Summer 2013

The Effects of Sensor Performance as Modeled by Signal Detection Theory on the Performance of Reinforcement Learning in a Target Acquisition Task

Nate Quirion
Embry-Riddle Aeronautical University - Daytona Beach

The Effects of Sensor Performance as Modeled by Signal Detection Theory on the Performance of Reinforcement Learning in a Target Acquisition Task

by

Nate Quirion

B.S. Embry-Riddle Aeronautical University, 2010

A Graduate Thesis Submitted to the

Department of Human Factors and Systems

in Partial Fulfillment of the Requirement for the Degree of

Master of Science in Human Factors and Systems

Embry-Riddle Aeronautical University

Daytona Beach, Florida

Summer 2013

Acknowledgements

The author thanks Dr. Dahai Liu for his help and guidance in the conceptualization and realization of this study. The author has learned much from him, and without his help this work would never have been completed. The author also expresses his gratitude to Dr. Liu as he has given the author a new professional passion and a large number of new skills.

The author would also like to thank his committee members, Dr. Albert Boquet and Dr. Andre Ludu, for their tremendous understanding, guidance, and patience during the duration of this work. No one could ask for a better group of mentors.

The author also wishes to thank his father and mother, David and Valerie Quirion, for all the life changing guidance and wisdom that have led the author to this point. Thanks are also given for the author's grandfather and grandmother, Benoit and Lorraine Quirion, who have shaped the author's life and choices in uncountable, incredible, ways. He would not be the same person without them.

The Effects of Sensor Performance as Modeled by Signal Detection Theory on the Performance of Reinforcement Learning in a Target Acquisition Task

by

Nate Quirion

This thesis was prepared under the direction of the candidate's thesis committee chair, Dahai Liu, Ph.D., Department of Human Factors and Systems, and has been approved by the members of the thesis committee. It was submitted to the Department of Human Factors and Systems and has been accepted in partial fulfillment of the requirements for the degree of Master of Science in Human Factors and Systems.

THESIS COMMITTEE:

Dahai Liu

Dahai Liu, Ph.D., Chair

Albert Boquet, Ph.D., Member

ALudu

Andrei Ludu, Ph.D., Member

Christina Frederick

MS HFS Program Coordinator

Department Chair, Department of Human Factors and Systems

Richard H

for Associate Vice President for Academics

Table of Contents

Abstract 1

Introduction 2

Literature Review 2

Unmanned Aircraft Systems. 3

Autonomy. 7

Reinforcement learning. 12

Sensor systems in UASs. 31

Literature Review Summary 37

Problem Statement 37

Hypotheses 38

Method 39

Experiment Description 39

Experiment variables description. 39

The environment. 40

Reinforcement Learning Model 41

Problem, state, action and reward formulation. 41

Markovian model of environment. 45

Target placement and interaction. 47

RL paradigm formulation. 48

Environment interaction. 52

Metrics 54

Agent Behavior. 56

Mission Performance. 58

TOP Map Comparison. 59

Results 61

Agent Behavior Results 62

Agent Rewards. 62

Incorrect Declaration Behavior. 69

Mission Performance 73

Agent Reward Analysis 76

Navigation Agent Reward Analysis 76

Search Agent Reward Analysis. 82

Incorrect Declaration Behavior. 88

Mission Time. 98

TOP Grid Comparison. 99

Discussion 103

Conclusion 105

General Remarks 105

Future Work 106

Closing Remarks 107

Appendix A: Mission Time Plots 115

List of Figures

Figure 1. The RL architecture ..……………………………………………………………….12

Figure 2. The hierarchical RL interaction diagram ……………………………………...... 26

Figure 3. Example probability density functions of target and noise signal values …………… 34

Figure 4. *:* Hit Rate and False Alarm rates for constant alpha values…………………….………. 36

Figure 5. The overall architecture of the experimental RL system ……………..………...…… 42

Figure 6. The Mission Time plot ………………………………………….………..………... 45

Figure 7. The transition model for the Search Agent ………………………..………………… 46

Figure 8. The position map …………………….………………………………………………. 48

Figure 9. An example of a Navigation Agent Reward plot ..…………………………………...… 57

Figure 10. Example of the Mission Time performance plot …………………………………… 59

Figure 11. The average reward for the first TD Navigation Agent ..…………………………... 63

Figure 12. All TD Navigation Agent episodic rewards ………………………….…….……… 64

Figure 13. All MC Navigation Agent episodic rewards …..………………………….…….…..… 65

Figure 14. All TD Search Agent rewards ………………………..…………………………...66

Figure 15. Smoothed TD Search Agent rewards ………………………...………………….. 67

Figure 16. All MC Search Agent rewards …………….……………………………………….. 68

Figure 17. Smoothed MC Search Agent rewards …………………..……..………………… 69

Figure 18. Average TD Incorrect Declarations………………………………………………. 70

Figure 19. Smoothed average TD Incorrect Declarations……..……………………………..… 71

Figure 20. Average MC Incorrect Declarations……………………………………………… 72

Figure 21. Smoothed average MC Incorrect Declarations……….……………………………... 73

Figure 22. Mission Time plot for first TD Agent ………………………...………………… 75

Figure 23. Box plot of the TD Navigation Agent Rewards …………………….…………… 77

Figure 24. Box plot of the MC Navigation Agent Rewards ………...………….………… 80

Figure 25. Box plot of TD Search Agent average rewards …...……...…………………...… 83

Figure 26. Box plot of TD Search Agent average rewards …………………….…………… 86

Figure 27. Box plot of TD Agent Incorrect Declarations …………….…………………... 89

Figure 28. Box plot of MC Agent Incorrect Declarations ……………………..…………… 92

Figure 29. Plot of the TD Agent declaration action values …………………..………………… 96

Figure 30. Plot of the MC Agent declaration action values ………..…………………………… 97

Figure 31. Perfect TD and MC Agent Mission Time plots ……………………...…………... 98

Figure 32. Fourth TD and MC Agent Mission Time Plots ……………………..……………… 99

Figure 33. Perfect TD Agent TOP Maps …..……………..……......……………………………. 100

Figure 34. First TD Agent TOP Map …………………………………………………………. 101

Figure 35. TOP similarity plots ………………………………...……..………………… 102

List of Tables

Table 1. Possible responses of a sensor with two possible states of the environment ………..... 35

Table 2. Experimental systems and their attributes……………………….……………..….. 40

Table 3. TOP state threshold definition ……………………………………………...... 43

Table 4. Sensor characteristic values and descriptions…………………………..……..… 53

Table 5. Conditional probability notation………………………..……………………... 53

Table 6. Experiment output and format list…………………………………………….. 56

Table 7. Mission completion rates…………………………………………………...… 74

Table 8. Average mission time decision step values of last 50 episodes…..…………………... 76

Table 9. TD navigation agent steady-state descriptive data ………………………………..… 77

Table 10. TD navigation agent ANOVA results …………..…………………………………... 78

Table 11. Tukey HSD test results………………………………………………………… 78

Table 12. MC navigation agent steady-state descriptive data………………….…………..… 79

Table 13. MC navigation agent ANOVA results………………………..………………..... 81

Table 14. MC navigation agent Tukey HSD results……………………………..……...… 81

Table 15. TD & MC navigation agent descriptive data……………………………........... 82

Table 16. TD search agent descriptive data …………………………………………………… 83

Table 17. TD search agent ANOVA results …………………………………………………… 84

Table 18. TD search agent tukey HSD results…………………………………………............. 84

Table 19. MC search agent descriptive data……………………………………....................… 85

Table 20. MC search agent ANOVA results……………………………………………….... 86

Table 21. MC search agent Tukey HSD results……………………………………..………. 87

Table 22. TD & MC search agent descriptive data…………………………………………..… 88

Table 23. TD & MC search agent ANOVA results……………………………………….…..... 88

Table 24. TD incorrect declaration descriptive data…………………………………..……. 89

Table 25. TD incorrect declaration ANOVA results…………………………………………... 90

Table 26. TD incorrect declaration Tukey HSD results ………………………………………. 90

Table 27. MC incorrect declaration descriptive data…………………………………………..… 91

Table 28. MC incorrect declaration ANOVA results …….…………………………………..... 92

Table 29. MC incorrect declaration Tukey HSD results …………………………................. 93

Table 30. TD & MC incorrect declaration behavior descriptive statistics ………………….….. 93

Table 31. TD & MC incorrect declaration behavior ANOVA results ………………….…….... 94

Table 32. Search agent state action values ……………………..…………………………... 95

Abstract

Author: Nate Quirion

Title: The Effects of Sensor Performance as Modeled by Signal Detection Theory on the Performance of Reinforcement Learning in a Target Acquisition Task

Institution: Embry-Riddle Aeronautical University

Year: 2013

Unmanned Aerial Systems (UASs) today are fulfilling more roles than ever before. There is a general push to have these systems feature more advanced autonomous capabilities in the near future. To achieve autonomous behavior requires some unique approaches to control and decision making. More advanced versions of these approaches are able to adapt their own behavior and examine their past experiences to increase their future mission performance. To achieve adaptive behavior and decision making capabilities this study used Reinforcement Learning algorithms. In this research the effects of sensor performance, as modeled through Signal Detection Theory (SDT), on the ability of RL algorithms to accomplish a target localization task are examined. Three levels of sensor sensitivity are simulated and compared to the results of the same system using a perfect sensor. To accomplish the target localization task, a hierarchical architecture used two distinct agents. A simulated human operator is assumed to be a perfect decision maker, and is used in the system feedback. An evaluation of the system is performed using multiple metrics, including episodic reward curves and the time taken to locate all targets. Statistical analyses are employed to detect significant differences in the comparison of steady-state behavior of different systems.

Introduction

There is a general trend in robotics and the Unmanned Aerial System (UAS) industry in particular to create systems with more autonomy and conduct a wider variety of missions (Clough, 2005; Singer, 2009). At present, one of the salient roles of military UASs is that of a reconnaissance asset (Singer, 2009). An autonomous system for the reconnaissance role would need to be able to effectively search an environment, maximizing its efficiency to locate targets given an imperfect sensor suite for target identification. The system would also need to be able to adapt to any perceived target behavior, especially in military applications. The ability of future unmanned systems to adapt to their operating environment has been identified as a key area for future development (Panella, 2008; Singer, 2009; Winnefield & Kendall, 2012). There are currently few studies on how the performance of a sensor system affects the capabilities of autonomous UASs. The objective of this study is to investigate the effects of sensor performance, more specifically, sensor sensitivity, on the ability of autonomous reconnaissance UASs and vehicles to complete their intended mission. A unique aspect of this study is the simulation features simulated feedback from a human supervisor. The human supervisor is modeled as a perfect decision maker in this study. The human decision maker never falsely confirms a real target nor misses a real target declared by the system.

Literature Review

The literature review is organized into four general sections. The first section outlines the capabilities and types of UASs in use today. The section also presents evidence of what is expected of UASs in the near and mid future, specifically the autonomous aspects that are expected of those future unmanned systems. The next section discusses the principals behind autonomy, and what capabilities and characteristics a system must exhibit in order to be classified as an autonomous system. Some examples of rule based systems used to achieve autonomous control are also presented. The third section of the literature review contains a review of Reinforcement Learning (RL) algorithms and their mechanics. The section begins with a general discussion of why and how RL algorithms behave and introduces some standard methods of implementation. The section ends with a review of RL experiments, simulations, and studies that implemented RL algorithms for the purpose of navigating a robotic vehicle in an environment without a priori information. The fourth section discusses current studies and experiments on sensors used in vehicles for autonomous task performance. The fourth section also introduces the Signal Detection Theory technique used to model the general behavior of a sensor in this research.

Unmanned Aircraft Systems.

Unmanned Aircraft Systems (UASs) can be applied to many different roles. Perhaps the most prominent and well known application of these systems is in the military. Less well known one is the fact that UASs could be used for civil and commercial purposes. Some possible civil applications are forest fire detection and tracking, search and rescue, and law enforcement. A potentially large and far reaching commercial application is the agricultural industry (Ackerman, 2013; Madrigal, 2009). A brief overview of the possible application and benefits of autonomous UASs in each of these areas is described as follows.

Military UAS applications.

Military UASs generally do the tasks that are too tedious or dangerous for human pilots. An example of such a task is surveillance and reconnaissance, where an aircraft operates over an area to detect and track targets or other elements of interest for typically long stretches of time

(Panella, 2008). Manned aircraft have an endurance problem largely caused by their human crews' susceptibility to fatigue. Unmanned systems do not have this problem. Current unmanned intelligence, surveillance, and reconnaissance (ISR) aircraft conduct missions "already beyond the effective endurance of a pilot in a manned aircraft" (Lake, 2012). The ability of UAS platforms to stay on mission for long periods of time has driven a growth in the number and types of platforms used by the US military services. This is reflected in the number of flight hours that have been collectively flown by these systems, which has increased dramatically since 2003. It is reported that this growth is expected to continue in the future as the capabilities of these systems increase (Winnefield & Kendall, 2012). One drawback of these ISR systems is that they create a large need for trained operators and data analysts (Shachtman, 2008). The reason for this drawback is that the systems are not capable of flying, managing, or analyzing mission data themselves.

Current UAS systems, such as the MQ-9B Reaper and MQ-1 Predator, fall into the category of remotely piloted vehicles, and therefore are not autonomous. These aircraft require a human being to constantly supervise the system and command it to take certain actions, such as selecting an area to search or engaging a target. Remotely piloted aircraft are mostly controlled from ground stations (Panella, 2008). These ground stations are critical for the mission capability of the vehicle, and the control signals may have to be relayed, causing a delay. A UAS control system that relies on input from a ground station runs the risk of losing its link through accidental relay loss, corruption or enemy action. Additionally, a UAS may be able to stay in the air for more than a day at a time but the operators need to be switched out for rest, which means there is no reduction in the number of operators required. In fact, given that the UASs are mechanically able to operate longer than the system operators' endurance, these systems can possibly increase the need for personnel. Another weakness with using remotely piloted UASs is that humans are needed to analyze the information that is collected by the sensor systems. Currently vehicles like the Reaper and Predator require a crew of two to conduct a reconnaissance mission. A possible solution to this human endurance and information overload problem is the use of autonomous systems.

Autonomous systems are capable of avoiding many of the problems found in remotely piloted vehicles. An autonomous system does not require a constant link to a control station or human operator (Singer, 2009). Autonomous systems are capable of simply being given an objective and the system determines the best way to accomplish the objective. Some systems that are partially autonomous in this way are already in use today. Excellent examples are the defense systems mounted on naval warships that shoot down incoming cruise missiles and aircraft (Singer, 2009). These systems, when turned on in a specific mode will automatically target enemy equipment (i.e. missiles and vehicles) and destroy it if it is perceived to be a threat to the ship, friendly forces, or is merely within range. Another such system is the modern naval mine. Some current mines are capable of identifying a passing military ship and launching an attack on it (Canning, 2006). These vehicle based systems do not require human operation other than to be activated and given a task. They are not susceptible to the same limitations as the remotely piloted vehicles mentioned earlier.

A drawback of a fully autonomous system is that they cannot always make the best decisions in a complex real-world environment. Computer algorithms will not always be able to determine the difference between military personnel or combatants and civilians. Currently, it is envisaged that autonomous systems would target only enemy equipment, not personnel (Canning, 2006). This avoids the possibility of a machine automatically targeting human beings.

Autonomous systems could also take a purely observational role in the mission space. The systems could notify a human supervisor of likely enemy activity, but require permission from the supervisor before attacking (Singer, 2009). A major drawback to this method is that human decisions take time to complete and the system is still dependent upon a communication link to be fully effective (Singer, 2009).

Agricultural UAS applications.

One prospective application for the widespread use of autonomous systems is in the agricultural industry. Farmers can benefit from information gathered by UASs to estimate crop yields and health. They can also be used to cheaply apply pesticides and fertilizer to crops, as well as control weeds. In Japan, this application of UASs has already been undertaken, with a dramatic reduction in area that was sprayed by manned aircraft (Ackerman, 2013). Two examples of these agricultural systems are the RMAX helicopter crop duster by Yamaha and the CropCam small UAS (Ackerman, 2013). Both systems are examples of remotely piloted vehicles, with the implication that trained operators are required for the systems to be efficient or effective. These operators would need to be paid, resulting in a long term recurring cost for the duration of use. An autonomous system would not require a trained operator, and therefore eliminates this recurring cost.

Some research on autonomous agricultural systems has been done in recent years. Many of these systems are designed for weeding applications. Many of these autonomous weeding systems rely on computer vision algorithms to differentiate between weeds and crops (Slaughter, Giles, & Downey, 2008). These systems are designed to operate without human intervention and save the farmers money. Autonomous weeding systems could even offer the possible advantage of eliminating the use of herbicides, as the autonomous systems could remove the weeds through mechanical or electrical means (Blasco, Aleixos, Roger, Rabatel, & Molto, 2002; Slaughter, et al., 2008).

Forest fire detection and containment.

Forest fire detection is another application of UASs that could benefit from autonomous technology. The reduced risk to human life and the effectiveness of the systems would be major benefits for this application. Research has been conducted on how UAS platforms could detect fires automatically and also interface with other vehicles and systems with the same mission (Casbeer, Beard, McLain, Li, & Mehra, 2005; Merino, Caballero, Martinez-de-Dois, & Ollero, 2005; Orello, Arrue, Martinez, & Murillo, 1993). The systems described in Merino, Caballero, Martinez-de-Dois, & Ollero, (2005) and Casbeer, Beard, McLain, Li, & Mehra, (2005) are autonomous in the sense that they complete their designed mission without human instruction. These systems could be used in conjunction with vehicle platforms to augment or replace manned aircraft that currently perform the same mission.

Autonomy.

The nature of autonomy.

Panella (2008) states that autonomous systems are systems that "can change their behavior in response to unanticipated events." Another definition is that "autonomy is characterized by the concept of learning through the use of own experience (Ribeiro, 2002)." A common theme in both of these descriptions is that autonomous systems are typically programmed to achieve a goal, but are not explicitly instructed on how to achieve that goal. They require no outside intervention to accomplish a task other than to be assigned the task. "An autonomous system is self-directed by choosing the behavior it follows to reach a human-directed goal" (Winnefield & Kendall, 2012). These definitions all describe the same general philosophy that a given autonomous control system is capable of some degree of optimization and/or useful decision making. The types of control systems most often found in the literature of autonomous search tasks can be placed into two general categories: rule based systems and machine learning systems. Rule based systems rely on preprogrammed behavior to take action in a given situation. Often the desired behavior is achieved through the use of a scoring function. A scoring function is built using metrics that the system programmer deems relevant to the effective performance of the system's designed mission, such as the relative worth of one target over another, the value of the vehicle itself, etc. These systems will always behave roughly the same in a given environment, and require a priori knowledge of their mission performance metrics to be effective. What differentiates machine learning systems from rule based ones is that machine learning systems are able to update and adapt themselves based upon previous experience and interaction with the environment. They are able to perceive patterns and trends in data, and then use this information to increase future performance (Alpaydin, 2010). This feedback mechanism is a critical component of machine learning systems, as it gives rise to the possibility that the autonomous system may be able to reach optimal performance without the need of the extensive human initialization found in rule based systems. In the following sections, the basic concepts of rule based systems and machine learning systems are reviewed.

Rule-based system experiments and simulations.

There are several approaches, methods, and algorithms that are used to achieve some level of autonomous operation in simulations. Rule-based systems have prebuilt decision-making algorithms and modes that direct the actions and behavior of the UAS. However, they do not have the ability to optimize their operations and behaviors to the environment. That is, rule based systems do not have any form of feedback so that they may change their behavior and increase their performance over time. The rule based system approach has been widely studied in a number of simulations and experiments to control the actions of a team of vehicles in search and tracking tasks. For example, in *Cooperative Real-Time Task Allocation among Groups of UAVs* (Jin, Polycarpou, & Minai, 2004) a system was designed to control multiple UAVs in the same operational space. Individual vehicles were assigned to missions that most suited their capabilities. One hundred runs were completed, with each run refreshing a randomized target placement. The UAVs flew in paths that were deemed optimal to reach assigned targets and search sectors. Vehicles were given tasks based upon their distance from an available assignment and their individual capability. The UAVs updated a global probability map using an imperfect sensor based upon Bayes' Rule. The performance of the sensor was not supplied and there was no analysis comparing system efficiencies at different sensor performance levels. No statistics or reports on the number of false positives are provided. It also seems that the value of alpha was computed in a simple ratio format. This study used two distinct metrics to analyze the performance of the system. The first of these is the target neutralization time. This was the time needed to complete all a priori tasks. The second metric was the time needed for every sector in the environment to be searched. The system used Jin et al. does not increase its performance over time (i.e. it does not learn from past experience) and the system would not be able to react to unanticipated stimuli, such as a new threat. To correctly interpret the new stimuli in terms of its impact on mission performance, the system designer would have to add a new set of rules to accommodate the new stimulus.

Another example of a programmed system can be found in Yanli, Minai, and Polycarpou's *Decentralized Cooperative Search by Networked UAVs in an Uncertain Environment* (2004). The method of control used in this experiment had UASs share information and plan their paths over a set number of future steps. This method is very similar in general architecture to that found in Jin et al. (2004), although the method found in the work of Yanli et al. (2004) includes other elements such as enemy defenses. Each UAS selects a path that maximizes a scoring function. The score function is made up of other separate functions. These functions each address different aspects of a proposed path: coordination, target confirmation, environment exploration, and threat avoidance. The problem with this approach is that the system will not improve its performance over time. If the threat function, for example, is set too low then the UAS control algorithm will not adjust this function to minimize future losses. The system in Yanli et al. is initialized with a given threat map and specified probabilities of survival for each location in the map. There is no feedback in this approach where the system can adjust its behavior based upon encountering new information in the environment, such as a higher than expected threat levels or new threat profile. This information would have to be entered into the system by a human operator.

Yet another example of these systems relies upon a network structure to coordinate the actions of a swarm of vehicles. The work of Nygard, Chandler, and Pachter (2001) used such a network that was solved through a linear programming format. All vehicles shared a common map of the environment. Every time the status of this map changes the linear programming optimization is performed to reassign units to tasks in the most efficient way possible. One of the major weaknesses of this study's approach is that "there is a large burden on being able to accurately specify cost functions (Nygard, Chandler, & Pachter, 2001)". This weakness is present in the other rule-based system studies as well. Nygard et al., also envisaged that the machines would be able to share all map information instantaneously. The study assumes that imaging systems are used to detect and classify targets. The article does not mention how this recognition system behaves at a detailed level and does not supply the actual values used for thresholds or sensor performance.

Machine learning systems.

Unlike rule-based systems that use pre-defined rules and lack of adaptation based feedback, machine learning systems increase their performance over time by identifying and examining the effects of their past behavior or example data (Alpaydin, 2010). They can adapt to new situations through their interaction with the environment. Some general machine learning methods that have been used for control of autonomous robots are Genetic Algorithms (GA), Neural Networks (NN), and Reinforcement Learning (RL) (Panella, 2008). All of these approaches can be categorized as machine learning systems, and can be applied to the search problem with varying degrees of difficulty and limitations.

Among these approaches, RL is perhaps the most popular approach for adapting the behavior of a robot to an environment. A number of studies have been conducted on the suitability of RL in robot navigation. Many of these experiments take the form of maze experiments or feature the agent vehicle navigating through hallways (Martinez-Marin & Rodriguez, 2007; Sutton & Barto, 1998). These examples typically contain one goal state with a corresponding reward. Other applications include job shop scheduling, network optimization, and effector control (Sutton & Barto, 1998). Before reviewing the experiments that utilize RL as a means of navigation control in a robotic vehicle it is necessary to undergo an introductory review of RL methods, concepts, analyses, and terms, as described in the next section.

Reinforcement learning.

Reinforcement Learning (RL) is a name given to a category of machine learning algorithms that “learn” about their operating environment through experience. The origin of RL algorithms is the concept that a living entity learns through experience and interaction with its environment (Sutton & Barto, 1998). For engineering and control applications, the overall concept of RL is that RL is able to learn using rewards that are generated by its operating environment. RL controlled entities take an action that changes the environment. The environment may then change from the perspective, of the RL controlled system and also receive a reward from the environment. The reward is an indicator of the usefulness of the action taken by the system. This reward is used to reinforce the likelihood of taking the action again. A diagram demonstrating this dynamic is shown below in Figure 1.

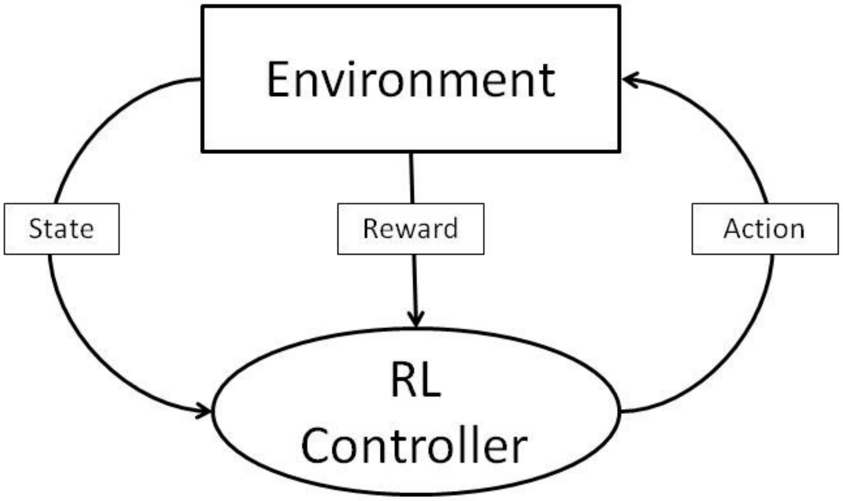


Figure 1: The RL architecture. A diagram of the basic interactions that are necessary for a RL implementation. Rewards are conceptually generated from the environment as they drive the controller’s action selection and performance. Reproduced from “Reinforcement Learning: An Introduction” by Sutton and Barto 1998.

The core mechanic in RL is that it must balance exploration with exploitation. During exploration the RL algorithm takes actions and transitions to states in way that features some degree of randomness. This enables the RL system to eventually explore its entire environment in the limit of an infinite number of exploratory steps. The exploration mechanic is a primary contributor to the utility of RL algorithms as it enables unsupervised learning. Unsupervised machine learning methods can be initialized in a completely unknown environment and are still able to reach an optimum level of performance, based on the reward received from the action taken. That is, they do not need to be trained or extensively initialize with behavior information to eventually become effective. The other side of RL is its ability to exploit the information it has gained. Exploitation relies upon the RL algorithm's past experiences acquired through exploration to maximize or minimize a given reward function. The strategies and methods used to carry out this mechanic are described in more detail below. It is worth noting that Sutton and Barto's 1998 text, *Reinforcement Learning: an Introduction*, seems to be a common resource for the studies discussed in this section. Additional information can be found in Alpaydin's 2010 *Introduction to Machine Learning*. Therefore these two texts are the main resources for discussion on the structure and behavior of the common RL algorithms in this section. The section ends with a review of studies and simulations of RL algorithms implemented in a navigation task. All RL algorithms feature some common components and mechanics. The decision of how these components and mechanisms function and interact with each other largely depends on the specific application (Krothapalli, Wagner, & Kumar, 2011; Sutton & Barto, 1998). The basic elements of a typical RL system are; the agent, the environment, states, rewards, the value function, the policy function, and the exploration mechanic.

The agent.

The agent is the decision-maker in the RL system. "An agent is the entity that communicates with and tries to control external processes by taking appropriate actions" (Ribeiro, 2002). The agent makes its decisions based upon policy functions and value functions. Value functions are methods that map the rewards that the agent has experienced to specific states and/or actions that it has tested. Policy functions dictate to what state the agent should move or what action it should select when presented with a given situation. A simple and often used policy is to select the state or state action pair that has the highest value (Whitehead & Long-Jin, 1995). Policies are also capable of incorporating other information not directly linked or processed by the value function (Costa & Gouvea, 2010). Policies can also be stochastic in nature, which aids in exploration (Sutton & Barto, 1998). In robotic applications the agent does not necessarily correspond to the robotic vehicle. One example of how an agent controlled robot can represent part of the environment is the energy or fuel level of the robot. The agent may or may not need to be aware of these parameters to achieve the desired performance in the application.

State construction and environment representation.

The agent perceives its environment through the use of states. A state is an instance of environmental parameters that an agent can perceive and act upon. A state provides a point to which an agent can associate information, and therefore enable decision making (Sutton & Barto, 1998). Position, fuel level, time remaining, threats, and distance traveled are only a few examples of environment parameters that can be represented through the use of states. A state in RL algorithms is usually denoted by s. One state represents one condition, or instance, of the environment. One of the major disadvantages of the state encoding process is commonly referred to as the curse of dimensionality (Kaelbling, Littman, & Moore, 1996; Krothapalli et al., 2011; Ribeiro, 2002; Sutton & Barto, 1998). This arises as the designer of the system tries to increase the performance of the RL algorithm by including more state dimensions or increasing the resolution of existing dimensions. A state dimension is a set of states that are used to perceive the current situation. While this increases the amount and precision of information the agent receives at any one time, it also greatly increases the number of states exponentially. The addition of a new state dimension results in, at minimum, a doubling of the number of possible states that the agent will likely have to explore. It is easy to generate a state architecture that features thousands or more states.

The consequences of this explosion of state combinations is that the agent now needs more decision time or runs to explore the state space to determine an optimum policy. The problem is further compounded if states and actions are considered separate entities in the value functions, also known as state-action values. An action is usually denoted as a, and RL algorithms that use states and actions are called state-action implementations. In state-action RL implementations possible actions are indexed in the same manner as an additional state dimension. The state action RL architecture is a common paradigm used in robotic control (Smart & Kaelbling, 2002; Sutton & Barto, 1998). The reason for the action selection mechanic is that a robot can have multiple choices of action in a given state, and the resulting state can be the same, but the rewards received are different. A specific state and action in RL algorithms are denoted as s_t and a_t, respectively. The notation t represents a state s_t or action a_t that is taken at time t. Future states or actions are noted as s_{t+i} and a_{t+i}, respectively.

An important aspect in the application of RL algorithms to decision-making and control problems is that operating space must be Markovian. The principle concern with the Markov property in RL is that the states that the agent can explore must contain all relevant decision making information (Sutton & Barto, 1998). Such decision problems are called Markov Decision Processes (MDPs). A process exhibits the Markov property if the future probabilistic outcomes of the process depend only on the current state and action selected by the agent (Alpaydin, 2010). The past states of a MDP do not have any bearing on the process in the future. The Markov property is sometimes referred to as the independence of path property. Non-Markov problems that rely on previous state and action information can be made Markov in nature through further state encoding being supplied to the agent, but this usually results in large and complex dimensionality problems (Ribeiro, 2002). Some success has been reported with the application of RL to hidden state MDPs, called Partially Observable Markov Decision Process or POMDPs (Sutton & Barto, 1998). It is worth mentioning that at the time of this writing the methods used to accomplish these tasks do not seem to be widely used and can vary widely in their performance (Sutton & Barto, 1998; Whitehead & Long-Jin, 1995).

Rewards and value functions.

The environment is the source of rewards (Sutton & Barto, 1998). Rewards, once mapped via value functions, provide the impetus to the RL algorithm to constantly improve its performance. Rewards indicate to an agent how desirable its current situation is. Rewards are received by the agent and processed into state or state action values by the agent's value function. It is possible for an agent to receive a negative reward, and is typically referred to as a penalty (Ribeiro, 2002; Sutton & Barto, 1998; Whitehead & Long-Jin, 1995). The reward and penalty system is one source of "design freedom" when implementing RL algorithms (RLA), and requires careful consideration. In algorithm formulation, a reward is usually denoted as r. If penalties are overly large relative to their actual severity then the agent may avoid a path or action sequence that would yield better real world results. The inverse is true for rewards. Therefore it is best to set reward and penalty values to levels that reflect the relative desirability of target states and actions.

In order to operate the agent must link states or state-action pairs to the rewards or penalties that they generate. Furthermore, states and state action pair values must also indicate if they have access to future rewards or penalties. The agent accomplishes this through the use of value functions. A value function calculates a numerical indicator of how "good" it is for the RLA to be in a given state or select a given action (Sutton & Barto, 1998), and is referred to as a state value or state-action value, depending on the implementation. When calculating the value of a state or action the agent is backing up the reward received from that state or action, essentially assigning "credit" for a given reward. The particular method used to accomplish the backup gives rise to different types of RL algorithms. The state or state-action values are typically stored as an index, where the RLA "looks up" its current state and the available actions that can be taken in that state. In the exploitation paradigm, the RLA would select the action that had the highest value. Typically for an agent using only states the value function is denoted as V and for a state action implementation as Q (Sutton & Barto, 1998; Kaelbling et al., 1996). The exact structure of a value function depends largely on a designer's choice and the specific task that the RL controller is meant to accomplish. When indicating the state value of a given state the term $V(s)$ is used for a state value and $Q(s, a)$ is used for a state-action value.

Depending on the value function paradigm a backup operation can be carried out online or offline. Online updates tend to produce better results as the agent has access to acquired information instantly, as opposed to waiting for the end of an episode or play before the value function map is updated (i.e. offline updating). Two of the most often used value function paradigms of RL used are Monte Carlo (MC) methods and Temporal Difference (TD) methods. Monte Carlo methods update state values iteratively through multiple sweeps of a state-space and wait until the end of an episode or play to update the state values. The updates generated by the Monte Carlo methods depend entirely on the reward generated by the current state and the rewards of states visited latter in the sweep (Sutton & Barto, 1998). For long episodes or continuing tasks future rewards generated during one sweep of the state space can be discounted (Alpaydin, 2010; Sutton & Barto, 1998; Whitehead & Long-Jin, 1995). The discount parameter, γ, is a value used to weight rewards based upon how many steps in the future they are expected to occur. The discount parameter allows rewards that occur in the future to be linked to the currently occupied state or tested state-action pair. Rewards that occur sooner are given a higher value than rewards that occur later. The discount parameter has a value somewhere between zero and one depending on the designer and application. A state encountered by the agent is given partial credit for any rewards encountered by the agent in the future through the discounting mechanic. The amount of partial credit given to a particular state is dependent upon the discount parameter raised to a power equal to the number of steps in the future the reward occurred. This discounting of future rewards forms a target for the value function of the current state. The target value is defined as *R*, and the specific state and action that the reward is assigned to is noted as $R(s_t, a_t)$. The target calculation for the MC paradigm is given below in Equation 1.

$$R_t(s_t, a_t) = \sum_{i=0}^{T} \gamma^i r_{t+i} \tag{1}$$

Here *t* is the current time step and *s* is the current state. The value *i* is the number of steps in future that the reward *r* has occurred, and may take a value between zero and one. The value of *r* can take any real value, positive or negative. When *r* is negative, it is frequently referred to as a penalty. *T* is the maximum number of steps into the future that the reward function is allowed to consider. This reward value calculation serves as a target for the value of the state or state action pair. The value of the state is then incremented towards this target by an update relation given in Equation 2. The value $Q(s'_t, a'_t)$ is the new value of the state action pair *s* and a at time *t*.

$$Q(s'_t, a'_t) = Q(s_t, a_t) + \alpha[R_t(s_t, a_t) - Q(s_t, a_t)] \tag{2}$$

The new value is calculated every time the state s_t is encountered. The value of α is adjusted according to the number of visits to a specific state action pair that have occurred, usually referring to the number of times that a given state or state-action pair has been encountered. This method is used for static tasks. For dynamic tasks and environments the value of α can be held constant. For static tasks the value of alpha is determined by Equation 3.

$$\alpha = \frac{1}{k_{s,a}} \tag{3}$$

Here $k_{s,a}$ is the number of times a given state action pair has been experienced. This value is initialized to one and is incremented without limit. The calculation of $k_{s,a}$ requires a separate list with the same dimensions of the state-action value index and is incremented by one for every visit to the state action value. In this way the value of the state or state action pair converges to its discounted true value. The process for implementing this method in a state action formulation is provided below (Kaelbling et al., 1996; Sutton & Barto, 1998):

```
For every state $s$ and action $a$ encountered by the agent at the end of the episode:
    Determine the discounted reward received by using the equation:
        $R(s_t, a_t) = \sum_{i=0}^{T} \gamma^i r_{t+i}$
    Update the number of visits to pair $s$, $a$ by incrementing the table $K$:
        $K(s, a) = K(s, a) + 1$
    Update the value of the state action pair $s$, $a$ by using the equation
        $Q(s'_t, a'_t) = Q(s_t, a_t) + \alpha[R_t - Q(s_t, a_t)]$
Repeat until state action list is exhausted
```

The method of state value updating in Temporal Difference (TD) learning contains an additional element when backing up its state values. The Temporal Difference method uses the estimated value of the next state instead of actual future rewards in its calculation of the state value. The difference between the estimated value and experienced value is calculated and the new value is incremented toward the experienced value (Conn & Peters, 2007; Sutton & Barto, 1998). The total reward of a given state in the TD method is the reward received in that state and the perceived value of the next state. In a state-action pair implementation the maximum value of all available actions in the next state serves as the estimate target. The next state or state-action pair's value is usually discounted to allow for bias towards more immediate rewards. This allows states and actions that are far from the high reward yield states to have a lower value than those states that are closer. In the case of state-action pair implementation the highest value of the next state action pair is used. The method used to calculate the difference for a one step state-action implementation is given below in Equation 4 (Sutton & Barto, 1998, p. 149).

$$\delta Q(s_t, a_t) = r_t + \gamma \max_a(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t) \tag{4}$$

Once the difference has been calculated then the state-action value is incremented by a portion of this difference. An increment is used to accomplish this in the same way as the MC method (Alpaydin, 2010). The increment α is calculated in the same manner for the TD implementation as it is for the MC methods. Therefore the formula for the update is shown in Equation 5.

$$Q({s'}_t, {a'}_t) = Q(s_t, a_t) + \alpha[\delta Q(s_t, a_t)] \tag{5}$$

If the formulation is fully expanded then the result is Equation 6.

$$Q({s'}_t, {a'}_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{\mathrm{a}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{6}$$

This formulation is one specific instance of a class of algorithms collectively referred to as TD(λ). λ is the degree to which the temporal difference method relies upon the estimated values of the future states. It is separate from a discount parameter as it assigns the credit of current rewards backwards through recently visited states. A value of zero signifies that the method relies only upon the current reward and the estimated value for the next state, whereas a value of unity is equal to the MC formulation (Sutton & Barto, 1998). The previous formulation is a one step method, which can also be called TD(0), meaning that it uses the current reward and the next step's value (Alpaydin, 2010). This pattern can be extrapolated further as λ moves towards one until the equation becomes the MC implementation. The λ parameter is used as a weight to determine the degree to which a state should be backed up based upon previous estimates and actual rewards (Alpaydin, 2010; Sutton & Barto, 1998). The value functions allow

RL algorithms to relate rewards to states and actions. Value functions converge as the number of visits to a given state or state action pairs goes to infinity. The MC and TD methods converge to different values. Value functions thus allow a RL agent to exploit its knowledge of the environment. However, to gain this knowledge, especially in applications with completely unknown environments or dynamics it is necessary for the agent to explore.

Exploration strategies.

RL algorithms work between two modes of operation. One is known as exploration. The other is exploitation. In its exploitative mode a Reinforcement Learning Agent (RLA) simply takes an action that is estimated to yield the most reward or least penalty. In the explorative mode the agent chooses from among a given set of actions that are available in the present state. The method used to select the explorative action is generally up to the designer of the algorithm. The most common versions are widely referred to as ϵ-greedy and Softmax exploration (Alpaydin, 2010; Sutton & Barto, 1998).

The ϵ-greedy function is a simple method used to select between exploitation and exploration. Every decision step, a random number is generated. If the number is greater than the value of ϵ the agent selects the exploitative mode of operation to determine the next action. If it is less than the value of ϵ the agent randomly selects any other action other than the one perceived to generate the best reward (Alpaydin, 2010, Sutton & Barto, 1998). The range, and thus the probability, that triggers exploratory actions can be reduced relative to the number of decision steps made in an episode (Sutton & Barto, 1998). This lets the algorithm explore more often at the start of an episode while increasing its exploitative behavior (i.e. behaves more greedily) near the end of the simulation. The general process for ϵ greedy action selection and implementation is given below (Alpaydin, 2010; Sutton & Barto, 1998).

```
Initialize $s$ and $a$ for entire set of states $S$ and actions $A$
Initialize $\epsilon$
For every step in episode:
    Generate a random number $n$
    If $n$ is greater than $\epsilon$:
        Take the highest valued action
        Once the action is complete update the value of the state
    Else if $n$ is less than $\epsilon$
        Take an exploratory action
Repeat until the end of the episode
```

Another class of strategies for choosing an exploratory action is the use of the softmax methods. The softmax methods take all possible actions in a given state and normalizes them relative to the estimated yield of the rewards those actions may generate if selected. Thus, more profitable actions have a higher probability of being selected and possibly yielding new optimal strategies for task completion (Kaelbling et al., 1996). A commonly used version of softmax exploration is called the Gibbs distribution. The Gibbs distribution normalizes the probabilistic selection by setting the state or state-action values as exponents (Sutton & Barto, 1998). The result is that actions with higher values have a much larger probability of being selected. The Gibbs method features a temperature variable that is used to adjust the relative probability of selecting less valued actions over time or the length of an episode. This is also referred to as simulated annealing in RL literature (Alpaydin, 2010). When the temperature variable is very large the probability of selecting any given action in an action set is relatively uniform. As the temperature variable τ decreases the relative differences of the estimated rewards between actions become more acute. A disadvantage with softmax methods appears when the perceived values of the next best action are relatively close. Multiple high state values result in a high probability of selecting the second best exploratory action throughout the course of the simulation. The formula for the Gibbs distribution is given below (Alpaydin, 2010; Sutton & Barto, 1998). The method generates the probability of selecting action $a_i$in state s is shown in Equation 7.

$$\mathrm{p}(a_i) = \frac{e^{Q(s,a_i)/\tau}}{\sum_{i=1}^{n} e^{Q(s,a_i)/\tau}} \tag{7}$$

Page 31 of Sutton and Barto (1998) states that “whether softmax action selection or ϵ-greedy action selection is better is unclear and may depend on the task and on human factors”. The use of “human factors” in this context seems to refer to the fact that people would find it easier to understand the implications of changing values of ϵ in the ϵ-greedy function (Sutton & Barto, 1998).

The behavior and performance of a RL algorithm also depends on how the state or state action value sets are initialized. If states or state action values are initialized with a high value then the agent will tend to visit unexplored states even while in an exploitative mode of operation (Sutton & Barto, 1998). Matignon, Laurent, and Fort-Piat (2006) demonstrated that an advantage to this behavior is that the agent can rapidly explore its environment. A possible disadvantage is that an agent may visit previously unvisited states or state action pairs latter in the simulation, likely producing suboptimal results. On the other hand, if the initial values are to low then the agent can perceive a truly undesirable state as a good state simply because these states were updated on a previous run. This mode of initialization effectively results in the agent visiting the same states or state-action pairs repeatedly (Matignon, Laurent, & Fort-Piat, 2006). Further work in Matignon et al. (2006) also demonstrated initializing state value functions in a manner that would enable the agent to quickly move towards more useful states and minimize the time and computational cost of exploration. A Gaussian curve was used when initializing the rewards in a maze task that allowed the agent to more quickly discover the goal state. A similar application of these “progress indicators” can be applied to state value functions in such a way as to mitigate the effects of a reward “sparse” environment mentioned in Kaelbling et al. (1996). The approach of Matignon et al. (2006) could possibly increase the learning rate of the RL controller. A disadvantage here is that initializing the values of states at or near the goal requires a priori knowledge of the operating environment. Therefore, the unsupervised learning aspect and advantage of RL controllers is mitigated.

State and complexity reduction in reinforcement learning.

A prevalent obstacle with RL implementation in real-world application is that many parameters and measures essential to the efficient operation of the robot are continuous in nature. To perfectly model a real world environment would often require a prohibitive number of states. Compounding this issue is that an individual state often represents values of multiple dimensions. Techniques and methods used to combat this issue have been explored in various simulations and experiments. It is possible to truncate the state-space so that a continuous function is approximated (Buck, Beetz, & Schmitt, 2002; Sutton & Barto, 1998). The truncation mechanism operates on the similarity between adjacent states. States that are operationally similar (e. g. two adjacent positions are free of obstacles) are combined to create a single state. The reverse of this operation can also be done where the agent subdivides a state given that the sensor readings or rewards from that state are not homogenous (i.e. a large amount of variance in the reward is encountered in a given state).

Yet another method is the use of the hierarchical agent architecture (Sutton & Barto, 1998; Yen & Hickey, 2004). Hierarchical control operates on the assumption that a single, possibly complex task can be decomposed into smaller subtasks. These hierarchical structures are essentially multiple RL controllers operating in “tandem” on a problem. A simple hierarchical setup is shown below in Figure 2.

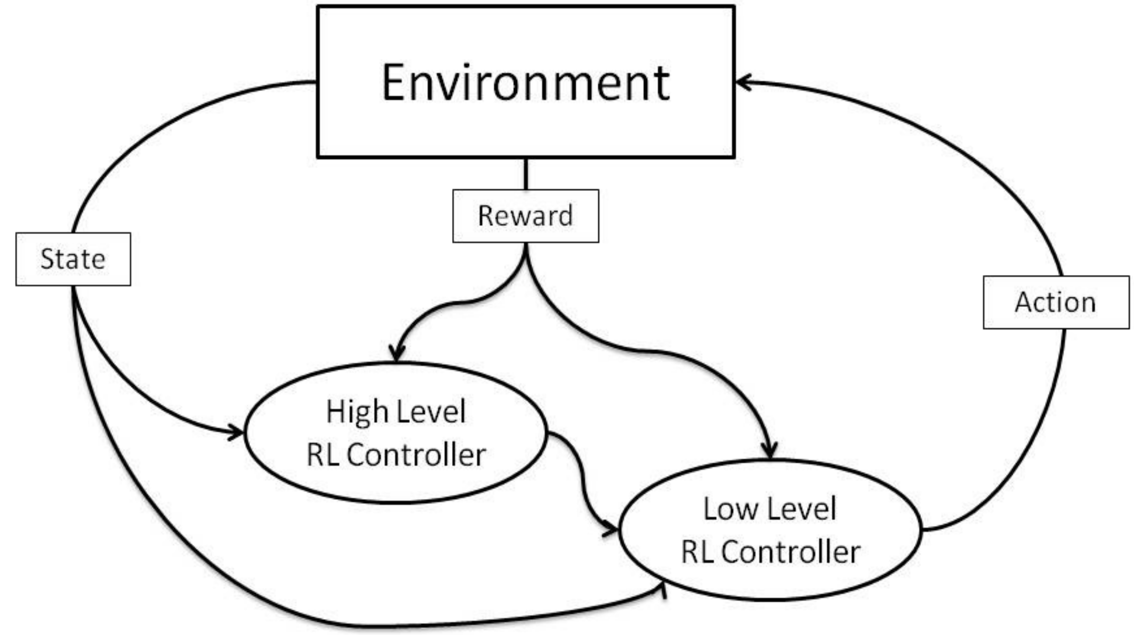


Figure 2: The hierarchical reinforcement learning interaction diagram. Both the low level and high level control receive state information directly from the environment. The low level controller also receives information in the form of a desired action from the high level controller. In this way the high and low level controllers are able to work on different state dimensions, thereby reducing the computational effort of the problem. This diagram is a work of the author.

The high level controller in the hierarchical framework works on the larger, overall objectives of a problem. A good example of this is a navigation task where the environment is extremely cluttered with obstacles. The high level controller may not perceive the layout of the vehicle's immediate surroundings, but only decide on the general direction it should go. Once the decision about intended direction of travel has been made the low level controller works out a way to navigate through the obstacles (Yen & Hickey, 2004). The hierarchical architecture allows each controller to reduce the number of state dimensions that it must explore. The hierarchical method seems to have a drastic effect on the performance of the algorithms when the size of the environment is large (Sutton & Barto, 1998, p. 260). Thus the amount of computational effort spent on exploring the state space is reduced, increasing the speed of learning.

Example applications of reinforcement learning.

Reinforcement Learning can be applied to many different problems and environments. One of these was the control of robotic vehicles with novel effectors. Effectors are the physical constructs that robots use to interact with their environments. Some examples of complex and/or nontraditional effectors studied with Reinforcement Learning control were; biologically inspired limbs and methods of movement (Lin, Xie, & Shen, 2009), reconfigurable vehicles (Valasek, Doebbler, Tandale, & Meade, 2008), and sensor interpretation (Stafylopatis & Blekas, 1998). One particular example of RL in nontraditional and complex effector control was the use of biologically inspired undulating fins to steer a submersible vehicle (Lin, Xie, & Shen, 2009). The fins were independently controlled and could operate at different frequencies. A RL algorithm was implemented to determine the best combination of frequencies to keep the robot oriented at a desired heading. The states used in this case were the current heading and rate of change of the heading. The action set that the agent could explore were different frequencies at which the fins could operate. 81 actions were selectable by the agent representing every combination of the control frequencies at which the undulating fins could operate. An ϵ-greedy action selection strategy was used to instigate exploratory behavior. It was concluded by the authors that the experiment successfully resulted in a policy to control the vehicle in every given state (Lin, Xie, & Shen, 2009). It should be noted that the ϵ-greedy strategy that was used resulted in a large amount of noise in the steady-state performance. This could have been mitigated by reducing the value of ϵ over the simulation time.

Another application for RL can be found in the work of Balakrishna, Ganesan, and Sherry (2010). This work applied RL methods to the task of predicting the time needed for aircraft to taxi out to their assigned runway at a major Unites States airport. The state variables used to make the predictions were performance measures that are usually obtained for queuing and processing systems (Blanchard & Fabrycky, 2011). Time of day was also used as a state dimension. The outputs of the algorithm were predictions of the time needed by individual flights and the average time needed for all flights to taxi out to the runway. The results of the experiment showed that the algorithm was able to predict taxi out times with an accuracy of 90%. An accurate prediction was assessed as being within 90 seconds of actual observed behavior. This work is a prime example of how RL algorithms can be applied to problems outside robot control.

Reinforcement learning applications in navigation control.

The utilization of RL algorithms to control robots navigating in an uncertain environment is not a new concept (Sutton & Barto, 1998). Many RL experiments take the form of an agent navigating a maze or moving around discrete obstacles to reach a goal position (Kaelbling et al., 1996). The mazes used vary in complexity based upon designer choice and computational limitations. Some maze applications have obstacles and traps that get the agent "stuck" or send it away from the goal position. One experiment features a cliff where the agent is sent back to the start if it falls into a set of penalty positions (Sutton & Barto, 1998).

Tian, Yang, Qi, and Yang (2009) conducted an RL experiment where multiple learners were present in the same space to complete a task. The uses of multiple robots simultaneously lead to the application of multiple agents in the same state space. A major advantage of this mode of operation was that the individual agents would be able to exchange information between each other, increasing the rate at which the state space was explored. Logically, the performance of such a system would be increased if the agents were initialized relatively far from each other in the state space. In Tian et al. (2009) simulated robots carried sensors to determine if a task was present at a given position and if the task was already being serviced by another machine. The machines had to travel to the task location and cooperate to complete at least one of the tasks in the simulation. No detail on if the performance of the sensors was supplied. Performance metrics were collected based upon the total amount of reward generated by the system and the amount of time needed to reach an optimal solution.

The work of Yen and Hickey (2004) investigated the capabilities of RL algorithms in dynamic environments. RL algorithms traditionally do not perform well in dynamic environments as learned state values become obsolete as the environment changes. In order to reduce the dimensionality of the problem the hierarchical agent architecture was employed. A similar approach can be found in the work of Perron, Hogan, Moulin, Berger, and Belanger (2008). For the operation of the RL algorithm in the dynamic environment a "forgetting" function was employed. This function was simply a coefficient with a value between zero and unity that drove the value of a given state action pair back to its initial value over time. The study proposed that the benefits of a distributed control are more efficient in larger environments as well as dynamic environments. The study also introduces a separate work where a user's input can be integrated into the system and change the systems behavior.

Another example of RL algorithms used in navigation can be found in the work of Costa and Gouvea (2010). The task of the agent in this experiment was to arrive at a certain location on a map while expending the least amount of energy to get there. The map in question was a simulation of three-dimensional terrain with peaks placed in random locations. What was unique in this simulation was that the exact position of the agent vehicle was not used as a state. The nearby terrain features were presented in a combinatorial fashion. Possible movements corresponded to grid locations. The agent's action choice consisted of the available vertical movements present in the current state (i.e. travel uphill, downhill, or maintain altitude). The policy used by the agent fused heuristics with the standard state action value formulation. This heuristic calculated the future distance of the agent vehicle from the goal position given a proposed action. In this way, the policy takes into account both the energy conservation metric and the distance metric.

Another application of RL in navigation can be found in Krothapalli, Wagner, and Kumar (2011). In this work, an experiment was run where the agent subdivided the state space on its own, which was called "variable grid sizing". Specifically, the agent vehicle was navigating a simple maze to a goal position. The agent created large states for operating regions with the same general characteristics. If a state was found to contain an obstacle then the state was split into smaller states and these are explored. No mention of the reliability of the sensor is made in this experiment. The study concluded that the variable grid sizing method was able to reduce the computational requirements of the traditional RL methods while still yielding optimal performance.

A study that attempted to shed light on the robustness of RL algorithms in dynamic environments can be found in Conn and Peters (2007). In this study, a TD algorithm was used to pilot a real-world robot through an environment. The environment also sometimes contained obstacles. The study investigated the reliability of the RL algorithm to complete its mission and how robust the RL controller was when operating in a dynamic environment. No mention of sensor performance or effects thereof is given in the work. A variable manipulated in this study was the step increment between the current state action value and the perceived value by the agent. The study concluded that an alpha value of 0.9 provided the best results when considering the number of episodes in which the agent made it to the goal state.

All of these studies show that RL can be applied to high level control problems. However, none of the experiments featured above address the impact of sensor performance on the capability of the system. Future UASs and other autonomous systems will require advanced sensor systems to operate in congested airspace or identify and track mission objectives and targets of interest. To study the problem of sensor performance in autonomous UASs requires some examination of current sensor technology and a mathematical framework to describe a sensor's characteristics.

Sensor systems in UASs.

Current sensor technologies that have been studied for use in autonomous systems for collision avoidance are active radar and electro-optical (EO) systems (Fasano, Forlenza, Tirri, Accardo, & Moccia, 2011; Lai, Ford, Mejias, & O'Shea, 2012; Luongo, S., Vito, V. D., Fasano, G., Accardo, D., Forlenza, L., & Moccia, A. 2011). Reliability of the sensor and control system for autonomous UASs has been identified as a critical area of research (Clough, 2005). , G., Accardo, D., Forlenza, L., & Moccia, A. (2011) showed that EO sensing systems feature probabilities of reporting false targets. The false alarm rate was reported to be 1.6%, but was originally as high as 10%. Another study by Jacques (2003) featured the possibility of false targets being picked up by swarming vehicles and being attacked. Sinopoli, Micheli, Donato, and Koo (2001) developed a system that relied on vision (i.e. eletro-optical) sensors to develop a lowest risk path to a preplanned target. The work concluded that imperfect sensor performance should be expected and taken into consideration when designing autonomous control systems.

Outside of UASs, a large amount of research in image recognition and classification is specifically targeted at identifying road vehicles as a way to monitor traffic systems. This type of research provides some hard numbers for the hit rate that can be expected in autonomous sensing and targeting systems. Artificial Neural Networks (ANNs) have been used in a number of experiments to classify vehicles based upon the vehicle's geometry. This approach has yielded recognition hit rates of over 90% when simply trying to detect vehicles (Gupte, Masoud, Martin, & Papanikolopoulos, 2002; Wei, Zhang, & Wang, 2001). Other research has generated detection and classification rates as high as 97% (Takeo, Yoshiki, & Ichiro, 2002). No reports were made regarding the false alarm rate or correct rejection rate of these experiments. The vehicle recognition research does suggest that a hit rate of 90% or above is possible. Research findings in the area of the sensor performance provides a basis for a Signal Detection Theory (SDT) model to be applied in the image based sensing system.

The current literature on sensor characteristics in autonomous UASs and other detection applications demonstrated that an investigation into the impacts of sensor reliability on autonomous systems behavior would be useful. The numerous types of sensor systems that can be used in autonomous UASs indicate that an abstract, high level model that can systematically address the effect of sensor characteristics on the agent performance is needed. The model should be able to capture the most salient aspects of the imperfect sensor systems described in the brief review above. Namely, the modeled sensor should feature hits, misses, and false alarms, and a unifying mathematical model that relates these possible outcomes. This will provide an abstract way to compare and quantify the performance of different sensor systems, regardless of their means of detection. One such model that does this is Signal Detection Theory (SDT).

Signal detection theory.

The origins of Signal Detection Theory (SDT) are rooted in psychology. SDT is considered to have originated from the work by Green and Swets (1966), *Signal Detection Theory in Psychophysics* (Macmillan & Creelman, 1991). The text derives a way to describe the ability of a decision maker or sensor, referred to as a "receiver", to distinguish between different stimuli and noises. The fundamental mechanic behind SDT is that a signal is separate from noise. Both the signal and noise are often modeled as Gaussian curves. The noise and the target signal elements of the environment do not coexist at the same time for any given state of nature (i.e. only one curve is present at a given time). A signal, whether generated from noise or from the target, will generate a value probabilistically. If this value is above a predefined threshold then the receiver indicates a target is present and vice versa. If a receiver becomes more sensitive, that is if the threshold is lowered so that the trigger value is more likely, then the receiver is more likely to mistake the noise signal for the true signal. An ideal observer in SDT is defined as a receiver whose threshold is set in such a way as the probability of generating a false signal is the same for the two possible states of nature (Macmillan & Creelman, 1991). An example plot of the noise and true signal probability distributions are provided below in Figure 2.

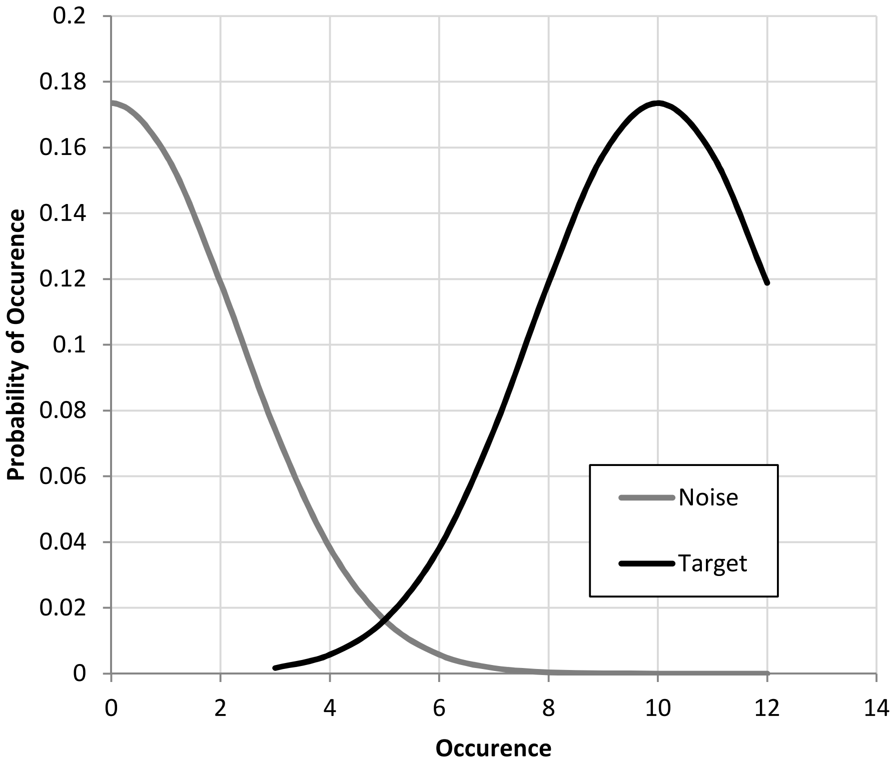


Figure 3: Example probability density functions of target and noise signal values. The regions where the two density functions cross and how much they cross determines the liklihood of generating false signals.

Given a two set discrimination task (i.e. two possible outcomes for two possible states of nature) there are four possible conditions. The first of the correct responses is when a receiver indicates that a target is present and the target is truly present. This is referred to as a “hit”. The second possible condition is the sensor truly indicating the absence of a target and is termed a “correct rejection”. The first of the false readings is when a target is present but the receiver reports no target present and is called a “miss”. The last condition is when a receiver reports a target when one is not truly present. This is referred to as a “false alarm”. Table 1 illustrates the possible states and responses.

Table 1

Possible responses of a sensor with two possible states of the environment

	Receiver Response	
State of Environment	Report Target Present	Report Target Not Present
Target Present	Hit (H)	Miss (M)
Target Not Present	False Alarm (F)	Correct Rejection (C)

A core concept in SDT is the Receiver Operating Characteristic (ROC). A receiver ROC curve is often determined by a difference of the probabilities of these responses. Another way to represent a ROC curve's "score" is in a ratio format, which is the preferred method of classification used in this study. The numerator of the ratio is the combined probability the receiver returns a true reading. The denominator of the ratio is the probability that the receiver returns a false reading. A higher ratio is associated with a receiver better able to return a correct reading for a given state of nature. The ratio, in Choice Theory, is often referred to as "α" (Macmillan & Creelman, 1991). A ratio of unity indicates a receiver that is as likely to return a true reading as a false reading given an equally random true state of the environment (Green & Swets, 1966; Macmillan & Creelman, 1991). The probability of a hit or the sensor declaring that a target is present when a target is truly present is denoted as $P(H)$. The probability of a false alarm or the sensor declaring a target is present when there is no target present is labeled as $P(F)$. The equation for determining the ratio is given below:

$$\alpha = \frac{\sqrt{P(H)(1-P(F))}}{\sqrt{P(F)(1-P(H))}} \tag{8}$$

The cumulative probability of all the responses to each state of nature must equal unity. Thus the probability of a "Hit" is the complement of the probability of a "Miss". This relationship is also present for the probability of a "False Alarm" and a "Correct Rejection". In this way, the ROC can be determined with only the Hit and False Alarm Rates being known.

A given alpha can return any number of possible combinations of $P(H)$ and $P(F)$. For example, a $P(H)$ of 0.6 and a $P(F)$ of 0.2 results in an alpha of approximately 2.45. If the $P(H)$ is 0.8 and the $P(F)$ is 0.4 then the same alpha is also computed. The sensitivity between the two pairs has not changed, but the bias for a positive response is greater in the second instance. These features give rise to the concept of the ROC curves. All possible combinations of Hit and False Alarm Rates for a given alpha can be plotted as a curved line (Green & Swets, 1966; Macmillan & Creelman, 1991). These lines are referred to as isobars as all points on the curve indicate the same receiver sensitivity. Figure 3 displays these curves at some selected alpha levels.

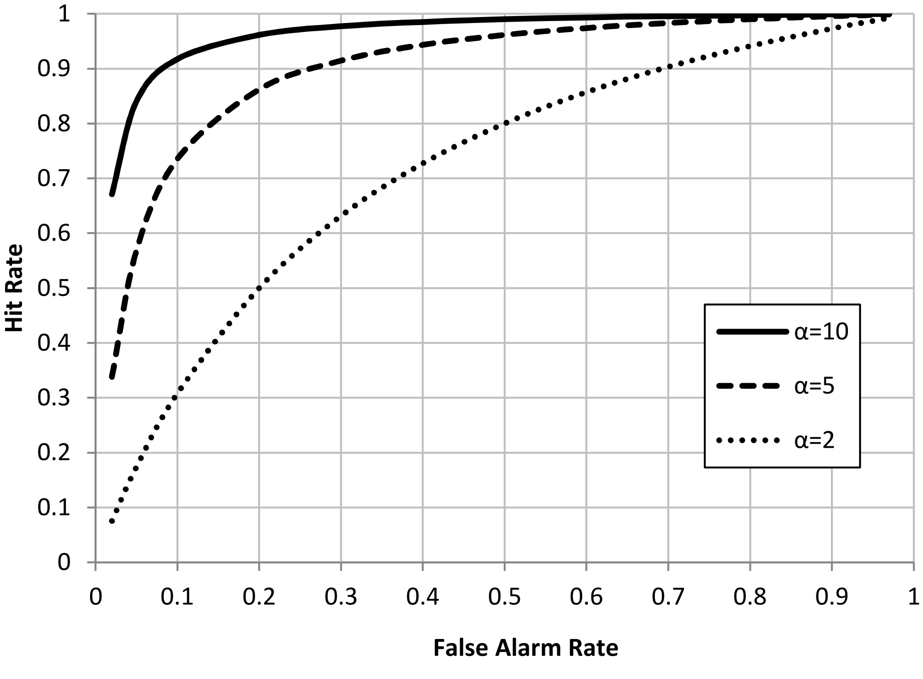


Figure 4: Hit Rate and False Alarm rates for constant alpha values. The constant alpha values are referred to as sensitivity isobars. As the alpha of the receiver increases, the performance of the receiver also increases.

Literature Review Summary

The previous studies showed that RL, in general, provides system designers a simple yet powerful way to shape the control functions and behaviors of a system in a complicated environment. However, RL is limited by the ability of the agent to perceive its environment and thus deduce its true state. Very little research has been done regarding the effects of a sensor's performance on the performance of the RL algorithms due to the fact that there is lack of theoretical model available to address the sensor characteristics. SDT provides a mathematical framework to effectively implement a model of the sensor behavior. SDT also provides a useful tool to abstractly and efficiently describe the performance of different types of sensors. Previous studies show that sensing mechanisms can produce Hit Rates as high as 90% or greater. The Hit Rate can be implemented in an SDT model and the sensitivity of the sensor can be adjusted to determine the effect it has on the performance of the system. Through implementing SDT within an RL simulation this research could benefit future designers of robotic systems that are governed and optimized through the use of RL algorithms.

Problem Statement

The objective of this research is to investigate the effects of sensor performance on the ability of an autonomous Unmanned Aerial System (UAS) controlled by a Reinforcement Learning (RL) algorithm to accomplish a target acquisition task. Little work has been conducted to assess the impact of sensor performance deterioration on the capabilities of autonomous systems using reinforcement learning algorithms. The reviewed methods and algorithms used to achieve autonomy for searching UASs often feature a simulated imperfect sensing or control mechanism, but to the author's knowledge, no reinforcement learning controlled systems studies have generated data where the sensor performance was explicitly changed and the resulting system behavior analyzed or compared. Some simulations simply do not incorporate an uncertain sensing mechanism. Even some real world experiments, such as hallway navigation, feature well defined environments so that a sensor can be assumed to behave perfectly. The performance of the sensor in this work was modeled through Signal Detection Theory (SDT). After the simulation was been run, an analysis of the individual systems' behavior was conducted to reach conclusions on the effects, if any, on the ability and efficiency of the systems to complete their objectives. The two commonly used forms of RL implementation, Temporal Difference and Monte Carlo methods were used to control the system. Conclusions were reached about the differences in system performance.

Hypotheses

There are two main hypotheses that are the focus of this study. The first hypothesis is that the systems with higher sensor sensitivity will locate targets more quickly and efficiently than the other systems during steady-state performance. In other words, it is believed that a sensor with a higher Receiver Operating Characteristic will locate targets faster. To test this hypothesis a number of metrics and figures of merit must be compiled to fully describe the system's mission performance. These metrics and figures of merit are described in more detail in the Metrics section of this study. The second hypothesis is that the systems using the TD methods will yield higher performance results than those that rely on MC methods. This hypothesis will be tested using some of the same data used to test the first hypothesis. More information on how the data is used to test the hypotheses is provided in the Metrics section of this study.

Method

Experiment Description

Experiment variables description.

The experiment featured two independent variables (IVs). One of the IVs was the performance of the sensor. The performance was manipulated through the value of α defined in SDT. There were four sensitivity values (i.e. levels of the IV) used in this study. Three of the systems simulated imperfect sensor equipment with alpha values of 4, 7, and 10. The hit rate of the systems equipped with imperfect sensor systems was set at a value of 0.9. The fourth system had a perfect sensor, implying an alpha value of infinity. The perfect sensor had a perfect hit rate and never generated a false alarm. The second IV was the form of RL algorithm applied. Two RL methods will make up the levels of this IV. The first method is the Monte Carlo method. The other method is the one step TD(0) method. Table 2 shows the layout of all the simulated systems.

Table 2

Experimental systems and their attributes

System	Agent Type	Sensor Alpha	Hit Rate	False Alarm Rate
System 1	TD(0)	4	0.9	0.36
System 2	TD(0)	7	0.9	0.1552
System 3	TD(0)	10	0.9	0.0826
System 4	TD(0)	Infinity	1	0
System 5	MC	4	0.9	0.36
System 6	MC	7	0.9	0.1552
System 7	MC	10	0.9	0.0826
System 8	MC	Infinity	1	0

There were three dependent variables (DVs) in this study. The first metric was the total episodic reward acquired by the Navigation Agents. The second metric was the total reward per trial acquired by the Search Agents. The third DV was the number of incorrect declarations that each system generates per trial when the system reached steady-state performance. Another metric that was used to support the findings found from the DVs was the mission completion rate of the agents. How these variables were used to quantitatively assess the performance of the systems is described in more detail in the Metrics section of this study. The TOP map was also compared across episodes of a given trial to see how the agents may have behaved differently.

The environment.

The simulation was carried out in a discrete environment consisting of a $n \times n$ grid. Each point on the grid represented a position that the agent vehicle could move to and search.

The grid was populated by three targets. Targets were placed randomly in specific regions in the environment and target regions did not overlap. Each position on the grid had a corresponding Target Occupancy Probability (TOP) value. The TOP changed when a position on the map grid was scanned. The change to the TOP value was changed using Bayes' Rule with the values supplied by SDT as the inputs. The TOP map was reinitialized between episodes and trials. At the start of each episode, the agent started in the same position and was presented with a uniform TOP map of 0.5, indicating that the status of the map is completely unknown. The agent was expected, after an initial learning period, to learn the simple behavior pattern of the targets present in the target regions.

Reinforcement Learning Model

Problem, state, action and reward formulation.

The RL architecture used in this study featured two agents operating in tandem, forming a hierarchical agent structure. Each agent was assigned to carry out a different aspect of the search task. One agent was referred to as the Search Agent. This agent decided if a position should be declared occupied by a target. The other agent was the Navigation Agent. The Navigation Agent determined where the vehicle should move and how it should get to that position. The Search Agent operated only on the TOP value of the current position and the Navigation Agent operated on the physical position in the environment. The problem was presented to the agents in the form of trials and episodes. Each trial is made up of a certain number of episodes. The locations of the target regions change every trial, but were static during episodes. The location of the targets in these target regions changed every episode. Multiple trials were conducted to eliminate the effects of exceptionally good or bad target placement in the system performance analysis.

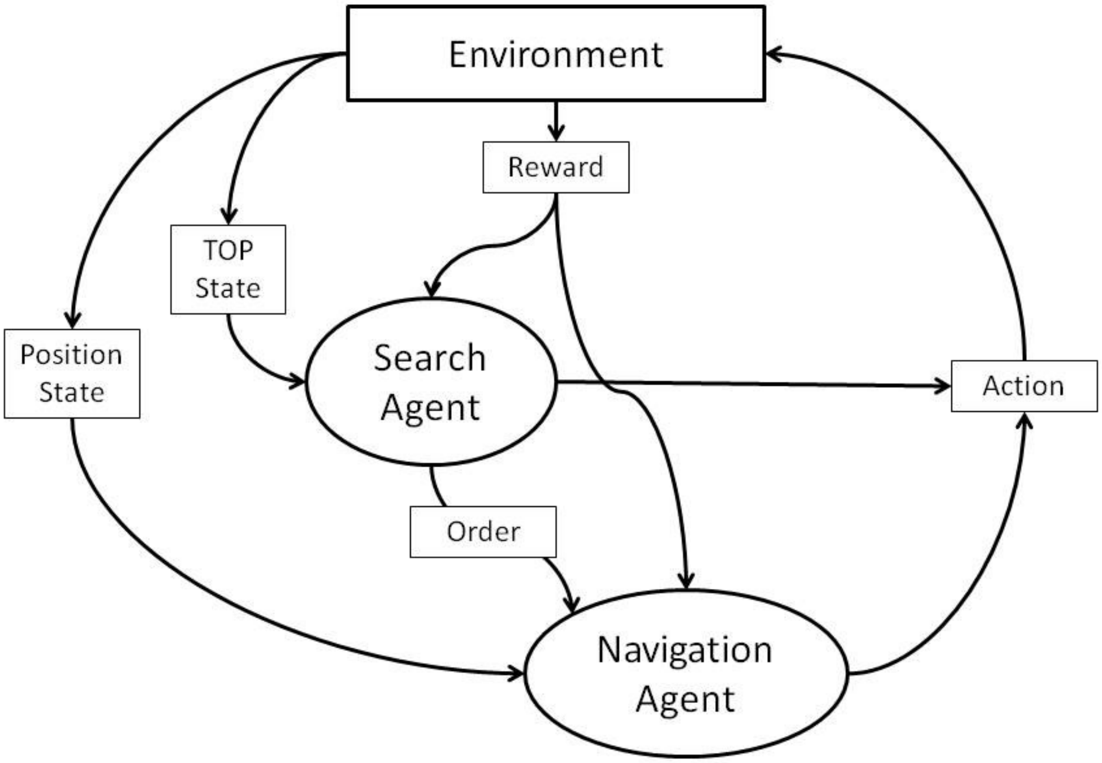


Figure 5: The overall architecture of the experimental RL system. The Search Agent works only on the TOP states. The Navigation Agent works on the position of the agent vehicle. The Search Agent decides when the Navigation Agent is allowed to move. Every movement by the Navigation Agent results in the Navigation Agent receiving a reward of -0.5. The "Order" signal tells the Navigation Agent to change its position.

Each agent operated on different state dimensions. The Search Agent operated on the TOP at any given position. There were a total of 11 possible TOP states that the agent could encounter. Ten of these states corresponded to different levels of TOP. The thresholds for each level were distributed in a non-uniform manner and are specified in Table 3. The eleventh state was used by the declarative agent to determine that a position can no longer be searched.

Table 3

TOP state threshold definition

TOP State	TOP Range
10	0 - 0.01
9	0.01 – 0.05
8	0.05 – 0.1
7	0.1 – 0.3
6	0.3 – 0.5
5	0.5 – 0.7
4	0.7 – 0.9
3	0.9 -0.95
2	0.95 – 0.99
1	0.99 –1.00
11	1 or 0

The Search Agent could choose from three different actions. The first of these was the "Declare" action. The Declare action was a way for the agent to confirm the presence of a target. The reward given by this action was determined by the veracity of the declaration. The second type of action was the "Move" action. This action selection was passed to the Navigation Agent and allows the Navigation Agent to select its next action. The last Search Agent action was referred to as "Loiter". This action disables the Navigation Agent's ability to move. The Search Agent was thus able to continue to scan a given position. When a position was in the eleventh TOP state the Search Agent is only able to select the "Move" action. The total number of state-action pairs that can be visited by the Search Agent is 33. The disabled eleventh state removes 2 possible state action pairs from the problem, resulting in a total of 31 operational state action pairs. The Search Agent's value function mapping was not reinitialized between trials or episodes.

The Navigation Agent moved the agent vehicle from cell to cell in the position grid. The vehicle was able to move in four directions; north, east, west, south. This agent's architecture used only state values as opposed to state-action values. The reason that the agent did not use state-action pairs was that the agent's interaction with the environment was deterministic, and had no other action other than to move to a different position. The adaptation of the Temporal Difference and Monte Carlo methods to a state value implementation is straight forward. The state-action value terms in the two paradigms are replaced with the state values. The Navigation Agent directed the vehicle to move in one of the allowed directions. Therefore if the agent wished to move from one cell to the next it could do so using only the position state information, with certainty that the agent would move to the intended position. The state map for the navigation function was reinitialized between trials, but not episodes.

The application of reward schemes has a large effect on the performance and "risk adversity" of reinforcement learning systems. These quantities needed to be carefully chosen so as the system did not unnecessarily avoid or take risks. The reward scheme chosen for this Search Agent was as follows; +5 for correctly declaring a target position, -1 for incorrectly declaring a target position, and -0.5 for moving or loitering. The reward scheme for the Navigation Agent was +5 for a position that contains a target and -0.5 for movement actions. This reward scheme was chosen after an initial reward scheme was used and the temperature variable was decreased across episodes. Figure 4 shows how the mission completion time for the Monte Carlo system with a perfect sensor varied across episodes.

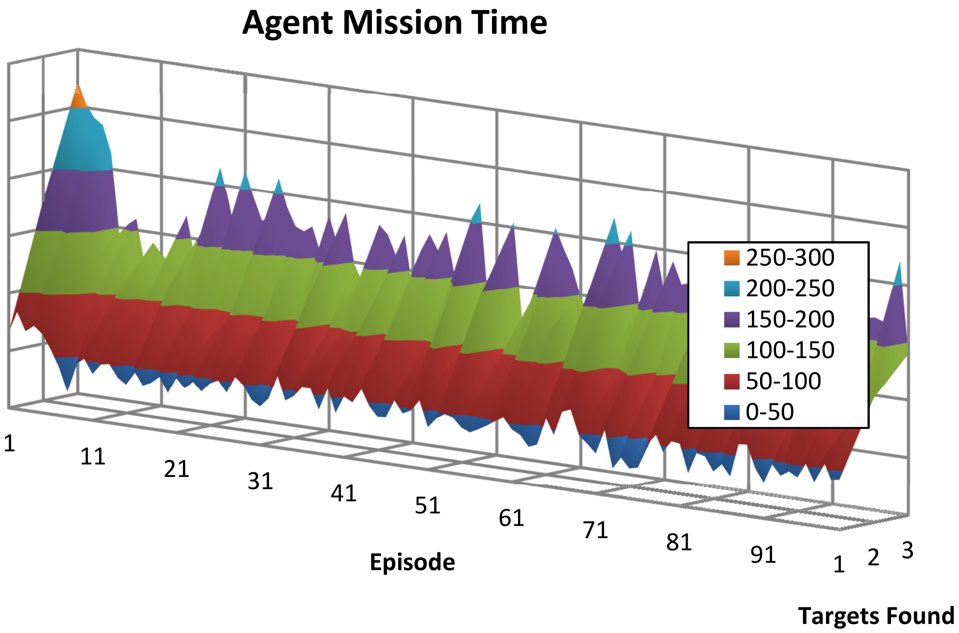


Figure 6: The Mission Time plot. A plot showing the time to mission completion for an initial reward scheme of 10 for finding a real target, -2 for an incorrect declaration, and -1 for all other outcomes. The time to locate all targets decreases for the system up to about episode 4 or 5, then it rises and attains a worse steady-state performance. The best performance was achieved when the temperature was equal to about two, indicating that the overall reward scheme should be factored by half to attain the best steady-state performance.

The time needed to locate all targets in a given episode was minimized when the temperature variable had a value of two, indicating that the reward scheme should be factored by half. This resulted in the minimum steady-state time to locate targets across episodes.

Markovian model of environment.

Two Markovian models were present in this simulation and allowed the RL architecture to function properly. The first of these is the TOP state model, which was what the Search Agent in Figure 4 interacts with to achieve its objective. The Markov property was present in this model as the Search Agent based its decisions purely on the level of TOP in a given position. The sequencing or number of previous scans did not matter. Bayes' Rule was used to update the TOP of the current position using a "likelihood" value provided by SDT. The application of

Bayes’ Rule resulted in a new, updated probability (TOP). Thus, the TOP level was the product of previous scans. The TOP level at a given position therefore encapsulated all necessary information for the Search Agents to make an “informed” decision. A partial example of how the agent can transition through the Markov Chain is given below in Figure 5.

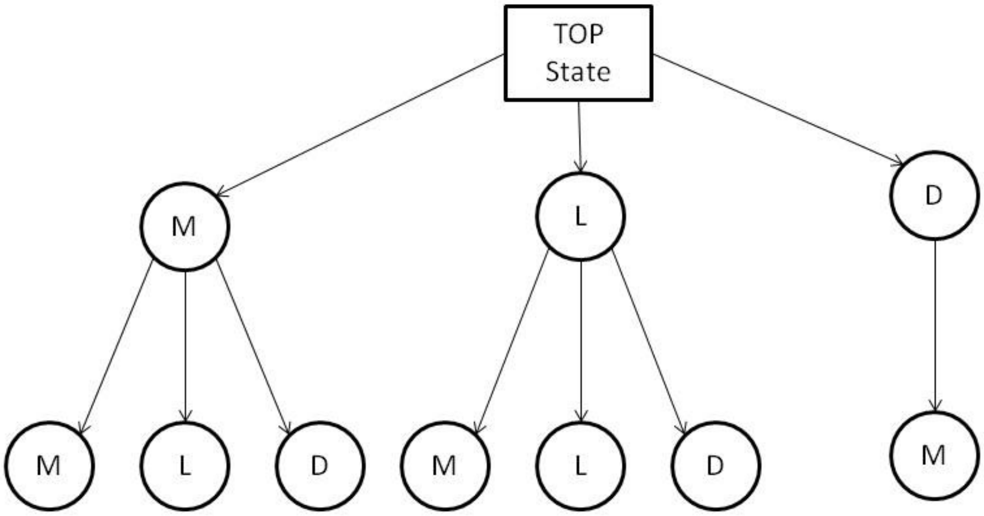


Figure 7: The transition model for the Search Agent. The Search Agent is able to select one of three actions for a non-declared position; Move (M), Loiter (L), and Declare (D). If the agent chooses to declare a given position is occupied the agent is then only able to move from that position. The other two actions result in a second choice with the same options as before.

The other Markov model present in this study was the navigation model. This model was Markovian and was frequently used to achieve maze and object navigation in simulations. The agent vehicle did not need to perceive the path it took to arrive at its current position. Its current position was known and that was all the information the agent vehicle needed to make effective decisions to navigate through its environment.

Target placement and interaction.

The zones for target placement were placed in the map at the beginning of every trial. At the beginning of every episode one of the targets was randomly placed in each of these zones. The effect of this implementation was that randomization was limited to certain areas of the map. Therefore these areas of the map were expected to be valued higher than others, thus the navigation agent would give these positions a higher value than other positions, enabling it to move to these areas rapidly. For the Navigation Agent areas far from where targets were typically found are less desirable than those closer to the other target zones. To maintain track of the targets that have been discovered three separate position state value maps were accessed and modified by the agent. Each position state value map corresponded to one target, and the value maps were updated only while that target had yet to be found (i.e. is “active”). Each map corresponded to the reward received by a different target. The current value of a position state was determined through the summation of all the state maps that corresponded to active targets. The target regions each consisted of 4 adjacent positions. An example of the environment with the target regions highlighted is shown in Figure 6. These target regions were placed randomly on the map at the beginning of every trial.

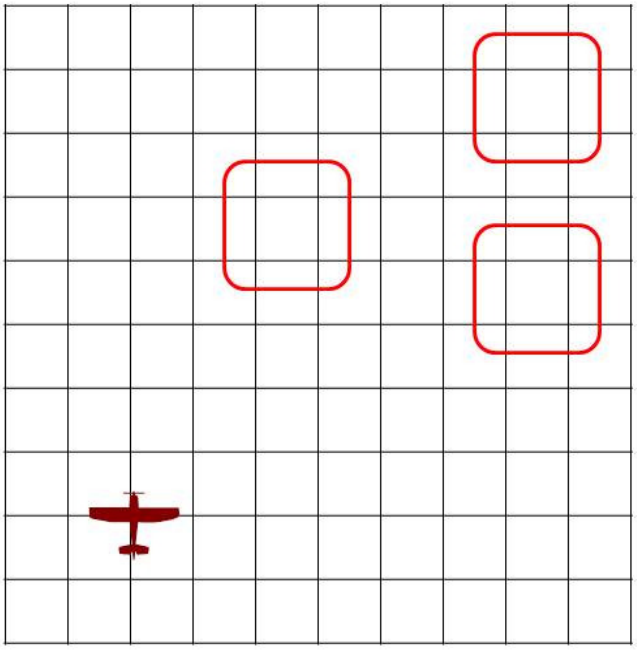


Figure 8: The position map. Example target regions highlighted. The location of the regions changes every trial. This allows a better understanding of the general behavior of the systems.

During an episode, when a target was found the corresponding position state value map was eliminated from the agent's future consideration. Therefore all rewards relating to the discovery of a particular target no longer impacted the agents' decision making process. This mechanic allowed the agent to base its future decisions only on the results of the past episodes' experience with the targets that had not been found in the current episode.

RL paradigm formulation.

The Monte Carlo method.

There were two implementations of RL that were studied in this experiment. The first of these was a Monte Carlo mode of operation. As the agent moved through the state action space a step list was maintained. Once the agent had met its objectives or the episode had otherwise terminated the reward function was propagated back through this list to update the state-action values. The agent was initialized in the same position for all simulation runs. The discount rate was set at 0.9, a value commonly used in other autonomous vehicle RL studies. This discount rate was used in both the Search and Navigation Agent implementations. Thus the update equation, once the episode ends, takes the following form:

$$Q(s'_t, a'_t) = Q(s_t, a_t) + \eta[R_t - Q(s_t, a_t)] \tag{9}$$

R_t is the summation of the discounted future rewards. Note that the step size parameter is represented by η, so as to avoid confusion with the α parameter used in SDT. For the Search Agent the value was incremented by the number of times a state-action pair had been visited. The value of η was based upon the standard equation:

$$\eta = \frac{1}{k_{s,a}} \tag{10}$$

The value of η for the Navigation Agent was fixed at 0.1. This fixed value allowed the Navigation Agent to cope with the somewhat dynamic aspects of the target positions across episodes. The result of a fixed η is that the position state values would never fully converge to a fixed value, but were allowed to constantly adapt according to their latest findings.

The MC methods require that an episode finish before an update is allowed to take place. For large problem spaces, a condition that triggers the end of an episode may not occur, especially in early episodes where the value functions are still relatively uniform. Limiting the number of decision steps and the inclusion of a negative reward for every step taken eliminates this problem. If the decision time runs out at the end of an episode and the agent has not located any targets then the state-action values of that operating region were decreased. The system will then be less likely to visit those state spaces in the next episode. The pseudo code for implementation is provided below:

```
For every decision step $t$ in the episode:
        Record the state and action taken and the received reward from
        that state action pair:
            $\begin{bmatrix} s_t & a_t & r_t \\ s_{t+1} & a_{t+1} & r_{t+1} \\ & \vdots & \\ s_T & a_T & r_T \end{bmatrix}$
        At the end of an episode update the value of “$Q(s, a)_t$” starting at
        the top of the list and using the equation:
        $R_t \ = r_t + \gamma \, r_{t+1} + \gamma^2 r_{t+2} \dots \gamma^k r_{t+k}$
        Until $t + k$ equals $T$ and then update using:
        $Q(s'_t, a'_t) = Q(s_t, a_t) + \eta[R_t - \ Q(s_t, a_t)]$
        repeat for the entire list
Repeat for every episode
```

The Temporal Difference method.

The other method that was implemented is the one-step Temporal Difference method. The method updated a state-action pair with the reward received from that state-action pair and the maximum state-action pair available from the next given state. The term $Q(s', a')$ is the value of the next state and action pair encountered. The formulation and implementation is given below:

$$Q(s'_t, a'_t) = Q(s_t, a_t) + \eta(r_t + \gamma \ \mathrm{max} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \qquad (11)$$

```
For every decision step $t$ in an episode:
    Record the reward $r_t$ received from being in $s_t$ and taking action $a_t$ and update the value function by:
    $Q(s'_t, a'_t) = Q(s_t, a_t) + \eta(r_t + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    Where $r_t$ is the current state action pair's reward
    Continue until the end of the episode
Repeat for every episode
```

The exploration mechanic.

For an exploration mechanic, all of the agents used a softmax method that is commonly referred to as the Gibbs distribution. The temperature variable was used to enable early exploration and later exploitation. This produced the effect that the agent, at the end of the trial for the search agent and episodes for the navigation agent, greatly favors the state action pairs that were perceived to have a higher value. The Gibbs distribution function is repeated below for convenience:

$$\mathrm{p}(a_i) = \frac{e^{Q(s,a_i)/\tau}}{\sum_{i=1}^{n} e^{Q(s,a_i)/\tau}} \tag{12}$$

The temperature variable for both the Search Agent and Navigation Agents for all of the systems was decreased in a nonlinear fashion over all of the episodes and trials. The initial temperature value equaled 100. The temperature variable decreased exponentially, and equaled unity in about the first 20 episodes. This provided the agents with an early set of trials and episodes where the agents had a higher rate of exploration. The temperature variable for the

Search Agents was decreased across trials and the Navigation Agents was decreased across episodes.

Environment interaction.

The Search Agent interacted with and changed its environment through the Target Occupancy Probability (TOP) map. The concept of the TOP map was taken directly from *Cooperative Real-Time Task Allocation among Groups of UAVs* (Jin, Polycarpou, & Minai, 2004). The TOP map was made up of a 10 by 10 array of cells. The TOP map was initialized at the beginning of each trial as a uniform distribution with a value of 0.5. The TOP of a given cell was the current probability that the cell contains a target as perceived by the sensor. Thus the initial value reflected the assumption that every cell had an equal probability of being occupied or empty. The TOP of a cell is adjusted by Bayes' Rule and determines the probability of actions taken by the Search Agent. The TOP update equations are given in Equation 15. The variable α represents the reliability measure of the sensors. To build the update functions, the sensor characteristics must be determined. These were found through the sensitivity equation from the SDT. The equation is repeated below for convenience.

$$\alpha = \frac{\sqrt{P(H)(1-P(F))}}{\sqrt{P(F)(1-P(H))}} \tag{13}$$

Through the use of this equation, if the sensitivity of a sensor system and the hit or miss rate is known, then all of the sensor's abstract performance characteristics can be determined. The notation for these characteristics, their certainty complements, and a description of each is given in Table 4 below.

Table 4

Sensor characteristic values and descriptions

Characteristic	Complement	Description
$P(H)$	$P(M)$	Probability that the sensor returns a hit when a target is truly present
$P(M)$	$P(H)$	Probability that the sensor falsely indicates that no target is present
$P(F)$	$P(C)$	Probability that the sensor returns a hit when a target is not truly present
$P(C)$	$P(F)$	Probability that the sensor correctly indicates that a target is not present
$P(T)$	$P(E)$	Probability that a target is present, is equal to TOP
$P(E)$	$P(T)$	Probability that a target is not present

With these characteristics it is possible to build the Bayes' Rule equations dependent upon the conditional probabilities. Table 5 lists these conditional probabilities, their computed equivalences found in Table 4, and their descriptions.

Table 5

Conditional probability notation

Conditional Probability	Table 4 Equivalence	Description
$P(H\|T)$	$P(H)$	Probability that the sensor returns a hit when a target is truly present
$P(R\|T)$	$P(M)$	Probability that the sensor falsely indicates that no target is present
$P(H\|E)$	$P(F)$	Probability that the sensor returns a hit when a target is not truly present
$P(R\|E)$	$P(C)$	Probability that the sensor correctly indicates that a target is not present

Note that in Table 5 all conditions are based upon the current state of nature and the return given by the sensor. Furthermore, the probability that the cell is unoccupied is the certainty complement of the occupancy probability. Thus, the conditional probabilities and the mechanics required for the implementation of the Bayes' Rule were fulfilled. The result is two equations. Each is actualized purely based upon the returned report from the sensor.

If the sensor indicated that a target was present:

$$P(T|H) = \frac{P(H|T)P(T)}{P(H|T)P(T) + P(H|E)(1 - P(T))} \tag{14}$$

If the sensor indicated that a target was not present:

$$P(T|R) = \frac{P(R|T)P(T)}{P(R|T)P(T) + P(R|E)(1 - P(T))} \tag{15}$$

Thresholds for TOP values determined the status and state of a given cell. Certain actions were allowed only if predetermined conditions were met. If a cell was declared occupied the agent received a reward based upon the veracity of the declaration and the TOP of the cell was appended to zero or one accordingly. This method of operation mimicked a real world scenario where the autonomous system "flags" a position as a likely target and a human operator or supervisor investigated the flag. The operator then made a decision whether the system has found a real target or not. This feedback mechanic then determines the reward the system receives. These interaction mechanics were not change across the different systems except for the value of α used to describe the sensor characteristics, and thus the TOP update mechanic.

Metrics

This section details the metrics that were used to assess the performance of the studied systems. Eight separate systems were simulated in this work. Four of the systems featured a TD(0) implementation for both the Search Agents and the Navigation Agents. The other four systems use the Monte Carlo implementation for both agent types. A system that has a sensor alpha value of 4, 7, and 10 was present for each type of implementation. The fourth system of each type used a sensor with perfect characteristics. The systems outlined in Table 2 were compared to one another in three major aspects. The first of these is referred to as Agent Behavior. The Agent Behavior section examines and compares the steady-state reward behavior of the Navigation and Search Agents. Agents that obtain a higher steady-state reward were considered superior. Also in this section is an analysis of the Incorrect Declaration Behavior of the systems. The number of incorrect declarations by each of the systems was tracked for each episode and trial. An analysis similar to that used in the assessment of agent rewards was used to compare and analyze the systems' Incorrect Declaration Behavior.

The second measurement was "Mission Performance". This metric contains an assessment of how well each of the system accomplished their mission objectives in terms of completeness and time. Both the Agent Behavior and Mission Performance sections are discussed in more detail below.

The final aspect of evaluation was a comparison of the TOP maps across the episodes of a given trial. At the end of an episode, the TOP map was saved and compared to the TOP map generated by the perfect version of that system at the end of the trial. This allows the systems to be compared with respect to their effects on their operating environments. This metric was presented as a plot. Table 6 lists all the results of the experiment, how the outputs were displayed, and how these outputs were analyzed.

Table 6

Experiment output and format list

Simulation Output	Output Type	Output Format	Analysis Method
Navigation Reward	Average Reward/Episode	Graph	ANOVA
Search Reward	Average Reward/Trial	Graph	ANOVA
Incorrect Declaration	Average Number/Trial	Graph	ANOVA
Mission Time	Decision Steps	Plot/Table	N/A
Mission Completeness	Percentage Complete	Table	N/A
TOP Map Behavior	Percentage Difference	Graph	N/A

Agent Behavior.

Following the established protocol found in Sutton and Barto (1998) in assessing the performance of RL algorithms, each agent had its total reward recorded with respect to the episode and trial number. Using the amount of reward obtained by an agent over a “play” or “sortie” is an effective way to compare the behavior of two different Reinforcement Learning systems. The total reward metric was recorded for every trial and episode. To assess the behavior of the Navigation Agents the average reward was calculated using a specific episode of all trials. The Search Agents were assessed in a similar manner, but using the average reward of all episodes across trials. The reason for this approach is that the Navigation Agent only improves itself over episodes and is reset every trial; while the Search Agent improves throughout the simulation (i.e. the Search Agents’ state-action values are never reset). Each plot was made up of one hundred data points (one hundred episodes for each trial for the Navigation Agent, one hundred trials for the Search Agents). A preliminary analysis of this data was carried out via a plot of the average return of a given episode or trial, depending on the agent. The steady-state data was used in an Analysis of Variance (ANOVA) test. Some noise was expected from the differences of each map layout presented in each trial, a constant update parameter, and the fact that the agent will not always choose the most optimal action in a given situation. An example of a reward plot is given below in Figure 6. The reward plot could take a negative value due to the presence of penalties. It should be noted that this particular plot was the average of the sum of all the reward encountered by the agent relative to the episode; it did not incorporate discounting or otherwise modify the rewards the agent encounters. This episodic reward plot was used to visualize the behavior of the Navigation Agents across episodes. Another plot relative to trials was used to visualize the behavior of the Search Agents across trials.

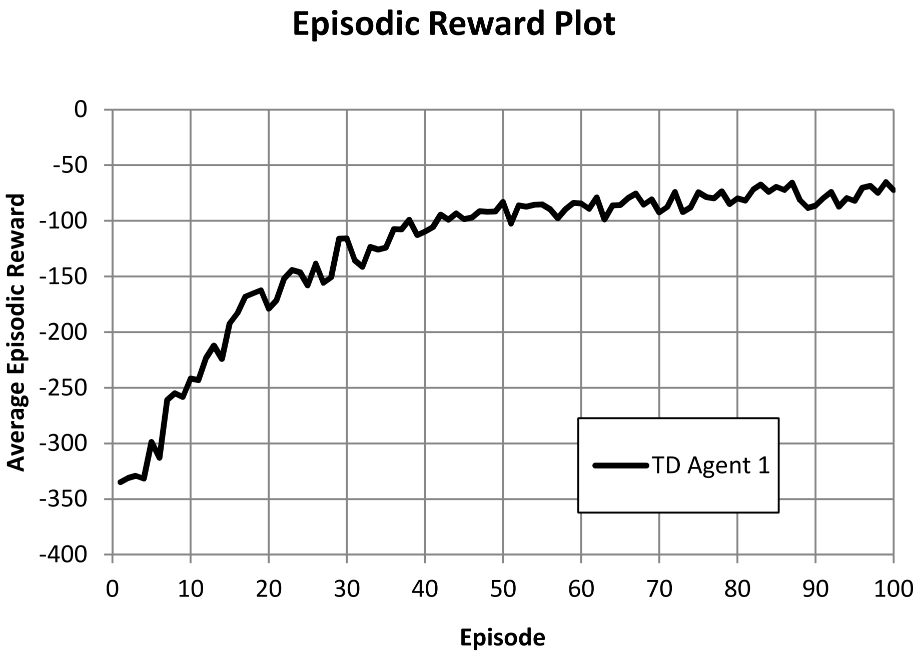


Figure 9: An example of a Navigation Agent Reward plot. The plot is the average value of the total reward obtained in a given episode, averaged across all trials. The noise of the data is from the Agents' exploration mechanic and the constant increment parameter used in the value funciton

It is noted that the system architecture studied here was highly coupled. The performance of one agent drastically affects the performance of the other agent. The reward plots should increase across episodes and trials. Peak performance of the systems is therefore expected during the latter episodes of latter trials.

This steady-state performance region was the subject of comparative analysis. A One-Way ANOVA was applied to determine the presence of significant differences, if any, between systems with different sensor performance. Upon discovery of a significant difference, a Tukey's HSD test was employed to assess the relationship between the systems. A *p* value of 0.05 is used as the significance threshold for both the ANOVA and the Tukey tests.

Mission Performance.

Another aspect of the systems that was compared is the time to complete the mission objectives. The number of steps to correctly declare one, two, and three targets was recorded for every episode of every trial. It was expected that the average time to locate targets will decrease over subsequent episodes within every trial due to fact that learning has occurred from previous episodes. The results of every episode are averaged across all trials and the results compared to the other systems. An example of how the data is displayed is provided below in Figure 10. This performance metric was not tested for significant differences. The reason for recording the mission time was to draw some conclusions about were differences between the systems lay and to reinforce and explain the findings of the Agent Behavior section.

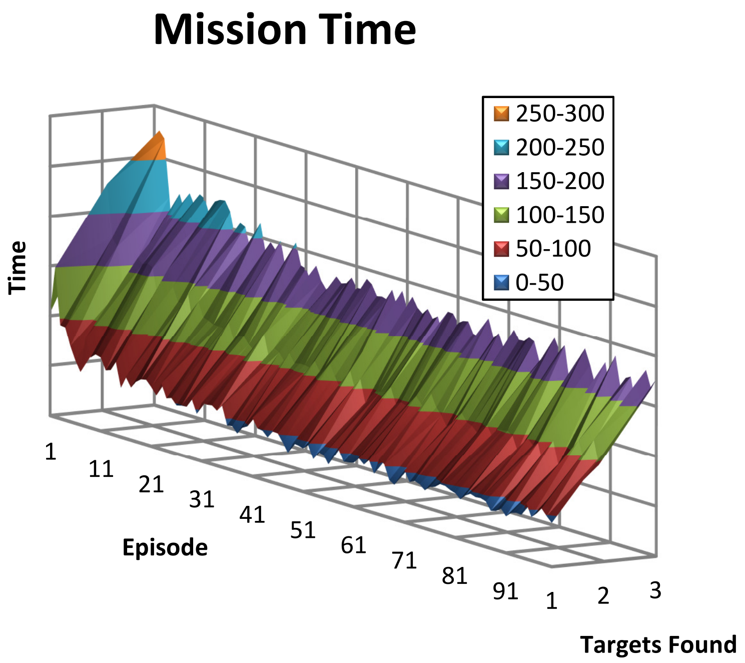


Figure 10: Example of the Mission Time performance plot. The number of targets found is on the depth axis. This plot is obtained by averaging the results across all trials. If a target is not found in a given episode, the epsisode is not used in the calculation of the average.

TOP Map Comparison.

The TOP map comparison was performed for every ten episodes of every ten trials. The value of $TOP_{x,y}^{exp}$ was the agent's perceived TOP and the $TOP_{x,y}^{act}$ was the final TOP map generated by the system with a perfect sensor. The process for generating the comparison metric is given below. The results of this data were then plotted across episodes in the same manner as the reward plots discussed earlier.

```
At the end of episode "$k$" for position "$x, y$"
    Determine the difference by using the following equation:
        $Difference = \left|TOP_{x,y}^{exp} - TOP_{x,y}^{act}\right|$
    Repeat this process for every position "$x$, $y$" in the TOP map
    Average the difference across the entire TOP map
    Average the difference metric across all "$k$" episodes
```

This method generated a value bounded by zero and one, with zero indicating the imperfect sensor system operated with the same effects on the TOP grid as the perfect system. Positions that had not been visited by either agent were inherently zeroed and eliminated from the metric. A downward trend was expected as the episode number increased. Correlation between the systems of the same Reinforcement Learning implementation was assessed via a plot.

Results

This section details the results and behavior outputs of the simulation. The section is organized into two parts. The first part displays the data that the system generated for every metric. The first subsection also includes the data on Mission Time and Mission Completion. The second subsection details the results of the statistical comparisons of the Navigation and Search Agent rewards, and the number of Incorrect Declarations each system produced. Systems were compared to one another based upon their sensor performance. Only systems using the same form of RL implementation were compared using sensor performance. The two types of RL implementation was compared using the performance data of all the systems of a given type, regardless of sensor performance. An ANOVA and Tukey post hoc test were used to determine significant differences. To verify that the assumption of homogeneity of variance was not violated for any of the comparisons each ANOVA was preceded by a Levene Test. The significance level used to determine the validity of the assumption of homogeneity of variance was 0.05.

The mission performance section details how effective the agents were at locating the three targets provided in each episode. The measures used to evaluate mission performance were the number of targets found and the number of steps taken to find targets. The number of targets found for each type of agent was represented by a ratio where a value of 1.0 indicates that all targets were found for every episode of every trial. A value of 0 indicates that no targets were ever found (e.g. a value of 0.98 indicates that 98% of all targets were found out of the 30,000 targets present in the environments from 100 trials and 100 episodes). This number does not indicate where in the simulation a system failed to detect targets (e.g. at the beginning of trials). The number of decision steps taken to find all targets was displayed via a three-dimensional color plot. The decision step metric was composed of how many decision steps were required to correctly declare the presence of a target in a given location. The number of incorrect declarations over time was presented in the form of a plot.

Agent Behavior Results

Agent Rewards.

This section details the results of the simulation from the perspective of the agents that make decisions on what the system should do. The reward acquired by each agent is a useful way to measure the effectiveness and efficiency of the RL systems as the reward functions are the same for both the TD and MC systems. The agent rewards are expected to converge to a steady-state value across episodes or trials. The possible rewards that the Search Agents can receive are 5, -0.5, and -1. These rewards occur when the Search Agent correctly declares a target, allows the agent to move or loiter, or incorrectly declares a target, respectively. The Navigation Agents can receive a reward of 5 when the Search Agent has found a target in a given location, and -0.5 every time it is commanded to move or the Search Agent incorrectly declares a target present. This section does not contain any analyses of the data. The section only presents the results of the simulation. The analysis of the results can be found in the Agent Reward Analysis section.

Temporal Difference Navigation Agent Rewards.

The average episodic reward for the first TD(0) Navigation Agent (with a sensor sensitivity of 4) is presented in Figure 11. This system used an imperfect sensor with an alpha value of 4. It can be seen by inspection that the Navigation Agent shown did increase its reward over episodes. The agent appeared to achieve steady state performance within the first 60 episodes for a given trial.

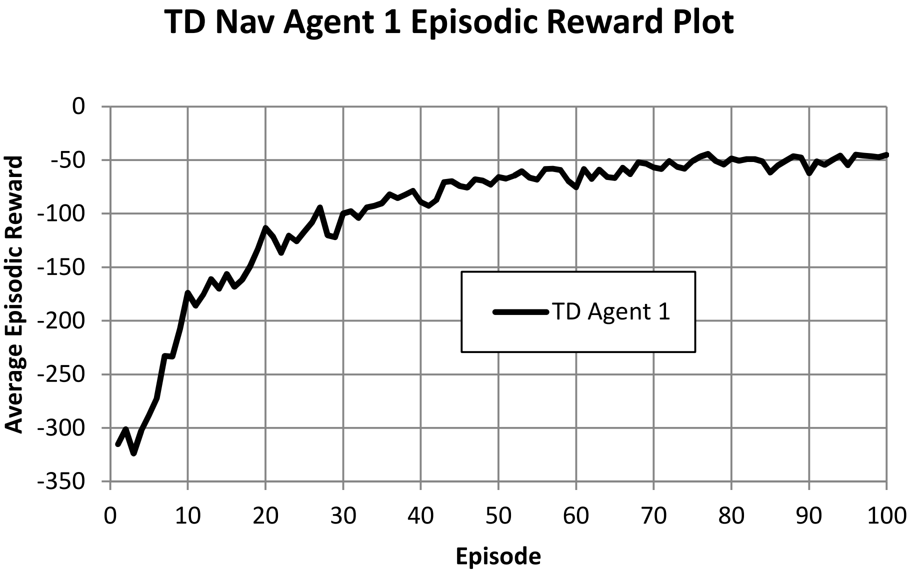


Figure 11: The average reward for the first TD Navigation Agent. This plot is formed by taking average episodic reward across all trials. The Navigation Agent's state value maps are rebuilt every trial as the regions of target occupancy change. Within a trial the Navigatoin Agent's state value map is constantly being updated and refined.

Figure 12 below shows a plot of all the TD Navigation Agents' averaged episodic reward. All the Temporal Difference Navigation Agents managed to achieve steady state performance after about 60 or 70 episodes. Agents 1, 2, and 3 are equipped with imperfect sensors with alpha values of 4, 7, and 10, respectively. Agent 4 is equipped with a perfect sensor. It can be seen that by inspection that systems using better sensors do seem to achieve higher steady-state performance than others. This preliminary observation was further tested using statistical methods to verify if there was a significant difference within this data. It was also noted from the plot that the system equipped with the perfect sensor (Agent 4) with a sensor alpha level of infinity) achieved the highest steady-state reward.

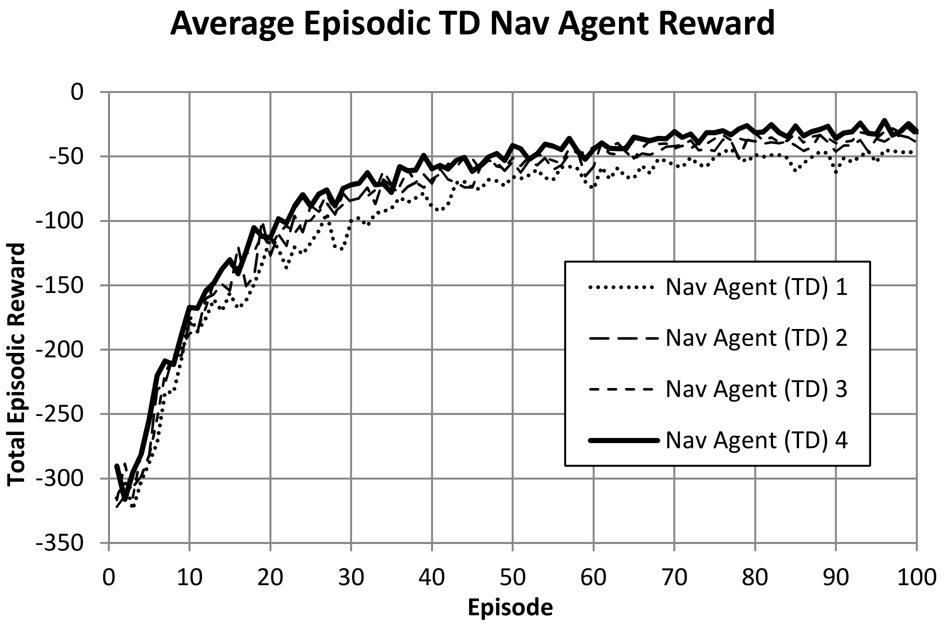


Figure 12: All TD Navigation Agent episodic rewards. The system equipped with a perfect sensor, Agent 4, has the highest reward. The worst agent, Agent 1, attains the lowest reward of all the systems. All agents seem to converge within 60 episodes. Agent 4 and Agent 1 attain a steady-state reward of about -30 and -50, respectively.

Monte Carlo Navigation Agent Rewards.

The MC Navigation Agents also exhibited a characteristic learning curve. The total reward obtained by the agents was also less than that obtained by the TD methods. Convergence seemed to occur within the first 60 episodes, the same as the TD methods. The results of all the MC Search Agents are plotted in Figure 12. Agents 1, 2, 3, and 4 use sensors that have an alpha levels of 4, 7, 10, and infinity, respectively. Again, Agent 4 (the system with a perfect sensor) clearly yields greater performance, converging to an average value of approximately -100. On first inspection, the systems equipped with sensors with a performance of 7 and 10 seem to be approaching the rewards obtained by the perfect system. Agent 1, with the worst sensor, clearly gained the least reward per episode in the last 20 episodes or so. What was less clear here was if the system with a perfect sensor (Agent 4) has significantly higher performance than Systems 2 and 3 (sensor alphas of 7 and 10, respectively). The Monte Carlo Navigation Agents do not gain as high a reward as the Temporal Difference Agents. This indicates the Monte Carlo Agents are "wandering" more than the Temporal Difference Agents. At this point in the analysis it was not clear if the increase in "wandering" behavior is caused by the Monte Carlo Agents' inherent inability to update online or if the drop in performance was due to the way the Monte Carlo Search Agents behave.

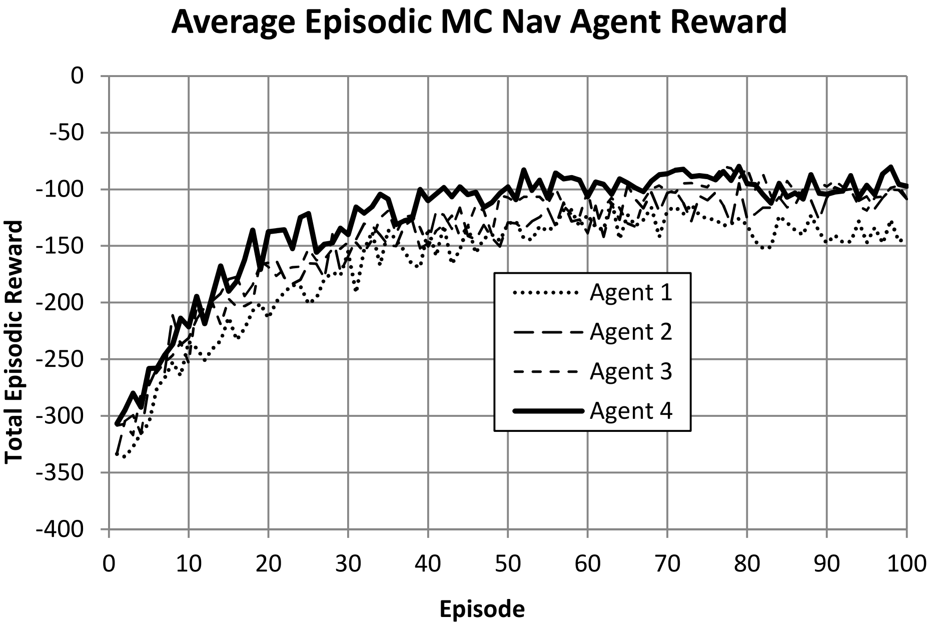


Figure 13: All MC Navigation Agent episodic rewards. The systems equipped with less sensitive sensors performed worse, mirroring the trend seen in the TD Navigation Agents' plots. Examination shows that the MC Navigation Agents, in general, have higher steady-state variance than the TD Agents.

Temporal Difference Search Agent Rewards.

The TD Search Agents also performed as expected, increasing their rewards over trials. The increase in reward over early trials is likely due to the temperature variable used in the softmax exploration mechanic.   They also feature a lot more noise as the location of targets changes from trial to trial. The Search Agent takes as many as 400 scans per episode, so learning would occur rapidly in the Search Agent system, and therefore most learning would take place within the first trial.   It is difficult to see visually any differences or average reward per trial within the systems in Figure 14. A moving average function was applied to the data to help identify trends. The moving average subset size is 5 units, and is shown in Figure 15. The reason that the moving average subset size was 5 units was that this size adequately separates the data within the plots, while still leaving enough detail to see the variation in performance across different trials.

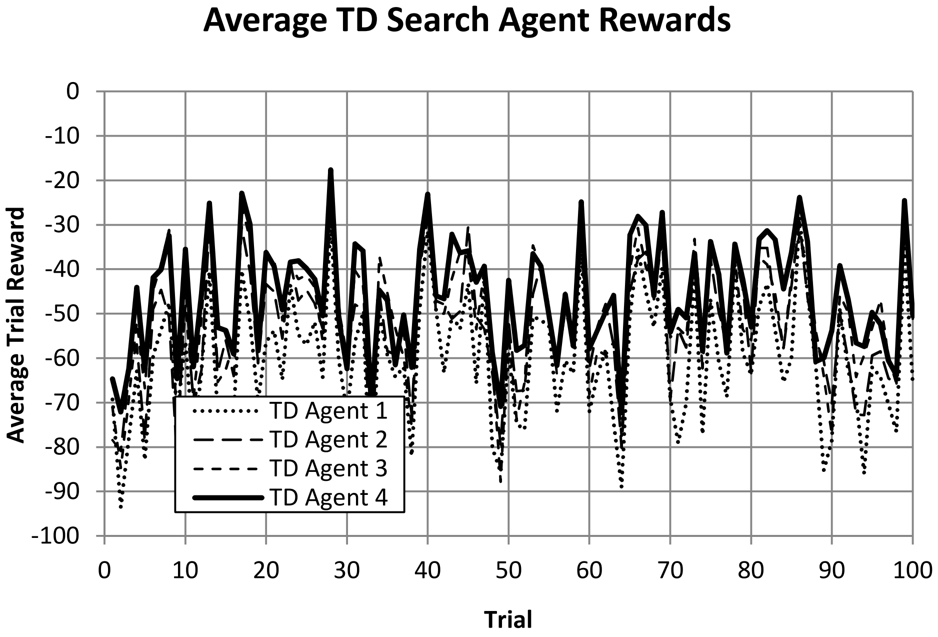


Figure 14: All TD Search Agent rewards. The plot shows that the agents tracked one another consistently. The large fluctuations in the data are due to the difference in the distances between the targets between trials. It is hard to identify trends in this data due to the variance between trials.

The TD Search Agents' rewards seem to fluctuate widely dependent upon the trial, as expected. It was clear that Agent 4 has the highest overall performance, which reinforces the findings from the navigation section. It was also seen that the system equipped with the least sensitive sensor is performing the worst out of all the other agents. Of the imperfect systems, Agents 2 and 3, with their higher performance sensors, seemed to perform the best.

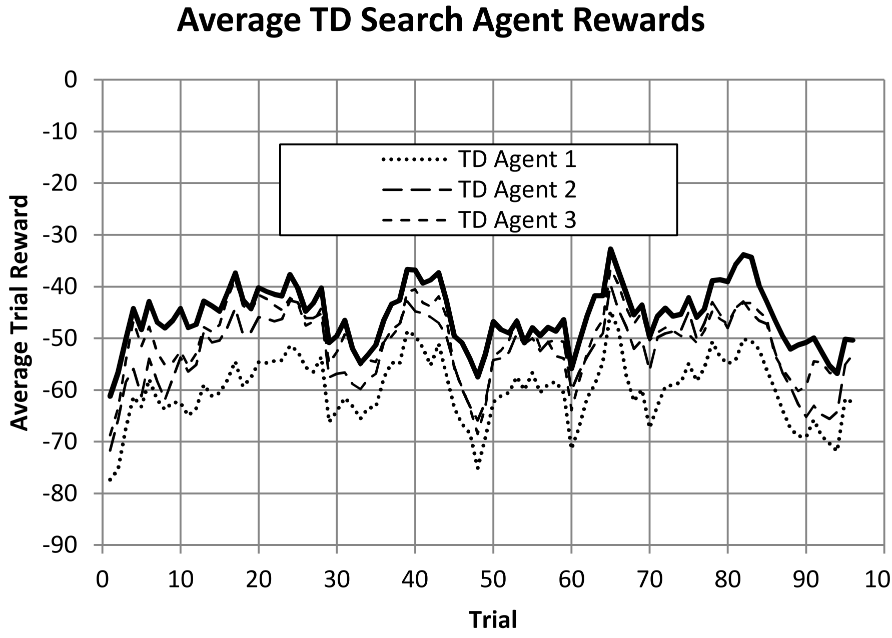


Figure 15: Smoothed TD Search Agent rewards. The data shows a trend that more sensitive sensors performed better across trials. The Agent with the perfect sensor attains a higher average reward than the other Agents.

Monte Carlo Search Agent Rewards.

Repeating the procedure found in the TD Search Agent Rewards section, the combined data for all of the MC Search Agents was plotted below for comparison. Looking at the raw data, in Figure 16, it seems that the perfect MC system performs better than the imperfect systems, but due to the high level of variance, other observations cannot be made. A moving average using a 5 unit subset was applied to more clearly display any trends in the reward data in Figure 17.

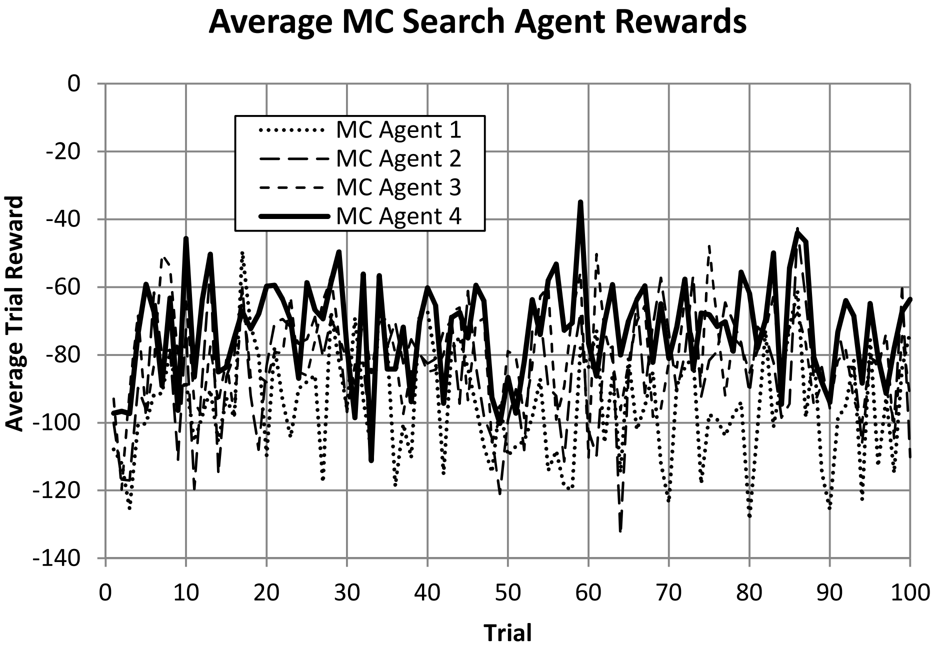


Figure 16: All MC Search Agent rewards. The data features the large variance found in the TD Search Agent data. Examination reveals that the rewards seem to be lower than those obtained by the TD Search Agents.

The superior performance of the perfect system was reinforced by examination of the moving average data in Figure 17. Among the imperfect systems, it seems that the worst system achieved much lower performance. The other two imperfect systems, however, do not clearly distinguish themselves from one another. Further analysis of the MC and TD Search Agent Rewards can be found in the Agent Reward Analysis section of this study.

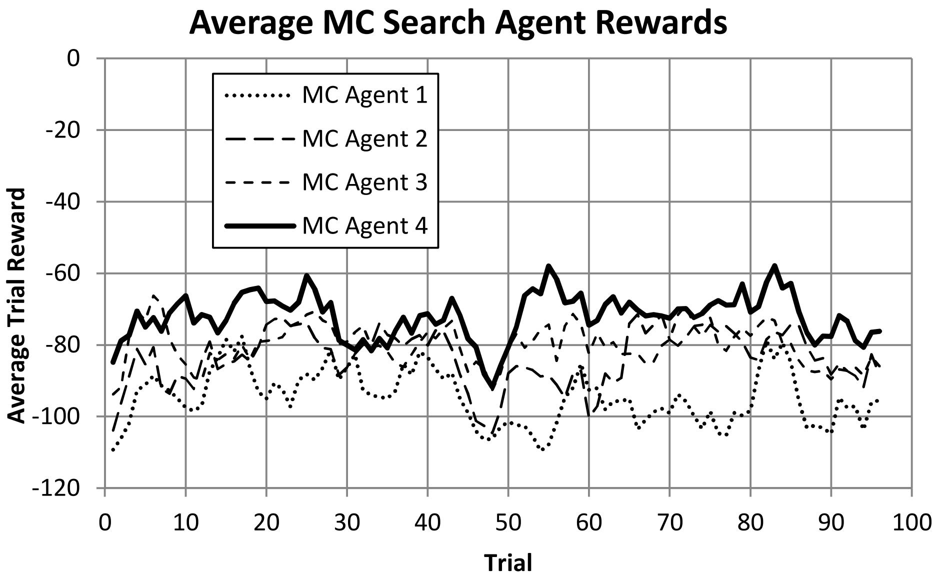


Figure 17: Smoothed MC Search Agent rewards. The data shows that the perfect system clearly did better than the imperfect systems. The system with the least sensitive sensor seems to function with a much higher rate fo variance and a lower average reward than the other systems.

Incorrect Declaration Behavior.

TD Search Agent Incorrect Declaration Behavior.

All systems managed to decrease the number of incorrect declarations over the course of the simulation. It was noted that the Search Agents seem to increase their performance only within the first few trials, and was probably caused by the application of the temperature variable in the exploration mechanic. This behavior was shown in the plots in Figure 18 and Figure 19. The data in Figure 18 was smoothed using a moving average method. The smoothing shows the differences between the agent performances more clearly. The subset size used for the moving average is 5.

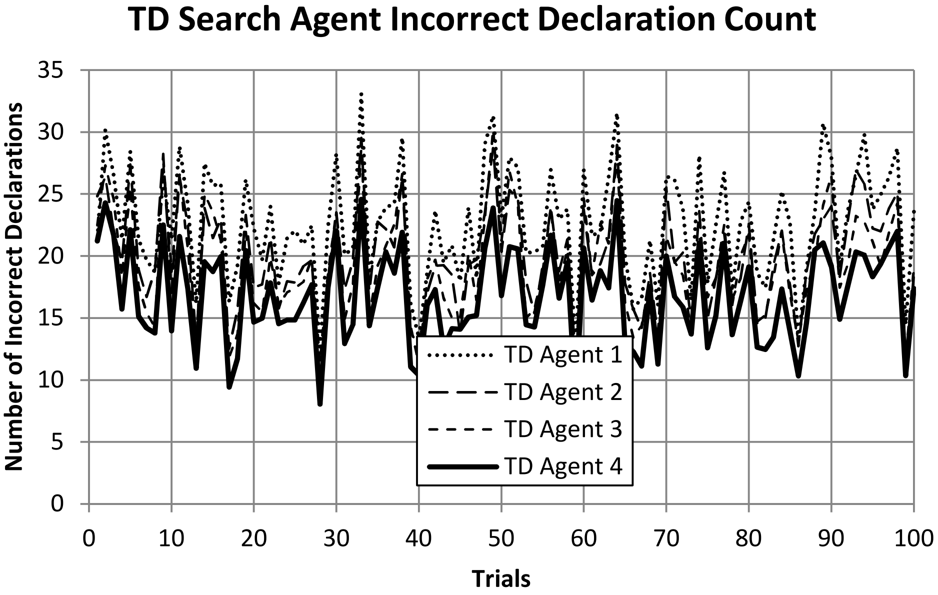


Figure 18: Average TD Incorrect Declarations. The TD Incorrect Declaration behavior exhibits the same high correlation relative to trials found in the Search Agent behaviors. As longer travel times between target areas offer more opportunities for incorrect declarations this behvior makes sense. We also see the early learning behavior. This learning period is of short duration, which corresponds to the observations of the Search Agents.

The systems equipped with a more sensitive sensor appear to perform better, with the system equipped with a perfect sensor clearly being superior to the other systems. The average number of incorrect declarations per trial seems to converge to a value of 15 to 20 for the perfect system, and about 20 to 24 for the imperfect systems. The systems equipped with sensors with a sensitivity of 7 and 10 seemed to have the same performance. A further analysis of the incorrect declaration behavior can be found in the Agent Reward Analysis section of this study.

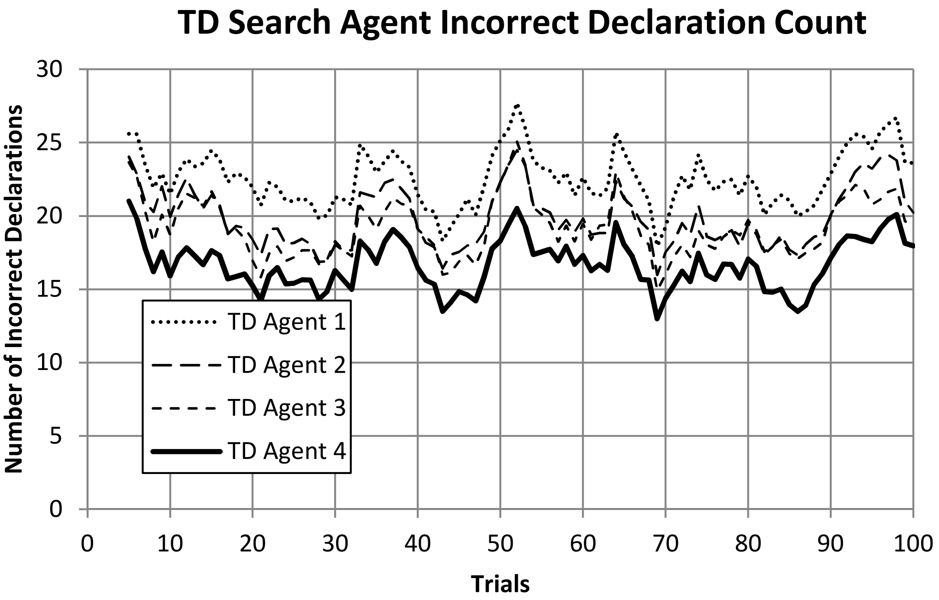


Figure 19: Smoothed average TD Incorrect Declarations. The smoothed data in this plot backs up the system ranking observations made in the Search Agent section. The systems with a sensor sensitivity of 7 and 10 are very similar in their performance, also following the trend seen in the Search Agent section. The perfect system appears to generate 2 to 3 less rewards than these systems, on average.

MC Search Agent Incorrect Declaration Behavior.

The MC Incorrect Declaration data was also recorded and analyzed. The perfect MC system yields a steady-state average declaration number of about 17, with the imperfect systems generating about 20 incorrect declarations per episode in a given trial. Figure 19 showed that the perfect system again exhibits superior performance. All Agents' incorrect declaration counts depended heavily on the trial being conducted. The smoothing method is applied to better see any differences between the systems equipped with imperfect sensors.

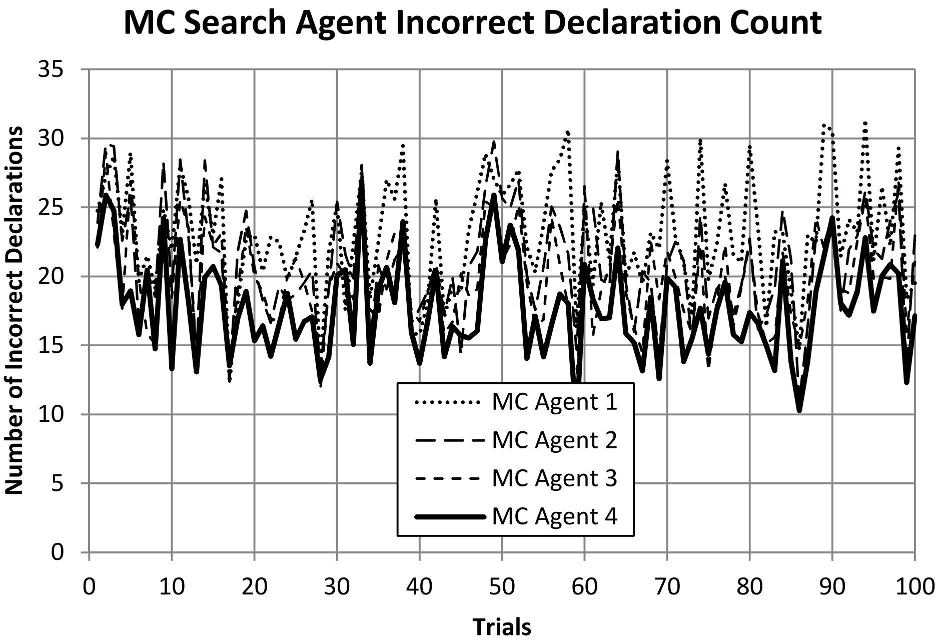


Figure 20: Average MC Incorrect Declarations. The data for the Incorrect Declaration behavior of the MC systems. The systems exhibit a period of early learning, but of short duration, the same as the TD systems. High correlation among the agents is present.

The results of the smoothing method are shown in Figure 21. The perfect MC system differentiates itself from the other systems clearly in this data. Again, the systems with a sensor sensitivity value of 7 and 10 (Agents 2 an 3, respectively) seem to achieve the same level of performance. The system with sensor sensitivity of 4 (Agent 1) performs the worst out of all the systems.

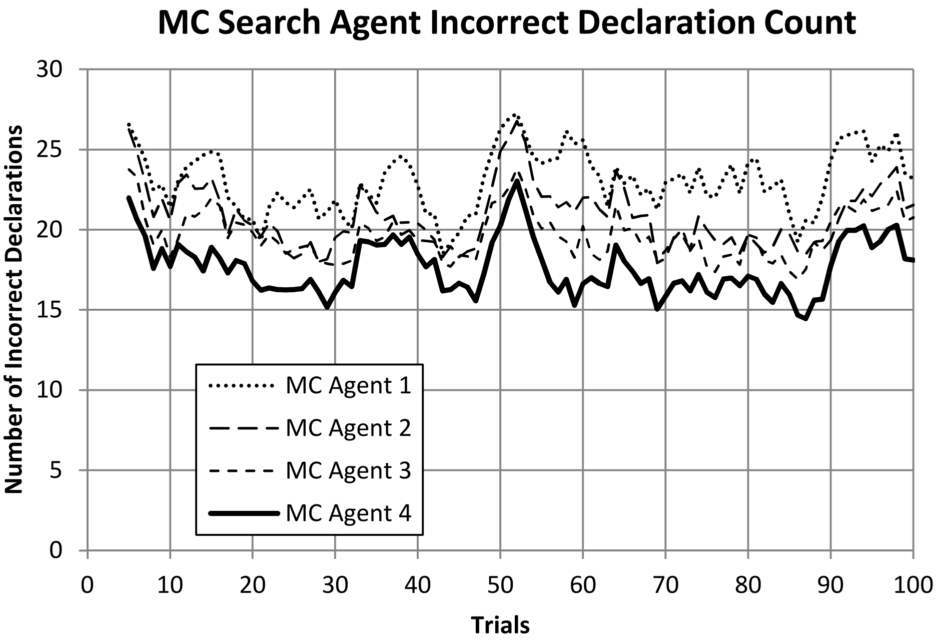


Figure 21: Smoothed average MC Incorrect Declarations. The smoothed data for the MC incorrect declarations shows the same general trends as that found in the TD system data. The perfect system generates the least number of incorrect declarations, while the most sensitive system generates the most.

Mission Performance

Mission Objective.

All agents were able to locate all targets 90 % of the time. The average Mission Completion results are displayed below in Table 7. A value of unity indicated that the agent was able to find all the targets every episode for each trial. Examination reveals that the Monte Carlo methods produced noticeably lower results than the TD implementations. It is also noted that the systems equipped with a perfect sensor did not greatly increase the ability of the systems to locate targets. The difference between the values of the TD and MC systems is most likely due to the fact that the TD agent learns the target behavior faster, resulting in a larger number of found targets in the early episodes.

Table 7

Mission completion rates

Alpha	4	7	10	Inf
TD	0.9794	0.9820	0.9827	0.9844
MC	0.9084	0.9252	0.9288	0.9379

A ceiling effect was almost certainly present here as the maximum number of targets the agents could possibly find was limited to three. Once all three targets had been located in a given episode the simulation ends. If the environment was further saturated with targets greater differences in performance could possibly have emerged. To accomplish this task would require a larger mission space and more targets.

Mission Time.

The time to locate all targets, as measured by the number of steps, decayed for all agents. After the fortieth episode of every trial the system begins to consistently achieve its mission objectives (i.e. identifying all the targets) in a minimum amount of time. The average time to locate the targets is plotted in Figure 21 for the TD Agent 1 (alpha equals 4). The plot shows the average time taken to find each to the targets across all trials. If an agent did not locate the required number of targets in a given episode the episode was not used in calculation of the average. The graphs of the other systems' mission performance can be found in Appendix A. Since it appears that mission performance plateaus after fifty episodes for all systems, the remaining episodes will be used to reach conclusions on the steady state behavior and performance of the systems.

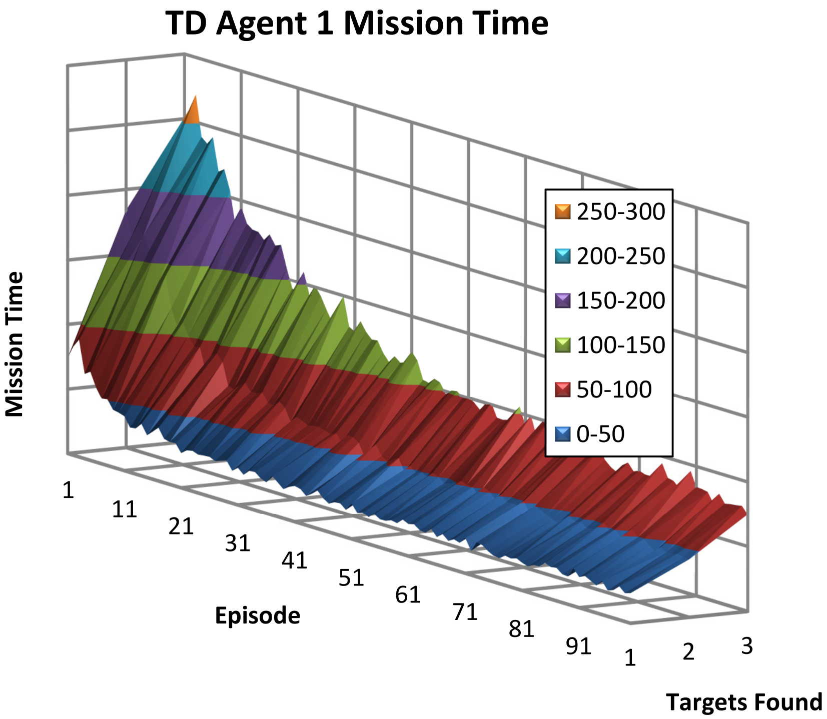


Figure 22: Mission Time plot for first TD Agent. The mission completion time decayed for all agents. Examination shows that after 50 episodes the time to complete the mission objectives is minimized and a steady-state behavior is achieved. Mission Time refers to the number of steps needed to locate 1,2, and 3 targets. Plots of the other systems can be found in Appendix A.

The averages of the last fifty episodes across all trials for all systems are displayed in Table 8. It can be seen that the TD implementations routinely achieved a lower mission completion time. It also appears that the perfect sensor systems for both the TD and MC implementation complete their missions quicker than the systems with imperfect sensors. Additionally, it can be seen that systems with a higher sensor performance metric achieved quicker target location times.

Table 8

Average mission time decision step values of last 50 episodes

Alpha	Number of Targets	4	7	10	Inf
TD	**1**	23.23	20.06	18.85	18.51
	2	47.91	41.58	39.12	36.26
	3	87.83	76.60	72.46	67.11
MC	**1**	33.39	29.62	27.71	26.32
	2	75.10	66.20	60.91	56.89
	3	136.39	125.16	119.24	110.45

Agent Reward Analysis

Navigation Agent Reward Analysis.

The last 20 data points for the average episodic reward for the TD Navigation Agents are used to compare the steady-state performance of the systems to one another (i.e. systems equipped with sensors with different alpha levels were compared). The last twenty data points used in the analyses of the Navigation and Search Agents are used as the effects of the temperature variables are minimal during these simulations and the Agents have acquired enough experience where their behavior is governed by relatively stable state and state-action values. The results of the different systems are compared to determine if the differences in sensor performance significantly affected the average episodic reward of the Navigation Agents. A significance value of 0.05 or less (i.e. the *p* value) is used to determine significance. SPSS was used to analyze the data. The descriptive data of this set is shown in Table 9.

Table 9

TD navigation agent steady-state descriptive data

α	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean Lower Bound	Upper Bound
Inf	20	-29.9030	3.86921	.86518	-31.7138	-28.0922
4	20	-50.4600	4.98323	1.11428	-52.7922	-48.1278
7	20	-39.4865	3.86040	.86321	-41.2932	-37.6798
10	20	-34.6580	3.58166	.80088	-36.3343	-32.9817
Total	80	-38.6269	8.66796	.96911	-40.5558	-36.6979

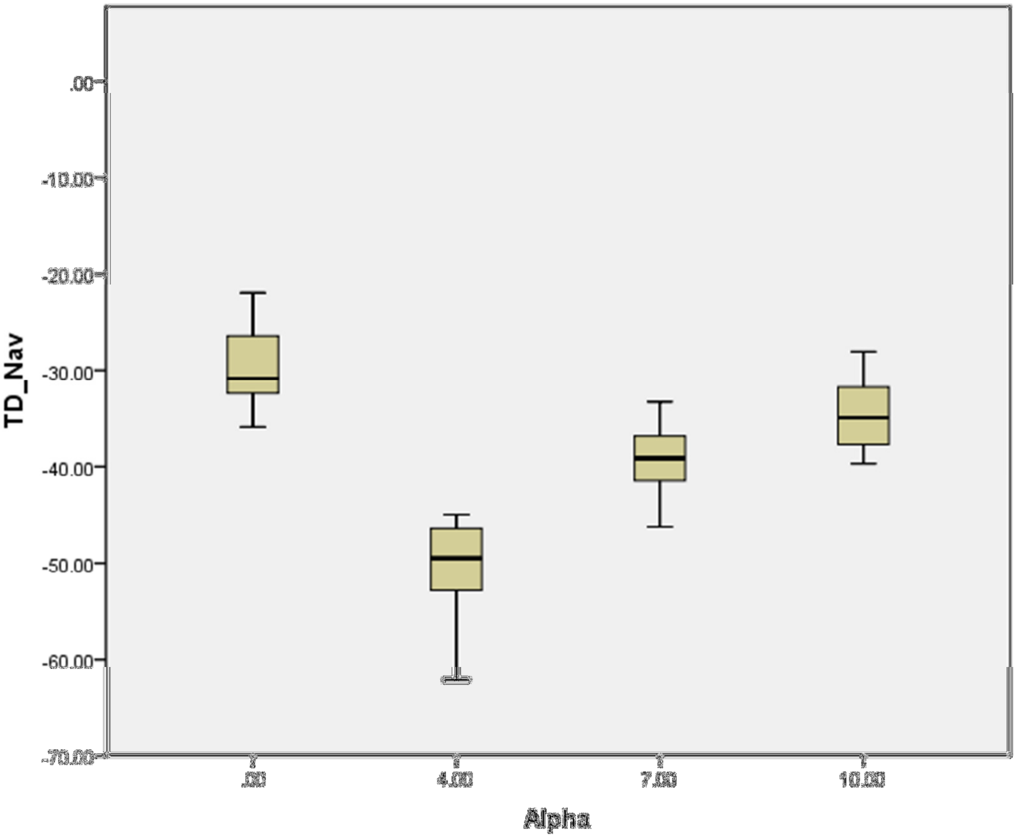


Figure 23: Box plot of the TD Navigation Agent Rewards. The populations used for the plot are the averages of the last 20 of all the episodes. The Alpha value of ".00" on the X-axis represents the output of the agent with the perfect sensor. The error bars represent the first and third quartile points. The whiskers show the values of the highest and lowest entries in the data set.

It can be seen that the mean decreased as the sensor performance increased, it is also noted that the standard deviation within each group is similar across groups. A significant difference was found in the imperfect sensor data sets. The results of the one-way ANOVA are shown in Table 10. A Levene statistic was calculated to test for the homogeneity of variance assumption required of an ANOVA. The Levene statistic showed that the homogeneity of variance assumption was valid as it was greater than 0.05 ($p = 0.737$).

Table 10

TD navigation agent ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	4652.395	3	1550.798	91.853	.000
Within Groups	1283.151	76	16.884		
Total	5935.546	79			

The ANOVA revealed that there was a significant difference between the groups ($p < 0.001$). It is therefore concluded that the type of sensor system does have an effect on the reward that the Navigation Agents receive. A Tukey's HSD test is used to assess the significance relationship between the different systems.

Table 11

Tukey HSD test results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	20.55700*	1.29937	.000
	7	9.58350*	1.29937	.000
	10	4.75500*	1.29937	.003
4	Inf	-20.55700*	1.29937	.000
	7	-10.97350*	1.29937	.000
	10	-15.80200*	1.29937	.000
7	Inf	-9.58350*	1.29937	.000
	4	10.97350*	1.29937	.000
	10	-4.82850*	1.29937	.002
10	Inf	-4.75500*	1.29937	.003
	4	15.80200*	1.29937	.000
	7	4.82850*	1.29937	.002

The Tukey's test reveals that all the systems are significantly different from one another. Based upon the results of the Tukey's HSD test and the descriptive data found in Table 7, it is concluded that the Navigation Agent episodic reward is highly sensitive to sensor performance.

The method used to compare the performance of the MC systems mirrors that used for the TD systems. The last 20 episodes are used for steady-state analysis. Significantly higher steady state rewards are considered to be superior systems.

Table 12

MC navigation agent steady-state descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Inf	20	-99.0835	8.49811	1.90023	-103.0607	-95.1063
4	20	-139.6030	9.21330	2.06016	-143.9150	-135.2910
7	20	-109.2790	8.07733	1.80615	-113.0593	-105.4987
10	20	-101.4005	8.27120	1.84950	-105.2715	-97.5295
Total	80	-112.3415	18.31005	2.04713	-116.4162	-108.2668

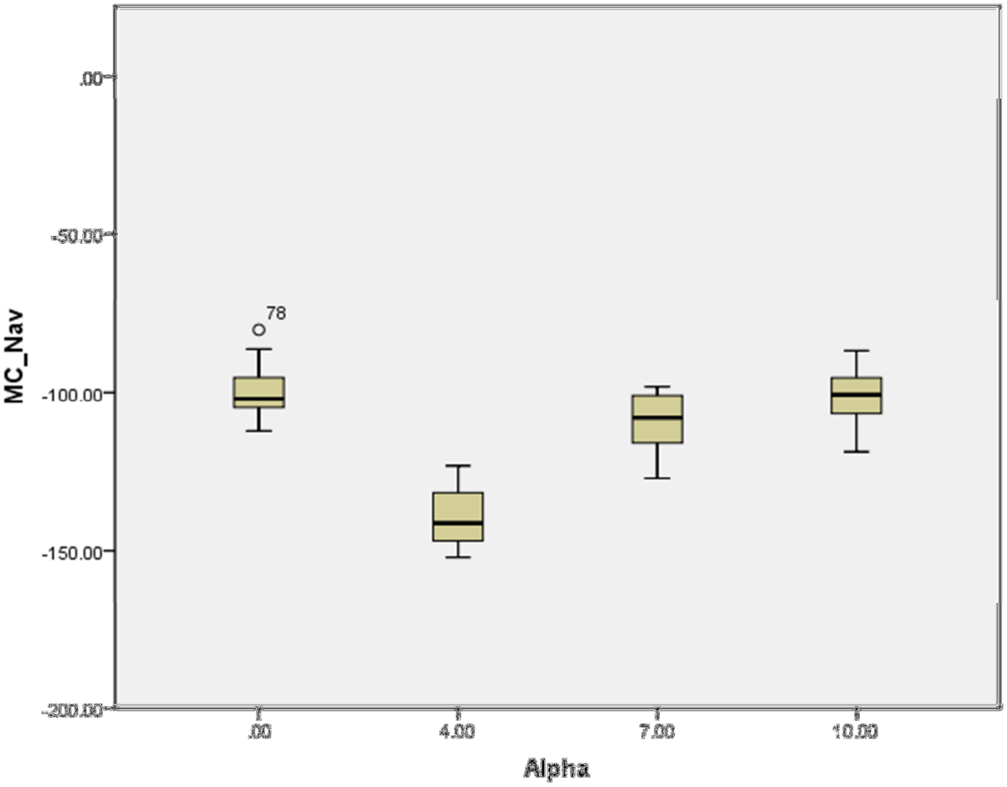


Figure 24: Box plot of the MC Navigation Agent Rewards. The populations used for the plot are the averages of the last 20 of all the episodes. The Alpha value of ".00" on the X-axis represents the output of the agent with the perfect sensor. The error bars represent the first and third quartile points. The whiskers show the values of the highest and lowest entries in the data set.

The data for the MC systems shows a lower reward average across the board. It also features higher variance levels. Again the standard deviation within each group is similar. The One-Way ANOVA shows that there is a significant difference ($p < 0.001$) between the MC systems. The Levene statistic revealed that the homogeneity of variance assumption was not violated ($p = 0.811$).

Table 13

MC navigation agent ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	20960.967	3	6986.989	96.121	.000
Within Groups	5524.414	76	72.690		
Total	26485.381	79			

A Tukey's HSD test is used to identify the source of significant differences found by the One-Way ANOVA. The results of this procedure are found below in Table 14. The analysis reveals that all systems were significantly different from one another except for the systems with a sensor sensitivity value of 10 and the perfect system. This result seems to indicate that the MC Navigation Agents were somewhat less sensitive to changes in sensor performance. Interestingly, this hints that there is most likely a diminishing return as a sensor approaches the infinite alpha value.

Table 14

MC navigation agent Tukey HSD results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	40.51950*	2.69610	.000
	7	10.19550*	2.69610	.002
	10	2.31700	2.69610	.826
4.00	Inf	-40.51950*	2.69610	.000
	7	-30.32400*	2.69610	.000
	10	-38.20250*	2.69610	.000
7.00	Inf	-10.19550*	2.69610	.002
	4	30.32400*	2.69610	.000
	10	-7.87850*	2.69610	.023
10.00	Inf	-2.31700	2.69610	.826
	4	38.20250*	2.69610	.000
	7	7.87850*	2.69610	.023

The last comparison is between the collective steady-state rewards of the MC and TD agents. The descriptive data for this comparison is found in Table 15. A homogeneity of variance test carried out in SPSS generated a significant Levene Statistic ($p < 0.001$). An ANOVA test carried out on this data would therefore not be valid. It can be seen in Table 15, however, that the combined TD Navigation Agents' episodic reward is much greater than that of the combined MC Navigation Agents. The difference in standard deviation between these two groups, while great enough to violate the assumption of homogeneity of variance, is not so great as lead to the conclusion that the Agents' episodic rewards are very likely the insignificantly different.

Table 15

TD & MC navigation agent descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean Lower Bound	Upper Bound
TD	80	-38.6269	8.66796	1.90023	-40.5558	-36.6979
MC	80	-112.3415	18.31005	1.84950	-116.4162	-108.2668
Total	160	-75.4842	39.63471	2.04713	-81.6726	-69.2957

Search Agent Reward Analysis.

The Search Agents were analyzed in a similar way as the Navigation Agents. The average reward across the last 20 trials was used to determine if there were any steady-state performance differences within the imperfect systems. The results for the TD Search Agents are shown below in Table 16.

Table 16

TD search agent descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Inf	20	-45.6103	12.78464	2.85873	-51.5936	-39.6269
4	20	-60.5410	15.58965	3.48595	-67.8372	-53.2448
7	20	-53.1910	13.83779	3.09422	-59.6673	-46.7147
10	20	-50.5940	12.45740	2.78556	-56.4242	-44.7638
Total	80	-52.4841	14.51087	1.62236	-55.7133	-49.2548

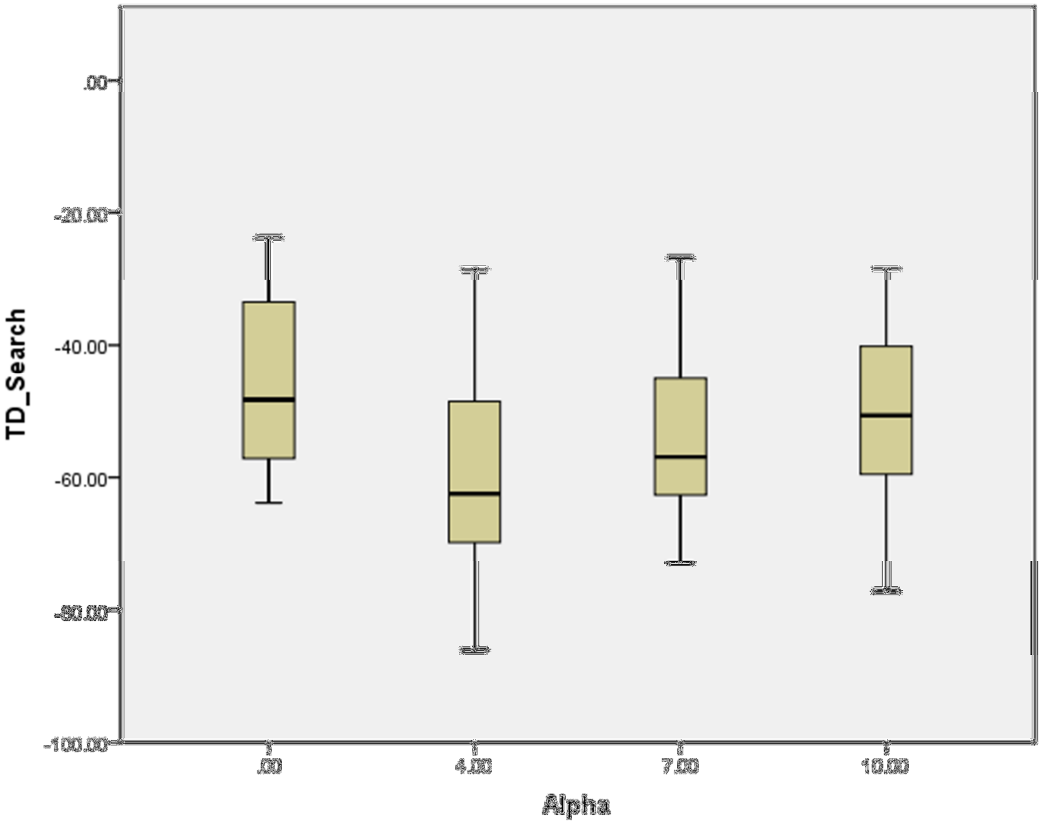


Figure 25: Box plot of TD Search Agent average rewards. Again, the whiskers represent the highest and lowest values found in a given set of data. The ".00" category depicts the results of the system with a perfect sensor.

The results of the ANOVA are shown in Table 12. A significant difference between the systems was found ($p = 0.009$). As before, a Tukey's HSD test is performed to determine what the relationship between the systems. The homogeneity of variance assumption was not violated (p = 0.850).

Table 17

TD search agent ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	2324.713	3	774.904	4.116	.009
Within Groups	14309.951	76	188.289		
Total	16634.664	79			

The Tukey's test revealed that only the worst system (with an alpha of 4) and perfect system exhibited a significant difference in their steady-state scores (p = 0.005). This is interesting as it signifies that the imperfect Search Agents seem to be able to mitigate the effects of sensors with different levels of performance. It also breaks from the pattern found in the TD Navigation Agent analysis.

Table 18

TD search agent tukey HSD results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	14.93075*	4.33923	.005
	7	7.58075	4.33923	.307
	10	4.98375	4.33923	.661
4.00	Inf	-14.93075*	4.33923	.005
	7	-7.35000	4.33923	.334
	10	-9.94700	4.33923	.109
7.00	Inf	-7.58075	4.33923	.307
	4	7.35000	4.33923	.334
	10	-2.59700	4.33923	.932
10.00	Inf	-4.98375	4.33923	.661
	4	9.94700	4.33923	.109
	7	2.59700	4.33923	.932

The analysis of the MC Search Agent rewards was carried out in the same manner as with the TD methods. The last 20 trials were used to compare the rewards of the MC Search Agents. The descriptive data is shown below in Table 19. As in the Navigation Agent Analysis section the MC Agents seem to feature lower mean performance values.

Table 19

MC search agent descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Inf	20	-72.0033	15.51416	3.46907	-79.2641	-64.7424
4	20	-93.2130	18.53801	4.14522	-101.8891	-84.5369
7	20	-83.1368	16.19789	3.62196	-90.7176	-75.5559
10	20	-82.8978	11.35186	2.53835	-88.2106	-77.5849
Total	80	-82.8127	17.07709	1.90928	-86.6130	-79.0124

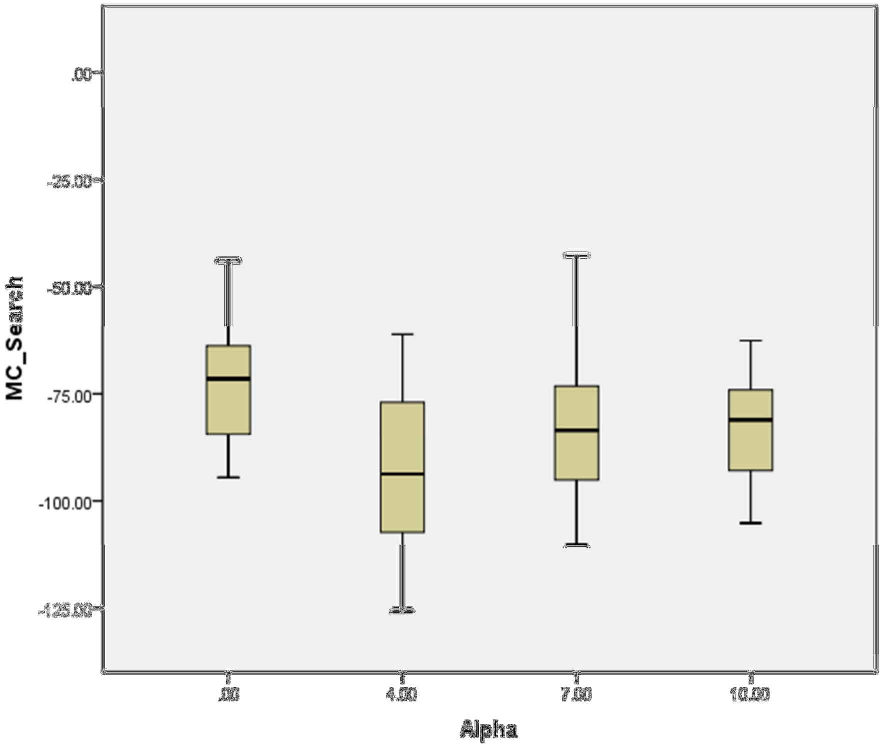


Figure 26: Box plot of the MC Search Agent average rewards. The ".00" category represents the data of the system with the perfect sensor. The whiskers represent the highest and lowest entries in the data.

The ANOVA for the MC Search Agent shows that there is a significant difference between the systems ($p = 0.001$). A Tukey's test is used to determine the specifics of the relations between the groups. Using the Levene test, the homogeneity of variance assumption was determined to not have been violated ($p = 0.211$).

Table 20

MC search agent ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	4502.454	3	1500.818	6.154	.001
Within Groups	18536.080	76	243.896		
Total	23038.534	79			

A significant difference was found between the perfect Search Agent and the Search Agent with a sensor sensitivity of 4 ($p < 0.001$). The first result mirrors that found in the TD Search Agent Analysis, the systems with sensor sensitivity ratings of 7 and 10 are not significantly different from one another or from the perfect system in regard to steady-state performance. This result again points to the conclusion that the RL architecture, in this case the MC system, is able to mitigate the effects of different sensor performance levels.

Table 21

MC search agent Tukey HSD results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	21.20975*	4.93858	.000
	7	11.13350	4.93858	.118
	10	10.89450	4.93858	.131
4	Inf	-21.20975*	4.93858	.000
	7	-10.07625	4.93858	.183
	10	-10.31525	4.93858	.166
7	Inf	-11.13350	4.93858	.118
	4	10.07625	4.93858	.183
	10	-.23900	4.93858	1.000
10	Inf	-10.89450	4.93858	.131
	4	10.31525	4.93858	.166
	7	.23900	4.93858	1.000

A comparison of the TD and MC systems' collective Search Agent rewards was completed using an ANOVA. The descriptive statistics used for this analysis are provided in Table 22. A homogeneity of variance test was completed before the ANOVA was conducted, and revealed that the homogeneity of variance assumption is valid for this data ($p = 0.360$). The results of the ANOVA shown in Table 23 indicated that there is a significant difference ($p < 0.001$) between the rewards obtained by the different Search Agent implementations.

Table 22

TD & MC search agent descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
TD	80	-52.4841	14.51087	1.62236	-55.7133	-49.2548
MC	80	-82.8127	17.07709	1.90928	-86.6130	-79.0124
Total	160	-67.6484	21.92988	1.73371	-71.0724	-64.2243

Table 23

TD & MC search agent ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	36793.020	1	36793.020	146.530	0.000
Within Groups	39673.198	158	251.096		
Total	76466.218	159			

Incorrect Declaration Behavior.

The Incorrect Declaration Behavior of the Search Agent was assessed in the same manner as the reward data. The last 20 trials were used as data points to examine the effects, if any, of the sensor sensitivity on the number of incorrect declarations occurring in a given trial. It is expected that the same general trends that were found in the Search Agent analysis section will be present in the Incorrect Declaration behavior.

The descriptive data for the TD incorrect declaration behavior is displayed in Table 24. It can be seen that the Agents using a higher performance sensor have a lower average number of incorrect declarations than the other agents. Again, the within group standard deviation is relatively uniform.

Table 24

TD incorrect declaration descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Inf	20	16.8175	3.70344	.82811	15.0842	18.5508
4	20	23.0265	4.85729	1.08612	20.7532	25.2998
7	20	20.2955	4.34889	.97244	18.2602	22.3308
10	20	19.3830	3.90774	.87380	17.5541	21.2119
Total	80	19.8806	4.70971	.52656	18.8325	20.9287

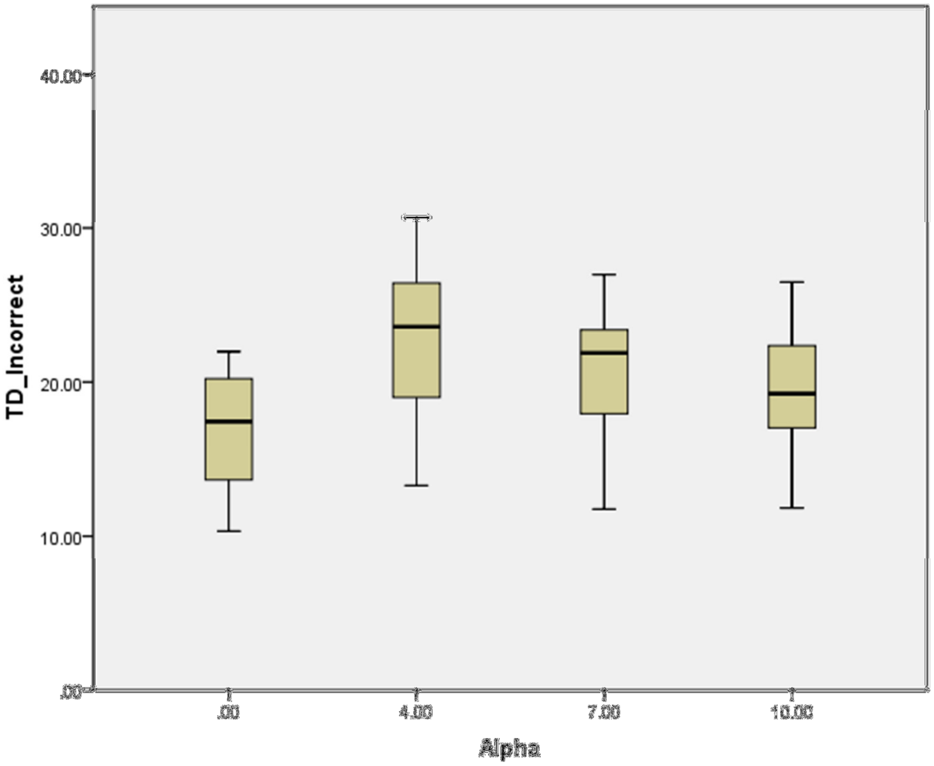


Figure 27: Box plot of TD Agent Incorrect Declarations. The ".00" category represents the data of the system using a perfect sensor. The whiskers represent the highest and lowest entries in each data set.

An insignificant Levene statistic was generated for the TD Incorrect Declaration data (p = 0.708), allowing an ANOVA test to be used to determine significant differences. The ANOVA test revealed a significant difference between the system data ($p < 0.001$). A Tukey's HSD test will be used to further analyze the systems' behavior in the same manner as the reward analyses. The results of the Tukey's test are shown in Table 25.

Table 25

TD incorrect declaration ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	393.980	3	131.327	7.348	.000
Within Groups	1358.347	76	17.873		
Total	1752.327	79			

The Tukey HSD test results are displayed in Table 26. A significant difference is found between the perfect system and the worst system (the agent with an alpha value of 4). This reinforces the analysis found in the TD Search Agent section. Interestingly though, there is a significant difference between the systems with sensor performance values of 4 and 10 as well. This relationship was not found in the TD Search Agent analysis. It is also found that there is no significant difference between the systems with higher sensor performance.

Table 26

TD incorrect declaration Tukey HSD results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	-6.20900*	1.33690	.000
	7	-3.47800	1.33690	.053
	10	-2.56550	1.33690	.229
4.00	Inf	6.20900*	1.33690	.000
	7	2.73100	1.33690	.182
	10	3.64350*	1.33690	.039
7.00	Inf	3.47800	1.33690	.053
	4	-2.73100	1.33690	.182
	10	.91250	1.33690	.903
10.00	Inf	2.56550	1.33690	.229
	4	-3.64350*	1.33690	.039
	7	-.91250	1.33690	.903

The descriptive data for the MC Agents is displayed in Table 27. The data reveals that the MC Search Agents, on average, did not generate a much larger number of incorrect declarations than the TD agents.

Table 27

MC incorrect declaration descriptive data

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Inf	20	17.6685	3.72033	.83189	15.9273	19.4097
4.00	20	23.1920	4.66682	1.04353	21.0079	25.3761
7.00	20	20.6725	4.21187	.94180	18.7013	22.6437
10.00	20	19.9625	3.48436	.77913	18.3318	21.5932
Total	80	20.3739	4.43647	.49601	19.3866	21.3612

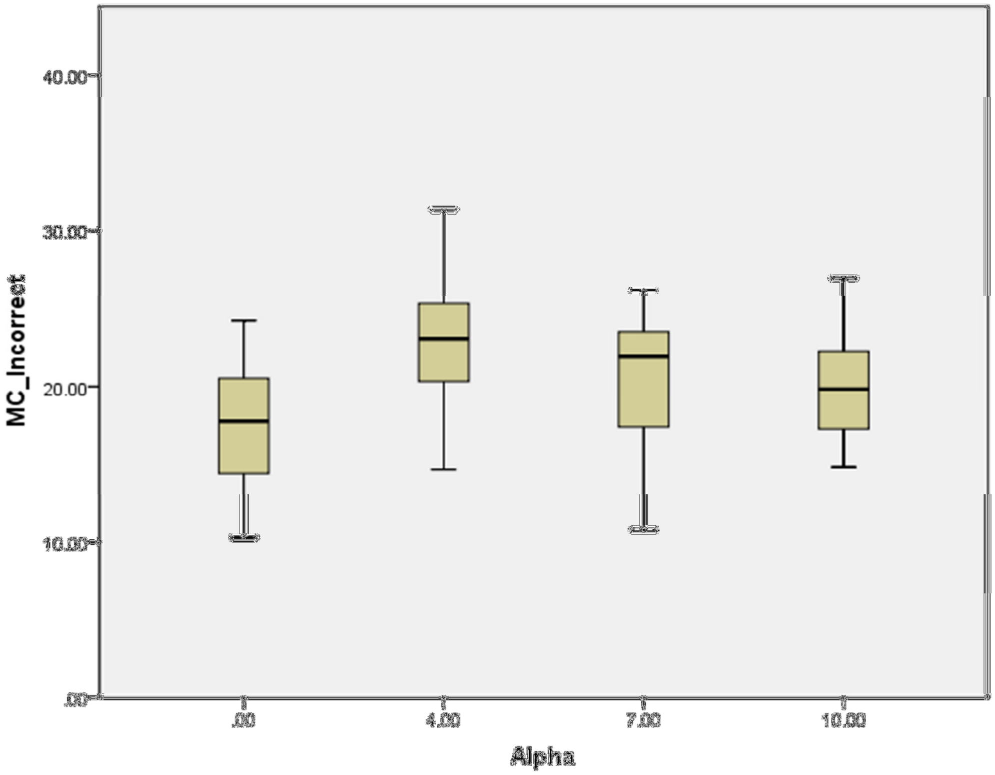


Figure 28: Box plot of MC Agent Incorrect Declarations. The ".00" category represents the output of the perfect system. The whiskers of each plot show the highest and lowest entries in the data set.

A Levene test was applied to the data and revealed that the assumption of homogeneity of variance was not violated (p = 0.806). The ANOVA results indicate that there is a significant difference between the numbers of incorrect declarations made by each system, dependent upon the performance of the sensor it uses (p = 0.001). A Tukey's HSD test is used to identify the source of the differences. The results of the ANOVA can be found Table 28.

Table 28

MC incorrect declaration ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	310.386	3	103.462	6.318	.001
Within Groups	1244.512	76	16.375		
Total	1554.897	79			

The test reveals that the significant difference is caused by the performance differences of the system with the sensor sensitivity of 4 and the perfect system ($p < 0.001$). This result mirrors the analysis found in the MC Search Agent section.

Table 29

MC incorrect declaration Tukey HSD results

Alpha	Alpha	Mean Difference	Std. Error	Sig.
Inf	4	-5.52350*	1.27965	.000
	7	-3.00400	1.27965	.096
	10	-2.29400	1.27965	.285
4	Inf	5.52350*	1.27965	.000
	7	2.51950	1.27965	.209
	10	3.22950	1.27965	.064
7	Inf	3.00400	1.27965	.096
	4	-2.51950	1.27965	.209
	10	.71000	1.27965	.945
10	Inf	2.29400	1.27965	.285
	4	-3.22950	1.27965	.064
	7	-.71000	1.27965	.945

An analysis of the incorrect declaration behavior between the Temporal Difference and Monte Carlo systems was also completed. The descriptive statistics for this analysis are provided below in Table 30. Examination of the data shows that the means for the two results are very close.

Table 30

TD & MC incorrect declaration behavior descriptive statistics

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
MC	80	20.374	4.43647	0.49601	19.3866	21.3612
TD	80	19.881	4.70971	0.52656	18.8325	20.9287
Total	160	20.127	4.56742	0.36109	19.4141	20.8404

An ANOVA was conducted on this data. The results of the ANOVA are shown in Table 31. An analysis for the homogeneity of variance in SPSS resulted in a Levene Statistic of 0.491, indicating that the homogeneity of variance assumption for the ANOVA was valid (*p* = 0.484). The ANOVA reveals that there were no significant differences in the collective number of incorrect declarations by the systems (*p* = 0.496).

Table 31

TD & MC incorrect declaration behavior ANOVA results

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	9.732	1	9.732	0.465	0.496
Within Groups	3307.225	158	20.932		
Total	3316.957	159			

The state-action value map of the Search Agents provided another way to view the behavior of the systems in terms of the number of incorrect declarations. The value mapping for all the Search Agents is found in Table 32. All TD agents seemed to converge to the same value for the declaration action, especially for the lower TOP states. TD Agents 3 and 4 sometimes have zeros for the declaration value of intermediate states. The cause of this was that the agent with a higher sensor performance tended to “skip” through the TOP ranges due to the nature of Bayes’ Rule for updating the TOP. The MC Agents also seem to converge to similar TOP state values, especially for lower TOP values.

Table 32

Search agent state action values

		TOP State									
Agent	Action	1	2	3	4	5	6	7	8	9	10
TD 1	Move	-0.695	-0.741	-0.690	-0.799	-0.880	-0.764	-0.803	-0.761	-0.783	-0.744
	Loiter	4.803	3.532	2.870	-0.453	-0.868	-0.800	-1.262	-1.289	-1.277	-1.274
	Declare	5.489	4.665	3.485	0.437	-0.771	-0.206	-0.957	-1.033	-1.090	-1.102
TD 2	Move	-0.761	-0.893	-1.264	-0.883	-0.998	-0.841	-0.873	-0.818	-0.857	-0.790
	Loiter	4.675	3.753	2.858	0.917	-0.993	-0.885	-1.361	-1.290	-1.322	-1.315
	Declare	5.315	4.169	4.176	1.408	-0.801	-0.585	-1.029	-1.014	-1.099	-1.108
TD 3	Move	-0.921	-0.500	-0.959	0.000	-0.932	0.000	-0.848	-0.882	-0.848	-0.805
	Loiter	4.395	0.000	1.988	-1.256	-0.854	0.000	-1.361	-1.371	-1.371	-1.369
	Declare	5.221	0.000	2.515	0.000	-0.377	0.000	-0.962	-1.033	-1.099	-1.110
TD 4	Move	-1.072	0.000	0.000	0.000	-1.050	0.000	0.000	0.000	0.000	-0.736
	Loiter	4.999	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	-1.294
	Declare	5.555	0.000	0.000	0.000	-0.821	0.000	0.000	0.000	0.000	-1.111
MC 1	Move	-3.178	-2.785	-3.107	-4.209	-4.518	-4.508	-4.639	-4.750	-4.727	-4.791
	Loiter	-1.190	-0.127	-1.247	-3.845	-4.668	-4.452	-4.792	-4.888	-4.902	-4.967
	Declare	2.030	0.932	-0.578	-3.570	-4.761	-4.409	-4.967	-5.165	-5.130	-5.226
MC 2	Move	-2.393	-2.698	-4.429	-3.720	-4.232	-4.406	-4.448	-4.636	-4.563	-4.702
	Loiter	0.573	-0.325	-2.020	-2.879	-4.419	-4.442	-4.668	-4.788	-4.792	-4.888
	Declare	1.756	0.406	0.844	-2.553	-4.543	-4.575	-4.857	-5.024	-5.002	-5.130
MC 3	Move	-2.688	0.074	-3.278	0.000	-4.174	0.000	-4.552	-4.345	-4.518	-4.641
	Loiter	0.590	0.000	-1.814	0.000	-4.143	0.000	-4.708	-4.594	-4.724	-4.831
	Declare	1.678	0.000	-1.312	5.000	-4.192	0.000	-4.843	-4.790	-4.961	-5.086
MC 4	Move	-2.703	0.000	0.000	0.000	-3.785	0.000	0.000	0.000	0.000	-4.314
	Loiter	1.127	0.000	0.000	0.000	-4.015	0.000	0.000	0.000	0.000	-4.567
	Declare	2.232	0.000	0.000	0.000	-4.193	0.000	0.000	0.000	0.000	-4.793

A plot of the declaration action values for the TD methods can be found in Figure 29. The plot reveals the TD agents are much more likely to declare the presence of a target if the TOP is greater than 0.5. The higher the TOP, the more likely the agent will declare the presence of a target. It is noted in Table 32 that the “Declare” value never gets substantially lower than the other action values for a given state. This means that at these low TOP values there remains a relatively high probability of that action being selected, increasing the number of false declarations. The number of false declarations would also inherently be greater for situations where the agent vehicle had to travel farther. To travel farther the agent would have to traverse more empty space, possibly selecting the "Declare" action. As value functions derive their values from the reward scheme used by an RL algorithm, the number of incorrect declarations could possibly be reduced by increasing the penalty for a false declaration or eliminating the ability of the Search Agents to declare target presence while in these TOP states.

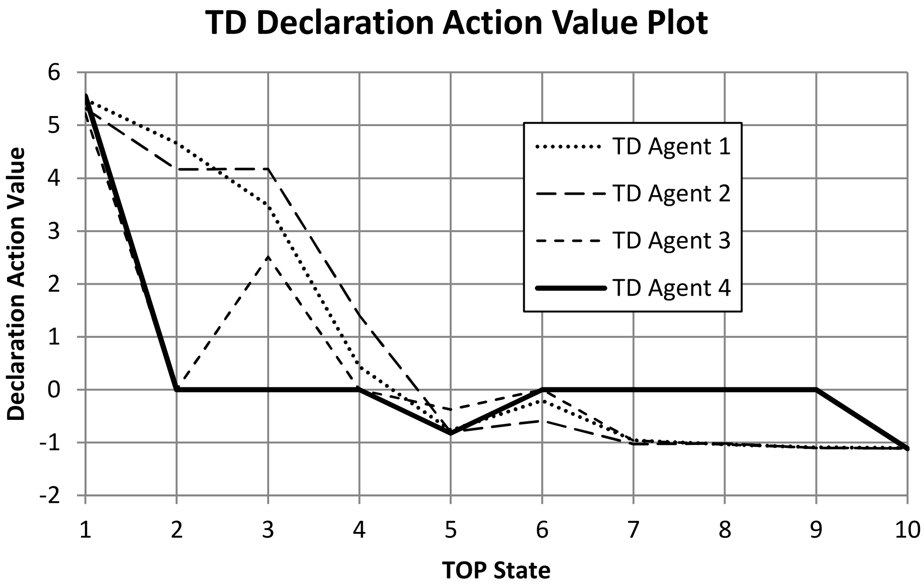


Figure 29: Plot of the TD Agent declaration action values. The plot shows that higher TOPs are more valued with respect to the declaration action. The perfect agent only experiences 3 operational TOP states, the highest, the lowest, and the once in which the agent is initialized at the beginning of every episode.

It is seen in Figure 29 that the state-action values for the Declaration action converge to the same number for all of the TD Search Agents. The parts of the value map that equal zero are states that were never encountered by a given system. The TD Agent 1 system seems to have the smoothest overall profile as it visited most of the states many times, allowing its value function to converge.

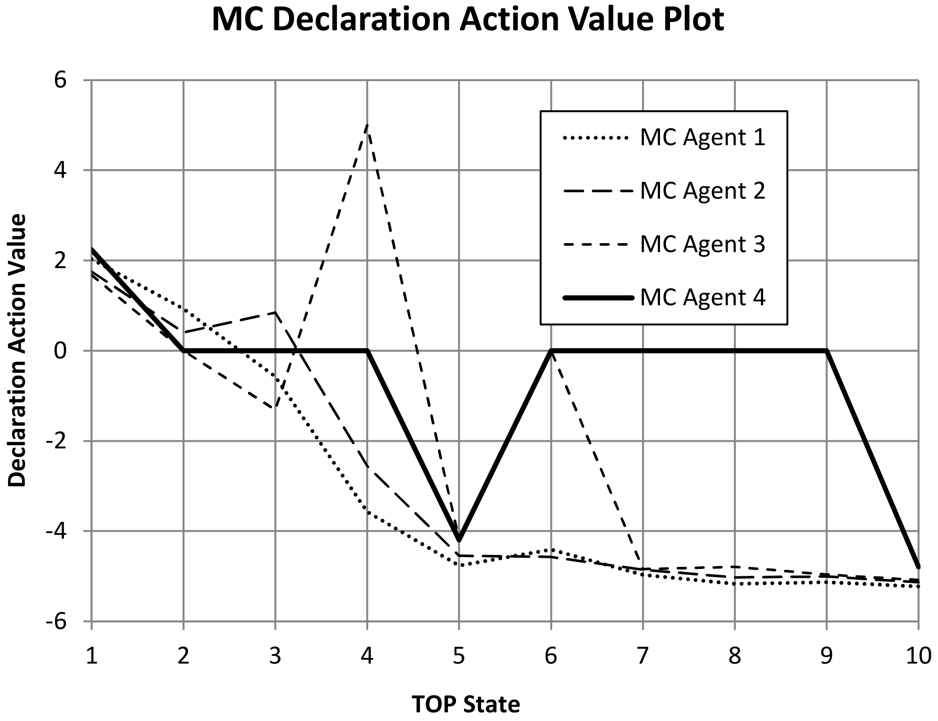


Figure 30: The plot of MC declaration values shows similar behavior to that found in the TD values. The states that have a value of zero have never been experienced by their respective systems, and therefore are never updated.

The same general trends are found in the MC Search Agent value maps. Figure 30 shows that the MC systems converged to the same values, although different values than those found in the TD Search Agent data. Again, the first agent, with the lowest sensor performance, has the smoothest curve due to the fact that it visited all of the states multiple times. A spike is present at TOP State 4 for the third system. Examination of the data reveals that this state was encountered only twice over the course of the entire simulation for this system. It is theorized that in order to generate a TOP in this state the system would have to generate a particular set of returns on an occupied position, with at least one return being a true result. The fact that this so rarely happens likely means that the position is occupied, thus resulting in the high value. This state value should thus be ignored as it was so rarely encountered.

Mission Time.

The time to mission completion decreased over the course of a given trial through repeated episodes. The rate of convergence for the minimization of target completion time was different between the TD and MC agents. The mission times for the TD and MC agents equipped with a sensor α value of 4 are shown below in Figure 31.

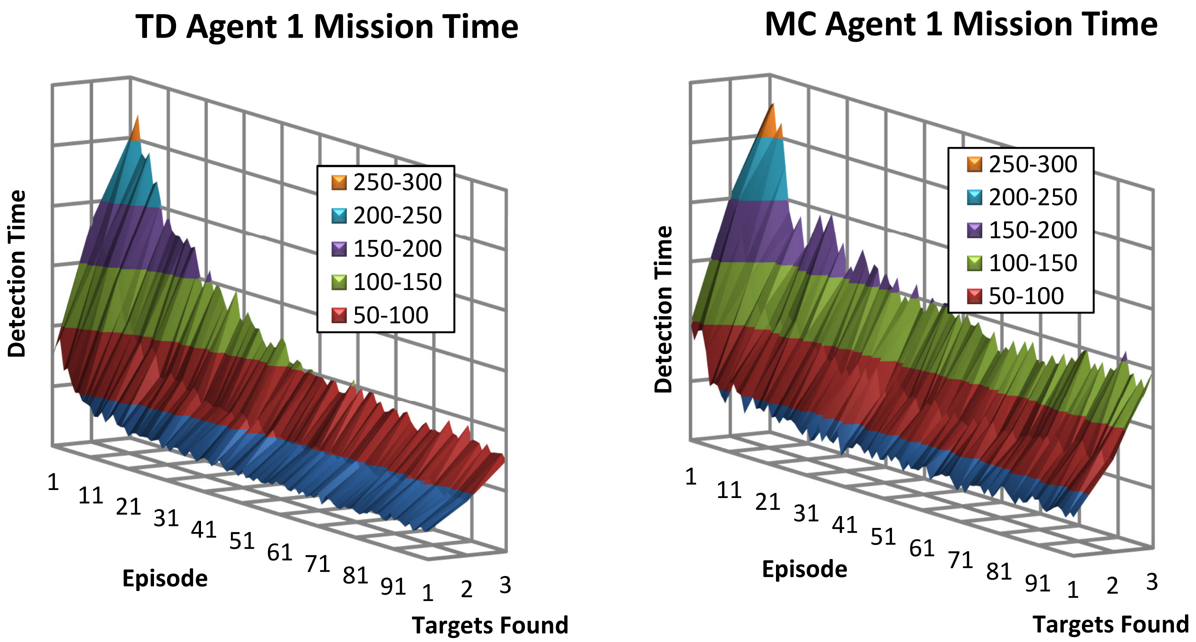


Figure 31: First (alpha = 4) TD and MC Agent Mission Time plots. The MC Agent's plot features an early period of rapid learning that lasts for the first 10 episodes, and then converges more slowly to a steady state value. The TD agent has a more gradual and longer learning period, and ultimately converges to a steady state mission time that is visibly lower than the MC system's result.

The graphs show mission times for the TD and MC agents with the worst sensor used in this study, but this behavior is typical of all the systems, including the perfect systems. The MC agents initially learn at an extremely high rate, which is seen in Figure 31. The TD agents learn more gradually, but eventually complete their missions much sooner than the MC agents. The results of the perfect systems are shown in Figure 32.

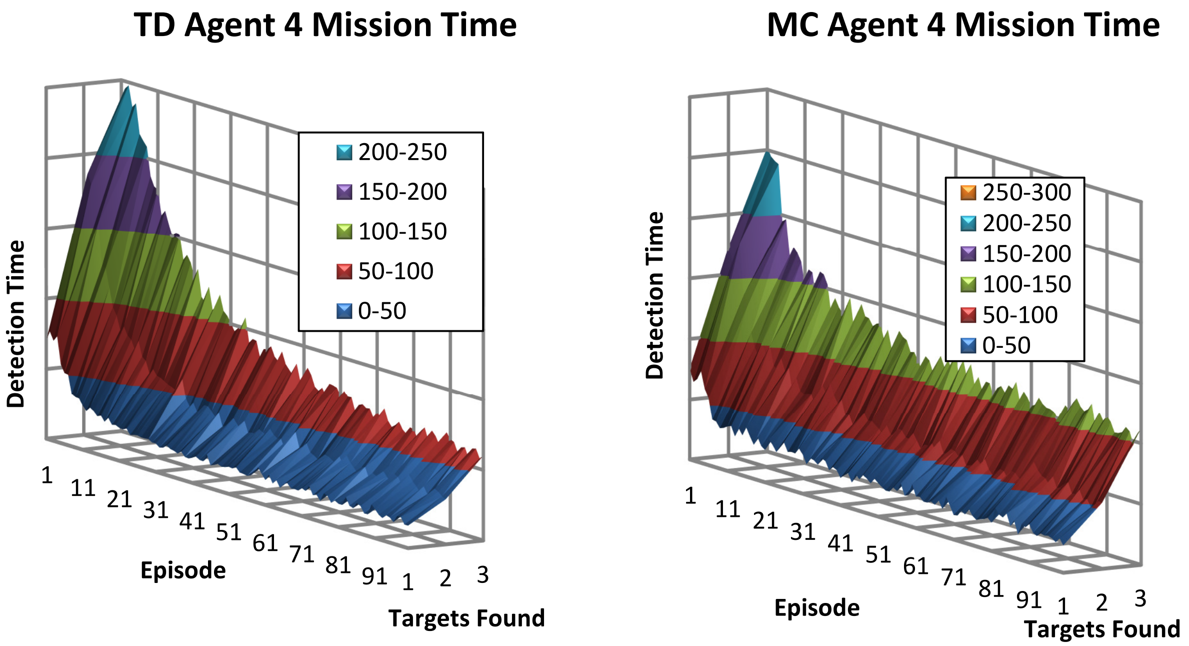
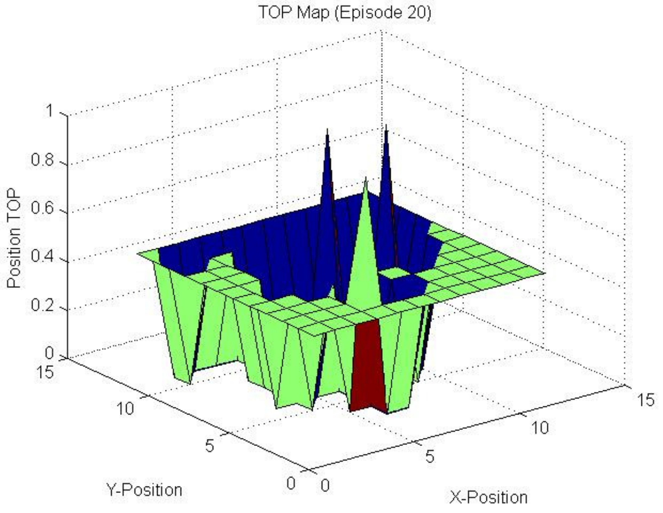


Figure 32: Perfect TD and MC Agent Mission Time Plots. The same trends as seen in the first systems are present in the perfect systems. The MC agent exhibits a period of rapid learning, but ultimately converges to a steady-state mission time that is higher than the results of the TD agent.

As can be seen in Figure 32, the initial rate of convergence for the perfect MC system is still faster than the equivalent TD system, even with a perfect sensor. Both systems show that the initial time to complete a mission is reduced compared to the imperfect systems. Additionally, the steady state performance of the perfect systems also improved compared to the imperfect systems. All system mission performance plots can be found in Appendix A of this study.

TOP Grid Comparison.

A TOP map comparison was performed on selected trials for all agents. The comparison is shown in Figure 35. The systems' TOP maps are compared to the map of the perfect system of a given agent type. Examples of the TOP plots are shown in Figure 33 and Figure 34. The plots were generated by the TD system with a perfect sensor (Agent 4) and the TD system with the worst sensor performance (Agent 1).



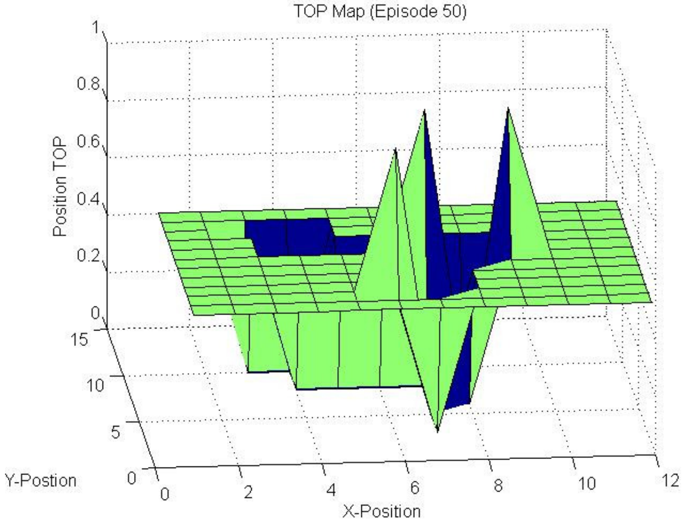


Figure 33: Perfect TD Agent TOP Map. The plot shows three spikes, which are equal to one, indicating that the agent found all the targets in this episode. A large portion of the map has a TOP value of zero. By episode 50, the agent rapidly travels to the likely target locations.

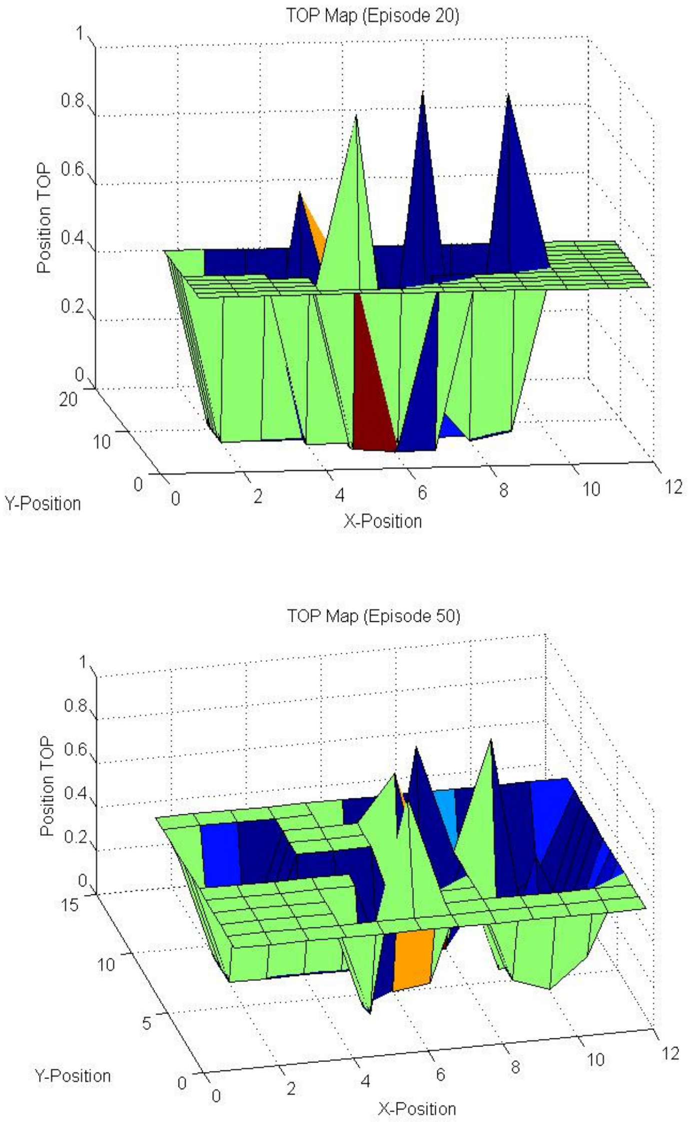


Figure 34: First TD Agent TOP Map. The TOP map generated by the TD system equipped with a sensor with a Hit Rate of 0.9 and a False Alarm Rate of 0.36. The agent finds all targets in both episodes, the same as the perfect system, but by episode 50 the agent hasn't reach the same efficiency with moving about the environment as the TD system with the perfect sensor.

The plots seem to suggest that the agents are reducing mission time by more quickly traveling to higher probability target locations. The perfect system generates a map with either a TOP of zero or one, with three spikes indicating target locations, showing zero False Alarm effects. The imperfect system's TOP map shows the results of the imperfect sensor on the TOP environment with TOP values between zero and one. To compare the systems more generally a procedure was devised where all of the systems were compared to the end result of their perfect versions across multiple trials. This procedure can be found in the TOP Map Comparison subsection of the Metrics section of this report.

Each map of a given system at a specific episode is compared to the map generated by the perfect system during the final episode. The results of this comparison are shown in Figure 35. The plots in Figure 35 are generated from the episodes of 10 trials averaged together.

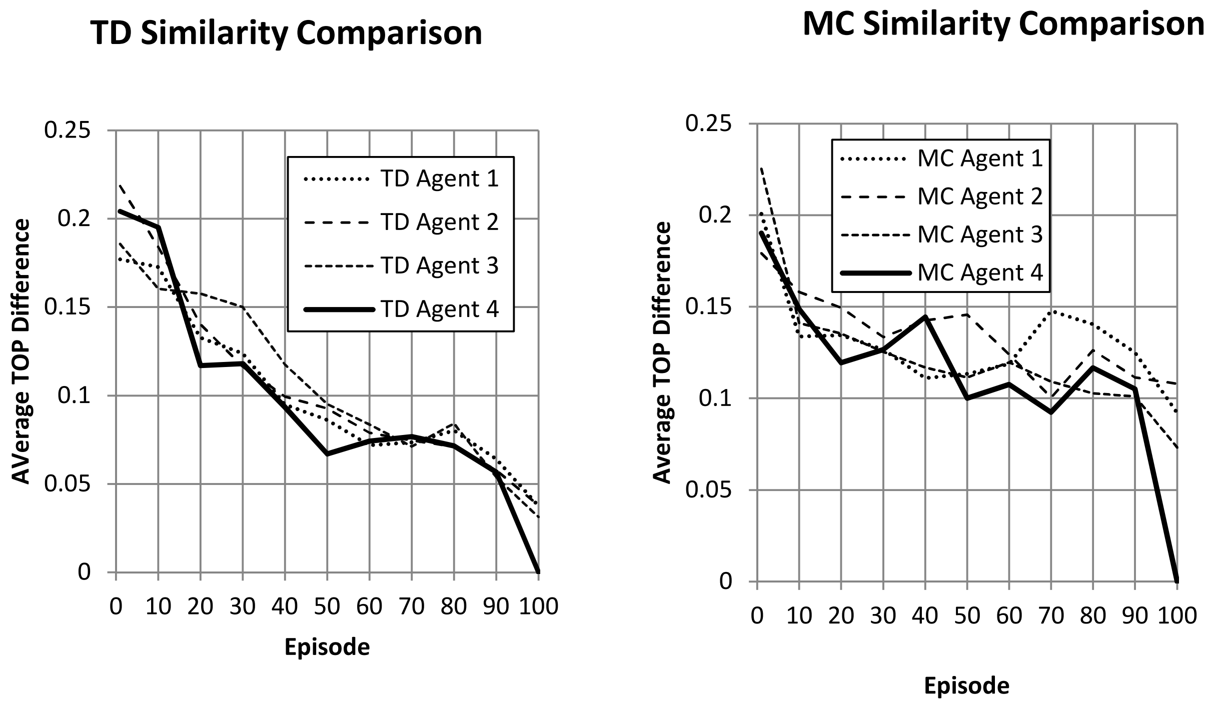


Figure 35: TOP similarity plots. The plot of the TOP map comparison data shows that, in general, the difference in the TOP maps decrease over the course of a trial. Interestingly, the TD agents seem to maintain very similar difference metric values over the course of the trial. The MC agents feature a large amount of variability in their metrics.

There is an overall downward trend in this data. This leads to the conclusion that the systems are converging to the final learned behavior exhibited by the perfect systems in each case. It seems that the TOP maps generated by the TD systems do not vary as much as the MC systems during their downward trend, meaning that their behavior is more uniform.

Discussion

The first hypothesis, that systems equipped with higher performance sensors will perform better than other systems, is confirmed by the evidence presented in this study. This is shown in the analyses and data of the incorrect declarations, mission time, mission completion, and reward plots. For the TD methods, the perfect Search Agent was only significantly different from the Agent with lowest sensor performance of 4, with no other significant differences. For the MC Search Agents the exact same relationships were found. The incorrect declaration analysis for the TD Search Agent systems resulted in the same conclusions, but the Search Agent with a performance of 4 (TD Agent 1) was significantly different from the perfect system and the system with a sensor performance of 10 (Agent 4 and Agent 3, respectively). These results were different from those found for the MC Search Agents with only the perfect and worst MC systems (Agents 4 and 1, respectively) exhibiting a significant difference in the number of Incorrect Declarations. For the Navigation Agent, all TD systems were significantly different from one another. The MC Navigation Agents had the same results, except Agents 2 and 3 (with sensor values of 7 and 10, respectively) where not significantly different. Overall, systems performed better with better sensors, but there is a diminishing return as the sensor sensitivity increases. This is shown by the sample means for the Navigation and Search Agent rewards and the number of incorrect declarations.

The mission times reduced according to the performance of the sensor, confirming the results of the Navigation Agent reward analysis. Small increases to the mission completion rate were observed, but overall the average mission completion rate was greater than 95% for all TD systems and 90% for all MC systems. The reduction of mission time as the performance of the sensor increased also shows that the agents with better sensors found their targets faster, reducing the number of penalties incurred and therefore increasing the average episodic reward.

The second hypothesis, that the TD systems would outperform the MC systems was proven through examination of the descriptive statistics for the Navigation Agents, and ANOVAs for the Search Agents and Incorrect Declaration Behavior. An ANOVA was not used for the Navigation Agents as the assumption of homogeneity of variance did not apply. The TD systems outperformed the MC systems in the Navigation Agent reward analysis by a factor greater than two. The Search Agent reward analysis also showed a clear superiority in the TD systems. Interestingly, the TD and MC systems produced similar incorrect declaration counts, but the mission completion rates for the MC systems were noticeably lower. The TOP map comparison reveals that both system types minimize the differences between their TOP grids and the final grid generated by the perfect system. The primary cause of this is believed to be the systems converging to the same paths between target regions.

The state-action values created by the Search Agents showed that the systems associated a higher TOP with greater value, as expected. The "Loiter" action also associates a higher value with higher TOP states, while the "Move" action values are relatively uniform across all TOP states. Interestingly, the Search Agents tended to give state-action pairs the same value between systems with the same implementation methods, regardless of the performance of the sensor that is used. This finding indicates that a cooperative setup could be used to transfer learning from one agent to another in a multi-agent operation, regardless of the type of sensor used. These values also provide a clue as to why both system types produced the same number of incorrect declarations. The values for the "Move" and "Loiter" actions are similar to those found for the "Declare" action. Since the "Declare" action is valued about the same as the other actions it has a high probability of being selected, even when in a low TOP states. The chief cause of this behavior is probably due to the reward scheme used for the simulation. Future work could change the reward scheme and/or implement a procedure where the Search Agents are not allowed to declare a target present unless the TOP is over a set value.

Conclusion

General Remarks

Very few studies have been published regarding the ability of RL systems to achieve mission goals given decreasing sensor performance. This study attempts to shed some light on the subject in an abstract way to maximize the potential applicability of its findings. Through the use of Signal Detection Theory, the sensor performance for the systems is quantified and described in way that was still applicable to wide variety of real world systems such as radar, image processing, infrared, and other sensing mechanisms. The use of Bayes' Rule allows probabilities of target occupancy to be updated in the simulation, and therefore a tradeoff mechanism through which the RL systems need to experiment with and learn. The targets were given a simple, random behavior for which the RL systems needed to adjust. The system worked with simulated, perfect human feedback to modify its behavior. The RL system was not able to plan ahead in this study, and had no way of knowing the TOP state of future positions. However, overall the agents were able to meet their objectives and did show a tendency to reduce the number of false declarations made by the system.

Future Work

Numerous opportunities for future development in this avenue of research exist. One of the simplest is to see how the reward scheme affects agent performance. If correct declarations are given higher values relative to the other rewards, the systems will become more risk tolerant and vice versa. Another research topic would be to have the systems for some set period take only the most optimal actions and analyze the behavior, especially for the incorrect declaration metric.

More work can be done to make the simulation better reflect real world situations. Target regions could move over time in some logical fashion. Initial TOP estimates could be made more realistic and non uniform. Another idea is to have the sensor sensitivity change with position or time in the environment. A real-world human operator could also be introduced to examine the effects of human error on the systems' ability to learn.

A third area of study could be to incorporate different entities into the environment. These can take the form of threats, different types of targets, and other allied vehicles. One possible idea would be to have a fully manned system operating in the environment and have the unmanned, agent controlled systems "observe" the situations and actions of the operator. Signal Detection Theory also allows for the possibility of multiple target types being detected. Combined with unique behavior it would be interesting to see how a RL system could adapt to the imperfect sensing mechanism in this case. A hybrid system could be built where certain types of states are updated using Monte Carlo methods in-between sorties and other states are updated using the Temporal Difference method.

Closing Remarks

This study should provide a good baseline for future developments in the area of operational systems using Reinforcement Learning algorithms to achieve a flexible and effective autonomous behavior. Based upon the findings of this research real world Reinforcement Learning systems will be impacted by the performance of their sensors, but a point of diminishing returns exists for higher sensor performance. Further research should be undertaken with more realistic scenarios and more sophisticated algorithms to increase knowledge of these systems' behavior is real-world, uncertain environments that will have to interact with a human supervisor or operator to accomplish a mission. Such research could aid in the development of a real-world deployable system.

References

Ackerman, S. (2013, February 5). Drone boosters say farmers, not cops, are the biggest U.S. robot market. *Wired*. Retrieved from: http://www.wired.com/dangerroom/2013/02/drone-farm/

Alpaydin, E. (2010). Introduction to machine learning. Cambridge, USA: The MIT Press.

Balakrishna, P., Ganesan, R., & Sherry, L. (2010). Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. *Transportation Research, 18(6),* 950-962. doi: 10.1016/j.trc.2010.03.003

Blanchard, B., & Fabrycky, W. (2011). Systems engineering and analysis. Upper Saddle River: Prentice Hall.

Blasco, J., Aleixos, N., Roger, J. M., Rabatel, G., & Molto, E. (2002). Robotic weed control using machine vision. *Biosystems Engineering, 83(2),* 149-157. doi: 10.1006/bioe.2002.0109

Buck, S., Beetz, M., & Schmitt, T. (2002). Approximating the value function for continuous space reinforcement learning in robot control. *Proceedings of the 2002 IEEE International Conference on Intelligent Robots and Systems, Switzerland, 1,* 1062-1067. doi: 10.1109/IRDS.2002.1041532

Canning, J. S. (2006). A concept of operations for armed autonomous systems. *3rd Annual Disruptive Technology Conference, USA*

Casbeer, D. W., Beard, R. W., McLain, T. W., Li, S., & Mehra, R. K. (2005). Forest fire monitoring with multiple small UAVs. *Proceedings of the 2005 American Control Conference, USA, 5,* 3530-3535. doi: 10.1109/ACC.2005.1470520

Clough, B. T. (2005). Unmanned aerial vehicles: autonomous control challenges, a researcher's perspective. *2nd AIAA "Unmanned Unlimited" Systems Technologies, and Operations Conference*, *USA, 1,* 1-15. doi: 10.1007/0-306-47536-7_3

Conn, K., & Peters, R. A. (2007). Reinforcement learning with a supervisor for a mobile robot in a real-world environment. *Proceedings of the 2007 International Symposium on Computational Intelligence in Robotics and Automation, USA,* 73-78. doi: 10.1109/CIRA.2007.382878

Costa, E., & Gouvea, M. (2010). Autonomous navigation in dynamic environments with reinforcement learning and heuristic. *2010 Ninth International Conference on Machine Learning Applications, USA,* 37-42. doi: 10.1109/ICMLA.2010.13

Fasano, G., Forlenza, L., Tirri, A., Accardo, D., & Moccia, A. (2011). Multi-sensor data fusion: a tool to enable UAS integration into civil airspace. *2011 IEEE/AIAA 30th Digital Avionics Systems Conference, USA,* 1-15. doi: 10.1109/DASC.2011.6096082

Green, D. M., & Swets, J. A. (1966). *Signal detection theory and psychophysics*. New York City, New York: John Wiley and Sons.

Gupte, S., Masoud, O., Martin, R., & Papanikolopoulos, N. (2002). Detection and classification of vehicles. *IEEE Transactions on Intelligent Transportation Systems, 3(1),* 37-47. doi: 10.1109/6979.994794

Jacques, D. R. (2003). Search, classification and attack decisions for cooperative wide area search munitions. *Cooperative control,: models, applications, and algorithms.,1,* 75-93. doi: 10.1007/978-1-4757-3758-5_5

Jin, Y., Polycarpou, M. M., & Minai, A. A. (2004). Cooperative real-time task allocation among groups of UAVs. In S. Butenko, R. Murphey, & P. M. Pardalos (Eds.), *Recent Developments in Cooperative Control and Optimization* (pp. 207-224). Norwell, Massachusetts: Kluwer Academic Publishers.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research, 4,* 237-285. doi: 10.1613/jair.301

Krothapalli, U., Wagner, T., & Kumar, M. (2011). Mobile robot navigation using variable grid size based reinforcement learning. *Infotech at Aerospace 2011, USA* 1-11. doi: 10.2514/MIAA11

Lai, J., Ford, J., Mejias, L., & O'Shea, P. (2012). See and avoid using onboard computer vision. In P. Angelov (Ed.), *Sense and avoid in UAS: research and applications* (pp. 256-272). Hoboken: John Wiley & Sons. doi: 10.1002/9781119964049.ch10

Lake, J. (2012, October). The unmanned future: great white hope or impossible dream? *Combat Aircraft Monthly, 13 (10),* 58-63.

Li, S.-M., Boskovic, J. D., Seereeram, S., Prasanth, R., Amin, J., & Mehra, R. K. (2002). Autonomous hierarchical control of multiple unmanned combat air vehicles (UCAVs). *Proceedings of the 2002 American Control Conference, USA, 1,* 274-279. doi: 10.1109/ACC.2002.1024816

Lin, L., Xie, H., & Shen, L. (2009). Application of reinforcement learning to autonomous heading control for bionic underwater robots. *Proceedings of the 2009 IEEE Informational Conference on Robotics and Biometrics, USA,* 2486-2490. doi: 10.1109/ROBIO.2009.5420445

Luongo, S., Vito, V. D., Fasano, G., Accardo, D., Forlenza, L., & Moccia, A. (2011). Automatic collision avoidance system: design, development and flight tests. *2011 IEEE/AIAA 30th Digital Avionics Systems Conference, USA,* 1-10. doi: 10.1109/DASC.2011.6096080

Macmillan, N. A., & Creelman, C. D. (1991). *Detection theory: A user's guide*. New York, USA: The Press Syndicate of the University of Cambridge.

Madrigal, A. (2009, October 19). Self-steered tractors and UAVs: Future farming is (finally) now. *Wired*. Retrieved from: http://www.wired.com/wiredscience/2009/10/precisionfarming/all/

Martinez-Marin, T., & Rodriguez, R. (2007). Navigation of autonomous vehicles in unknown environments using reinforcement learning. *2007 IEEE Intelligent Vehicles Symposium, Istanbul,* 872-876. doi: 10.1109/IVS.2007.4290226

Matignon, L., Laurent, G. J., & Fort-Piat, N. L. (2006). Improving reinforcement learning speed for robot control. *International Conference on Intelligent Robots and Systems, Beijing*, 3172-3177. doi: 10.1109/IROS.2006.282341

Merino, L., Caballero, F., Martinez-de-Dois, J. R., & Ollero, A. (2005). Cooperative fire detection using unmanned aerial vehicles. *Proceedings of the 2005 IEEE International*

Conference on Robotics and Automation, Barcelona, 1884-1889. doi: 10.1109/ROBOT.2005.1570388

Nygard, K., Chandler, P., & Pachter, M. (2001). Dynamic network flow optimization models for air vehicle resource allocation. *Proceedings of the 2001 American Control Conference, 3, Arlington, VA,* 1853-1858. doi: 10.1109/ACC.2001.946006

Orello, A., Arrue, B. C., Martinez, J. R., & Murillo, J. J. (1993). Techniques for reducing false alarms in infrared forest-fire automatic detection systems. *Control Engineering Practice, 7 (1),* 123-131. doi: 10.1016/S0967-0661(98)00141-5

Panella, I. (2008). Artificial intelligence methodologies applicable to support the decision-making capability on board unmanned aerial vehicles. *ECSIS Symposium on Bio-inspired Learning and Intelligent Systems for Security, Edinburgh,* 111-118. doi: 10.1109/BLISS.2008.14

Perron, J., Hogan, J., Moulin, B., Berger, J., & Belanger, M. (2008). A hybrid approach based on multi-agent geosimulation and reinforcement learning to solve a uav patrolling problem. *Winter Simulation Conference, Austin, Texas,* 1259-1267. doi: 10.1109/WSC.2008.4736198

Ribeiro, C. (2002). Reinforcement learning agents. *Artificial Intelligence Review, 17 (3),* 223-250. doi: 10.1023/A:1015008417172

Shachtman, N. (2008, April 21). More spy drones, less information? *Wired*. Retrieved from: www.wired.com/dangerroom/2008/05/defense-secreta/

Singer, P. W. (2009). *Wired for war*. New York City, New York: Penguin Group.

Sinopoli, B., Micheli, M., Donato, G., & Koo, T. J. (2001). Vision based navigation for an unmanned aerial vehicle. *Proceedings of the 2001 IEEE International Conference on Robotics & Automation, 2, Seoul, South Korea,* 1757-1764. doi: 10.1109/ROBOT.2001.932864

Slaughter, D. C., Giles, D. K., & Downey, D. (2008). Autonomous robotic weed control systems: A review. *Computers and Electronics in Agriculture, 61(1),* 63-78. doi: 10.1016/j.compag.2007.05.008

Smart, W. D., & Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. *Proceedings of the 2002 IEEE International Conference on Robotics & Automation, 4, Washington, DC,* 3404-3410. doi: 10.1109/ROBOT.2002.1014237

Stafylopatis, A., & Blekas, K. (1998). Autonomous vehicle navigation using evolutionary reinforcement learning. *European Journal of Operational Research, 108(2),* 306-318. doi: 10.1016/S0377-2217(97)00372-X

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. Cambridge: The MIT Press.

Takeo, K., Yoshiki, N., & Ichiro, M. (2002). Preceding vehicle recognition based on learning from sample images. *IEEE Transactions on Intelligent Transportation Systems, 3(4),* 252-260. doi: 10.1109/TITS.2002.804752

Tian, Y.-T., Yang, M., Qi, X., & Yang, Y. (2009). Multi-robot task allocation for fire-disaster response based on reinforcement learning. *2009 Internation Conference on Machine Learning and Cybernetics, 4, Baoding,* 2312-2317. doi: 10.1109/ICMLC.2009.5212216

Valasek, J., Doebbler, J., Tandale, M., & Meade, A. (2008). Improved adaptive-reinforcement learning control for morphing unmanned vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, 38(4),* 1014-1020. doi: 10.1109/TSMCB.2008.922018

Wei, W., Zhang, Q., & Wang, M. (2001). A method of vehicle classification using models and neural networks. *Vehicular Technology Conference, 4,* 3022-3026. doi: 10.1109/VETECS.2001.944158

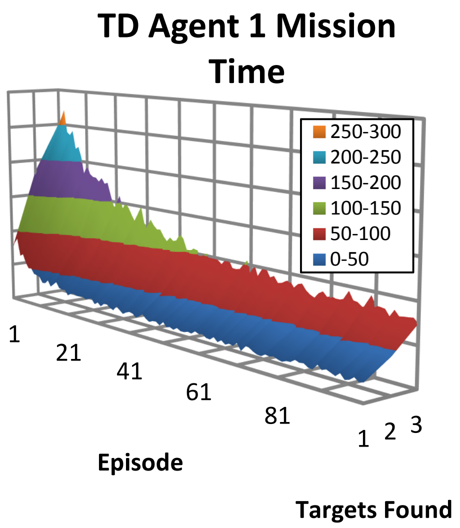
Whitehead, S. D., & Long-Jin, L. (1995). Reinforcement learning of non-Markov decision processes. *Artificial Intilligence,73(1-2),* 271-306. doi: 10.1016/0004-3702(94)00012-P

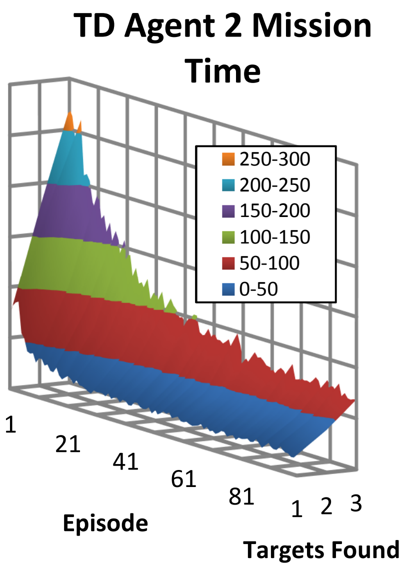
Winnefield, J. A., & Kendall, F. (2012). Unmanned systems integrated roadmap FY2011-2036 (reference number 11-S-36130). Department of Defense.

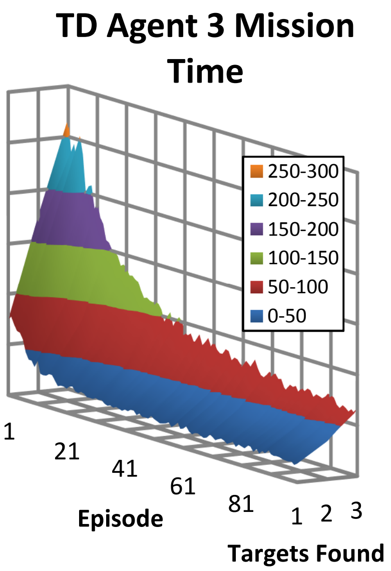
Yanli, Y., Minai, A. A., & Polycarpou, M. M. (2004). Decentralized cooperative search by networked UAVs in an uncertian environment. *Proceedings of the 2004 American Control Conference,6,USA,* 5558-5563.

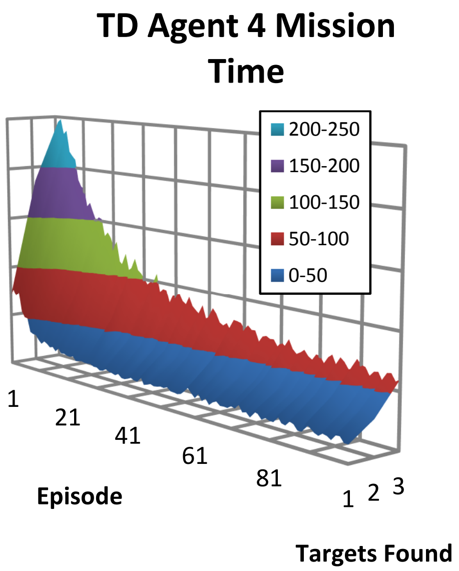
Yen, G., & Hickey, T. (2004). Reinforcement learning algorithms for robotic navigation in dynamic environments. *ISA Transactions*, *43(2)*, 217-230. doi: 10.1016/S0019-0578(07)60032-9

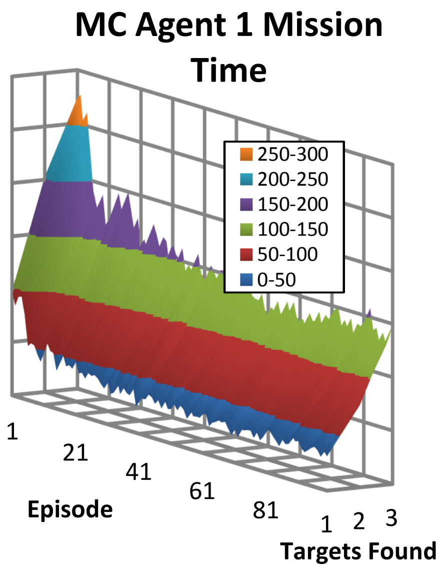
Appendix A: Mission Time Plots



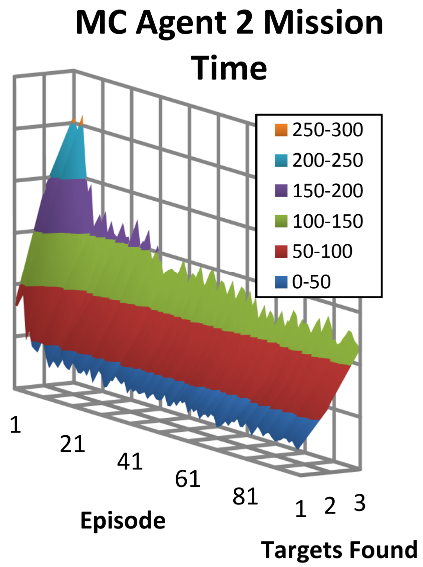




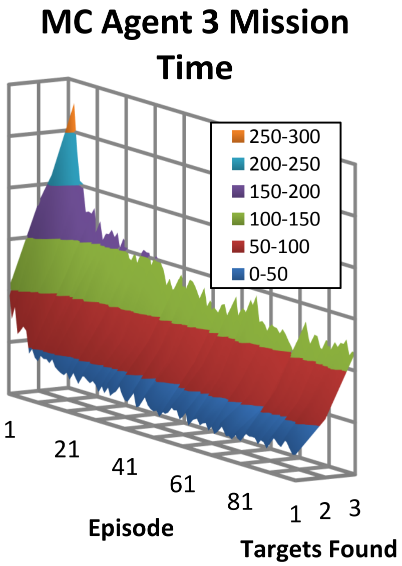




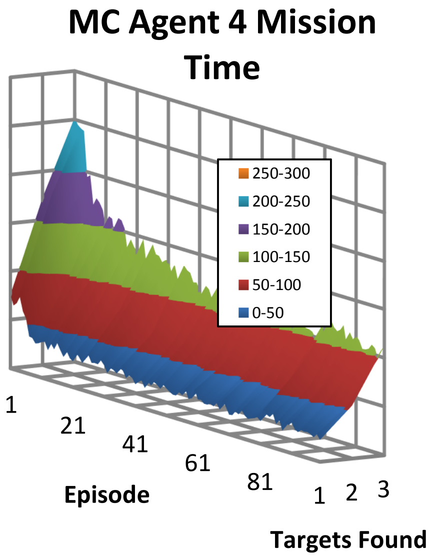
MC Agent 1 Mission Time



MC Agent 2 Mission Time



MC Agent 3 Mission Time



MC Agent 4 Mission Time