: Learning Phase I

The learning rule for WA1 afferent connections is:

$$\begin{aligned}
\Delta w_{ji} &= \mu \cdot a_j \cdot in_i \\
\Delta w_{jh} &= \eta \cdot a_h^{t-1} \cdot (a_j^t - a_j^{t-1}) \\
\mu &= 0.3225 \Big/ \left(1 + e^{(epoch/1000-0.4747)/0.0213}\right) \\
\eta &= 0.6307 \Big/ \left(1 + e^{(epoch/1000-0.8561)/0.0217}\right)
\end{aligned}$$

where all other terms are as defined above. The variable t is the current phoneme number in the input sequence (for the first phoneme, a_j^0 is 0). All incoming weight vectors to neural elements in WA1 are normalized after each learning step.

Verbal Input: Learning Phase II

The learning rule for WA2 afferent connections is:

$$\begin{aligned}
\Delta w_{kj} &= \mu \cdot a_k \cdot a_j \\
\Delta w_{kh} &= \eta \cdot a_h^{t-1} \cdot (a_k^t - a_k^{t-1}) \\
\mu &= 0.3225 \Big/ \left(1 + e^{(epoch/1000-0.4747)/0.0213}\right) \\
\eta &= 0.6307 \Big/ \left(1 + e^{(epoch/1000-0.8561)/0.0217}\right)
\end{aligned}$$

where all other terms are as defined above. The variable t is the current phoneme number in the input sequence (for the first phoneme, a_k^0 is 0). All incoming weight vectors to neural elements in WA2 are normalized after each learning step.

Verbal Input: Learning Phase III

All connection weights throughout the path other than those learned during Phases I and II are randomly initialized between 0 and 1.

Learning from WA2 via AG to AE occurs through RPROP with delta-values as follows:

$$\delta_n = a_n \cdot (1-a_n) \cdot (a_n^{target} - a_n)$$
$$\delta_m = a_m \cdot (1-a_m) \cdot \sum \left(\delta_n \cdot w_{nm}\right)$$

The measure of its correctness is the proximity of an element in AE to its target after information has propagated through the entire network.

Spoken Output: Architecture and Activity Dynamics

The SF (speech features) region [1x28] acts as the input region to the spoken output pathway of the neural networks. Each of the 28 input nodes represents a feature of the concept to be expressed in output as a string of phonemes and /stop/ values amounting to a multi-word statement, concluding with two /stop/ values.

BA [15x15] received activity from SF, fully connected. BA nodes follow the activation rules:

$$sum_j = \alpha_1 \sum (in_i \cdot w_{ji}) + \alpha_2 \sum (a_h^{t-1} \cdot w_{jh}) + \alpha_3 \sum (a_k \cdot inh) + \alpha_4 \sum (a_m \cdot exc) + \alpha_5 \sum (a_n^{t-1} \cdot w_{jn})$$

$$a_j = \frac{1}{(1+e^{-g(\sum(sum_j)+bias_j)})}$$

where a_j is the activation of node j in BA, $bias_j$ is the bias value for BA node a_m, and logistic function gain g is 4. Parameters α_1, α_2, α_3, α_4, α_5 are .5, .2, .3, .3, and .2, respectively, while *exc* and *inh* are 0.2 and -0.4, respectively.

The phoneme output region [1x20] receives activity from BA, fully connected. M1 nodes follow the activation rule:

$$a_n = \frac{1}{(1+e^{-g(\sum(a_j \cdot w_{nj})+bias_n)})}$$

where a_j is the activation of node j in BA, $bias_n$ is the bias value for M1 node a_n. and logistic function gain g is 4.

When two /stop/ phonemes are produced in a row, then the network ceases production.

4.5. Higher-level Command-based Decisions

4.5.1 Agent Understanding of the Environment

The agent exists in a three-dimensional environment. It is given commands and has knowledge of its global position, as well as the global outlines of the city and its districts, but the only information it receives from this environment comes from its vision window. The agent knows nothing initially of the layout of the streets, the buildings or what objects it will encounter, and has to judge that information by processing and segmenting the view snapshot of the 3D environment. This leaves the agent with several options of how to use the sequence of snapshots of the environment to navigate most efficiently. These options cover a spectrum of how much processing is done with the image sequences and how much memory the agent employs in its reactions to them.

In the rest of the chapter, three different variations of the agent are compared. In each case, the agent has a different amount or type of working memory, as follows. The three basic categories of movement behavior memory are termed: no memory (NM), local memory (LM), and cumulative memory (CM).

4.5.2 No Memory Movement

This movement strategy is named “no memory” movement because it only makes use of the 2-dimensional image of the present time step. It does not remember the previous images or their information, and it will not remember this image by the next time step. However, the agent still retains memories of commands, goal locations, dangerous positions, and waypoints. “No memory” here simply refers to the information the agent acquires about the structure of its environment through experience.

NM movement takes the information provided by a single 2-dimensional image of the 3-dimensional scene at any given time step and uses it to potentially generate a waypoint. It does this by using self-organizing maps to locate all the 32 by 32 pixel cells in the image considered to be streets. It then discards any of the street locations as aberrant if they are not contiguous with the lowest central cells, which are usually street cells. An agent will maintain and approach the waypoint until it is unable to move, or it is in the square of size 0.2 centered at the waypoint.

The agent has the option either to go straight ahead or to make a turn. If the angle of the agent’s location relative to where the agent wants to go is less than 0.01 away from 0 degrees, then the agent moves straight ahead. The waypoint is thus placed in the furthest street position in a straight line in front of the agent. Otherwise, if the angle is negative, the agent decides to make a right turn, and if the angle is positive, the agent decides to make a left turn. In either case, the agent looks for the street cell that is closest to the agent, but farthest in the direction (right or left) that the agent decides to go, where there is at least one intervening impassable cell between this street cell and the bottom of the screen. This implies a street is available at that point for the agent to turn onto and change its direction. Figure 4.10 demonstrates an example of this.

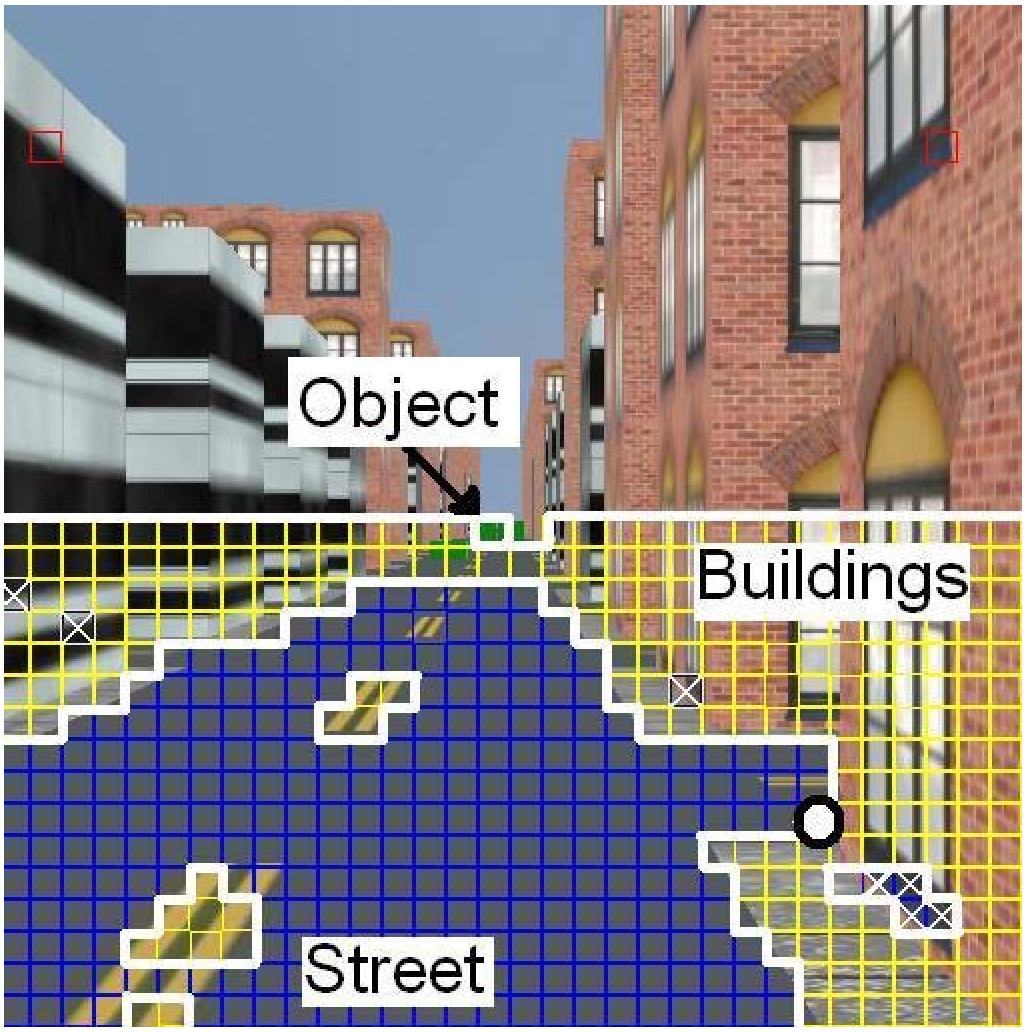


Figure 4.10. Agent's processing of the environmental view. Most cells that overlay areas of the screen correctly identify them (e.g. cells over streets are determined to be streets). Contiguous blocks of cells identified as the same have been outlined in white and labeled. There are scattered areas that remain unidentified and some that are misidentified. Street cells that are not contiguous with the cells are marked with an X, and are taken to be aberrations. Indeed, these tend to occur on buildings. Should the agent wish to make a right turn, the filled cell (marked by a black circle) would be where it would drop its waypoint according to the behavior rules.

Once a cell is chosen to be the location of a waypoint, the agent estimates its location in the environment. Using the same method of computing the location of objects described above, the agent is able to estimate the location in the environment of the central pixel of the cell and can establish a waypoint at that location.

Waypoints are not generated under certain circumstances. Should no street cells that fit the desired movement pattern be discovered in the processing, generating a waypoint is delayed until a later time step, and the movement at this time step will be purely reactive. Additionally, if a waypoint is already present, then the agent will not produce a new one. Finally, if the agent has achieved its goal (the evasion behavior has expired or the agent has reached or patrolled the desired region), then no waypoint is generated for that command.

4.5.3 Local Memory Movement

The advantage of the NM movement is that it requires no extra manipulation of the image data or any memory of what was seen at previous time steps. However, because of this, it frequently delays placing waypoints or places them in aberrant locations. Additionally, the estimation of locations in space from the viewpoint snapshot is not completely accurate, a condition which increases the problems of placing waypoints using this method. Many of these problems can be addressed by adding a memory of what is seen in the surrounding environment and converting some of the information into retained knowledge of the surrounding space for navigation. The LM movement strategy performs this function.

In addition to using the knowledge of the environment from the view window at every time step, when using this movement type, the agent also keeps a local memory (LM) of the area immediately surrounding it. This is a five by five two-dimensional overhead map designed as a cellular space, that is built by the agent as it moves. It is centered on the agent, but aligned with the global grid. Each cell is given a different value depending on what the agent has estimated exists in that area from the visual information.

As in creating waypoints in the NM movement, some conversion of snapshot information to a 2-dimensional area map is required. Pixel cells and their locations in the environment are computed the same way as in the NM method, but what is done with the information is different. For each pixel cell along the terrain level (approximately the lower half of the snapshot view), the location of the cell is determined relative to the agent. If this estimated location falls in a cell of the agent's LM, then it contributes to generating a memory of the environment surrounding the agent. A sample local map is depicted in Figure 4.11.

The LM cells are determined by the agent to be one of four possible categories. If a cell has not been seen yet, then it is labeled "unknown." If an agent is in the cell—or was in a location within the cell at a previous time step—then the location is set to "been there." Otherwise, the agent determines whether the cell is "passable" or "impassable," depending on the number of pixel cells it has seen corresponding to the LM cell that are determined to be streets or objects (signifying an area is passable) or that are determined to be buildings or water (signifying an area is impassable).

The counts of the number of pixel cells associated with the LM cell that are considered streets, objects, or buildings are accrued over many time steps. If the number of building pixel cells seen exceeds the number of object and street pixel cells for a particular LM cell, then that cell is considered impassable. If the number of street pixel cells exceeds the number of building pixel cells, then the corresponding LM cell is considered passable, and a waypoint can be placed there. However, if the agent has been in the location previously, then the LM cell will be set to “been there.” These counts and determinations shift as well, as the agent moves along the grid. In this way, when an agent moves into a cell it has recognized as “passable” in its LM (i.e. when it becomes a “been there” region), then all the information shifts to the appropriate LM cells, while some old information is lost to the LM for cells no longer in range, and new “unknown” areas appear in LM cells that are now close enough.

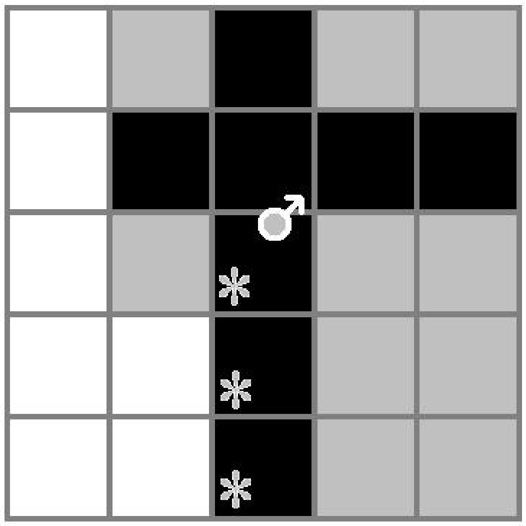


Figure 4.11. Representation of an example local memory map produced by an agent in the simulated environment, taking up a five by five grid. The agent is represented by the circle with a directional arrow in the central cell. Black cells are places estimated by the agent to be streets. Gray cells are places estimated by the agent to be buildings. White cells are unknown, meaning the agent has not acquired information about them. Black cells marked with an asterisk are areas the agent remembers passing through. As the agent moves, the center of its local memory map moves with it.

The 3-dimensional environment is not flat. Also, the process that converts the pixel of the snapshot of the 3D scene to an estimated 2-dimensional point on the map assumes that the point in the view is on the ground (an assumption that is made because there is no accurate and trivial way to judge the position and height of buildings). As a result, the agent will sometimes mistake “passable” areas as “impassable,” if a building is obstructing the view of the passable area. However, because there is a bias toward areas being passable in the summation of the pixel cells for their LM cells, the agent is able to correct the misjudgment as it draws closer to the cell and sees more of the cell in question.

With this information, the process of creating waypoints is somewhat easier. Because the LM tells us exactly which locations in the environment are “passable” and which are not, we can position a waypoint in the middle of an adjacent passable LM cell with reasonable certainty that the agent can head to it. An exception to this case is when an area has mistakenly been judged passable, which is a rare occurrence given that the accumulated sums of pixel cells over time tends to make the judgments with regard to streets fairly accurate. Another exception occurs when the waypoint is too close to an object. In this latter case, if there is an object known to be in the cell where we wish to put the waypoint, we simply drop the waypoint in the cell at some distance away from the object, so it will not interfere.

The different possibilities of the cell states of the LM allow for an order of preference in determining which direction to take in the case of multiple options. An agent will tend to opt for a “passable” area first. If none is available, it checks possible “unknown” adjacent cells to see if any of them are in fact “passable.” If it has no other options, it will choose a “been there” cell. This gives the agent the impetus to explore areas it has not seen, a tendency that is useful for acquiring information about the environment.

4.5.4 Cumulative Memory Movement

CM movement makes use of the same techniques that appear in the LM movement, but it adds the ability for the agent to use a global map it can construct from its cumulative LM states over the course of a simulation to generate a sequence of planned movements from its present location all the way to the final goal. This method makes use of much more information than both the previous methods.

Like the LM, the global map is a cellular space aligned with the world’s grid. Unlike the LM, it is calibrated to be the same size as the world. Also, it is not centered on the agent, but the agent moves through it. This means the agent, which knows its own global position, is able to compute the appropriate global map grid cell from any LM cell. As LM is adjusted from “unknown” to estimated states, the corresponding positions on the global map are also updated. The global map only tracks positions as “unknown,” “passable,” “impassable,” “dangerous,” and “dangerous & passable,” which is set when the agent estimates a hostile object in a certain position. An example map built by the agent is depicted in Figure 4.12.

When the agent follows a command that requires movement, an appropriate movement is plotted out as a queue of waypoints across the map, in both known and unknown areas. This queue is computed using an A* search algorithm where the search space is the agent’s grid map, the start is the agent’s current cell, and the goal is either a specific cell or any cell in a specified location. Areas known to be impassable are considered invalid, and known passable areas do not get priority over unknown areas, because there may be more direct routes that pass through unknown territory. Therefore, the agent is not willing to discount paths through such territory.

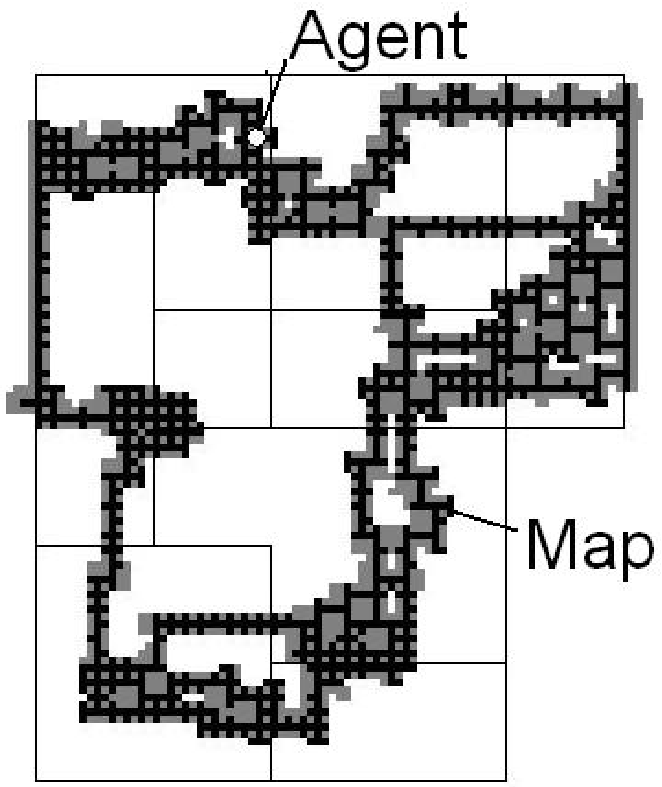


Figure 4.12. Representation of an example cumulative memory map produced by an agent after touring parts of the simulated environment. Black cells are places estimated by the agent to be streets. Gray cells are places estimated by the agent to be buildings. White cells are unknown, meaning the agent has not acquired information about them. The map also includes lines denoting the district boundaries of the city, which are known to the agent.

This queue of waypoints is never completely fixed. If the agent discovers an obstruction in an area that was previously unknown, it will adjust the waypoint strategy to fit the new knowledge of the map. Additionally, if a hostile object has emerged in a previously passable territory, the corresponding global map cell will be set to “dangerous.” If the waypoint path goes through this cell, it will be adjusted to give it a safe berth, depending on the sort of danger the object poses. Cells in the chart between the obstacle’s computed location and the agent are set to a type called “dangerous & passable,” since they are cells in the line of sight of the dangerous obstacle. In this case, there are no obstructions, but there is a risk passing through those locations.

One caveat that bears mentioning is that the cellular space in the chart happens to line up with the environment. This was done for the sake of simplification, but it is certainly possible to have a much finer grain cellular space chart that allows for a richer environment where buildings and streets are not aligned with the cellular space, but can still be closely and accurately outlined in the agent’s local and cumulative memories.

4.5.5 Movement Mechanics

Each of the command types that influences movement (go, go to, patrol, and evade) requires a different set of rules for operation, and most of these commands require a further differentiated set of rules depending on which of the movement behaviors with their varying degrees of environmental memory are implemented. The command “go” is the same for every strategy set, since it only requires a general momentum in a particular direction. For “go to” and “patrol,” however, knowledge of the environment is more useful as recognizing specific turns becomes key to succeeding. Similarly, this is also true for the “evade” command where a greater knowledge of the environment will allow the agent to more successfully sidestep hostile entities with greater success.

4.5.5.1 Mechanics of "Go"

When an agent is following a "go" command that specifies a direction rather than a region (e.g. "go west"), it moves following the standard rules of reflexive navigation described above by using the self-organizing maps that recognize obstacles in the closest visible regions. Yet, it is given an additional acceleration in the direction specified in the command. This results in an agent that will tend to move in that direction, when there are no intervening obstructions. This command has the same mechanics regardless of the movement behavior employed, since the command is essentially independent of the environment, whereas the other commands are all relative to some environmental feature.

4.5.5.2 Mechanics of "Go to"

Unlike the simpler "go" commands, a command to "go to" a specific district does vary depending on which movement behavior is implemented. These commands direct an agent to head to a specific location on the map. In this case, the agent already possesses both knowledge of its global position and the location in space where it needs to head. Because of how the commands are received, some global knowledge has to be assumed, even in the case where the agent does not preserve memories of what it has seen in previous time steps.

For the NM movement, the agent follows a system of waypoints it generates with the ultimate destination being the specified region, and when it reaches that region, it states it has accomplished the command. If a waypoint is present, an agent is drawn towards it, ignoring any reflexive navigation and stopping only if obstructed physically.

When it is sufficiently close (described above) to the waypoint, the waypoint is removed and the agent will turn in an angle that will point it in the direction of the ultimate goal. This helps to align the agent in the environment, so it can more accurately judge where to set waypoints.

If a waypoint is not present, then there is potential for a new one to be added. The angle of the agent's view vector relative to the desired goal position is compared. If the angle is very close to zero, then the agent is heading in the correct direction toward the position it needs to go, and a waypoint is placed at the furthest recognized passable position in view that is straight ahead. If the angle is sufficiently greater than zero, the agent needs to make a left turn. If the angle is greater than 90 degrees, then the agent will simply make a left turn in the environment, since its goal area is behind it. Otherwise, it will attempt to find a valid waypoint position for a left turn as described above. If the angle is less than zero, then a similar set of rules apply, but just for right turns as opposed to left turns.

For the LM movement, the agent will again generate waypoints one at a time. The choice of waypoint position now depends on the agent's local memory, as opposed to just what it is looking at presently. Again, if a waypoint is present, the agent will approach it until it is sufficiently close, and it will turn toward its goal area when the waypoint is reached.

When a waypoint is absent, a new one will be generated based on what the agent remembers seeing around itself. The agent has an order of preference that it always follows. Its first choice is always in a direction that leads toward the goal area. If the adjacent local memory cell in that direction is labeled "passable," then it will set a waypoint in the center of the cell in that direction. If the direction is labeled “unknown,” the agent sets a waypoint in the direction of that cell, but right next to itself. This allows the agent to turn toward the unknown cell and gain information about it for the next time step, when it will likely be able to make a judgment as to whether it is “passable” or “impassable.” Should the agent remember having been in the cell or the cell be labeled “impassable,” the agent then will opt for a second or third choice. These choices will not directly approach the goal area, but neither do they back away. Instead, they act in a side-stepping manner, a tactic useful when the agent is obstructed by a building or a passable area containing too many or dangerous objects. Again, known “passable” areas are given priority, and if there are none, the agent will check “unknown” memory cells. If none of these options are available, the agent will select the cell where it has been (labeled “been there”) that has the highest preference for a waypoint. This means the agent has a strong inclination against backtracking. In the rare case where none of these strategies have produced a successful waypoint, the agent will then be allowed to wander reflexively, so it can find a way out of being stuck.

For CM movement, the agent produces a sequence of waypoints, one per map cell, in the shortest, safe and unobstructed path to the goal area. Waypoints are also potentially placed in areas “unknown” on the map if the shortest path requires this, but a known “passable” path of equal length will be equally considered. As the agent moves from waypoint to waypoint, its local memory may revise the map if it misjudged an area earlier (e.g. believed a “passable” area was “impassable”). This potential revision could lead to the waypoint sequence being updated, should it produce a shorter path. Additionally, if unknown dangerous objects along the path are discovered, the agent will adjust its path through the map to avoid them as it sees them. Finally, if the agent discovers “unknown” areas feature obstructions, the path is altered to accommodate this new knowledge. As before, the waypoints are eliminated when the agent is sufficiently close and the next waypoint at the head of the queue is the agent’s new target, until it has exhausted waypoints and is in the desired region. Unknown and passable cells are given equal costs when computing a path, while cells that are dangerous and passable cost six times as much (to deter the agent from taking that path if a safer one is available).

In some cases for the CM and LM strategies, when the goal region is a single cell, the agent will misinterpret the territory for that cell from a distance and believe it to be impassable. This can cause problems when searching for a path or making a decision to turn into the region. Thus the goal region, if it is a single cell in the environment, is always assumed to be passable.

4.5.5.3 Mechanics of “Patrol”

Unlike the “go” commands, patrol requires some map-memory at every level to function at all. The implication of the command is that an agent can travel or see as much of the specified region as possible. The metric used here is the number of environmental cells the agent either has been in, or has been in the agent’s field of vision up to a distance of 2 environmental cells away. This means that it would be nearly impossible, without maintaining some larger memory of what it has seen, to complete this task successfully. Therefore, an agent, when using the NM or LM strategies, does have access to a larger map that it is converting from “unknown” to various known states. However, we continue to limit its knowledge of the environment as before when it comes to making choices for navigation with these strategies.

In the NM movement strategy for the “patrol” command, there are two possible behaviors depending on agent position. If the agent is not in the desired region, then waypoints are generated much in the same way that the “go to” command generates them for this strategy. If the agent is in the desired region, then the agent is allowed to explore using nothing more than reflexive movement. Unable to remember what it has seen, the agent will move more or less randomly, albeit determined in part by the obstacles that appear in its view. If it happens to wander outside the desired region, then waypoints once again direct it back to the region.

In the LM movement strategy for the “patrol” command, there are again two possible behaviors. If the agent is not in the desired region, then waypoints are generated to direct it to the region, as in the “go to” command. If the agent is in the desired region, then the agent follows a set of rules that determine which of the places preserved in its local memory it will move toward. It has a preference for placing its next waypoint in the direction it is facing, but if it has been there, if it is impassable, or if it is blocked by a dangerous object, then the agent will seek another path. First priority is assigned to “passable” or “unknown” regions to the right or left; second priority is assigned to the area behind the agent. If there are no “passable” or “unknown” regions in those directions, then the agent sets a waypoint in the most preferred area which is denoted as “been there.”

In the CM movement strategy for the “patrol” command, waypoints are generated much as they are in the “go to” command, and the goal state changes much as it does in the NM movement. As long as there are still unknown cells in the area to patrol, the agent will compute a path of waypoints to the closest unknown cell. The patrol terminates, when there is no longer any unknown cells left in that area.

4.5.5.4 Mechanics of "Evade"

The agent responses to the "evade" command and corresponding visual stimuli vary greatly depending on its memory of the environment. As in some previous commands, some memory is required in all cases for success. In this case, the agent invariably has at least temporary access to the estimated location in the city of the viewed object to be evaded. Additionally, the agent has a special "fear" behavior that is activated temporarily as well, in which all other commands for movement are put on hold, while it attempts to escape or side-step the object it saw. The agent will evade a visible or remembered obstacle as long as it is sufficiently close to it, but the radius for proximity is typically quite high (i.e. 20 cells). Obstacles are assumed to be dangerous as long as they remain in line of sight and not too far away.

In the NM movement strategy, the agent simply moves in a velocity vector that takes it in a direction opposite to the direction of the evaded object, provided that the object is in the range considered close enough to be dangerous. The agent continues to move in this direction, allowing for reflexive navigation when obstructed, until the time limit for the memory is reached.

In the LM movement strategy, the agent has access to a memory of its immediate surroundings, so it is able to achieve a more complicated behavior. This again involves setting waypoints based on preference. The agent's first choice is to side-step the path with the dangerous object. If those directions are “impassable” or “been there,” then the agent will opt for the direction that will backtrack. Lowest preference is given the areas that will take it closer to the dangerous object. If none of these areas are “unknown” or “passable,” then the agent will select the most favored direction labeled “been there” to set its waypoint. Once the time limit is reached for the “fear” behavior and the obstacle memory, the agent again moves normally, having moved to a potentially new path.

In the CM movement strategy, the agent has access to a memory of the entire map. Therefore, it has a more accurate picture of the environment surrounding it. From this map, the agent creates a smaller map of the area immediately surrounding it, and it uses this information to create waypoints in order to evade the obstacle. This strategy functions much as the LM movement strategy for “evade,” but makes use of the chart information rather than the less comprehensive local memory.

4.6 Results

4.6.1 Experimental Methods

In the following results, the training of the agent’s neural network architecture is examined first. The network’s self-organizing maps for interpreting an image sequence is trained on 2000 random images of the environment. For the verbal input, the multi-winner self-organizing maps are trained 100 iterations for each of the first two phases (the encoding of words and the encoding of sentences). The third phase of the verbal input is trained for 1000 iterations using resilient error-backpropagation [Riedmiller, 1993]. For the visual object input, the multi-winner self-organizing maps of the first phase are trained for 100 iterations. The second phase of the visual input is trained for

1000 iterations using resilient error-backpropagation. For the spoken output, the networks are trained for 1000 iterations using resilient error-backpropagation.

What follows this is a series of scenarios where the success and speed of the agent using the different working memory strategies (NM, LM, and CM) are collected, measured, and compared. For each memory strategy, the agent is given several different tasks or scenarios. The different scenarios examined include three cases: (1) a case where the agent is told to go to a single location, (2) a case where an agent is told to go to a district and look for an object there and then return to a home base, and (3) a case where an agent is told to go to multiple different locations.

The above scenarios are run twenty times for each memory strategy. Mean completion times, along with standard deviations and t-tests for significance, are computed for the scenarios. Though individual runs within a set of twenty could have different initial conditions, the same set of twenty was used to compare each of the memory strategies.

For the case where the agent is told to go to a single location, the initial conditions for all twenty runs are the same. The maximum time limit is 1000 time steps for this scenario. For the case where the agent is told to find an object while patrolling a district, the initial position is different for each individual run. The maximum time limit of this scenario is 8000 time steps. For the case where the agent is told to go to a sequence of locations, the initial position is consistent for all runs, but the order of the locations to be visited is different for each individual run. The maximum time limit of this scenario is 20,000 time steps.

The maximum time steps were chosen for these scenarios so they would have a sufficient amount of time to complete the scenarios, which could have very different lengths from one another, but also prevent them from running too long since individual runs take a long time to complete.

In addition to the above scenarios, the improvement due to creating a map of the environment in the CM strategy is examined in a scenario where the agent is required to repeatedly tour a series of goal locations throughout the city. This scenario is comparable to the type of scenario used in the previous chapter.

4.6.2 Neural Network Results

Each of the network paths are trained separately, and in different numbers of phases. The visual input path has two components. First, it involves processing of the raw image, breaking it down into patches, individually recognized as buildings, roads, or objects. The success of the learning along this path is examined in the scenarios following this section. In these, information taken from the sequence of first-person perspective images can be used to build the working memory of the agent and influence its behavior in the environment. The other visual input path is for object recognition. The first phase of learning produces a unique representation for each of the different images. This means that there is now a unique pattern for each object to be used as input for the second phase. After 1000 iterations of error-backpropagation, these unqiue patterns are trained on their associated features, and the root mean squared error for the learned output to their target is 7.77×10^{-17}.

The verbal path has three phases of learning. The first phase produces a unique representation for each word in the agent vocabulary with multi-winner self-organizing maps. The second phase uses a sequence of these to produce a unique representation of a sentence using another multi-winner self-organizing map. Figure 4.13 depicts a typical encoding for different words (the self-organizing map WA1) and for a sentence consisting of those words (the self-organizing map WA2). The unique patterns produced for WA2 are used as input and trained on their associated features. The root mean squared error for the learned output to their target is 0.0236.

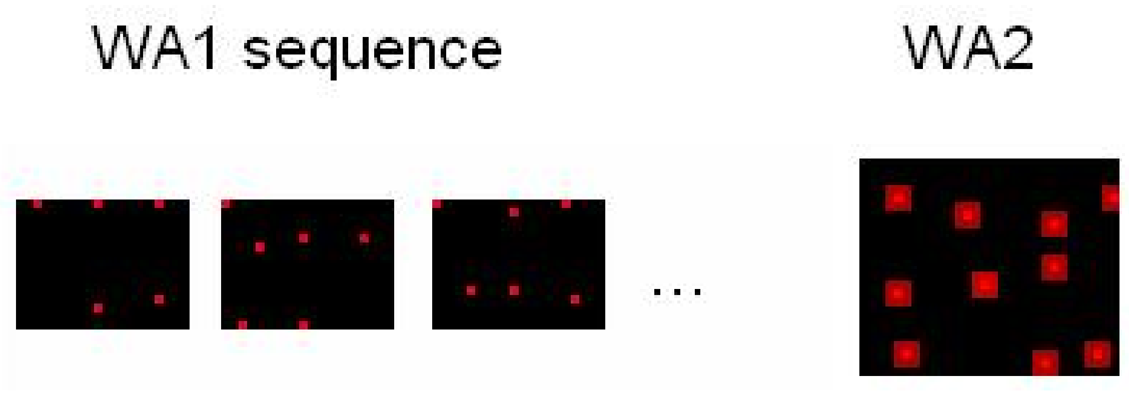


Figure 4.13. Depiction of typical unique patterns created during the first phase of verbal input training for region WA1. These are used to produce a unique sentence pattern for region WA2 in a second phase of learning with a multi-winner self-organizing map.

For the spoken output path, the features of the desired sentence are each trained to produce a different sequence of phonemes (forming a sentence of words broken by /stop/ markers) for 1000 iterations using resilient error-backpropagation. When these sequences of motor features are matched to the closest phonemes, the fraction of erroneous phonemes produced by the network over all sentences is 0.0169.

4.6.3 Going to a Location

In this scenario, the agent is assigned the relatively simple task of going to a specific location that is nearby. This scenario is performed twenty times for each strategy under identical initial conditions. The only feature varied is the memory movement strategy the agent uses (either NM, LM, or CM). Because the scenario is not guaranteed to terminate, the maximum time limit of 1000 time steps is imposed for this task.

Figure 4.14 displays the mean time taken to complete the task for each memory strategy. Note that CM and LM take a comparable amount of time on average. NM requires several times as many time steps. Additionally, the number of time steps the agent was "stuck" was recorded. An agent is considered stuck when it cannot move because its next intended step forward is blocked by an environmental obstacle. This value was 0 for CM and LM agents, but the NM strategy has a mean time steps where an agent is "stuck" of 36.15.

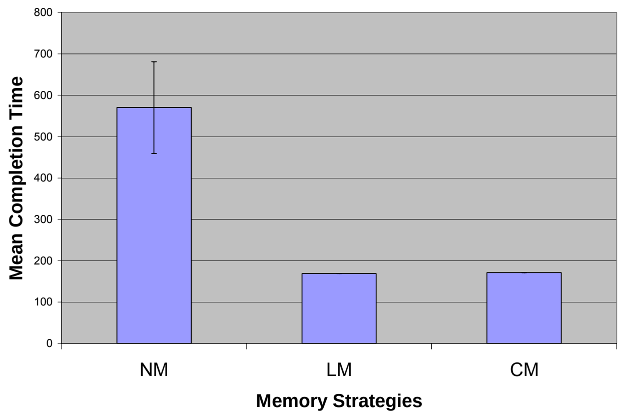


Figure 4.14. Results for the command of an agent instructed to go to a specific nearby location. The error bars represent a 95% confidence interval. This interval is very small for the LM and CM strategies.

4.6.4 Looking for an Object in an Area

In this scenario, the agent's starting location, or "home base," is situated at an arbitrary position outside, but not necessarily close to, the Historic District. It is then given the commands "Patrol the Historic District" and then "Find a van." If the agent eventually sees the van in the Historic District, it informs its commander, who then tells the agent to return to its home base. Because the scenario is not guaranteed to terminate, the agent has a maximum of 8000 time steps to complete the tasks. This number is increased from the previous scenario due to the added distances and complexity. These tests are run twenty times for each memory strategy.

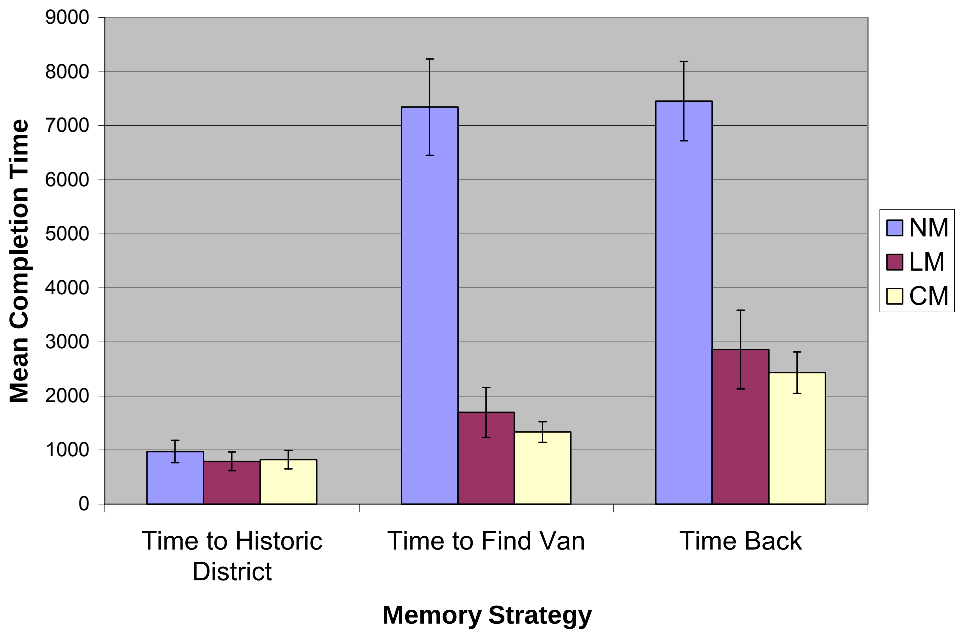


Figure 4.15. Results for the scenario when the agent is commanded to "Patrol the historic district" and "Find a van." The error bars here represent a 95% confidence interval.

Figure 4.15 shows the times for this scenario. NM fails the task in 90% of the trials (though it always found its way to the historic district). The maximum time limit of 8000 time steps is factored into its results. A paired t-test reported $P < 0.05$ for the LM and CM strategies for both the “go to” and “find” commands. Thus the average time to the historic district is significantly lower in the LM strategy than the CM strategy, but the CM does significantly better when finding the van.

4.6.5 Sequence of “Go to” Commands

In this scenario, the agent begins in a location on a street that leads into the city. This position is its home base for the scenario. The agent then receives a sequence of five commands telling it to go to various locations on the map. These commands are issued in an arbitrary order, and the agent only receives a new command when it successfully completes its present one. Between the commands, the agent is told to return to home base. Due to the distances involved, the maximum time limit for this scenario is increased to 20,000 time steps. The order of these “go to” commands were varied for each of the twenty tests run to acquire these averages. The locations of the coordinates for each district and the location of the home base were not varied.

Figures 4.16 and 4.17 show the results for each of the different memory strategies. The only strategy to show improvement is the CM, which is not surprising, since it is the only strategy that makes use of the map the agent constructs in its movement. For CM, there is a significant improvement (a paired t-test reporting $P < 0.05$), in all commands when compared to the first when the agent has no knowledge of the environment. Figure 4.16, shows the time taken varied wildly for the NM strategy, but remained fairly consistent for the LM strategy. NM was the only strategy to have any failure in completing commands in the time limit, failing to finish all five commands in 35% of cases. For Figures 4.16 and 4.17, the commands not completed were not factored in.

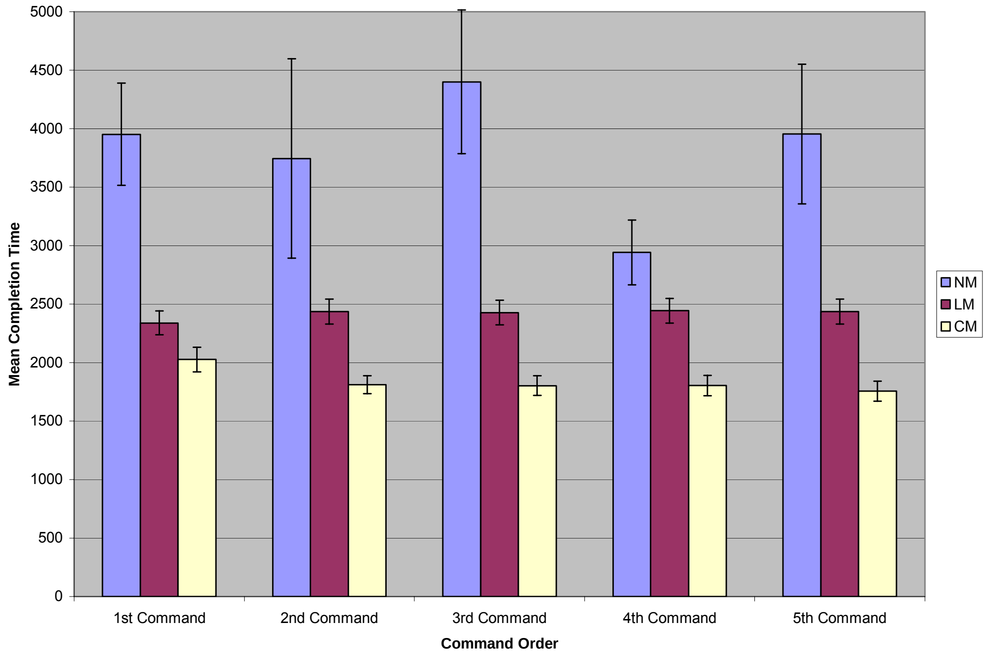


Figure 4.16. This chart shows the average performance of each strategy for the commands by order given. Times here include time to location and time back to base. A full command is considered to be both arrival at the specified destination and the return to home base. The error bars here represent a 95% confidence interval.

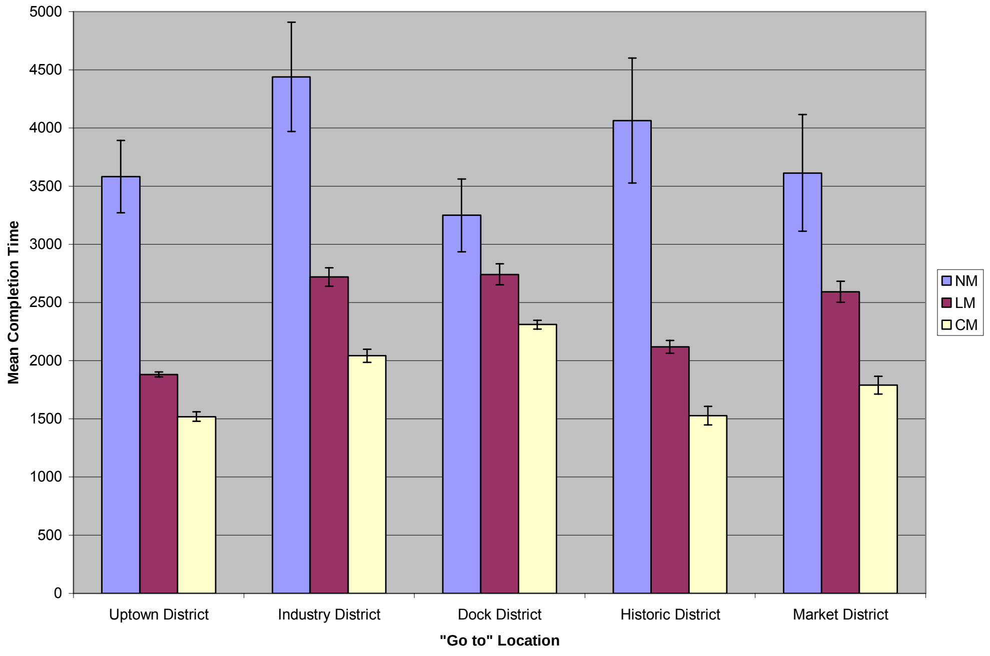


Figure 4.17. Chart for the same scenario as Figure 4.16, but separated by the location specified for the agent to reach. Times here include time to location and time back to the base. The locations are labeled by district names, but they are each only specific 1x1 cells for the purposes of the scenario. The error bars here represent a 95% confidence interval.

4.6.6 Cumulative Memory Tours of the Environment

The final scenarios examine a long tour of the entire city, with particular locations specified along the way. These scenarios are limited to the CM memory strategy, since the comparison between the three strategies has already been made in the case where terrain is covered multiple times. Because this strategy is the only one to benefit substantially from a map, it is the only one that has the opportunity to experience improvement due to memory. Two scenarios are looked at, one without enemies and one with nine enemy obstacles located in places the agent is likely to travel. The same home base and goal positions in the city are used in both scenarios.

These scenarios take the agent to a location in every district of the city in succession, and then return to its home base before making the next circuit. There are a total of ten tours it makes through the city. Each loop affords it a new opportunity to refine its knowledge of the location of buildings, streets, and (in the case where there are enemies) dangerous locations in the city.

Figure 4.18 displays the time taken to perform each of the ten tours of the city in both scenarios. There is improvement in both cases. The dramatic improvement in the case with enemies is due to the agent making a poor decision to follow a street that leads away from the city before returning. During the last five tours of the city, the agent follows the same path, having found a path in both scenarios that takes close to the same amount of time. Because the initial conditions and goal locations are the same in runs of this scenario, the cumulative memory strategy will always make the same decisions each time it is run. The process of making ten complete tours of the city is very computationally expensive, so it was only run once for each strategy to graphically demonstrate the improvement over time that cumulative memory allows. This behavior is confirmed, and the improvement is shown to be significant (a paired t-test reporting $P < 0.05$), in the multiple trials of the scenarios in Section 4.6.5.

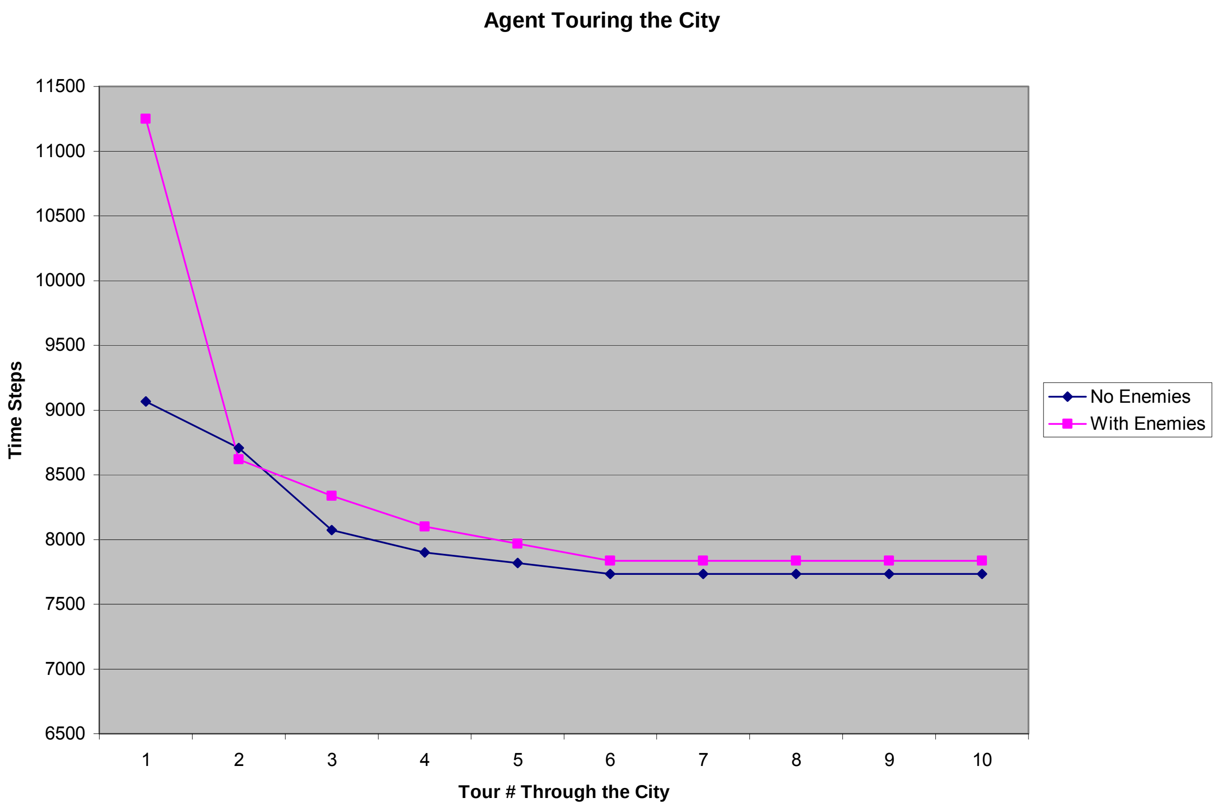


Figure 4.18. Results for the CM scenarios where the agent tours the entire city, for a case with enemies and a case without enemies.

4.7 Discussion

This chapter presented an agent architecture with a qualitatively more advanced behavior and a model urban environment with greater complexity than in Chapter 3. The architecture consisted of neural networks trained with various strategies (e.g. self-organizing maps, error-backpropagation) to process the incoming external visual and auditory information, while the high-level control was managed by finite-state automata and the influence of verbal and visual memories. This allows the agent a richer variety of behaviors to engage in. The environment has been enhanced to be a 3-dimensional model of a city, with streets and buildings with various textures. Perhaps the most crucial difference was changing the agent's perspective from complete overhead knowledge of the area within a certain radius surrounding it at any given time step to a limited angle of vision and a more realistic first-person perspective where the agent can see much further, but has a visual input that can be obstructed by buildings and obstacles.

There are three main conclusions to emerge from the computational experiments described in this chapter. First, the results support the hypotheses that the conversion of the data in the image of the 3-dimensional scene into a 2-dimensional map, either local or cumulative, will increase the efficacy and efficiency of the agent in the environment. Second, when the agent covers the same ground in the environment multiple times, there is a significant benefit over time to maintaining a cumulative memory map of the environment. Third, while both local and cumulative memory strategies are efficient, local memory strategies tend to produce agents with more flexible behavior and cumulative memory strategies tend to produce agents that prefer the same paths again and again.

As the results demonstrate, the agent completed its tasks far more quickly and with greater reliability when the visual information taken from the environment is kept in working memory, even if only locally and for a short period of time. The agent also performed more quickly and reliably when the visual information is transformed into an overhead map to allow the agent to navigate. The simplest demonstration of this is the fact that the frequency of collisions in the environment due to particle-like movement control and the length of time required for the NM agent to complete even the simplest task is far worse than in either the LM or CM strategy. The LM and CM strategies experience no collisions with the environment due to the accumulation of information, which allows them to more accurately judge where and when turns are appropriate and estimate the location of waypoints. What makes this particularly difficult for the NM strategy is that its limited view angle easily tells it what is in front of it, but when it is too close to an obstacle, it has difficulty judging proximity when such a limited perspective is combined with an inability to remember what is below it, let alone to the right or left of it. When a building or obstacle is sufficiently close, it fills the entire view and the agent with the NM strategy gains no relative information on which to base a judgment of proximity. The addition of memory and converting that memory into a 2-dimensional map (either just the immediate area or the entire environment), where the agent knows its position and can estimate the position of obstacles after acquiring visual information in successive time steps, helps to alleviate this context problem.

As expected, when an agent is covering the same ground over and over, this ability to retain visual information and convert it into 2-dimensional maps improves the performance. In particular, the CM performs much better than either of the other strategies when the task requires remembering what area has been covered (such as a patrol, where it is helpful to have seen or traveled as much of a district as possible). The CM performs better when attempting to find an object in the environment. This is because it systematically eliminates areas that it has seen as not containing the object and moves on to other regions, whereas the LM and the NM have a greater inclination to return to ground covered earlier.

The LM strategy in turn has a greater success rate than the NM, because even a limited local memory allows it to know that it prefers an adjacent area it has not visited to one that it has passed through. The NM has no such knowledge, and thus spends much of the time covering the same ground over and over again when making repeated tours between locations. Additionally, when the agent is simply passing through familiar areas more than once, the CM performs better, and the final set of results indicates that this is an improvement gained over time until the agent finds a path that is unobstructed and completely familiar.

Though a working memory makes the agent more resistant to mistakes in judging the environment, it is not completely immune to error. This is indicated by the better performance of the LM strategy when the agent is told to go to a location of which it has no foreknowledge. Because the agent is more apt to misjudge locations in the distance, plotting a complete course to the goal sometimes will make the agent's route needlessly longer, usually because it misjudged a street as being blocked by a building or, in the final scenario, misjudged a hostile obstacle. The strategy to take the most immediately expedient path employed by the agent with the LM strategy is resistant to these problems because misjudged areas further away on the map are apt to be corrected as the agent draws closer. This is also why the agent with the LM strategy in some scenarios has a higher accuracy (demonstrated in Figure 5.6 in the following chapter). It tends to approach areas that it has misjudged, and corrects them. In scenarios where the same ground is covered multiple times, it has a better chance of seeing terrain from different perspectives, a factor which also can increase accuracy. This gives a certain flexibility to the LM strategy. This is in contrast to the CM strategy, which tends to avoid areas it considers blocked for known clear paths and thus has a tendency to take the same route again and again, if it is proven to be clear.

Chapter 5:

Memory Management Strategies for Communicating Agents in a Simulated Urban Pursuit Scenario

5.1 Motivation

In Chapter 3, the effect and benefits of working memory were examined when added to a simple particle system in an abstract environment. In the previous chapter, the architecture of an agent operating in a simulated city with a first-person perspective (a 2-dimensional image of a 3-dimensional scene) was described and the results of different memory management strategies were examined for multiple scenarios that made use of the agent's various potential actions. It was demonstrated that while both the local memory (LM) strategy and the cumulative memory (CM) strategy improved upon a strategy with minimum memory (NM) needed to complete the tasks, various tradeoffs existed between these two successful methods.

In this chapter, I use a similar urban environment but now with multiple interacting agents, each having a capability and range of behavior comparable to the agent in Chapter 4. The goal here is to extend the results of Chapter 3 in three ways. First, the work here involves a more complex agent capable of more complex interactions with its neighbors. Second, agents are placed in an environment of greater realism to provide them more of a challenge than the abstract environment. Third, the task is enhanced from simple point-to-point movement to a task that can encompass the new agents' possible behaviors. To take advantage of the different and more sophisticated actions available to the agents, a more challenging pursuit scenario is employed where a team of agents chase and seek to capture a moving target with capabilities on par with theirs. This classic scenario is of interest because it requires multiple agents to accomplish. As defined below, no lone agent would be able to capture the target. Further, it potentially can be facilitated through communication between agents, coordination of behavior, and experiential knowledge (working memory) of the environment.

The following questions are asked in this chapter: Given the urban pursuit scenario, the limitation of agent behaviors and the potential for interaction with one another and the environment, can improvement in the task be gained from such interactions? Can acquired knowledge of the environment improve performance, even when limited as in Chapter 3? Will different strategies of using working memory influence the efficiency of their movements? And finally, what combinations of these strategies will be optimal?

It is hypothesized that communication is crucial for the agents' efficacy, just as in the simple world of Chapter 3 where simpler agents communicated their position and velocity locally to one another, while coordination of agent behaviors and experiential knowledge (i.e. working memory) will both provide significant improvements in the ability of the team to capture the target. It is further hypothesized that, when the agents coordinate their behavior based on local information of one another's positions and planned paths, even a limited memory can yield significant improvement in agent behavior, as demonstrated in the simpler situation of Chapter 3. Additionally, based on the results of Chapter 4, it is hypothesized that while both cumulative and local memory strategies will prove successful for this scenario, local memory strategies will be more helpful in situations requiring flexibility, such as the agents' searching the environment for the unknown moving target. In contrast, cumulative memory behavior is expected to be more successful in cases where the agents are required to coordinate their movements with one another, with the position of the target, and with the remembered features of the environment, such as when agents are attempting to surround the target.

5.2 Description of Pursuit Scenario

The pursuit scenario, also known as the “pursuit domain” and the “predator-prey domain,” is a highly studied multi-agent system that is particularly of interest in the study of cooperative behavior between agents (see Section 2.1). It can be used with a variety of intelligent agents and in different environments. It usually features multiple predators (traditionally colored blue) and one prey (traditionally colored red), respectively called agents and targets later in this work. In the original formulation, these agents exist on a grid. The predators are able to move only one cell vertically or horizontally at any time step, while the prey has the same limitation, but moves randomly [Benda, 1986]. Most of the interest tied to this domain has been in discovering optimal configurations of predators or cooperation strategies [Korf, 1992; Lenzitti, 2005; Zhao, 2005].

This chapter describes two implementations of the pursuit scenario. The first is a pilot study developed in a 2-dimensional environment where agents exist as cells on a grid. This was done so features of the scenario could be explored in a computationally fast environment before extending them to the more complex agent and environment of Chapter 4. The second is in an urban environment similar to that described in the previous chapter, where agents and the target move in continuous space and have the same first-person view of a simulated environment as the agents in Chapter 4 have.

5.3 Pilot Study: 2-dimensional Pursuit Scenario Model

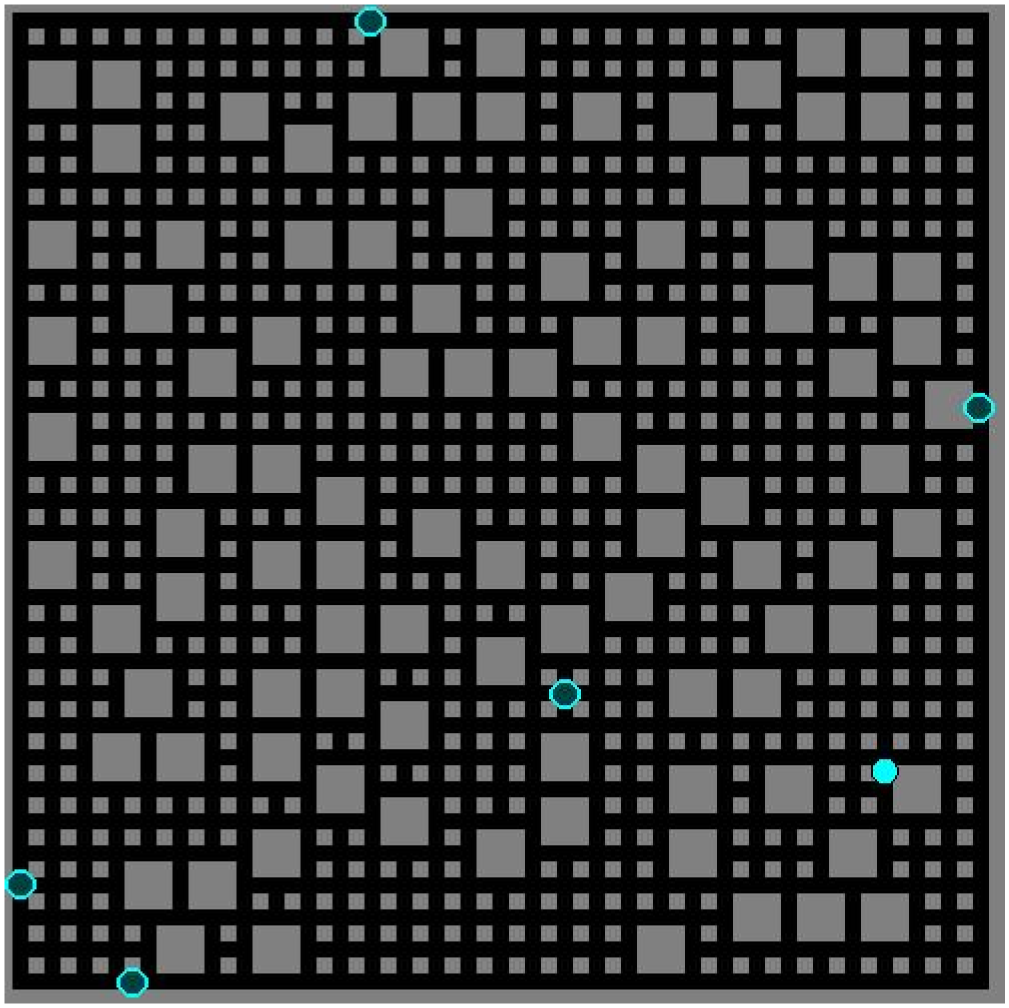


Figure 5.1. Sample pilot-study environment. The light-filled circle represents the target, the dark-filled circles represent the agents seeking to capture it, gray cells represent buildings (environmental obstacles that obstruct agent vision), and black cells represent streets.

This first model was developed to explore the effect of coordination and map knowledge in a simple and computationally fast environment where many repeated trials are possible for different parameter settings. This was done for the express purpose of discovering what behaviors might be useful for the more advanced agents and environment through initial testing in a simpler setting that captures the structure and relationships of the more complex setting. A sample environment is depicted in Figure 5.1. The environment is a discrete grid space measuring 64 by 64 cells. A discrete space was chosen because it allowed for a convenient method of determining a captured state for the target (i.e. when the target is surrounded vertically and horizontally by either agents or buildings) while preserving the structure of a grid which is superimposed over the continuous space in the more complex environment.

Cells in the pilot study are occupied by one of the following: street, building, agent or target. These four cell types have the following properties. Street cells (black) are the open spaces in the environment. The target and agents can move freely onto street cells. They also have open visibility, meaning that the target and the agents can see across these cells for some distance unobstructed. Building cells (gray) are the obstacles in the environment. These obstruct the vision of agents and are impassable.

In this cellular environment, there are five agents (dark-filled circles) and a single target (light-filled circle). At each time step, they choose to move onto a valid cell. The only valid cells in the environment are vertically or horizontally adjacent street cells. Agents are not allowed to share the same cell as one another, and they may not share the same cell as the target. Both the agents and the target can see as far as 20 cells, provided there are no obstructions (buildings or other agents). This view can be in all vertical or horizontal directions, or it may be limited to only the direction in which the agent is moving.

The target moves before the agents and exhibits a simple evasive behavior where it chooses a velocity vector that is away from the average position of all observed agents. It then reduces this vector to the closest valid movement cell.

The goal of the five agents is to trap the target by surrounding it such that it has no valid movements. When this happens the scenario ends. This condition includes situations where the target may be surrounded by both agents and buildings. So long as there is no valid movement available to the target, the scenario ends with the agents' success. There are no dead ends in the environment. Thus, at least two agents are required to capture the target, and it is impossible for a single agent to capture the target alone. If a simulation has run longer than 10,000 time steps, the scenario ends with the agents' failure.

The agents have the ability to communicate locally (within a radius of 20) with each other information about the target's position when it is observed, coordinate their movements given the proximity of their positions, and use knowledge and memory of the environment. Initially, in the scenario, there may be no communication, if the target is not in the line of sight of any of the agents. Once the target becomes visible to an agent, it communicates the target's location and velocity vector to all other agents within its broadcast range, a radius of 20 cells.

The coordination of agent movement is affected by various influences. An agent may choose its course based on what is visible, the known locations of other agents, the broadcast location of the target, and the predicted path the target will take. Agents have the potential to exhibit collective movement using the forces described in Chapter 3 (avoidance, matching velocity, and centering). These allow the agents to coordinate their movements based on one another's position and direction.

Additionally, because agents can locally broadcast the target's position and direction when they see the target and because the target moves in a simple evasive behavior attempting to avoid its pursuers, the agents have the ability to make predictions about the target behavior. When this capacity is used in the scenarios, the agents will predict the target's movement patterns up to 15 cells away, based on the target position and velocity broadcast by an agent who has the target in view. Agents predict the target will move in a straight line until obstructed by an obstacle in the environment. At that point, the target could potentially turn one of two ways, and the agent chooses a prediction randomly. If there are many obstacles, the prediction may be inaccurate, but because the path is not too long, the agent will at least be heading in the correct direction, and may update predictions depending on what future broadcasts it receives regarding the target's location. After creating a predicted path, an agent can then find the closest location along the path that it can reach before the target arrives there. The agent then can set its own velocity vector in that direction. If such a position does not exist, the agent then chooses a direction vector toward the target's last known location. If the target is in view, the agent ignores the coordination behavior and only moves based on the target's position. The following equation shows how the desired vector of movement is computed for agent *j*.

$$
\begin{aligned}
&\overrightarrow{f_j} = \left(\frac{w_c}{N}\sum_{i=1}^{N}(\overrightarrow{x_i} - \overrightarrow{x_j}) + \frac{w_{mv}}{N}\sum_{i=1}^{N}(\overrightarrow{v_i}) + \frac{w_a}{M}\sum_{i=1}^{M}(\overrightarrow{x_j} - \overrightarrow{x_i})\right) \\
&\overrightarrow{p_j} = (\overrightarrow{p'} - \overrightarrow{x_j}) \\
&\overrightarrow{p'} = \arg\min_{\vec{y}\in P}\left(\begin{cases}(\vec{y} - \overrightarrow{x_i}) - (\vec{y} - \vec{g}), |\vec{y} - \overrightarrow{x_i}| \le |\vec{y} - \vec{g}| \\ \vec{g}, |\vec{y} - \overrightarrow{x_i}| > |\vec{y} - \vec{g}|\end{cases}\right) \\
&\overrightarrow{t_j} = (\vec{g} - \overrightarrow{x_j}) \\
&\overrightarrow{v_j} = (1-c)(\overrightarrow{f_j} + b(\overrightarrow{p_j})) + (c)\overrightarrow{t_j}
\end{aligned}
$$

In these equations, *N* is the number of neighboring agents within communication range of agent *j*. *M* is the number of neighbors considered to be too close to agent *j*. The

w values are the weights for the centering (c), matching velocity (mv), and avoidance (a). P is the path agent j predicts the target will take given knowledge of its location and direction, g is the last known location of the target. b is 0 when agent j is not receiving a broadcast of the target's position and velocity and 1 when it is receiving a broadcast. Finally, c is 0 when the target is not visible and 1 when agent j sees the target.

In the above equations, f is the sum of all the forces due to centering, matching velocity, and avoidance. p' is the coordinates of cell y on agent j's prediction for the target's path P that is closest to j. If the agents are not allowed to make predictions about the target's movement, then $p' = g$ always. p is the vector between p' and j's position, while t is the vector between g and j's position. v is the velocity vector for agent j based on the presence or absence of the target in the agent's view, the agent's knowledge of the target's location, and the agent's prediction about the target's movement.

Once the velocity vector is computed, the agent must convert it into a valid orthogonal move in the environment. It chooses the closest orthogonal movement to the vector and updates its position in that direction.

The agents can also be given a further vision limitation, stipulating that they may only see along the direction they last moved. This limitation forces the agents to operate under conditions more similar to the agent presented in Chapter 4, in which the agent only has a window of visibility that looks ahead and has no peripheral vision. Both this method and a method where agents can see in all orthogonal directions are implemented.

Agents were also given varying degrees of knowledge of the environment in the scenarios. Agents could be given full knowledge of the map at the outset of a scenario, or they could acquire memory as they explored the map, adding knowledge of obstacles to their individual maps when they were within a range of 1 cell of the agent's position.

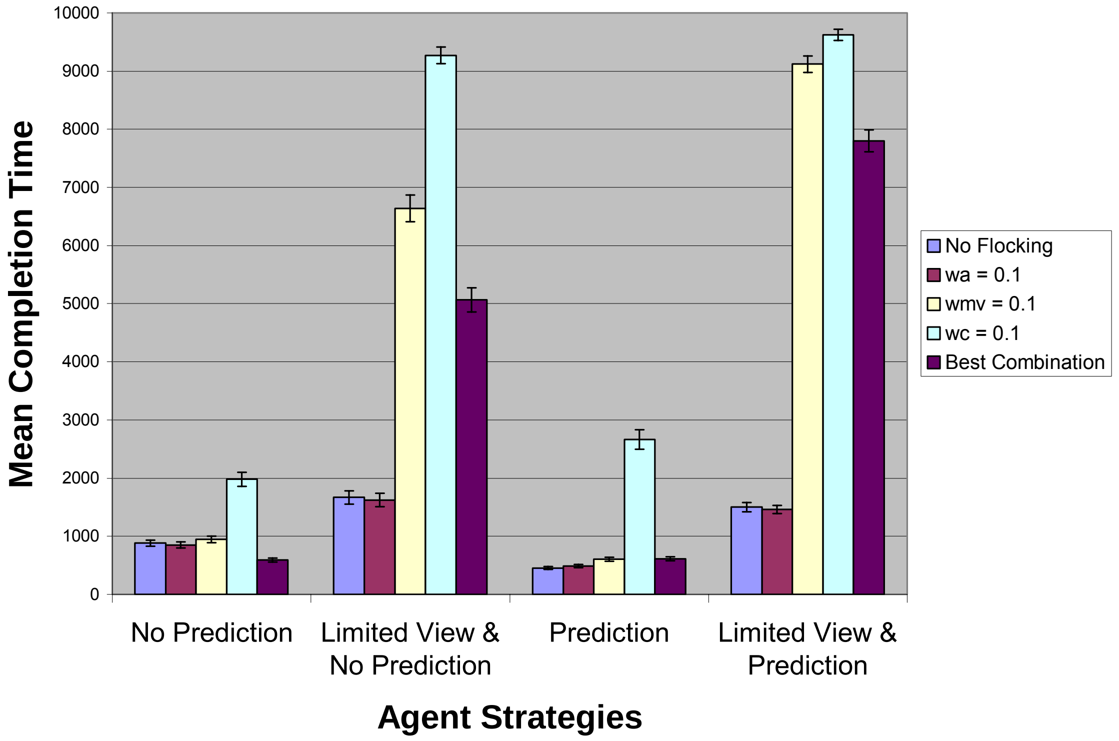


Figure 5.2. Results for the different strategies employed by the agents in the pilot study pursuit scenario where the agents do not initially have any knowledge of the map, but may acquire it. In the strategies described here, the agents can either predict or not predict the path of a target when they hear a broadcast of its location. Additionally, agents can either see in all unobstructed vertical and horizontal directions, or they can be limited to seeing in the direction they are moving (i.e. limited view). w_c, w_{mv}, and w_a refer to the flocking forces of centering, matching velocity, and avoidance, respectively. While the overall time increases significantly for agents with a limited view, there is also a significant worsening when the flocking parameters of w_c are used. In the case of a limited view, w_{mv} also has an enormous detrimental effect. In all other cases, there is no significant difference between scenarios featuring the flocking parameters and those that do not. The error bars in this figure represent 95% confidence intervals. The best observed combination of the flocking parameters, although better for the case of no prediction, was significantly worse ($P < 0.05$) in the cases where the agents had limited views.

Figure 5.2 displays the mean task completion times for the agents over 1000 trials for different coordination parameters in cases where agents were either limited to a view along their direction of movement or could see in all orthogonal directions and either could predict the target’s future position or lacked the ability. This demonstrated that while the problem was significantly more difficult in the domain with a limited agent view, the strategies that employed the use of coordination by traditional flocking forces either had no significant impact or had a significant detrimental effect. This is probably due to the cramped nature of the environmental obstacles. These forces are usually employed where agents have a great deal of room to maneuver and see other agents, whether or not they exist in discrete or continuous space. When extending this scenario to the more complex agents, there is an added difficulty in that these collective influences could clash with the subgoal-driven behavior of the waypoint sequences produced in the cumulative memory. Instead of these influences, a method for coordinating agent paths will be necessary for the more complex agents in order to achieve the desired effect.

Although in most scenarios the prediction of the target behavior produces a significant improvement, this too will not necessarily be compatible with the scenario of agents in the more realistic environment. The target in this environment has a behavior that is much easier to predict when it is in view. Its behavior is to move as far away from the pursuing agents as possible. When there is only one pursuer, this means the agent will likely only move in a straight line away until obstructed. In this scenario, while the agent may make inaccurate predictions, the points where it deviates from the actual target’s path are only at obstacles. In the more realistic environment, the target has a greater tendency to turn when being followed in an attempt to confound and lose its pursuers. This makes accurate prediction much more difficult for any agents receiving broadcasts of the target position. Yet with a new method for coordinating agents and the ability to remember the environment as they are exposed to it, significant improvement is possible.

Among the main results of this pilot study, it was learned that when restricted to a grid environment with many obstacles, agents did not significantly benefit from using the traditional coordination as employed in Chapter 3. This became even more pronounced when the agents were given a limited line of sight (akin to the limits the agents who receive 2-dimensional images of the 3-dimensional environment experience). In some cases certain influences (those of centering and matching velocity) worsened the performance. This indicates when dealing with a multi-agent system of more complex agents in the more realistic environment that some of these forces are not relevant for this task and new methods for coordinating agents are required.

5.4 More Realistic Pursuit Scenario Model

In order to accommodate the multiple agents in the scenario and have them accomplish the new desired goals, some alterations to the model environment and agent structure described in Chapter 4 had to be implemented for the sake of computational efficiency. Some of the features that follow have been simplified in consideration of time and memory constraints in some cases, but the parts that have been changed were intended from the beginning to be either adjustable or modular pieces. Any of the simplified features could be retained, and implemented again, if running many trials in a reasonable amount of time were not an issue. In contrast, other features, such as inter-agent communication and coordination, have been added because they only apply to scenarios where there are multiple cooperating agents and also opponent agents.

5.4.1 Scenario Specification and Environmental Changes

The pursuit scenario when implemented in this model has the majority of the same features seen in the previous section. Again, there are five agents and a single target. They move in real-valued space, but as in Chapters 3 and 4, there is a grid of environmental features overlaid on this space. This is a 32 by 32 grid. Because agents exist in both real-valued space and grid space, they can potentially occupy the same cell as one another, but they are restricted from passing through one another since each agent has mass in the environment. The scenario ends when one of two criteria is met. If the time counter has reached 5000 time steps, the scenario ends in failure for the agents and success for the target. At any point before that, if the agents manage to surround the target such that there is either a building or agent within one cell north, south, east, and west of it, then the target is said to be captured and the scenario ends with success for the agents and failure for the target.

In the environment of the previous chapter, the city was made up of different types of buildings separated by roads and surrounded by a wide-open grassy countryside on three sides and a body of water on the other. This scenario requires a more contained setting in order to limit the run time of the scenario. Additionally, it means that the target cannot escape the world, and must evade the agents throughout the scenario. Only buildings and roads are used, and the city perimeter consists of an unbroken wall of buildings. As in previous scenarios, buildings are impassable and roads are passable. As a natural result of using a first person perspective for each agent, agents cannot see through buildings but can potentially see any distance along the road, limited here only by their ability to recognize an object. Figure 5.3 displays a snapshot of the new environment.

5.4.2 Simplification of Visual and Verbal Inputs

Several changes were necessary to the model to make it run more quickly and efficiently in the case with multiple agents. This was necessary for two major reasons. First, because there were now five agents (plus a target agent) running concurrently in the environment, this meant that the image creation from the camera, which took the longest time of any operation at any time step, would take several times longer. For an agent in Chapter 4, a typical time step takes about 500 ms. This would approximately require 3 seconds per time step for the new scenario. Second, the task to be performed here for the agents has a high termination maximum (5000 time steps in this model) and there could potentially be a lot of variance in performance because agents are pursuing a moving, and evasive, target. Sometimes they might make a capture quickly by chance, and other times take a long time or have no success. Thus, many trials may be needed to get accurate means for the different strategies explored.

The image resolution produced by the camera for each agent in the environment has been reduced from 512 by 512 pixels to 128 by 128 pixels. This means that the image processing for each agent will be much faster, and the total real time devoted to a scenario time step would be reduced. A typical time step for the entire system, the target and all agents, takes about 470 ms, on the order of the system in Chapter 4. This reduction does not have a very detrimental effect on the agents' ability to segment the images either, as the resolution of the scene-segmenting grid (visible in Figure 5.3b) is reduced from 32 by 32 to 8 by 8. Since each cell is based on the average pixel values within it, these values do not experience much change in the reduction of the image, and the agents can still effectively detect which areas in the image are streets, buildings, and objects.

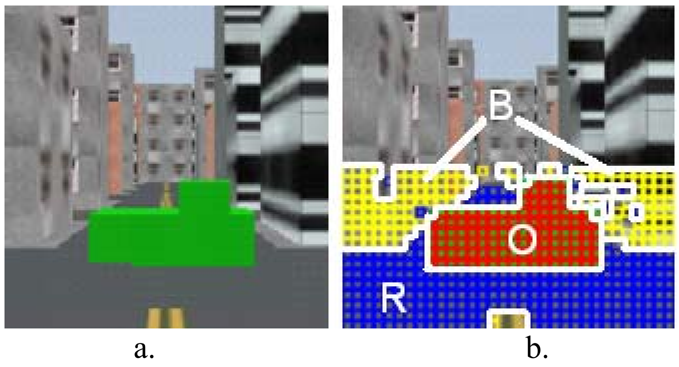


Figure 5.3. Sample snapshots of the pursuit environment from the perspective of an agent. The agent receives the raw image shown in (a) and processes the image using its trained self-organizing maps to segment the scene into buildings (labeled "B"), roads (labeled "R"), and the target object (labeled "O"). Contiguous blocks of cells of the same type have been outlined in white.

The complexity of passing verbal and visual information through many networks also consumes a great deal of time in this model. For the purposes of running these tests, many of the modules that handled this were simplified to allow information to enter and exit the agents more quickly. The accuracy can be trained to be very high for these tasks using neural networks. Trained networks similar to those that appear in the previous chapter could be retained and trained to do the processing of vision and commands. The visual simplification, which is possible in this model environment, involved allowing the agent to detect whether an object was another agent or the target from just unique color information, as opposed to processing a "close-up" thumbnail to reach the same result. Agents are a slightly different color than the target, and the agents in this scenario are allowed to immediately recognize the difference. The verbal simplification allowed agents to directly convert command sentences received into the appropriate concept memories and behaviors. It also allowed them to transmit information directly to one another, which, while not a feature in the model of the previous chapter, relates to their spoken output. Agents will be communicating a good deal of information to one another, so this would be very time consuming to model with neural networks over many trials.

The agent environmental step sizes per time step in the model described in the Chapter 4 were fairly small (0.05 in the real space). Increasing this value, allows the agents to move much more quickly, while still being able to accurately interpret their environment. Agents now use step sizes of 0.25, while the target moves slightly faster with step sizes of 0.26.

There were also some changes needed so that the model could function. In the environment of Chapter 4, there was only one agent and thus no need to model it physically in the space because the agent would never see itself. However, in this model, it becomes crucial that agents have a physical form, not just so they do not pass through one another, but so that they are able to see one another in their first-person perspective images and are able to recognize the identity and location of the other mobile objects. The agents and the target were given forms which are placed in the environment, centered at the location specified by their coordinates and oriented in the appropriate direction (Figure 5.3a shows a target from the point of view of an agent, and the outline of this object is labeled "O" in Figure 5.3b). As the agent or target position or angle is updated in each time step, the corresponding object in the environment is updated as well.

5.4.3 Agent and Target Behaviors

The agents and target have behaviors dictated by the commands issued to each of them at the beginning of a scenario. The agents are each issued two commands. They are told to patrol the city and to find and pursue the target. Patrolling the city functions as described in Chapter 4, but which memory strategy is employed determines the specific movement of the agents. Finding the target once again causes the agent to announce it has seen the target when the target is visible, but there is an added set of behaviors that occur when this happens, including pursuit.

If agents are given the ability to communicate, then when an agent sees a target, it broadcasts the target's location to all other agents within a broadcast range (15 environmental cells in these scenarios). It will also adjust its normal patrol behavior. When an agent sees the target, it will follow the target in a manner depending on what memory strategy is in use (described below). The agent also remembers seeing the target, so if the target should turn onto a side street, the agent will continue to the last estimated location where he saw the target and then turn to match the last remembered angle of the target. If this succeeds in returning the target to view, the agent will continue the process over again. If it fails, then the agent returns to a regular patrol of the environment, looking for the target it lost, unless it receives a broadcast from another nearby agent about the target's current location.

When an agent receives a broadcast of the target's location, as long as the agent receiving the broadcast does not also see the target, he will not behave as described above. Instead, the agent treats this as a command to go to the broadcast location to assist in trapping the target. It will then move depending on the agent memory strategy in use for this scenario. The planned movement of other agents and remembered portions of the environment's layout may also affect how the agent moves.

The target is pre-programmed to follow two basic commands, which determine its behavior for the entire scenario. It is essentially told to patrol the city and evade the agents. The target is oblivious to the communication between agents and cannot communicate with them. Essentially, the target is playing the scenario as the solitary agent in the previous chapter, albeit with moving enemy units. Unlike the agents, the target behavior rules are the same across all tests, which allow for a baseline in determining how the changes in communication, coordination, and memory affect the agent team. Whereas some memory strategies for the agents below will give them limited chart memories, the target has an unlimited memory, although its memory of obstacles is temporary since agents rarely will stay in the place the target saw them for long. These two commands in concert ensure the target is always moving and attempting to get out of sight of its pursuers.

5.4.4 Agent Strategies

The agents have different strategies that they can use to work together or improve their performance in capturing the target. These include an ability to communicate information to one another, such as the position of the target. They also can coordinate their movements in the environments when attempting to surround the target, making a capture more likely. In addition to working together, they can have advantages given by different strategies of managing memories, such as the successful local and cumulative memory management strategies described in the previous chapter.

5.4.4.1 Agent Communication

Basic agent communication consists of broadcasting the location of a target when the target is visible to all other agents within its broadcast radius. This broadcast location is computed from the first-person perspective image pixels as the approximate location of the object in the environment as described in the previous chapter (though the k in the equation is adjusted to 238.9 for the smaller resolution in this scenario). While not a completely accurate position of the target, it is usually quite close when the agent is close to the target. If the agent is far away, the accuracy is reduced, but this broadcast position can still point ignorant agents in broadcast range toward the correct general area, giving them a greater chance of finding the target and making their own broadcast.

The other forms of communication that constitute coordination are described below.

5.4.4.2 Local Memory vs. Cumulative Memory

Agents retain the ability to use local and cumulative memory strategies to help them navigate the environment smoothly, with very few or no collisions with the buildings. This potentially has the added bonus of giving the agents the ability to plan routes using their acquired knowledge of the map in the case of the cumulative memory strategy.

When the agent is using a strategy of local memory, the method of pursuit is fairly simple. If the agent has the target in view, it computes the direction of the target and places a single waypoint in a position that will get it closest to the target. If the agent recently saw the target and remembers its last known location and direction, the agent will continue in the process of generating and approaching waypoints toward that location until it gets close enough (less than half a cell away). When it reaches the last known location of the target, it matches the target's last known direction angle. The agent does this because this operation gives it the best chance to see if the target is still visible and continuing on the same heading that the agent last remembers seeing the target choose.

If the agent does not see or remember the target, but hears a broadcast of its current location, the agent treats this situation as a command to go to that location. The agent drops a waypoint in the closest adjacent cell that is valid and takes it to the broadcast target location. As the target's position updates in the broadcast, assuming another agent is in pursuit of it, then the agent hearing the broadcast will continue to move toward waypoints that are approaching closer to the changing target location until it either sees the target or moves out of range of the broadcast. If the broadcasts cease, but the agent remembers a broadcast target location, it will continue to approach that location until a new broadcast is issued, or until it sees the target.

At this point, a small change was made to the range of local memory around the agent, reducing it from five by five to three by three. This was done to facilitate the pursuit of the target. When there is a smaller window, memories are only added when they are most necessary and more accurate, that is, when they are closer to the agent.

When the agent is using a strategy of cumulative memory, the method of pursuit is different. If the agent has the target in view, it computes a path of waypoints to the target's position using the method described in the previous chapter and follows them. If the agent recently saw the target and remembers its last known location and direction, the agent generates a path of waypoints to that location. When the agent reaches the last seen position of the target, it matches its last known direction angle.

If the agent does not see or remember the target, but hears a broadcast of its current location, the agent generates a path of waypoints to that location that it will follow. Because the broadcast target location the agent receives will potentially update as an announcing agent tracks it through the environment, the waypoint paths generated by the agents relying on the broadcast information also are permitted to update. Again, if the broadcasts cease, but the agent remembers a broadcast target location, it will continue to that location until a new broadcast is issued or the target is visible.

5.4.4.3 Agent Coordination

While the agent's computation of its next waypoint and paths of waypoints is not altered in the previous sections from what was described in the previous chapter, because there are now multiple agents in the environment, they can coordinate their movement. In Chapter 3, this coordination was achieved through accelerations that allowed the agents to move in a flock together. However, because of the cramped nature of the current city environment, where agents are less apt to share the same roads as the other agents, this coordination is less likely to prove beneficial. This is particularly the case with the very limited view angle of each agent. This view angle, when coupled with the many building obstacles, essentially means that the agent can only see straight ahead. The results of the pilot study, particularly when agents have their view limited to straight ahead in that domain, support this.

Nevertheless, the avoidance acceleration could prove useful in this scenario, particularly in cases where the agents might collide and have difficulty navigating around one another. It could also help to separate them, allowing them to spread out more and cover more of the environment. The radius of this avoidance influence is typically kept quite low. If too high, it can cause agents to divert unnecessarily. Because this could interfere with the pursuit of a target, the avoidance influence is only factored in when the agent does not see the target and is not approaching the last place it remembered seeing the target. If the agent is close enough to another agent to experience this influence, then it alters its course to a waypoint that is in a valid adjacent cell that is closest to its avoidance vector. The avoidance vector is computed in the same manner as is described in Chapter 3, with the exception that the agent only avoids the closest agent in its avoidance radius.

In addition to this influence, there is another method of agent coordination that can be implemented in the case of a cumulative memory strategy. Whereas the influences described in Chapter 3 cannot be applied to agents that are computing paths, the agents can attempt to coordinate their paths so that they can better capture the target. The chances of a successful outcome are increased if the agents are approaching the target from different directions. Put another way, the agents are able to do better if they do not take the same paths toward the agent when they hear a broadcast. If the paths can be coordinated so they cross as little as possible, then the agents' performance should improve. This can be achieved by having an agent broadcast its planned path when the agent has planned a path to the target's location, whether it was seen, remembered, or heard through a broadcast. Agents in range can then take into account the paths of other agents when planning their own paths. When planning a path, the cost value of waypoints increases when a location is on another agent's path as well. Instead of the base cost value of 1, location costs 6 to add to the agent's path, making it more likely the agent will attempt to find another direction from which to approach the target. Additionally, the cost of any cell at the end of another agent's path that is adjacent to the broadcast target's current location has a cost of 16, increasing the chances that agents will look for valid paths that approach the target from another direction, even if they are substantially longer.

5.4.4.4 Memory Limitation

In addition to the strategies intended to improve the performance of the agents, a memory capacity limit from Chapter 3 can be examined here as well. In Chapter 4, and so far in this description, an agent with a cumulative memory is able to remember everything it has seen, and an agent with a local memory is able to remember only things that are at most one cell distant from it. Agents can also be given a limited memory that randomly eliminates memories from the cumulative map storage over time as it updates with new memories. This resembles the behavior of the agents of Chapter 3 that were able to experience improvement with even a small, limited working memory. The hypothesis is that given the ability to make the cumulative memory plans of waypoint paths, the agents with a limited memory can do as well as one that remembers the entire map. Further, it is hypothesized that a significant improvement will be found due to memory and not just the coordination described in the previous section.

5.5 Results

5.5.1 Experimental Methods

The following section describes the results for three scenarios. The first two scenarios are single agent scenarios meant to explore the agent behavior for two key commands in the pursuit scenario. These were run using a single agent in the environment as described in Chapter 4. Unlike the agents in the pursuit scenario, the single agent in each of these still operates using the modular neural network to govern incoming commands and visual information, and the image resolution is 512 by 512.

The first scenario is a case where the agent is told to patrol a district, and the three possible memory strategies (NM, LM, and CM) are tested. There are twenty runs for each. The initial position is varied for each individual run, but always remains a position on the district's perimeter. The same set of initial positions are used for sets of runs for each memory strategy. The maximum time limit of this scenario is 5000 time steps.

The second scenario is similar to the scenario in Chapter 4 where the agent is told to go to different locations, returning to a home base between commands. The difference is that in this scenario there are hostile obstacles blocking certain paths. Agents using each memory strategy experience twenty different runs. The initial position is consistent for all runs of this scenario, but the sequence of locations to be visited varies across the twenty runs. The sets of runs where different memory strategies are applied are identical, and the maximum time limit to this scenario is 15,000 time steps.

The third scenario is the multi-agent pursuit scenario. This features the scaled-down agents described in Section 5.4.2. Each agent receives a sequence images at a resolution of 128 by 128 pixels. Each set of trials consists of 100 separate runs of the scenario (each with an allowed maximum of 5000 time steps). While the setup for individual runs differs from one another, the same set of trials is used for each of the agent teams tested. Additionally, the agents and the target are not placed completely randomly in the environment. The target always begins in the center of the map, and one agent is always placed so that it can see the target from the beginning. This eliminates the need for the agents to find the target from the beginning. This scenario can vary greatly in how long it would take, though in scenarios it frequently happens that the agents lose the target and have to find it again. This is done for the purpose of making the scenario run more quickly. The other four agents are each placed randomly within the four different quadrants of the city, allowing them to be spread out from the beginning. None of these other agents begin the scenario seeing the target.

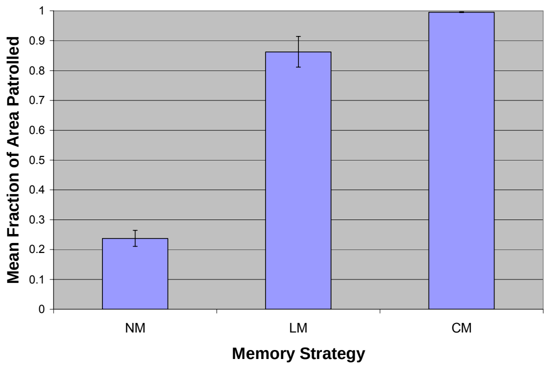
5.5.2 Single Agent: Patrolling an Area

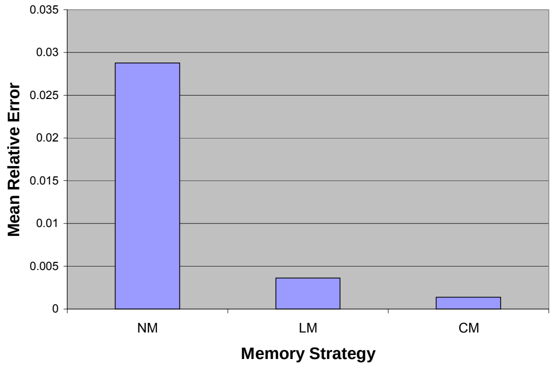
In this scenario, a single agent begins at an arbitrary position near the periphery of the Historic District of the city. It is then instructed to "Patrol the Historic District." The agent interprets this command as it should cover as much area in the district as possible, and, if possible, all reachable locations of the district. This scenario is performed twenty times for each strategy under identical initial conditions. The only difference is the strategy the agent employs and the memory of the environment available to it. Because the scenario is not always guaranteed to terminate for some strategies, a maximum time limit of 5000 time steps is imposed.

Figure 5.4a displays the fraction of the Historic District's area that was patrolled within the time limit. Along with this, an accuracy measure is also computed. This measure is accumulated from the visual information available to the agent, in a manner identical to the way the CM agent constructs its map. At the end of the scenario, it represents a picture of an overhead view of the environment at the resolution of cellular space. This is identical to the CM agent's chart. This "picture," though collected from the visual information of all agents, is not available to the LM or NM agents to affect their movement strategy, but is instead used for the purpose of external assessment of what the agent thought it saw. This accuracy is a measure of the percentage of area in the Historic District that was identified correctly (such that cells marked streets are streets in the environment, and those marked buildings are buildings). From this measure, the chart also displays the relative error. This is an error computed with the following equation:

$$(a - a')/a$$

where a is the number of cells mapped (i.e. no longer labeled unknown) and a' is the number of cells correctly mapped. This allows us to see how accurate the strategy was regardless of how much area was mapped.





a. b.

Figure 5.4. Results for the command of an agent instructed to "Patrol the Historic District." **a.** Displays the mean fraction of area in the historic district patrolled with each of the strategies. The map accuracies of the different strategies (not shown) are very close to these values. The error bars here represent a 95% confidence interval. **b.** Displays the relative error of the different strategies for this scenario.

The CM strategy was the only strategy to finish within the time limit, and ended with close to 100% accuracy and 100% of the relevant area patrolled. The cells missed are interiors of larger buildings of which the agent has seen the entire outside skin and, thus, cannot find a path to the cell. On average, this strategy took 2611 time steps. The LM strategy covered less, and the NM did much worse than either of them in both area patrolled and accuracy. The NM agent's relative error is also a power of ten higher. This high error is related to the NM agent's tendency to collide with buildings. During collisions, the agent's view is blocked by a building and the agent can mistakenly judge more distant locations as buildings when they are in fact passable and just not visible.

It should be noted that this scenario is naturally inclined to be best suited for the agent able to map the environment (using the CM strategy), because this is the only strategy guaranteed to terminate. It is possible, though increasingly unlikely as time goes on, that the LM agent would never complete the task. Because of its total lack of memory, it is highly likely that the NM agent would never complete the task. The NM agent often just covers the same corridor of the district over and over again because it has no memory of being in locations it has even recently left. The low fraction of area covered by the NM strategy indicates it is a poor choice for this command when compared to either the LM or the CM strategies. The other two strategies however, will be examined in the pursuit scenario.

5.5.3 Single Agent: Sequence of "Go to" Command with Hostile Obstacles

This scenario operates much the same as the scenario in 4.6.5, but the crucial difference is that three stationary "hostile" obstacles, or "enemies," have been placed at points in the environment likely to be in the agent's path. A sequence of commands is issued, along with instructions to return to the home base between each command, but the agent is also instructed to evade the hostile obstacles. The number of locations where the agent must visit has been reduced to three in this scenario, and the maximum time limit has been dropped to 15,000 time steps. The order of the "go to" commands was varied for each of the twenty tests run to acquire these averages. The locations of the coordinates for each district and the location of the home base were not varied.

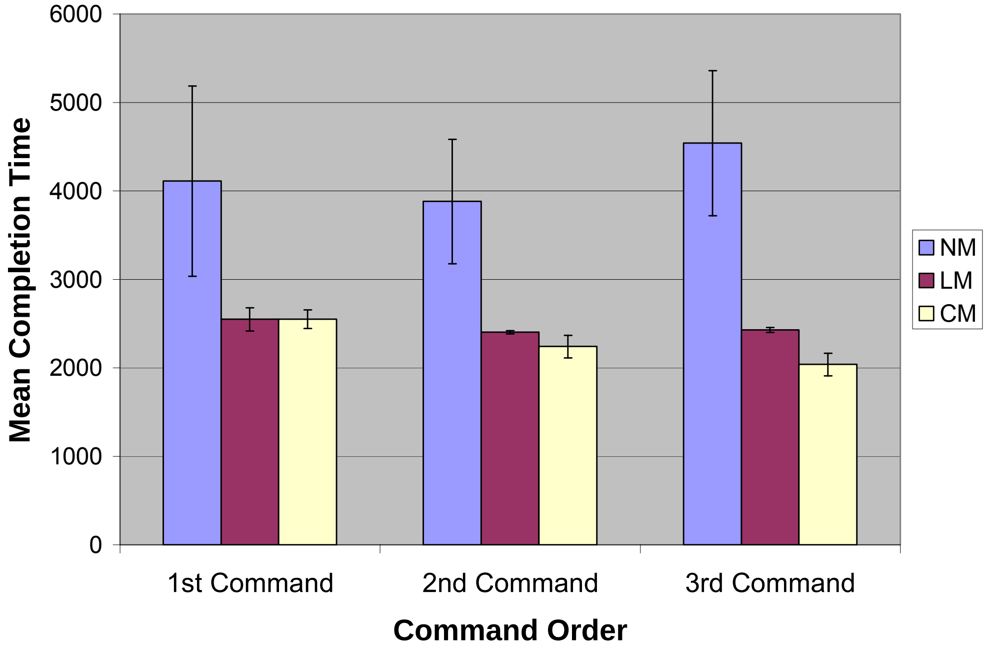


Figure 5.5. This chart shows the scenario where the agent has been given "go to" commands similar to the two previous figures' scenario, but the agent is also required to "evade enemy tanks." The error bars represent a 95% confidence interval.

Figure 5.5 displays the results for the three different movement strategies for the successive commands to three locations. The CM strategy shows significant improvement from one command to the next as it learns the map. The NM fails to complete all three commands within the time limit in 55% of its trials. Only the successful trials are included in the figure's data.

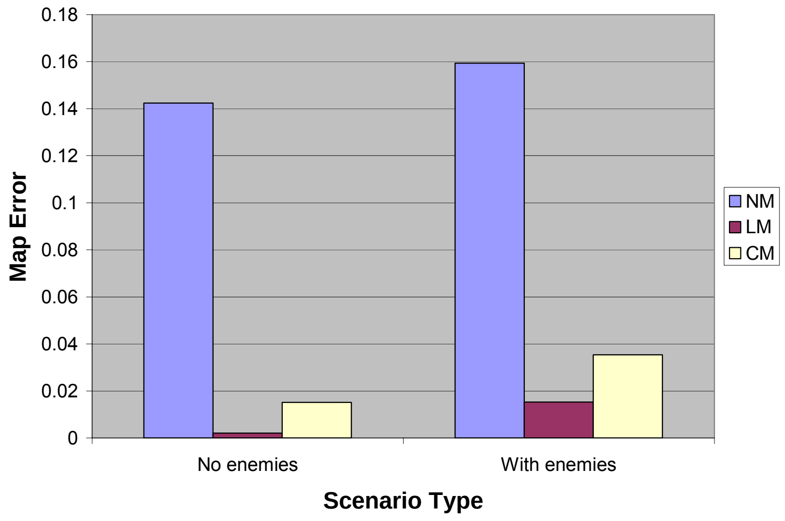


Figure 5.6. This figure shows the error rates of the maps produced in each of the three memory strategies for the "go to" command sets, with and without enemies to evade. An error is when the agent maps a location incorrectly (i.e. maps a building where a road should be).

In both this scenario and the related scenario from Chapter 4 without hostile objects, we can compare how well the agents did at interpreting the environment, using the same picture collected during the patrol scenarios. Again, in these cases, though this chart of the environment was being produced from visual information, the LM and NM strategies did not have access to the information to influence their movement. Figure 5.6 shows the relative errors (computed as in Section 5.5.2) for the different strategies in the two scenarios. In both cases, the LM shows the lowest error, while the NM shows a much higher error.

Finally, Figure 5.7 displays the measure of peril the agents faced from the hostile obstacles. The obstacles are considered to be dangerous if the agent is in their line of sight within a range of 20 cells. The obstacles in this scenario were taken to be tanks, which would be dangerous from a great distance. In these results for this scenario, the NM strategy varies wildly, the CM strategy shows significant improvement, and the LM strategy has the worst mean. Note that the higher the value, the more frequently the agent is in view of the hostile obstacles. This tells us that the target providing the most challenge in evasive behavior will use a cumulative memory strategy for that command.

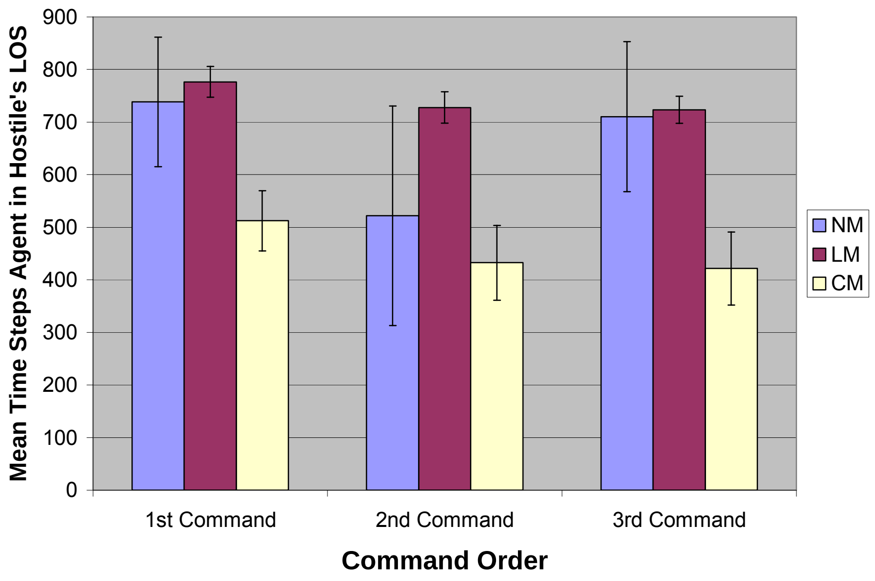


Figure 5.7. This shows the average number of time steps the agent was in the line of sight of a tank, ordered by the commands given. The error bars represent a 95% confidence interval.

5.5.4 Multiple Agents: The Pursuit Scenario

For the agents receiving a sequence of images of the more realistic environment as input, various pursuit strategies were employed to test how well the agents would perform in their environment. These were meant so show a gradual improvement in performance as certain features were included. These features were communication between the agents, changing the memory strategies used, and adding coordination.

While these features were varied between the agents, the target remained the same in all cases, following the behavior detailed previously.

While the strategies for the agent team varied, the target's behavior is consistent across all trials. The target will evade agents using a cumulative memory strategy. As the results demonstrated in Section 5.5.3, this method is more effective for staying out of view than the local memory strategy. However, for its patrol behavior, the goal of which is to give the target some semi-random movement through the environment when not pursued, the target uses a local memory strategy. This gives its movement more flexibility since it is not trying to chart the city and does not care about backtracking over observed territory.

The baseline scenario featured five completely independent agents that were assigned the goal to find and capture the target while patrolling the city. There is no communication between agents in this scenario. Therefore, there is no coordination of agent positions or paths. Agents may visually perceive one another in the environment, but it does not influence their behavior. The agents use a local memory strategy for their patrol and for their pursuit behaviors. Given the scenario limit of 5000 time steps to complete the task, the agents were able to capture the target in 70% of the trials. The mean completion time for this set is 2904 time steps. This indicates that while agents can solve the problem if each is given an individual goal, there is still room for improvement.

The first addition made to the agents, hypothesized to be crucial for their efficacy, is communication. Once enabled, agents that see the target can broadcast the location and agents that hear it have some knowledge of where to intercept the target. This is the only difference between this scenario and the previous one, and the results here were much improved, with the agents experiencing a 97% success rate and a mean completion time of 804 time steps, which is a significant improvement (a paired t-test reporting $P < 0.05$) over the original mean of 2904. A slight improvement in the mean completion time is also gained when a small agent avoidance influence of radius 1 is included in this scenario. The accuracy rises to 98%, and the mean completion time becomes 744.

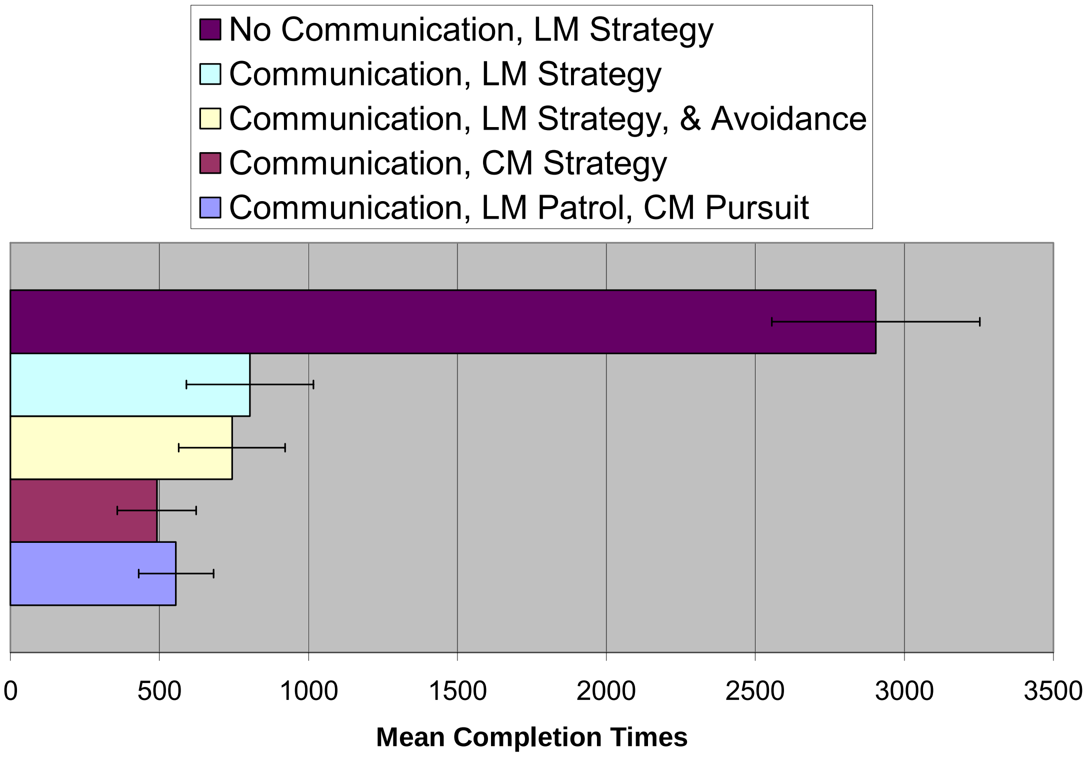


Figure 5.8. Chart of the mean completion times for agent teams using several different strategies. The worst features no communication between agents (70% success rate). There is significant improvement when agents adopt communication. There is further improvement when the agents adopt a cumulative memory strategy for either all their behaviors or just the pursuit behavior. The error bars here represent 95% confidence intervals.

The scenarios described so far have been using local memory strategies for both the patrol behavior and the pursuit behavior when an agent either sees the target or knows of the target position. Either or both of these strategies can be changed to using a cumulative memory strategy. While a cumulative memory strategy is not predicted to help the patrol much, it should improve the performance of the pursuit, because agents in this behavior are actively trying to get somewhere as quickly as possible, as opposed to simply exploring. When the cumulative memory strategy is applied to both patrol and pursuit behaviors, the accuracy increases to 99% and its mean completion time decreases to 491 time steps. When the cumulative memory strategy is applied to only the pursuit behavior, all trials successfully complete before the time limit, and the mean completion time is 556 time steps. Both of these mean completion times are a significant improvement over the local memory strategy, but do not have a significant difference from one another. As expected, given what was learned in Chapter 4, a cumulative memory strategy was able to improve upon a strategy where agents were able to acquire less information. However, it is apparent that this is also useful in a scenario where multiple agents are performing a task together. Figure 5.8 shows these compared results.

The question remains as to whether coordination is a useful feature for the agents, so the scenarios that used cumulative memory were also tested to see what improvement could be gained from using an avoidance influence and from allowing agents to broadcast their paths so that they could coordinate in an attempt to surround the target. Figure 5.9 shows the results for these influences on these scenarios.

The results here differ greatly depending on the memory strategy used. In the scenarios where cumulative memory is used for both of the two major behaviors of the agents, adding these coordination methods, either had no effect (as with the path coordination) or made the performance significantly worse (as in those cases with an avoidance influence). However, when a local memory strategy is used for the patrolling behavior, both coordination methods and their combined use yield a significant improvement. The benefit of these coordination methods appears to be mitigated when using a cumulative memory patrol. For the path coordination, this is likely due to the interference between patrol paths and pursuit paths, which influence each other though they themselves do not benefit from the interaction. The avoidance influence also evidently interferes with the patrol behavior, probably because the agents create distant goals for themselves when patrolling using cumulative memory. When paths cross, it is more difficult for two agents with different goals to reconcile them with just this influence. This situation is less likely to occur when agents are in pursuit of the target, because if they are in close proximity to one another, they are likely heading in the same direction and do not have conflicting paths.

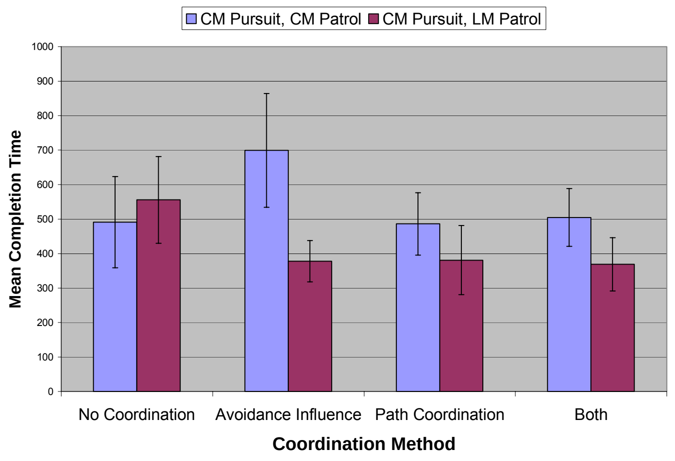


Figure 5.9. This figure shows the mean completion times for the different coordination methods in scenarios with different memory strategies involving cumulative memory. The first columns feature no coordination of the agents apart from communication, the second columns add an avoidance influence, while the third columns adds the ability to generate paths dependent on other agents' paths and positions. The fourth column set is for scenarios where both coordination methods are employed. The first column in each set features cumulative memory pursuit with local memory patrolling, while both behaviors use cumulative memory in the second column in each set. 100% of these results completed the task within the time limit except for the cumulative memory pursuit and patrol strategy with only the avoidance influence, where 99% of the tasks were completed. The error bars here represent 95% confidence intervals.

The above results demonstrate that there is an enormous benefit to communication and significant benefit to coordinating the agent behavior and to using a memory strategy with more information. This upholds the advantages of agents with a greater memory capacity observed in Chapters 3 and 4. Yet some questions still remain. First, while the avoidance influence is independent of the effects of individual memories since it is based on the proximity of agents to one another, path coordination is much more inclined to rely on these memories, which help determine the shape of the paths taken. The question that remains is whether the benefit seen with that coordination is due only to the coordination, or whether the inherent presence of memories, influencing the paths, adds to the improvement. This can be tested by running a scenario that features the minimum amount of memory necessary for the agents, which includes four cells (i.e. knowing their location and the three cells in front of them), against scenarios with increasing memory capacities, all of which use the path coordination.

A scenario was tested in which agents used local memory strategies for patrolling and cumulative memory strategies with path coordination for pursuit, but with a requirement that a random old memory be removed when adding new a memory to the agent map based on the visual data if the memory capacity is exceeded. This effectively kept a limit on the size of the agent memory, though the contents could change and be updated as the agent moved through the environment. During this test, as the capacity was increased by a factor of two, the mean dropped. By the time the threshold for removing old memories is 32, there was a significant improvement (a paired t-test reporting $P < 0.05$) over the scenario with the minimum memory and no significant difference between its performance and that of the scenario with unlimited cumulative memory. These results, depicted in Figure 5.10, uphold what was observed in Chapter 3, that a limited, incomplete memory of the environment could improve performance.

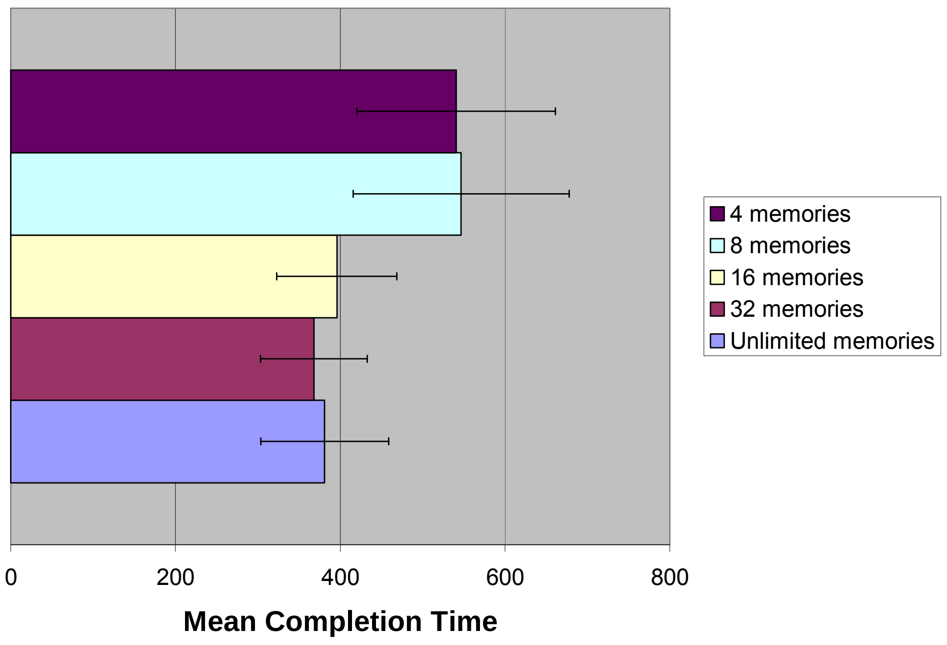


Figure 5.10. Results of the scenarios featuring limited memory. These results improve as the agent memory capacity is increased, allowing the agent to remember more cells. Once the agents can remember 32 cells, it is performing on the same level as the agents with unlimited memory. All trials of these scenarios completed within the given time limits. Not included in this chart are the results where the agent has a limit of no memory cells. These agents perform much worse, with below 100% accuracy and a mean completion time in the thousands. The error bars here represent 95% confidence intervals.

Figure 5.11 displays a sample composite map showing a single time step of the remembered cells of agents with limited memories, which cover a portion of the environment around them. Even though agents do not tend to remember distant locations given the frequent updates of memory, the collective memory has an impact because the agents are often trying to find ways to effectively spread out and surround an area they have recently explored and where they are told the target is estimated to be located.

Knowing more than a few features is very useful for this, and shows that a memory strategy located somewhere between what I have termed local and cumulative, which resembles the simple memory of Chapter 3, is just as effective as the CM strategy.

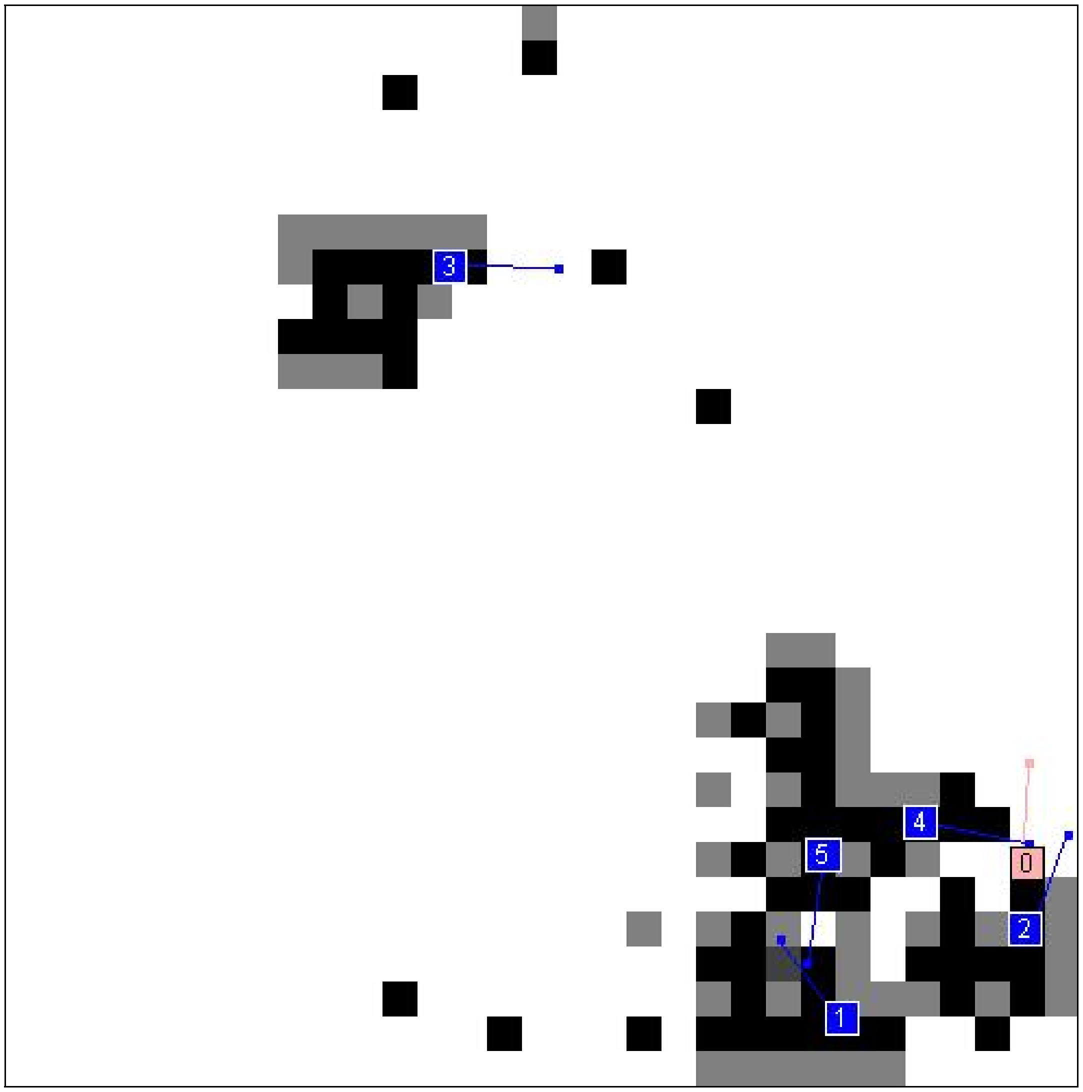


Figure 5.11. This is a composite depiction of the maps constructed by each of the agents in the environment. The memory limit is 24 in this case. The target's memory, which is always unlimited, is not displayed. Black areas are streets remembered by at least one agent, gray areas are buildings remembered by at least one agent, and white areas are unknown to anyone. Some areas that are in between the normal shades of black and gray are places where the agents have differing memories of what is there. Agents are indicated by numbered boxes outlined in white. The box numbered 0 is the target. The agent numbered 2 is pursuing the target, and agents 1, 4, and 5 are in its broadcast range.

5.6 Discussion

There are three main conclusions to emerge from the computational experiments of this chapter. First, the results support the hypothesis that adding an individual working memory of the obstacles encountered by an agent team can significantly improve both its success and efficiency in accomplishing the pursuit scenario, even when extended to this environment of greater realism and for a team of agents with environmental information limited to a sequence of 2-dimensional images of the 3-dimensional scene. This was found to be the case even when the agent team was already benefiting from communication and coordination. Second, the results also support the hypothesis that even when the size of this individual working memory is limited, the agent team still experiences a significant improvement in its efficacy and efficiency. Third, the results indicate that a different memory strategy for different tasks works best for the agent team. Local memory is found to be better for the patrol behavior, while cumulative memory strategy is found to be better for the pursuit behavior.

This chapter presented a pursuit scenario for two models that combined features presented in the earlier chapters to show how they would work together when applied to a scenario requiring multiple agents. The first model is a simple discrete space 2-dimensional model and was used to do preliminary testing for what might or might not work in the more complex model. These included features such as communication, agent coordination, and path prediction. While communication was extendible to the more realistic model, most coordination influences and prediction could not be extended, given the limitations on agents' knowledge of the environment and the difficulty in tracking the more evasive target in the more realistic model.

The bulk of the chapter was concerned with the more challenging environment viewed in a sequence of images, an extension of the previous chapter's model. First, the efficiency of the strategies was examined for some key behaviors in the pursuit scenario. It was determined that the cumulative memory strategy was most effective at covering an area during a patrol, but that the local memory strategy, although it could not remember all places it visited, also covered a wide area. Yet with no memory, the agent was unable to effectively patrol much of the environment except for the same corridor again and again. This indicated that the local and cumulative strategies had potential to be very useful in the pursuit scenario, while the strategy with no memory would have little success.

The cumulative memory strategy was also demonstrated to have a substantial advantage when it came to remaining out of the line of sight of hostile obstacles in the environment. This is because it remembered the location of these obstacles even after it has seen and successfully evaded them. In the case of "no memory" and "local memory" strategies, these objects are evaded and then forgotten. They can potentially return to a road where they are seen without seeing the hostile obstacle and remain in its line of sight without realizing it. Yet the agent with a cumulative memory strategy is able to recall the positions of the obstacles when they are distant, and it can evade them even without seeing them again. While some forgetting is required for the strategy when the obstacles are not stationary, this feature can be added to the cumulative memory, whereas the other strategies still forget too quickly, and the cumulative memory was chosen as the memory strategy for the target's evasive behavior in the following pursuit scenario.

The model was extended to multiple agents operating in the more realistic city environment, attempting to capture an evasive target. They were given a variety of methods to improve their performance. As expected, a local communication radius, where agents are able to broadcast the estimated location of the target, improved performance greatly, allowing agents to converge on the target more easily.

The scenario also further demonstrated the advantages of the cumulative memory strategy for pursuit over a local memory. Again, this is not surprising given the agent is following a moving target and clearly can find advantages in knowing the environment if it has to retrace its steps at any point. These strategies can be linked conceptually to the observations described in Chapter 3, if one considers that the local memory allows for the immediate effects of memory observed there, while cumulative memory allows for both the immediate effects and an impact in the long run when the agent needs to pass through previously visited territory. Even more interesting than the advantages of the cumulative over the local memory strategy is the fact that a combination of strategies tended to work quite well. In cases with coordination, it worked much better for the pursuing agents, where they would patrol using a local memory strategy, but pursue the target with a cumulative memory strategy. This gave them flexibility in their searching, while allowing them to make decisions, potentially informed by memories, when trying to quickly reach the target's location upon hearing a broadcast.

Two types of coordination were examined in the model, path coordination and an avoidance influence between agents with a radius of 1. While not consistent in improving the performance, they did make significant improvements in the cases where two different memory strategies were used for the pursuit and the patrol. While this is not surprising with the path coordination, which was designed so the agents would tend to surround the target, it is surprising that the avoidance influence had such a significant impact. It should be noted that when the radius of this influence is increased, the effect is detrimental, so it proved effective mostly as a means of ensuring agents did not get stuck when they collided in the environment.

Finally, it was shown that even a limited memory of the environment could give the agents a significant improvement, even when improvements were already present due to communication, path generation, and coordination. More surprisingly, the memory is not required to be very large. A significant improvement was gained when each agent was limited to remembering only a square root of the number of cells in the environment. There is perhaps a tradeoff in the benefits of using a complete map in that agents do not create maps with 100% accuracy. This can lead to agents being forced to take less efficient paths when better ones are available in the environment. Yet when memory is limited, erroneous locations stored in an agent's chart have a greater possibility of being eliminated, thereby giving agents greater flexibility. Balanced with the decreased knowledge of the environment, this could explain why the results are not significantly different. Another more likely explanation for the similarity of the results is that agents only need to have a few memories of the environment and the rest are redundant. The improvement witnessed here supports Chapter 3 and extends it to a model of greater complexity, demonstrating that a simple, limited memory of its environment distributed among the agents is still relevant.

Chapter 6:

Discussion

6.1 Summary and Limitations

Traditional particle systems are not only difficult to control and direct in performing specific tasks, but they are also limited to rather simple conditions, rules, and environments [Reynolds, 1987; Huth, 1994; Reynolds, 1999]. Extending these relatively simple systems to allow for behaviors beyond mere reflex responses to presently observed stimuli is an important step toward developing a significant collective intelligence. There has been initial work looking into adding a top-down control of behavior to the systems [Reynolds, 2000; Rodriguez, 2000] and there have been efforts to extend particle swarm systems with local interactions to perform problem solving behavior both in simulation [Rodriguez, 2005; Lapizco-Encinas, 2005] and robotics [Jones, 2003]. However, while the particles' local interactions give rise to interesting and useful behaviors, they are not typically designed to learn the initially unknown features of their environment or to remember them. Instead, they react to presently observable stimuli or forces pulling them to locations in the environment, and in some recent instances, a high-level controller. Yet, granting each individual a limited-capacity working memory of environmental features offers the potential to preserve the dynamics of the particle system's local interactions, while improving the efficiency of the system when exhibiting goal-driven behavior as the particles acquire information about their world.

The self-organizing collective behavior of particle systems frequently studied in abstract environments has also been extended to work in robotics and other more real-world settings [Hosokawa, 1998; Jones, 2003]. The ability for agents to maintain and move in these settings based on an individual working memory also may be extended to agents of greater complexity in scenarios and domains of greater realism. Map formation with autonomous robots has been an area of interest in recent years. One study looked at coordination techniques to reduce the time required to create maps [Burgard, 2000]. Another study examined robot system where robots built and integrated a map of the environment. They were subsequently used to locate an object in the map and guard it in fixed positions [Konolige, 2004]. A recent study used a centralized algorithm for integrating maps created in a multi-robot system, using manifold representations of the maps rather than planar representations to facilitate this process [Howard, 2006b]. The goal in these systems is usually map construction and synthesis. They do not examine the influence of these maps as a working memory on the system and the way the collective can gain improvement through memory that remains distributed in tasks unrelated to map making that require travel through the environment. The systems also frequently rely on centralized processing, which is not true in this dissertation.

Methods in robotics for acquiring information for building environmental maps can vary from using sonar [Mataric, 1990; Konolige, 2004] to laser range-finders [Burgard, 2000; Howard, 2006a]. However, the agents of interest in this dissertation are restricted to observing their environment via a sequence of camera images. In this case, building and maintaining the individual working memory remains beneficial for efficiency and becomes important for efficacy.

In this dissertation, I have proposed and demonstrated that extending particle systems to have a limited memory distributed among its members leads to significant benefits in performing simple tasks, while preserving all of the basic behaviors of a particle system. I then defined and designed a more complex agent that takes the agent from the abstract 2-dimensional universe and places it in a more realistic environment, where the agent receives a sequence of 2-dimensional images of a 3-dimensional urban environment, forcing the agent to be responsible for gathering information about the environment relevant to its behavior from this sequence. Here and in a final system of multiple agents of this kind, certain benefits of working memory became crucial to the system. Strategies that preserved a local or cumulative memory were required to increase the success rate of commands, as well as improve the efficiency. The benefits due to memory witnessed in the particle system, where the improvement in efficiency was observed to be both gradual and immediate, remained relevant in the multi-agent system.

In a particle system, it was hypothesized that memory would provide a significant benefit to agent behavior in scenarios where agents were making multiple tours of an environment wherein obstacles were scattered. The expectation was that performance would improve gradually as the collective traveled the environment, was exposed to obstacles, and remembered them during future tours of the same area. It was discovered through experimental trials that the agents did indeed gain a significant improvement in many scenarios.

A surprising result was the discovery that the collective could also gain an immediate benefit from the addition of memory. This meant that memory not only could improve the behavior over time, but could provide a benefit immediately as well. This was due to the added influences the memory would immediately exert on an individual as soon as it was added, which, in influencing the individual’s movement, indirectly influenced the movement of the entire collective, other members of which did not necessarily share the memory. This helped minimize the time spent by the collective trapped in “blind alleys” as it tried to pursue its goals.

It was also observed that as the agents made more and more tours of the environment, their collective was generating a rough map of the environment, essentially an outline of the relevant parts of the terrain obstacles (see Figures 3.10 and 3.11, for example). While no individual agent knew the entire map and each only had a scattered knowledge of the locations of a few obstacles throughout the environment, they collectively did have an internal picture of what the environment looked like. In particular, the combined map of the agent collective demonstrated those features of the environment that were pertinent to the tours, that is, the places where agents were likely to be obstructed on their journey through the simulated world.

In addition to the visually interesting nature of the map generated by the particle swarm’s superorganism, though the map was distributed among its individual parts, each memory had the potential to affect the behavior of each particle, either directly (if an individual agent had the memory) or indirectly (if an individual agent updated its movement based on the position and velocity of a neighbor, either with the memory, or itself influenced by the memory). This remained true even in the case of the multi-agent system in the urban environment, where agents maintained cumulative memories and coordinated their paths. Though no agent necessarily knew the entire map, the knowledge of other agents’ paths could influence their movements. Should a neighboring agent choose a particular path influenced by its memories, if the paths were coordinated, another agent’s decisions would be affected. Therefore, though memories were individual to the members of the collective system, they could exert an influence on the entire collective.

Additionally, the results from both the particle system scenarios and the multi-agent pursuit scenario indicated that complete knowledge of all environmental features is not necessary for success. Agents limited to accumulated environmental observations, and even those limited to a constantly updating memory that could hold only a few locations out of the entire observed environment, still gained a significant benefit from the capability. This demonstrates that while even a simple limited memory could improve the efficiency of a simple particle system, it remained relevant even when applied to a complex multi-agent system, where control and coordination were more complicated than the simple interactions of the particle swarm.

Having some capacity for memory was also shown to be crucial for efficiency in this pursuit scenario multi-agent system. Simple reactive behavior between agents was no longer able to efficiently guide the agents through an environment cramped with obstacles having thin paths between them, where the agent had only a very limited view of its environment at any given time step. Because each image at a time step provides no outside context for what surrounds the agent and in some cases makes even gauging distances difficult when an agent is close to an obstacle, the ability to accumulate information to form a map of the environment, much as internal maps are built in the particle system scenarios, allows these more realistic agents to move based on these individual internal maps.

The different strategies of local and cumulative memories also reinforce the results seen in the particle system that memory can have both an immediate and a gradual effect. The local memory taken alone shows the immediate benefit granted to the more complex agents as the accumulated memories allow the agents to avoid collisions with obstacles and traverse the environment more efficiently. Yet, this has no gradual effect as the map of these surroundings is forgotten by the time the agent has moved on to a new area. When returning there would be no added benefit over what was immediate. On the other hand, the cumulative memory, which does preserve this information, receives the immediate benefits witnessed in the local memory, as well as the gradual improvement, when it is required to plot a path across an area it has previously charted. This demonstrates that this result, noticed in the abstract particle system domain, appears in the scenarios of the more complex multi-agent systems too. Even when the complexity of the environment—and consequently the agent behavior—is increased, the importance and impact of memory either remains intact or grows even more significant.

Despite the successes of these agents in their environments, there are some limitations that were necessary in order to quickly and efficiently test the behavior in Chapter 5. The communication interpretation and production for this multi-agent system was simplified so that processing time did not have to be devoted to the constant stream of information that would have to be processed at every time step by the neural networks as agents communicated both the observed location of the target in the environment and their own planned paths. This information in turn would likely require much larger networks because information being passed would include a wide variety of possible numbers for coordinates in the environment. The amount of information that would need to travel in this manner would slow the agents, so it was simplified to a straightforward transfer of coordinates and paths between agents in range of one another.

The other key limitations of the simulated environment were those of the object segmentation and identification and the movement detection. Object recognition was simplified for this agent model in such a way that the agent was able to recognize an object from the background, but required a contrived method of distinguishing them from one another by using thumbnails representing the objects at “close range.” This simplification was used because of the difficulty of object segmentation, an open problem that is not a central issue of this dissertation. For similar reasons and because of the quick movement steps taken by the agents and the target in the environment, movement detection was simplified so that the agent could know by observing the target, the angle of its velocity.

6.2 Contributions

By extending simple particle systems with a limited distributed memory capacity and by building upon and applying these techniques to a more complex multi-agent system, this work made the following general contributions:

- *I demonstrated the immediate and gradual benefits of adding a limited distributed working memory to a self-organizing particle system with goal-driven behavior.* In this way, agents given memory are able to influence one another’s movement via their own avoidance of obstacles and their coordinated accelerations. This simple addition gave a significant improvement to the efficiency of agents moving in the environment, making tours of various goal states, both in long-term effects as the agents remembered previously seen obstacles at a distance and in immediate effects of indirectly communicating to the entire system observed memories through changes in movement. It was also demonstrated that the collective was able to obtain a rough picture of the relevant environment, even though no individual knew the entire map.

- *I created a new agent using self-organizing maps and other neural network methods that can process a sequence of 2-dimensional images as it moves to acquire visual information from a simulated 3-dimensional environment, interpret sentence-length commands, and produce sentence-length observations.* Using self-organizing maps and neural networks, the agent was trained to recognize environmental features and objects. Neural networks were also trained to recognize sequences of phonemes constituting full sentence commands. This involved generating unique representations for the words and then the complete sentence with multi-winner self-organizing maps and ultimately using these to encode the sentences into their features. Networks were also trained to produce sequences of phonemes constituting full sentence observations. This then required the design of specific behaviors in the agent at a higher level to achieve appropriate reactions to manifold incoming stimuli, as well as the design of memory management techniques to keep track of its knowledge of the environment.
- *I demonstrated through experimental simulations that the more realistic agent with a working memory was capable of performing tasks in the urban environment efficiently*. I established the advantages given to the agent when operating in the environment while using its working memory of the observed

locations of obstacles and objects over a method using no environmental memory. Using memory prevented collisions with environmental features, a greater success rate for the different commands, and improved efficiency in executing them. I determined that in addition to an improvement given by a cumulative memory of all environmental features observed, the agent's performance could even be helped by a local memory of recently observed stimuli.

- *I developed a system of multiple agents, capable of communicating with one another, coordinating their movement behavior, and utilizing the memory strategies developed in a simulated urban environment.* In a pursuit scenario making use of the different behaviors these agents could perform, this work established through experimental simulations that a combination of different memory strategies for different tasks proved most efficient and that a limited memory of the environment distributed among the agents could produce significant improvement in the efficiency of the agent collective, even independent of the benefits of communication and coordination. Additionally, I demonstrated that of the strategies developed for the management of the agents' memory, a combination of local and cumulative memory ultimately proved to provide the most benefit.

6.3 Future Work

This dissertation has explored the effect of adding working memory to agents of increasing complexity. The main purpose here has been to demonstrate that these effects remain significant even in more challenging environments. However, there is still a great deal of room for future work and exploration of this topic for these and other agents.

At present, memory in the agent system plays a direct role in influencing the behavior, but it does not change the nature of the behavior of the agent which is trained beforehand to recognize the environmental features. Allowing an agent to learn on-line as it experiences the environment and acquires memories would make the agent more adaptable to scenarios where it is necessary to enter an environment with as few expectations as possible. In these situations, memory has the potential to be even more crucial as the agent can use relevant examples experienced in a scenario to train and determine its behavior for later. In this case, memory would no longer be limited to knowing a specific feature of the environment at a location, but previous experience could be used to extrapolate a judgment on an observation at a new location based on the similarity to what has been stored in memory. If, for example, an agent has determined that an area was an obstacle, and if an image via the camera appears bearing resemblance to the remembered location, the same determination could be made here. This allows working memory to be extended to more than just recollections of specific environmental features observed and commands issued. Memory could now consist of features independent of some context where the agent is able to apply them to novel circumstances in order to make decisions. To make this sort of extension of the agent more successful, the processing of the sequences of input images may require improvement, using more sophisticated methods of image segmentation apart from neural networks, such as clustering or histogram-based methods. Finally, how the agent plans its long term movements could adapt other algorithms, for instance probabilistic roadmaps

[Kavraki, 1996; Nissoux, 1999; Bayazit, 2002]. Though these typically assume environmental information, they could be adapted to fit the limits of an agent's visual information and environmental memories.

In addition to giving the agent a greater ability to adapt its behavior, other features for the particular scenario could be explored. At present, agents using the sequence of images have limited environmental information at any given time step. They can essentially only see straight ahead. In a task where searching for a moving target is required and spreading out to locate it is beneficial, the maximum coverage of the space is very different here than in an abstract 2-dimensional environment where agents have a complete view all around them limited only by some set radius. The distance agents see with their first-person perspective is limited only by the ability of their self-organizing maps to recognize objects at any position in the image, regardless of distance. Instead of the traditional methods of having agents try to spread out their positions to search, it makes more sense for agents to spread out their visual space with a minimum amount of crossover. In this way, agents that are looking down the same stretch of space would be considered more in need of influencing one another's movement away from each other than agents who were close in position but looking at two totally different angles of the environment.

Another method for coordination that adjusts the behavior of the simpler models for more complex agents planning their paths through the environment is to change the nature of what is being coordinated. Instead of having agents update their acceleration for the next time step based on the position and velocity of neighbors, which no longer applies when agents are planning paths, the paths themselves could be treated as particle systems. With this approach, the goal would be to create a minimum amount of overlap, while preserving the endpoints of the starting location and the goal and a consistent path of waypoints between them. Alternately, other methods, such as probabilistic road maps, are viable for work in concert with agent desires for a minimum path to the goal, a limited knowledge of environmental obstacles, and a limited knowledge of neighboring paths, which could also be treated as obstacles.

Finally, the neural networks that controlled and processed the visual and verbal information, removed for computational efficiency in the final pursuit scenario, could be integrated back into the more realistic agents for the multi-agent system. While this increases the cost and complexity of the system substantially, it also means that the agent system would be more adaptable to be trained for other scenarios. The agents here can be viewed as a step toward a hybrid architecture, where neural networks and self-organizing maps process and extract features from the environmental data and transmit this to a higher level system to make decisions based on the information presently available and information maintained in memory. This sort of architecture has been shown to be successful in several scenarios here, and its hierarchical nature suggests it can be applied to any manner of scenario requiring one or more communicating agents, where complete knowledge of the environment is not available to individuals but is approximated by the collective.

References

[1] Alahakoon, D., and Halgamuge, S. “Dynamic self-organizing maps with controlled growth for knowledge discovery,” *IEEE Transactions on Neural Networks*. 11 (3), 601-614, 2000.

[2] Alcazar, J. “A simple approach to the multi-predator multi-prey domain,” *International Conference on Complex Systems*. 2004.

[3] Andrecut, M., and Ali, M. “Competitive learning of fuzzy models,” *International Journal of Modern Physics B*. 16 (30), 4621-4639, 2002.

[4] Atherton, J., Hardin, B., and Goodrich, M. “Coordinating a multi-agent team using a multiple perspective interface paradigm,” *Proceedings of the AAAI*. 2006.

[5] Auda, G., and Kamel, M. “Modular neural networks: a survey,” *International Journal of Neural Systems*. 9 (2), 129-151, April, 1999.

[6] Ayers, J., Davis, J., and Rudolph, A. (eds.), *Neurotechnology for Biomimetic Robots*. Cambridge, MA: MIT Press, 2002.

[7] Baddeley, A. “Recent developments in working memory,” *Current Opinion in Neurobiology*. 8, 234-238, 1998.

[8] Baddeley, A. “Working Memory,” *Science*. 255, 556-559, 1992.

[9] Baddeley, A. *Human Memory: Theory and Practice*, Psychology Press, 1997.

[10] Baldassarre, G., Nolfi, S., and Parisi, D. “Evolving mobile robots able to display collective behaviors,” *Artificial Life*. 9, 255-267, 2003.

[11] Baldi, P. and Hornik, K. “Learning in linear neural networks: a survey,” *IEEE Transactions on Neural Networks*. 6 (4), 837-858, July, 1995.

[12] Banzhaf, W., Nordin, P., Keller, R., and Francone, F. *Genetic Programming: An Introduction.* San Fransisco, CA: Morgan Kaufman Publishers, 1998.

[13] Bayazit, O., Lien, J-M., and Amato, N. “Roadmap-based flocking for complex environments,” *10th Pacific Conference on Computer Graphics and Applications*. 104-113, 2002.

[14] Bedau, M. “Artificial life: organization, adaptation and complexity from the bottom up,” *TRENDS in Cognitive Sciences*. 7 (11), 505-512, Nov. 2003.

[15] Benda, M., Jagannathan, V. and Dodhiawalla, R. “On optimal cooperation of knowledge sources,” Technical Report, Boeing Advanced Technology Center, Boeing Computer Services, Seattle WA, 1986.

[16] Best, S., and Cox, P. “Programming an autonomous robot controller by demonstration using artificial neural networks,” *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing*. 157-159, 2004.

[17] Bonabeau, E., Dorigo, M. and Theraulz, G., *Swarm Intelligence*. New York: Oxford Univ. Press, 1999.

[18] Bonarini, A., Aliverti, P., and Lucioni, M. “An omnidirectional vision sensor for fast tracking for mobile robots,” *IEEE Transactions on Instrumentation and Measurement*. 49 (3), 509-512, June 2000.

[19] Bosman, R., van Leeuwen, W., and Wemmenhove, B. “Combining Hebbian and reinforcement learning in a minibrain model,” *Neural Networks*. 17, 29-36, 2004.

[20] Bousquet, F., Barreteau, O., Le Page, C., Mullon, C., and Weber, J. “An environmental modelling approach: the use of multi-agent simulations,” in *Advances in Environmental and Ecological Modelling*. Blasco F. (ed). Elsevier, Paris, 113-122, 1999.

[21] Burgard, W., Fox, D., Moors, M., et al. “Collaborative multi-robot exploration,” *Proceedings of IEEE International Conference on Robotics and Automation*. 1, 476-481, 2000.

[22] Caelli, T., Guan, L., and Wen, W. “Modularity in neural computing,” *Proceedings of the IEEE*. 87 (9), 1497-1518, Sept. 1999.

[23] Camazine, S., Deneuboug, J., Franks, N., et al. *Self-Organization in Biological Systems*. Princeton: Princeton University Press, 2001.

[24] Carlisle, A., and Dozier, G. “Adapting particle swarm optimization to dynamic environments,” *Proceedings of International Conference on Artificial Intelligence*. Las Vegas, NV, 429-434, 2000.

[25] Chen, S-H. “Modular recurrent neural networks for Mandarin syllable recognition,” *IEEE Transactions on Neural Networks.* 9 (6), 1430-1441, Nov. 1998.

[26] Chong, W. “Reflective reasoning.” Diss. University of Maryland, 2006.

[27] Clerc M., and Kennedy, J. “The particle swarm,” *IEEE Transactions in Evolutionary Computation*. 6, 58-73, 2002.

[28] Corchs, S., and Deco, G. “Large-scale neural model for visual attention: integration of experimental single-cell and fMRI data,” *Cerebral Cortex.* 12, 339-348, April, 2002.

[29] Couzin, I., Krause, J., Franks, N., and Levin, S. "Effective leadership and decision-making in animal groups on the move," *Nature*. 433, 513-516, 2005.

[30] Cowan, N. "The magical number 4 in short-term memory: a reconsideration of mental storage capacity," *Behavioral and Brain Sciences*. 24 (1), 87-114, 2001.

[31] Di Caro, G., and Dorigo, M. "AntNet: distributed stigmergic control for communications networks," *Journal of Artificial Intelligence Research*. 9, 317-365, 1998.

[32] Dorigo, M. "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*. 1 (1), 53-66, 1997.

[33] Dorin, A. "Physically based self-organizing cellular automata," in *Multi-Agent Systems—Theories, Languages and Applications*. Zhang, C., Lukose, D. (eds), Lecture Notes in Artificial Intelligence 1544, Springer-Verlag, 74-87, 1998.

[34] Drogoul, A., and Ferber, J. "Multi-agent simulation as a tool for modeling societies: application to social differentiation in ant colonies," *MAAMAW '92*. 3-23.

[35] Durstewitz, D., Seamans, J., and Sejnowski, T. "Neurocomputational models of working memory," *Nature Neuroscience*. 3, 1184-1191, 2000.

[36] Edwards, L., Peng, Y., and Reggia, J.. "Computational models for the formation of protocell structures," *Artificial Life*, 4, 61-77, 1998.

[37] El-Bakry, H., Abo-Elsoud, M., and Kamel, M. "Fast modular neural nets for human face detection," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. 3, 320-324, 2000.

[38] Elfes, A., Dolan, J., Podnar, D., Mau, S., and Bergerman, M. "Safe and efficient robotic space exploration with tele-supervised autonomous robots." *Proceedings of the AAAI Spring Symposium*. 104-113, 2006.

[39] Gerstner, W. and Kistler, W. "Mathematical formulations of Hebbian learning," *Biological Cybernetics*. 87, 404-415, 2002.

[40] Girosi, F., Jones, M., and Poggio, T. "Regularization theory and neural networks architecture," *Neural Computation*. 7, 219-269, 1995.

[41] Griffith, S., Goldwater, D., and Jacobson, J. "Self-replication from random parts," *Nature*. 437, 636, September 2005.

[42] Haarmann, H., and Usher, M. "Maintenance of semantic information in capacity-limited short-term memory," *Psychonomic Bulletin*. 8 (3), 568-578, 2001.

[43] Haykin, S. *Neural networks: a comprehensive foundation, 2nd edition*. Prentice Hall, New Jersey, 1999.

[44] Haynes, T., and Sen, S. "Evolving Behavioral Strategies in Predators and Prey," *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*. 1996.

[45] Haynes, T., Wainwright, R., Sen, S., and Schoenefeld, D. "Strongly typed genetic programming in evolving cooperation strategies," In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*. 271-278, San Francisco, CA, 1995. Morgan Kaufmann Publishers, Inc.

[46] Heppner, F., and Grenander, U. "A stochastic nonlinear model for coordinated bird flocks," *The Ubiquity of Chaos*. S. Krasner (ed.), Washington DC: AAAS, 1990, 233-238.

[47] Hicks, R., and Bajcsy, "Reflective surfaces as computational sensors," *CVPR99, Workshop on Perception for Mobile Agents*. 1999.

[48] Hodgins, J. and Brogan, D., "Robot herds," *Artificial Life IV*. R. Brooks & P. Maes (eds.), MIT Press, , 319-324, 1994.

[49] Horn, D., and Usher, M. "Parallel activation of memories in an oscillatory neural network," *Neural Computation*. 3, 31-43, 1991.

[50] Hosokawa, K., Tsujimori, T., Fujii, T., et al. "Self-organizing collective robots with morphogenesis in a vertical plane," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*. Leuven, Belgium, May, 1998.

[51] Howard, A. "Multi-robot simultaneous localization and mapping using particle filters." *The International Journal of Robotics Research*. 25 (12), 1243-1256, 2006.

[52] Howard, A., Sukhatme, G., and Matarić, M. "Multi-Robot Mapping using Manifold Representations," *Proceedings of the IEEE - Special Issue on Multi-robot Systems*. 94 (7) , 1360-1369, 2006.

[53] Hutchins, E., *Cognition in the Wild*, Cambridge, MA: MIT Press, 1995.

[54] Huth, A., and Wiesel, C. "The simulation of the movement of fish schools," *Journal of Theoretical Biology*. 156, 365-385, 1992.

[55] Ilmonen, T., Takala, T., and Laitinen, J. "Soft edges and burning things: enchanced real-time rendering of particle systems." *The 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision '2006*. Plzen, Czech Republic, 33-38, 2006

[56] Iwata, A., Kawajiri, H., Suzumura, N. "Classification of hand-written digits by a large scale neural network 'CombNET-II'," *IEEE International Joint Conference on Neural Networks*. 1021-1026, Nov. 1991.

[57] Jennings, A. *The Invisible Matrix: The Evolution of Altruism, Culture, Human Behavior, and the Memory Network.* Veritas Books, Mountain View, CA, 2006.

[58] Jim, K., and Giles, C. "Talking helps: evolving communicating agents for the predator-prey pursuit problem," *Artificial Life*. 6 (3), 237-254, 2000.

[59] Jones, C., and Mataric, M. "Adaptive division of labor in large-scale minimalist multi-robot systems," *Proceedings of the IEEE Internet Conference on Intelligent Robots and Systems*. 1969-1974, 2003.

[60] Kaelbling, L., Littman, M., and Moore, A. "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research.* 4, 237-285, 1995.

[61] Kangas, J., Kohonen, T., and Laaksonen, J. "Variants of self-organizing maps," *IEEE Transactions on Neural Networks*. 1 (1), 93-99, 1990.

[62] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M.. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automaton*. 12 (4), 566-580, 1996.

[63] Kennedy, J. "The particle swarm: social adaptation of knowledge," *IEEE International Conference on Evolutionary Computation*. 303-308, 1997.

[64] Kennedy, J., Eberhart, R., and Shi, Y. *Swarm Intelligence*. San Diego, CA: Academic, 2001.

[65] Kim, D. "Self-organization for multi-agent groups," *International Journal of Control, Automation, and Systems*. 2 (3), 333-342, September 2004.

[66] Klavins, E. "Automatic synthesis for controllers for distributed assembly and formation forming," *Proceedings of the 2002 International Conference on Robotics and Automation.* 3, 3296-3302, 2002.

[67] Kohonen, T. "The self-organizing map," *Proceedings of the IEEE.* 78 (9), 1464-1480, 1990.

[68] Kohonen, T., and Somervuo, P. "How to make large self-organizing maps for nonvectorial data," *Neural Networks*. 15, 945-952, 2002.

[69] Konolige, K., Fox, D., Ortiz, C., et al. "Centibots: very large scale distributed robotic teams." *9th International Symposium on Experimental Robotics (ISER-04).* Singapore, June 2004.

[70] Korf., R. "A simple solution to pursuit games," *Proceedings of the 11th International Workshop on Distributed Artificial Intelligence*. Glen Arbor, MI, Feb, 1992.

[71] Kurihara, K., Nishiuchi, N., Hasegawa, J., and Masuda, K. "Mobile robots path planning method with the existence of moving obstacles," *10th IEEE Conference on Emerging Technologies and Factory Automaton, 2005*. 1, 195-202, 2005.

[72] Lapizco-Encinas, G., and Reggia, J. "Diagnostic problem solving using swarm intelligence," *Swarm Intelligence Symposium*. 365-372, 2005.

[73] Lee, T., Ching, P., and Chan L-W. "Isolated word recognition using modular recurrent neural networks," *Pattern Recognition*. 31 (6), 751-760, 1998.

[74] Lenzitti, B., Tegolo, D., and Valenti, C. "Prey-predator strategies in a multiagent system," *Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception*. 184-189, July 2005.

[75] Luke, S., and Spector, L. "Evolving teamwork and coordination with genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference*. Stanford, July 1996.

[76] Mataric, M. "Environment learning using a distributed representation." *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*. 1, 402-406, 1990.

[77] Mataric, M. "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems*. 16, 321-331, 1995.

[78] McCook, C., and Esposito, J. "Flocking for heterogeneous robot swarms: a military convoy scenario." *39th Southeastern Symposium on System Theory*. Macon, GA, 2007.

[79] Melin, P., Gonzalez, C., Gonzales, F., and Castillo, O. "Face recognition using modular neural networks and fuzzy Sugeno integral for response integration," *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*. 1, 349-354, Aug. 2005.

[80] Miller, E., and Cohen, J. "An integrative theory of prefrontal cortex function," *Annual Review of Neuroscience*. 24, 167-202, 2001.

[81] Mishkin, M., Suzuki, W., Vargha-Khadem, F., and Gadian, D. “Hierarchical organization of cognitive memory.” *Philosophical Transactions: Biological Sciences*. 352 (1360), 1461-1467, 1997.

[82] Muller, S., Marchetto, J., Airaghi, S., and Kourmoutsakos, P. “Optimization based on bacterial chemotaxis,” *IEEE Transactions on Evolutionary Computation*. 6, 16-29, 2002.

[83] Nissoux, C., Simeon, T., and Laumond, J.-P. “Visibility based probabilistic roadmaps,” *Proceedings of the International Conference on Intelligent Robots and Systems*. 1316-1321, 1999.

[84] Nolfi, S., and Parisi, D. “Neural networks in an artificial life perspective,” *International Conference on Artificial Neural Networks*. 733-737, 1997.

[85] Nourbakhsh, I., Sycara, K., Koes, M., Young, M., et al. “Human-robot teaming for search and rescue.” *IEEE Pervasive Computing*, 72-78, 2005.

[86] O’Reilly, R., and Frank, M. “Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia,” *Neural Computation*. 18, 283-328, 2006.

[87] Pagac, D., Nebot, E., and Durrant-Whyte, H. “An evidential approach to map-building for autonomous vehicles,” *IEEE Transactions on Robotics and Automation*. 14 (4), 623-629, 1998.

[88] Pan, G., Dou, Q., and Liu, X. “Performance of two improved particle swarm optimization in dynamic optimization environments.” *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications*. 1024-1028, 2006.

[89] Pearlmutter, B. “Gradient calculatons for dynamic recurrent neural networks: a survey,” *IEEE Transactions on Neural Networks*. 6 (5) , 1212-1228, 1995.

[90] Reggia, J., Tagamets, M, Contreras-Vidal, J., et al. “Development of a large-scale neurocognitive architecture. Part 1: a conceptual framework,” *TR-CS-4814, UMIACS-TR-2006-33*. University of Maryland. June, 2006.

[91] Reynolds, C. “Big fast crowds on PS3.” *Proceedings of Sandbox*. Boston, MA, July, 2006.

[92] Reynolds, C. “Flocks, herds and schools,” *Computer Graphics*. 21, 25-34, 1987.

[93] Reynolds, C. “Interactions with groups of autonomous characters,” *Proceedings of Game Developers Conference*. 2000.

[94] Reynolds, C. “Steering behaviors for autonomous characters,” *Proceedings of Game Developers Conference*. 763-782, 1999.

[95] Riedmiller, M. “A direct adaptive method for faster backpropagation learning: the RPROP algorithm.” *Proceedings of the IEEE International Conference on Neural Networks*. 586-591, San Francisco, CA, 1993.

[96] Rodriguez, A. and Reggia, J. “Extending self-organizing particle systems to problem solving,” *Artificial Life*. 10, 379-395, 2004.

[97] Rodriguez, A., and Reggia, J. “Collective-movement teams for cooperative problem solving,” *Integrated Computer Aided Engineering*. 12, 217-235, 2005.

[98] Rueckl, J., Cave, K., and Koslyn, S. “Why are ‘what’ and ‘where’ processed by separate cortical visual systems? A computational investigation,” *Journal of Cognitive Neuroscience*. 1, (2), 171-186, 1989.

[99] Rumelhart, D., and Zipser, D. “Feature discovery by competitive learning,” *Cognitive Science*. 9, 75-112, 1985.

[100] Sahin, E., Labella, T., Trianni, V., et al. “SWARM-BOT: pattern formation in a swarm of self-assembling mobile robots,” *2002 IEEE International Conference on Systems, Man, and Cybernetics.* 4, October 2002.

[101] Schulz, R., and Reggia, J. “Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps,” *Neural Computation*. 16, 535-561, 2004.

[102] Serpen, G., and Corra, J. “Training simultaneous recurrent neural network with resilient propagation for static optimization,” *International Journal of Neural Systems*. 12 (3&4), 203-218, 2002.

[103] Shillcock, R., Ellison, T., and Monaghan, P. “Eye-fixation behavior, lexical storage, and visual word recognition in a split processing model,” *Psychological Review.* 107 (4), 824-851, 2000.

[104] Singh, K, and Fujimara, K. “Map making by cooperating mobile robots.” *Proceedings of the IEEE International Conference on Robotics and Automation*. 254-259, Atlanta, GA, 1993.

[105] Sipper, M. “Studying artificial life using a simple, general cellular model,” *Artificial Life.* 2 (1), 1-35, 1995.

[106] Smith, A., and Turney, P. “Self-replicating machines in continuous space with virtual physics,” *Artificial Life.* 9, 21-40, 2003.

[107] Spector, L., Klein, J., Perry, C., and Feinstein, M. "Emergence of collective behavior in evolving populations of flying agents," *Genetic and Evolutionary Computation Conference*. 61-73, 2003.

[108] Sutton, R. and Barto, A. *Reinforcement Learning*. The MIT Press, Cambridge, MA, 1998.

[109] Tagamets, M., and Horwitz, B. "A model of working memory: bridging the gap between electrophysiology and human brain imaging," *Neural Networks*. 13 (Special Issue), 941-952, 2000.

[110] Tan, K.C., Tan, K.K., Lee, T., Zhao, S., and Chen, Y. "Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning," *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*. Vancouver, Canada, Oct. 27-30, 2002.

[111] Tang, Z., Wang, X., Tamura, H., and Ishii, M. "An algorithm of supervised learning for multilayer neural networks," *Neural Computation*. 15, 1125-1142, 2003.

[112] Taniguchi, T., and Sawaragi, T. "Adaptive organization of generalized behavioral concepts for autonomous robots: schema-based modular reinforcement learning," *Proceedings 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Espoo, Finland, June 27-30, 2005.

[113] Terzopoulos, D. and Rabie, T. "Animat vision: active vision in artificial animals." *Videre: Journal of Computer Vision Research*. 1 (1), 2-19, 1997.

[114] Tu, X. and Terzopoulos, D. "Artificial fishes: physics, locomotion, perception, behavior," *SIGGRAPH*, 1994.

[115] Vail, D., and Veloso, M. "Multi-robot dynamic role assignment and coordination through shared potential fields," *Multi-Robot Systems*. 87-98, 2003.

[116] Wang, Q., Liu, L., Xie, G., and Wang, L. "Learning from human cognition: collaborative localization for vision-based autonomous robots," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China, Oct. 9-15, 2006.

[117] Weems, S. and Reggia, J. "Simulating single word processing in the classic aphasia syndromes based on the Wernicke-Lichtheim-Geschwin model," *Brain and Language*. 98 (3), 291-309, Sept. 2006.

[118] Weems, S., Winder, R., Bunting, M., and Reggia, J. "Running memory span: a comparison of behavioral capacity limited with those of an attractor neural network," *Submitted*.

[119] Wegner, D. “A computer network model of human transactive memory,” *Social Cognition*. 13, 319-339, 1995.

[120] Weiss, G., and Dillenbourg, P. “What is ‘multi’ in multi-agent learning?” in *Collaborative-learning: Cognitive and Computational Approaches*. P. Dillenbourg (ed.), Oxford : Elsevier, 1999.

[121] Winder, R., Cortes, C., Reggia, J., and Tagaments, M. “Functional connectivity in fMRI: a modeling approach for estimation and for relating to local circuits,” *NeuroImage*. 34, 1093-1107, 2007.

[122] Zhao, D., and Jin, W. “The study of coorperative behavior in predator-prey problem of multi-agent systems,” *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*. 90-96, April 2005.