Using Intelligent Anticipation to Improve Error-Prone Communication in Social Robots

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Juan F. Marulanda

Major Professor: Terence Soule, Ph.D.

Committee Members: Dean Edwards, Ph.D.; Robert Heckendorn, Ph.D.;

Ata Zadehgol, Ph.D.

Department Administrator: Terence Soule, Ph.D.

May 2020

Authorization to Submit Dissertation

This dissertation of Juan F. Marulanda, submitted for the degree of Doctor of Philosophy with a major in Computer Science and titled "Using Intelligent Anticipation to Improve Error-Prone Communication in Social Robots", has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor _____Date_____
Terence Soule, Ph.D.

Committee Members _____Date_____
Robert Heckendorn, Ph.D.

_____Date_____
Dean Edwards, Ph.D.

_____Date_____
Ata Zadehgol, Ph.D.

Department Administrator _____Date_____
Terence Soule, Ph.D.

Abstract

This dissertation is centered on the study of anticipation behaviors based on Artificial Intelligence (AI) strategies as a method to improve the behavior of groups of autonomous ground robots by inferring the missing messages that get lost due to a noisy outdoor environment. These missing messages represent the information that the robots share as a group in order to stay synchronized. Also, this dissertation includes the study of an AI strategy as a method to recover corrupted phonetic messages. This research is inspired on the results of my Master thesis and expands on it to include more advanced anticipation techniques with a more complex navigation and sensors system for large environments; this research includes the use of GPS and compass for navigation and a (simulated) outdoor environment that is more realistic for many robotic applications and makes the inputs to the navigation system of the robot much noisier and more complex. This research also uses a small fleet of social robots whose collective behavior is less predictable and more complex. These issues make accurate communication between the robots a key and important feature and increase the need for anticipation as an aid to recover from faulty messages and events. A faulty event occurs when the communication between the robots gets interrupted and a message does not reach its destination.

In addition, this work expands the study of anticipation from two robots (one leader and a follower) to larger groups of robots (one leader and multiple followers). In this case, the followers attempt to stay in formation while the group navigates between waypoints. This means that this group of robots must coordinate their navigation control with their formation control in a complex, simulated outdoor environment. The anticipation models presented in this research are based on Fuzzy Logic and Artificial Neural Networks models (ANN). The last one was trained using methods like Backpropagation and a Genetic Algorithm. We also designed the anticipation models based on two structures that are commonly used in System Identification

Theory: AutoRegressive with eXogeneous input (ARX) and AutoRegressive Moving Average with eXogenous input (ARMAX). This research includes tests of how well anticipation works to improve coordination in complex environments.

Also, we introduced an additional approach to message error correction based on syntax and phonetic inference as a complementary tool to message anticipation. This is based on the idea that some messages are not lost completely but their content can get corrupted. This approach uses an inference-based approach by implementing fuzzy logic theory in order to fully recover these corrupted messages. To test this idea, an approach based on human-robot interaction through voice commands was implemented. Here, external noise or even a user with a strong accent can confuse a speech engine and produce bad data. We define this as corrupted data and we use our solution to fix it.

The results showed that the ARMAX models were more successful in reducing the distance error between both a pair of robots and a larger group of robots than the ARX models while the Leader Robot run missions included in the training data. On the other hand, the ARX models were more successful than the ARMAX models in developing a generalization behavior while the Leader Robot run missions that were not similar to the training data. Also, we observed that the ANN models that were trained with Genetic Algorithms had better results than the ANN models trained with Backpropagation. We also noticed that the robots were able to recover their formation from collisions that happened during their path. But they still can only take a certain amount of collisions before running out of time to reorganize themselves. In addition, the results from the speech experiments showed that our approach was successful in recovering corrupted messages created by a speech engine after generating homophone words in a voice command.

Acknowledgements

I would like to thank the Office of Naval Research for their support and funding during the first 4 years of the Ph.D. through "Configurable UUV Sensor Network", ONR N00014-14-1-0662. Also, I would like to thanks Schweitzer Engineering Laboratories for their support and funding during the last year through "Air-core reactor inter-turn fault detection, using magnetic field sensors".

In addition, I would like to thank John Canning for his advise on sensor design and navigation systems which helped to develop the models for the real and simulated environments, and Travis DeVault for his advice on Unity environments that were used to optimized the robot behaviors in the simulated environment.

Dedication

I would like to thank my wife, Veronica Fernandez, for her emotional support and company during the time I was pursuing my Ph.D. at University of Idaho. Also, I would like to thank my parents for always believing in my goals.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In research and the industry, there is a growing interest in groups of cooperating autonomous robots. In AI and Robotics research, this is part of a major field of study called social robotics. Research in this field of study shows that a multiple agent system must work under one main principle: individual decision making based on shared information generates an emergent group/cooperative behavior that can solve complex problems that a single agent would not be able to solve [2, 3]. Swarm-inspired algorithms are very popular techniques that are used to generate cooperative behaviors. They are normally based on large numbers of agents with simple rules and with the ability to share local information that can be propagated through the entire population depending when necessary. Using task allocation methods imitating the foraging in swarms for decentralized control problems is a good example of how an emergent cooperative behavior can solve a real-time complex problem just by agents sharing local information [4, 5]. This implies that information and information sharing plays a major role in the success of multi-agent systems. Regular communication between the agents in this type of system allows each agent to have a better understanding of its surroundings and better coordination [6]. Based on the group behavior, cooperation between the agents can be established and this generates an implicit coordination between the group members. But this coordination is very sensitive to changes in the message frequency when communicating and examples like the one described in [7] shows how in the same environment poor communication can negatively affect the coordination of a group of agents.

One approach to address the problem of coordinating multiple autonomous vehicles in complex, unstructured environments, especially when communication is noisy, is anticipation. In most cases, anticipation is considered as a behavior that emerges from autonomous organisms/systems because they inherently interact and adapt to

their environment. According to the authors in [8], autonomous systems use the principle of anticipation to generate anticipation behavior when trying to interact with a dynamic environment while being aware that time is also a major factor for every decision that is taken. An organism that is self-aware of its environment is able to respond faster and more efficiently than another organism that is not aware of its environment [9]. This can generate systems that are self-directed through evaluating their interaction flow and modifying their interaction process to achieve their goals [8]. Organisms that use these abilities properly can develop behaviors that can be defined as an anticipation behavior; where the organism creates a prediction of an upcoming event in the environment - i.e. they anticipate the event. Now, we can try to use this idea of anticipation with computers/robots. We understand that machines are able to process lots of information much faster than humans. If a robot can build a representation of its surroundings accurately, a task like navigation can be fulfilled by combining self-awareness and anticipation successfully [10]. This means that robots capable of anticipation can be used as an aid for humans on tasks that require synchronized decisions. The goal is to make the robots that are more aware of future events further in advance and with more accuracy in order to give robots better resolution for making decisions.

The previous stage of this research introduced an anticipation model that could support an indoor navigation system for two ground robots, a leader and a follower, that coordinate their movements by sharing information of the current progress of the leader. Both robots were supposed to look for the same goal, defined by two colored balls placed over a finish line but apart from each other, and reach it at the same time. The anticipation module in the follower robot was used to detect missing messages from the leader and replace it with a new message that represents an inference of how the leader was preforming previously - i.e. an anticipated message. In this research, this idea is expanded into an outdoor environment where more variables and noisy

signals need to be considered and to systems with more than two robots.

The overall goal of this research is to study the application of anticipation behaviors using Artificial Intelligence on Autonomous Vehicles by expanding the idea that was originally presented in the 2013 thesis for my M.S. degree "Anticipation Strategies based on Artificial Intelligence Techniques for Communication between Two Groups of Agents". This research will introduce more complex and larger environments, the use of more instrumentation to acquire more information from the environment, and the use of a larger fleet of robots that require larger and more complex group formations. The goal of the research is to test whether the use of anticipation can aid each robot's ability to coordinate their movement with the other robots as a group; specifically their ability to maintain a group formation.

A second portion of the research project aims to expand the research on robot-to-robot interaction to include human-to-robot interaction. This portion of the project focuses on the idea of using AI to restore corrupted messages during human to robot communication instead of dropping the message completely. This is a complementary tool to anticipation in social robots. The strategy for this part of the project is to use Fuzzy Logic Theory as an approach to Natural Language Processing (NPL) in a human-robot interaction with multiple robots. Here, a robot must identify and validate voice messages from a person and decide if the message is intended to be received by it and what command the human speaker intended it to perform based on the content of the message. The challenge includes making the robot not only identify the correct message format that represent a valid command but also, recognize and correct cases with homophone words that can be misinterpreted by the underlying speech engine.

The study and use of autonomous robots is growing rapidly and they are becoming an important tool to aid people with many tasks. Autonomous robots are a major focus of artificial intelligence research. Large companies and research groups

are currently working in this area to develop autonomous robots for ground, water, air and space. Companies like Google, SpaceX, Boston Dynamics, Amazon, among others have dedicated research groups that are focused on creating the next generation of autonomous vehicles [11, 12]. A major problem for autonomous robots is the difficulties that they face when used in complex, outdoor environments. To solve this problem, it is normally required to design robust systems that make robots more aware of their surroundings.

Today, autonomous robots are being tested in many different outdoor environments. This includes self-driving cars running through the regular traffic in some cities around the world [13], autonomous robots making decisions while driving on the surface of Mars [14], and underwater vehicles navigating using AI while using sensors to scan the environment [15]. Companies like Waymo (a Google subsidiary) and Zoox are currently running tests in areas with regular traffic using their own autonomous vehicle models. In general, these environments involve difficult challenges that the vehicle must identify and react/solve on the run because a bad decision can generate a terrible outcome. Artificial Intelligence (AI) has become a common tool in the market because of its learning flexibility to identify similar scenarios and most of the companies pursuing research in this area have their own AI team to address these issues [16].

Outdoor environments are particularly challenging because they add a level of uncertainty and random noise that can easily cause a failure or an accident on the robot. A robust sensor systems inside the robot can help it avoid external errors while it continues running its process unaffected. To develop an autonomous robot, its sensors must retrieve accurate information from the environment which can be used by the robot effectively to make decisions. In some cases, sensors loose accuracy due to specific external conditions, like GPS sensors trying to acquire data while driving between buildings. Normally, additional processing is added to support the

system by adjusting the acquired data to normal values while the sensors return to the operational range. Sometimes, filtering the data is sufficient; other times, it is necessary to implement more complex tools, like Kalman filters [17].

In this project, we implemented an algorithm to aide the GPS sensor of the ground robots. This is an error correction algorithm that keeps track of the previous and new data and the speed of the robot to calculate the next location. This model implements the idea of detecting when the new acquired location is further than expected and calculate a new position in the same direction based on the actual speed. This method was not necessary nor implemented in the simulated environment because the GPS sensor was simulated without the real noise.

This dissertation is structured as follows: Chapter 2 presents an overview of the current state of the art of anticipation research and related areas that are relevant to this project. Chapter 3 presents the research that was done on anticipation strategies in a large environment following a trail of multiple waypoints while using multiple navigation controllers and multiple sensors. Chapter 4 introduces a complementary approach for anticipation using Fuzzy Logic Theory where corrupted messages (audio messages, in this case) are analyzed based on their syntax in an attempt to fix the error. In this part of the research, we consider that, for a group of robots receiving voice commands, not all messages get lost but some of them get corrupted. We use our approach recover the original message from these corrupted messages instead of dropping it. In Chapter 5, the anticipation strategy presented in Chapter 3 is expanded to a larger group of robots to analyze how anticipation behaves in pairs of robots versus larger groups of robots where the chances of collision inside the group increases. Formation control is not included as part of this research because our goal is to analyze anticipation as a way to keep a group of robots together and check for emergent behaviors in their formation. Finally, Chapter 6 summarizes the conclusions of this work based on the results that we presented in previous chapters.

Chapter 2: Background

This project is a continuation of research that started with my Masters thesis at University of Idaho in 2012 [1]. It has been funded mainly funded by Office of Naval Research as part of a larger project that involves the research and development of Autonomous Underwater Vehicles (AUV) for multiple tasks including mine-countermeasure (MCM) and magnetic signatures assessment (MSA) missions [6, 7, 15, 18, 19, 20, 21, 22, 23]. These missions are run by a fleet of two to five AUVs that must maintain fairly constant communication in order to keep precise group coordination. A major problem in these missions is the low bandwidth and potentially unreliable nature of underwater communications. Standard underwater acoustic communications work on a low frequency bandwidth with an operational range between 10Hz and 1kHz [24, 25]. Frequencies outside this range are not commonly used because they become even less reliable. The overall AUV project at University of Idaho is being developed by an interdisciplinary group that involves teams from Electrical Engineering, Chemical Engineering, Mechanical Engineering and the Laboratory for Artificial Intelligence and Robotics (LAIR) from Computer Science.

The results of my Masters thesis demonstrated some concepts that this dissertation will expand upon and analyze more in depth [1, 26]. The previous research focused on the structure of trained anticipation modules and evaluating which training configuration creates the best results. In particular, the previous work determined which configurations are optimal to train a Multi-Layer Neural Network to obtain accurate anticipation behaviors under different noisy conditions. The configuration included the number of historical messages that are used as inputs to the Neural Network to accurately infer missing messages. The results of this research concluded that, for this type of system, the best results were obtain by an anticipation module that just kept track of the last two historical messages. Keeping more messages in the history list added more error to AUV behavior. In addition, we concluded that

the neural network models with best results had a configuration with 10 neurons in the hidden layer. We used this principle as a base for the research.

As a strategy, anticipation has been used for other types of problems such as imitating the natural language of humans to correct acoustic messages between Unmanned Underwater Vehicles (UUV) [27]. This approach uses previous, but not necessarily full, knowledge about the desired behavior of the robot to increase anticipation performance. The anticipation approach that was used in [27] is based on a linguistic logic that analyses the structure of a binary string using syntactic, semantic, pragmatical and behavioral logic, which is very different from the approach used in this research.

Other studies have focused on trying to understand human anticipation as humans respond to familiar events under different conditions. For example, the authors in [28] analyzed human body motion and reaction while subjects tried to navigate in a virtual environment. Their idea was to compare this behavior with human behavior in a real environment and analyze if humans can use inherent knowledge to adapt themselves in new environment with similar conditions (Real vs. Virtual environments). Other research in human behavior has focused on evaluating anticipation based on daily routines and its role in causing stress before specific stressful events occur [29]. The authors in [29] ran a series of tests and surveys on adults across different ages and showed that, when a stressful event was coming up, the subjects suffered large spikes of stress at the start of the day when the stressful event was to occur, but not during the preceding night. The goal of the study was to analyze the anticipation in humans from a cognitive perspective.

Other studies in human behavior have focused on the use of inference, a form of anticipation, for speech comprehension. For example, the authors in [30] ran tests on hearing-impaired subjects to analyze how difficult it was for them to process inferential information. In this paper, inferential information is defined as an input

that requires additional reasoning before understanding the information that it's being transmitted. For some of the subjects, it was more difficult to process inferential information than literal information. But the results also showed that it was harder for the subjects to identify predictive inferences than associative inferences. Even though this research focused on ways that devices or methods could improve inference for speech comprehension in hearing-impaired people, the authors also described how humans use inference based on speech syntax.

Research in Robotics has also become the focus of studies on anticipation. One example [31] in which the authors worked to make a robot aid pedestrians by trying to anticipate the next position of the pedestrian and trying to meet them at an estimated point along their path. This anticipation strategy used data from a camera and a set of motion sensors to detect a person in the environment. Another approach was presented in [32] where an anticipation method based on image processing was implemented to detect the next possible move that a human was likely to do. Using a set of images, the robot could identify a particular action from the test subject by recognizing the movement of their body and other objects in the environment and then, by analyzing the sequence, the robot was able to predict the next action of the subject. In both cases [31, 32], the authors used image processing as a part of the data for the anticipation strategy and both require a predefined model of likely behaviors to generate an anticipated behavior. These two examples also show a common behavior for anticipation in living organisms which can be summarized as identifying the target and analyzing its behavior before deciding what action to take. In contrast, the authors in [33] presented another approach for anticipation in autonomous robots operating in environments with a lot of people. Their goal was to have the robots anticipate behavior of the people inside this environment based on their routines. Their approach focused on training the robots using a statistical analysis that creates clusters of people over a generated spatio-temporal model. According to the authors,

this gives them the option to focus their tests during regular active hours and train the robots with the new data overnight in order to update their model.

Also, the authors in [34] presented an introduction to anticipation focused on robotic systems. This article highlights the importance of the implementation of a telemetry system that starts by picking and installing the right sensors to design an accurate controller that guides the robot to fulfill a determined task. The model for anticipation is based on a state-space model that picks a particular state based on the information from the sensors. The model requires the user to identify specific behaviors to create states that will define the robot behavior. This approach helped the robot to behave accurately while running under optimal conditions, but its behavior would be weak if it ends up working in the grey areas between states. This means that the robot can fail or be limited on perceiving its environment and/or it can generate inaccurate decisions. The anticipation approaches used in this research were based on Neural Networks, which are known to help bridge grey areas if the right data is used during training. Anticipation helped in some specific cases but these grey areas were difficult to solve.

When working with multiple robots as a collective group, the design of the entire group can be defined as homogeneous or heterogeneous based on the hardware components of each robot [35]. Social robots are defined as homogeneous when the robots are built with the same components and they are capable of solving the same task. This type of group is typical in exploration and navigation problems where the robots have to deal with a large scale environment and it would take a lot of time for one robot to complete the task. A heterogeneous group refers to a robots that are built with different components or capabilities and/or they are programmed to focus on separate simple tasks that are part of the group's overall mission. This difference is explored and developed in [35] where flying drones are combined with ground robots to solve a mission that requires the group to follow a specific sequence of actions.

This research also makes it clear that the size of the group can affect the decision of the type of components that should be used. Based on the results described in [36], the authors propose to use simple and cheap robots with few sensors when it is necessary to run a large number of them. But they preferred to use a smaller group when it was required to use more complex and expensive sensors.

Based on the review of the previous literature, and especially the results from my thesis research, five anticipation models are implemented in navigation problem of this dissertation. One of them is based on Fuzzy Logic and the other 4 are Neural Network models. Two of the models were trained using Backpropagation and the other two were trained using a Genetic Algorithm. In addition, the neural network models were trained using two different structures that are based on the ARX and ARMAX models for system identification [37]. These models are explained in detail in Chapter 3 and the results of the first set of experiments with these models are presented in that chapter. The models are tested again in Chapter 5 but under more complex conditions and with larger groups of robots. Chapter 3 also presents a detailed description of how the entire system was designed; starting from the implementation of the sensors and the design of a combined multi-controller system for navigation in large areas. Chapter 4 introduces an inference system that corrects corrupted audio messages by evaluating the sound proximity of the input words with validation words. This last phase of the project is intended to be a complement of the anticipation strategies that considers missing messages and corrupted messages as dropped messages whiles the inference system will focus on recover the corrupted messages.

Chapter 3: Anticipation Strategy for Two Ground Robots in an Outdoor Environment

3.1 Introduction

This chapter continues the research that was started in my Master thesis "Anticipation Strategies based on Artificial Intelligence Techniques for Communication between Two Groups of Agents" in the chapter "Anticipation Strategy for Speed Control in a Commodity Off the Shelf (COTS) Robot Fleet". This research extends that research by focusing on the implementation of anticipation in additional environments and testing the performance of anticipation strategies in real-world environments where external noise and environmental complexity can affect the system. My Masters study demonstrated how anticipation successfully aided two robots in coordinating their behavior while having faulty communication. When using anticipation they were able to drive in parallel to reach the goal at the same time in a small indoor arena (3-meter distance).

In that research one of the robots, the leader, sent messages to the second robot, the follower, every 250 milliseconds. The message describes the current status of the leader by using a fuzzyfied value representing the leader's progress towards reaching the goal at a specific time. From this data, the follower can determine if the leader is ahead, behind, or on schedule to reach the goal and adapt its current speed to try to match the behavior of the leader in order to reach the goal at the same time as the leader. The frequency of the message reception was decreased to simulate the loss of messages during a mission. When a message is lost, the follower tries to infer the possible current behavior of the leader by looking at the immediate history of received messages with a fuzzy logic controller to create an anticipated message that fills the role of the missing message. This work showed that the follower could infer

the behavior of the leader and adapt its current behavior to reach the goal at the same time as the leader. When the leader is behind schedule, the follower is able to reduce its speed enough that the leader can catch up. In a similar way, if the leader robot is ahead of schedule, the follower increases its speed and catches up the leader. Then, the follower can adapt to the current speed of the leader to drive in parallel and reach the goal at the same time.

While the previous research demonstrated that anticipation was successful at generating accurate, anticipated messages that helped two robots to drive in parallel to each other, the experiments were limited to only two robots in a very small, very simple environment. To determine whether the previous successes with anticipation would apply to a more complex environment, this chapter tests a similar group of robots in a much larger environment on a more complex problem. For this new approach, it was necessary to consider and design a more robust navigation controller. The previous controller was based on an blob detection method that uses the camera on a phone placed on top of the robot. The main limitation is determined by the maximum effective distance between the target and the camera. The robot cannot use the blob detection farther than this distance because there are not enough pixels in the image to identify the object as a correct target. To solve this problem, this chapter present a multi-controller system that activates the correct controller after detecting that the robot has crossed some predefined thresholds, defined by the operational ranges of the sensors. Additionally, a total of 5 anticipation models are presented in this chapter in order to compare what type of anticipation structure and training produces the best results.

This chapter is organized as follows: Section 3.2 presents a brief state of the art that is relevant to this research. Section 3.3 presents the implementation of the entire robotic system as a group which includes the preliminary studies that were done on sensors and how they were used on the robots. Section 3.4 includes the

description of the anticipation modules and how they were designed and, trained. Section 3.5 presents the results for different missions using every combination of the robot behavior that were run to test all 5 anticipation modules and validate the accuracy. Finally, section 3.6 discusses the behaviors that were observed and the data that was analysed as part of the conclusion section.

3.2 Background

According to the author in [38], navigation in a complex outdoor environment requires the use of multiple sensors in order to guide a robot successfully. Thus, in moving from working with a simple indoor environment to a more complex, larger, outdoor environment, it is necessary to add additional sensors to the robot. The key for successful performance is the selection of proper sensors for specific tasks. For example, some sensors, like a GPS sensor, are very accurate while the robot runs outdoors, but once it drives inside a building it becomes useless because the satellite signal drops immediately. The authors also emphasizes that the environment can limit the performance of some sensors [38]. For example, a compass is very useful to help the robot with navigation but, in the presence of additional magnetic fields such as from an on-board electric motor, this sensor is no longer reliable. This means that the author also recommends evaluating both the sensors and the conditions of environment where the robot will be used to build a better and more robust fault-tolerant system.

In [39], the author states that the sensing technology for navigation has evolved greatly in the last decade. This paper recommends the implementation of remote sensing for large arenas or outdoor environment. Here, remote sensing is defined as "any non-contact technique whereby the object space can be observed". New and improved technologies have given easier access to methods for navigation that were very expensive in the past. For example, the use of satellite GPS has increased

because there are a large number of satellites flying around Earth that grant low cost, public access for location services.

The authors in [40] emphasize that robot navigation must always be used along with some method for self-localization; otherwise the robot will not be able to effectively reach its target. The self-localization method can implement global information, like GPS information, or local/relative information, like landmarks. The authors present a system based on a Hidden Markov Model (HMM) to make the robot recover itself when it gets lost before continuing with its planned path. For this work, the robot was deployed in an indoor environment. For our work, it is important to consider a recovery method for the robot because it will run in a larger environment and there will be more space to move in and to get lost in.

One recent study presents a different approach to solve indoor robot navigation without making the robot build an internal map of its environment [41]. Here, the authors take an approach based on reinforcement learning to help a robot navigate in a room while learning the path through the room in the process. The authors claim that this method allows the robot to solve the problem of both navigation and obstacle avoidance at the same time. Also, the authors highlight that this strategy allows the robot to recover from a fail state or a location which the robot will need to avoid in the future. The approach presented in this research does not use an AI-method like this one as a navigation strategy, but other methods are used to help the robot with the GPS sensors and to create an adequate estimation of the path that is being followed.

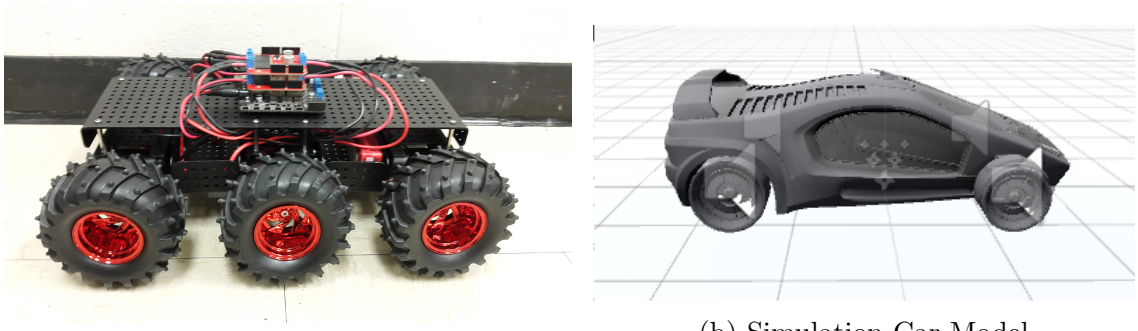
In addition, some work in robotics can be linked to anticipation. For example, the authors in [42] presented a study using anticipation for collision avoidance. The model is based on a fusion between a sensor system capturing images with a deep neural network model that is trained using static and dynamic obstacles/samples for obstacle avoidance. The authors compared the results of training the neural network

with two different sets of data: one set with all the static and dynamic samples combined and separate sets with no combined data. The results showed that it was more effective to train the neural network using separate sets than using a combined set and that the robot was able to identify the dynamic samples, which were the most challenging tests.

The approach that is presented in this chapter uses a simulated environment in Unity that allows the implementation of a multi-sensor and multi-controller robot system with large testing arenas. To implement this simulated model properly, initial tests were run using real robots with the selected sensors. The r data from these tests were used to build an equivalent model in Unity. A side-by side comparison is presented to explain the physics that needed to be considered in Unity to obtain a similar environment with similar behaviors. In addition, a system with two robots with their navigation system is described in detail including the communication system that is used to keep the robots synchronized. Finally, we present and test five different anticipation models whose main task is to aid communication with anticipated messages when true messages are lost or not received on time.

3.3 Experimental Platform

For this part of the research, we are interested in determining whether anticipation is still effective in a more complex environment. The testing environment will follow a similar idea from the indoor robot experiments. There will be two robots, one leader and one follower, that will navigate between waypoints to a final location while sharing information to synchronize the behavior of the robots. The arena will include more than one waypoints which will create a path that defines a mission for the robots. Just like the goal mark in the previous experiment, the waypoints will be represented by visible objects (landmarks) that the robot can identify. One of the big differences with these new experiments is that the waypoints will placed too far



(a) Ground Robot Model

(b) Simulation Car Model

Figure 3.1: **(a)**: Robotic car that was build for outdoor testing. **(b)**: Simulation car built in a Unity environment.

apart from each other and the camera controller, that has been used before, would find difficult to identify the next target. This will require to redesign and introduce an improved version of the navigation controller.

Unity was selected to model the robots and the testing arena. But we also run some tests with real robots and real sensors in order to understand the way they operate and how we have to deal with the raw data before it used by the robot properly. The simulation model will try to follow the physics features of the real model as closely as possible in order to create a navigation model that can be easily moved between the simulated and the real environment and to make it easy to design the controllers. In addition, Unity allows to change the shape and type of the terrain just in case we consider on building additional cases. Figure 3.1 shows the models of cars (real and simulated) that will be used for both the real and simulation tests. In a simulation environment, it is possible to avoid (or increase) the added noise that is introduced by sensors in a real environment. Running the models in both environments will allow to improve the navigation controllers on both models and to prove that the anticipation model can be easily transferred between systems.

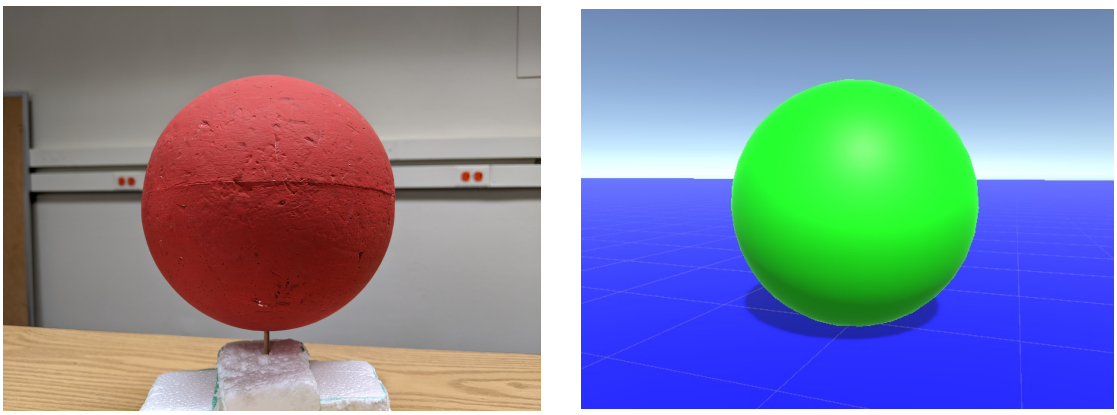
Based on the detailed description of sensors for mobile robots that is presented in [43], robot sensors can be divided in 2 categories: relative position measurements and absolute position measurements. Specific sensors can be categorized as follows:

- Relative position measurements: odometry, inertial navigation
- Absolute position measurements: magnetic compasses, active beacons, global positioning systems, landmark navigation, model matching

3.3.1 Short-distance controller

This research will implement sensors related to the second category; specifically focusing on landmark navigation, magnetic compasses, and global positioning systems. The landmark navigation method is represented by the camera and this type of sensor was tested in the previous stage of this research project. For those indoor arena experiments, the robots used a camera and an object detection routine to recognize the goal landmarks (green balls specifically). The landmark navigation category defines these objects as artificial landmarks because they are man-made objects that can be placed in an environment with the sole purpose of navigation and they are designed to optimize the contrast for the visual system of the robot. Generally, the size and shape of these objects are known in advance and they are used to make precise calculations regarding position during tests. The downside of this type of sensor is that its effectiveness is highly dependable on distance and ambient conditions, such as lighting. The author recommends using this category of sensor for navigating over short distances [43].

This camera controller was introduced for the experiments during my Masters thesis initially. The design is built using the OpenCV framework to capture the images directly from the camera and convert them into an object containing the RGB information of all pixels. The controller implements a blob detection method which basically check for a desired color in the image and, once it is identified in the image, it tracks for a similar shade of color in the neighboring pixels until it fully detects the circular shape of the ball. This algorithm is based on the blob detection



(a) Real Environment Target (b) simulation Environment Target

Figure 3.2: Navigation waypoints that were created for real and simulation environments

function that is part of OpenCV but, at the LAIR research group, we created our own version of this algorithm in order to extract more information of the image. In this case, the algorithm also finds the longest line in the blob to determine its diameter; assuming that the targets are defined by colored balls. Then, it calculates the number of pixels of this line and this value is used to estimate the distance to the ball. This method was tested in both simulation and real environments with accurate results using targets with known sizes (approximately 14.7cm or 5.8" in diameter) (Figure 3.2). And after running some tests placing this ball at different distances, the linear equation that converts the radius of the ball in pixels to distance is:

$$distance = \frac{202.47}{\text{diameter}^{0.844}} \tag{3.1}$$

After some preliminary tests, it was established that these marks can be recognized by the camera within a 5-meter range accurately.

The blob detection algorithm also runs an additional analysis where it tries to identify the location of the detected target in the image. This is done by dividing the image in 3 vertical sections and calculating which section contains the most amount of pixels of the target. this can be translated later in the heading of the robot (left, right

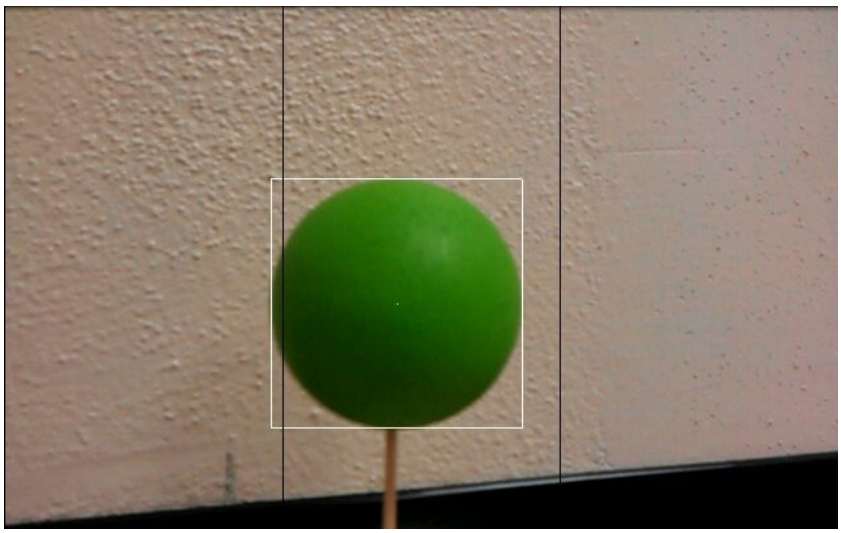


Figure 3.3: Screen division that the camera controller uses with the blob detection algorithm to generate the heading of the robot

or forward) (Figure 3.3). With the heading and the distance values being returned by the blob detection algorithm, the robot has the necessary information to focus on the target and drive towards while correcting its path.

3.3.2 Long-distance controller

According to the authors in [43], the magnetic compass is recommended solution for heading control for the navigation of autonomous vehicles. The only disadvantage is that the signal of the compass can be distorted by power lines or steel structures. For this reason, the authors recommends using this sensor for outdoor navigation tasks only. Global positioning systems are also recommended for outdoor navigation. Under optimal conditions, this system returns x-, y- and z- coordinates which allows this system to be used almost anywhere on the planet. The disadvantage of these systems are interruptions of the GPS signal because of tall or high-level object blockage, like buildings, high hills and thick foliage.

To solve the limitation problem of the viewing range of the camera, the robots need to have access to additional sensors that allow them to be aware of points of interest

in a far distance. A GPS sensor and a programmable compass are used as inputs to the robot system in order to improve the navigation in an outdoor environment. A digital compass can be created by combining the data from a 3-axis accelerometer and a 3-axis magnetometer. Using the GPS and the compass combined compensates the limitation of the navigation controller that is used with the camera for larger distances.

During this part of research, it was found that raw GPS data contains a lot of noise that, at some point, it can spike up to a 15-meter error using incoming satellite data and up to 200-meter error using cellphone tower data. Thus, implementing GPS data requires additional processing methods to not just ignore these spikes but to smooth the GPS signal based on the actual movement of the robot. Figure 3.4a shows GPS data without any post-processing and filtering method. Here, the robot receives bad data which makes its behavior erratic. A filtering and navigation correction module was added to try to reduce these big jumps and to smooth the continuity in the GPS data. In general, this module analyzes new GPS data by calculating how far this new location should be from its previous one based on the current robot speed. If the new GPS data is inside this inferred area, it is considered as accurate data. Otherwise, the module will calculate an inferred new position in between the new and previous locations. Figure 3.4b shows the result of using this new approach.

However, using GPS for location and navigation purposes involves additional calculations that must help dealing with GPS data in two specific situations: driving across long distances and running tests in different locations across the planet. This method is based on the WGS-84 model which consider earth as an ellipsoid [44]. Basically, it defines that the radius of the earth is not the same for every location on earth and that the altitude of the current location needs to be consider as part of the radius. This method returns a more accurate result while estimating a location on terms of latitude and longitude values. This method is used as part of the current navigation

(a) Raw GPS data (b) corrected GPS data

Figure 3.4: **(a)**: This figure was plotted directly from the GPS data without activating any filter or navigation estimation process. **(b)**: This data shows GPS samples while implementing a distance estimation process to improve the navigation of the robot.

system that was coded for the AUVs and it will give a more general behavior to the robot making the experiments more hardware-independent and location-unbound.

The simulation environment can easily simulate a GPS sensor by using its global mesh as a location tracker for every robot moving on the surface that is generated. Also, the 3D vectorial libraries that come with Unity allows to considered the changes in the Z-axis which can be use to simulate the effect of changes in elevation. These libraries can also be used to calculate the estimated distance between two points in a 3D environment which will fill in for the WGS-84 model that was used in the real robots.

The implementation of the programmable compass normally requires of an initial configuration that involves detecting the current location of the robot. This is necessary to calculate the offset that the robot needs to use to compensate for the detection of the magnetic north according to the robot's current location. For example, for Moscow ID, the magnetic north has an offset of approximately 11 degrees from the geographic north. It is important to add this compensation in this sensor before using this data alongside the GPS data to have both sensors use the geographical north as reference.

On the other hand, the programmable compass generates high-frequency noise mainly due to both the accelerometer. To reduce this noise, the signal of each axis on each sensor must pass through a low-pass filter. This filter is defined by the equation:

$$Y_t = Y_{t-1} + ALPHA * (X_i - Y_{i-1}) \tag{3.2}$$

The low-pass filter helps to smooth the signal by adjusting the current sample based on the error with the previously adjusted sample. A constant ($ALPHA$) is needed to adjust the spacing between samples. This constant is very important and must be implemented properly. $ALPHA$ can be a value between 0 and 1. If $ALPHA$ is 0, the filter will not allow data to pass. If $ALPHA$ is 1, the filter will let everything pass. For this problem, six $ALPHAs$ must be adjusted to improve the signal of each axis of the accelerometer and the magnetometer. Based on some field experiments, a value of 0.8 was set for every $ALPHA$.

The simulation environment allows to imitate a compass in a much simple way. In contrast to the programmable compass with the real robot, there is no need to use accelerometer sensors in the simulation which are the main cause of noise generation in the compass. For the simulation, we simulate a vector with a very large magnitude that represents the reference for the geographic north. In addition, every robot generates a vector that always point to the direction that they are facing. By combining these two vectors and calculating their difference, we can generate a similar behavior that is seen in a compass.

3.4 Methods

For these experiments, the navigation routine of the robots was designed based on the structure that was introduced in my Masters thesis. The navigation model runs two controllers that are used in all robots (Message Generator, Speed Increment Cal-

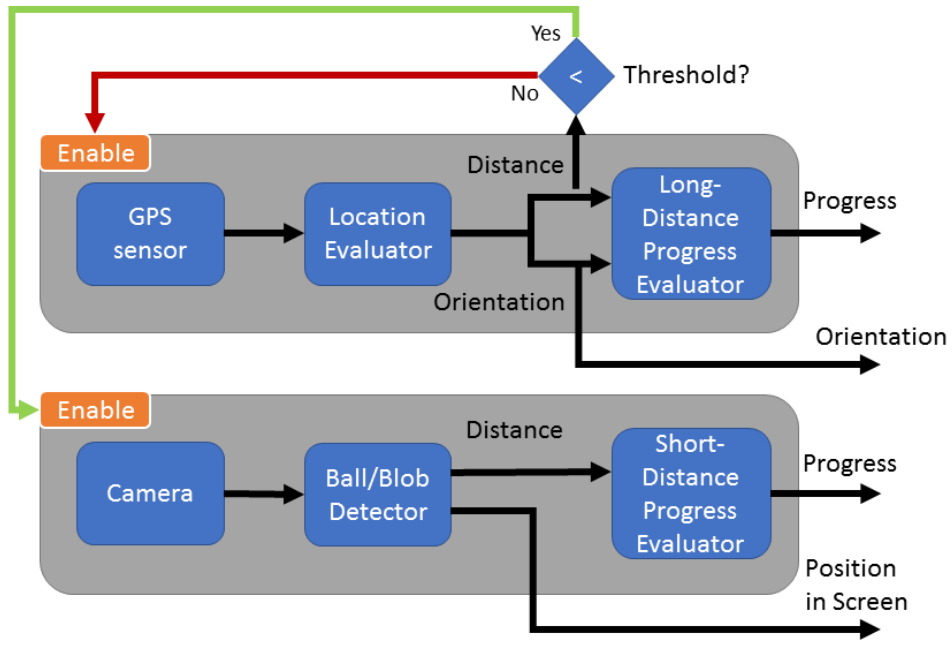


Figure 3.5: Block diagram to control the different sensors for navigation

culator) and one more controller that is run only by the Follower Robot (Anticipated Message Estimator). The Message Generator and the Anticipated Message Estimator modules use Fuzzy Logic controllers to generate their respective outputs. For the new approach in this research, we configured a Message Generator module at the end of each sensor controller, the Long-Distance Progress Evaluator and Short-Distance Progress Evaluator, to keep the signal of these two controllers tied to their own individual distance ranges in which they operate (Figure 3.5). But, in the end, both modules generate a message that represent the status of the robot during the mission.

The goal of the navigation system as a whole is to generate an iterative process that allows the robots to switch between navigation controllers based on the remaining distance to each target. This way, the robot can switch from GPS based control to finer and more accurate distance samples with the camera when it gets closer to the target/waypoint and later switch back to GPS based control when it needs to move away from the current waypoint and begin moving to the next waypoint. Figure 3.5 shows how it is possible to switch between navigation controllers by keeping track of

Figure 3.6: Block diagram with variables that are needed control the motors on the robot

the estimated remaining distance. It also describes the flow of data from the sensors to the outputs of the navigation controller that are used to drive the robot.

The Speed Increment Calculator module was also redesigned for this research and it is referred as the Navigation Controller (Figure 3.6). For the experiments that were presented in my Masters thesis, this module was in charge of the speed of the robots only because the first missions were configured for a straight line. This new version returns both the speed and the heading to make this controller more suitable for missions with more freedom in a 3D environment.

3.4.1 Message Generator Module

First, the message generator module calculates the input to the Fuzzy Logic controller by using the current speed of the robot and its current distance to next target and the distance it has left to the next waypoints after that. Using its current speed, the robot calculates an estimated time to the final waypoint.

The second step is to calculate the progress time. This variable represents, in milliseconds, how much time the robot is off from its schedule for reaching the goal. It uses a variable defined as *estTotalTime*, which is a constant calculated at the beginning of the mission that represents the time the robot will take to reach the final waypoint at its regular or "on schedule" speed. An additional variable is *elapsedTime*, which is a timer that starts at the beginning of the mission. The final equation is:

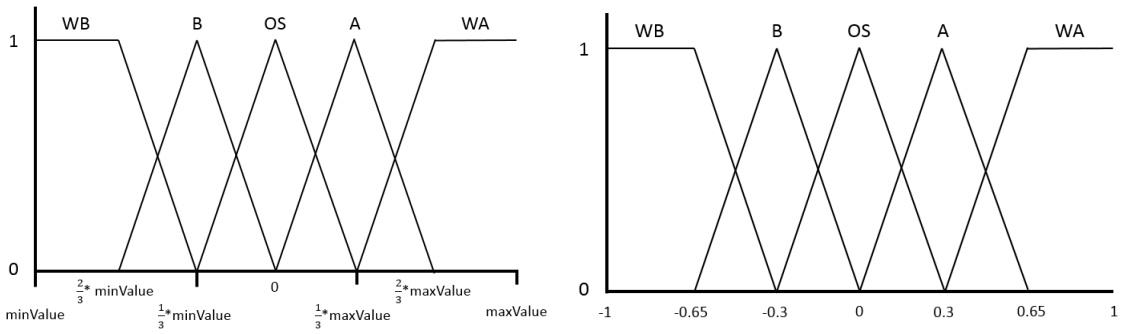
$$progressTime = estTotalTime - elapsedTime - remainingTime \tag{3.3}$$

Finally, *progressTime* is used as the input to the Fuzzy Logic controller which uses the following fuzzy sets: WayBehind (WB), Behind (B), OnSchedule (OS), Ahead (A) and WayAhead (WA). The Fuzzy logic controller uses these sets to generate a message consisting of five fuzzy values representing how far ahead of, behind, or on schedule the robot is. However, these new experiments do not have a fixed arena with one waypoint like the indoor experiments in my Masters thesis. This environment can be defined by multiple waypoint that can change in numbers and in distance between each other depending on the selected mission. Having a non-static arena changes the way the fuzzy memberships must be defined for the Message Generator because these parameters are time-dependent. By running preliminary tests using the minimum and maximum speed of the Leader Robot and different distances between waypoints, we created a set of linear equations that defines the range for the input values for the Progress Time variable and this range was used to define all the memberships evenly at the start of the mission.

$$\begin{aligned} maxA &= -0.0038 * totalWaypoints + 0.2807 \\ maxB &= 0.1538 * totalWaypoints + 0.0841 \\ maxValue &= maxA * totalEstimatedTime + maxB \end{aligned} \tag{3.4}$$

$$\begin{aligned} minA &= 0.0237 * totalWaypoints + 2.305 \\ minB &= 0.2714 * totalWaypoints + 2.9499 \\ minValue &= -minA * totalEstimatedTime + minB \end{aligned} \tag{3.5}$$

The output of the Fuzzy Logic controller uses the same names (WayBehind (WB), Behind (B), OnSchedule (OS), Ahead (A) and WayAhead (WA)) for its fuzzy sets but the limit values are different from the input fuzzy sets and the sets are evenly distributed. Both input and output fuzzy set are shown in Figure 3.7. The Fuzzy Logic Controller evaluates the input value by using the rules described in 3.1 and its



(a) Progress Time Input

(b) Message Output

Figure 3.7: Fuzzy Sets that describes the values of the input and output of the Fuzzy Logic controller in the Message Generator module. The fuzzy sets are: Way Behind (WB), Behind (B), On Schedule (OS), Ahead (A), Way Ahead (WA)

Table 3.1: Fuzzy Rules for Message Generator

Scheduled Time				
WB	**B**	**OS**	**A**	**WA**
WB	B	ON	A	WA

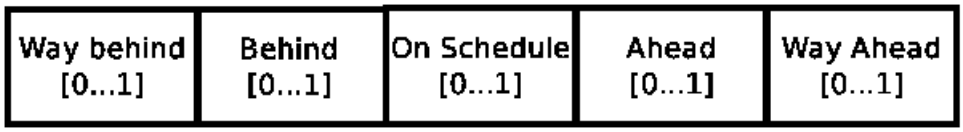


Figure 3.8: Message format consisting of five fuzzy values that represent the progress of the robot.

main objective is to set the progress messages inside a normalized region between -1 and 1 and to generate an equivalent distribution of the progress. The output of the Message Generator is returned as a set of five values generated by the Fuzzy controller and this set of five values is the progress message (Figure 3.8).

The experiments in this chapter were run in a simulated environment created in Unity that was created to include the important features of a real environment. For example, the physics that can affect a real robot, gravity, inertia, skid, collisions, etc, are included and can also affect the robots in the simulation. We expect that the leader messages will include more noise compared to the data that was used in the simulation presented in my Masters research work [1]. In our previous research,

Table 3.2: Example of a sequence of 10 Leader Messages taken from the On Schedule behavior of the Leader that was part of the simulation in the AUV research, the messages are identical because in this simplified simulation environment, with the On Schedule behavior there was no source of noise[1].

WB	**B**	**OS**	**A**	**WA**
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0
0	0	0.8666667	0.1333333	0

the AUV simulation offered a much more controlled environment which allowed the Leader to create and broadcast more stable messages (Table 3.2). This is, of course, a behavior that is not close to most real scenarios. Thus, for this part of the research because we implemented a more realistic model we expect messages with more variation (Table 3.3).

We hypothesize that even in this more realistic, complex environment the neural network responsible for generating anticipated messages (described below) can identify the implicit behavior of the Leader Robot while it speed changes during each mission allowing the follower robot to maintain progress. This is an additional challenge that is tested in this research.

3.4.2 Navigation Controller Module

This module uses a much simpler approach that was initially proposed for the AUV simulation in my Masters thesis. First, the controller calculates the difference between the Fuzzy sets of two progress messages. One progress message represents the reference message and the second message is the message obtained from the Message

Table 3.3: Example of a sequence of 10 Leader Messages taken from the On Schedule behavior of the Leader that is part of the simulation the current research in Unity. The variation in messages reflects the increased realism, and hence complexity, of the Unity environment.

WB	**B**	**OS**	**A**	**WA**
0	0.1630379	0.8369621	0	0
0	0.1643994	0.8356006	0	0
0	0.07823301	0.921767	0	0
0	0.04382417	0.9561758	0	0
0	0.0560173	0.9439827	0	0
0	0.06191415	0.9380859	0	0
0	0.1681354	0.8318645	0	0
0	0.1011745	0.8988255	0	0

Generator Module. If this module is being run by the Leader Robot, the reference message is obtained from the mission planner. If the module is being run by the follower Robot, this message is the progress message that is being broadcast by the Leader Robot. The adjustment value is calculated by the difference between the Fuzzy sets except the On Schedule set. Previously, it has already been tested that having a difference in the other sets means that the two robots are not synchronized. The result of the adjustment value is a decimal number in the range of [-1, 1]. The closer the result is to the limits of this range, the bigger the adjustment will be on the current speed (positive or negative). For example, if there is a positive difference in the Ahead set or Way Ahead set, this means that the follower is behind the leader (the leader is further ahead) and it should increase its speed. If the difference is positive in the Behind and Way Behind sets, the follower is ahead of the leader and it should decrease its speed.

$$\text{diff}_{\text{messages}} = \text{message}_{\text{Reference}} - \text{message}_{\text{MessGen}} \tag{3.6}$$

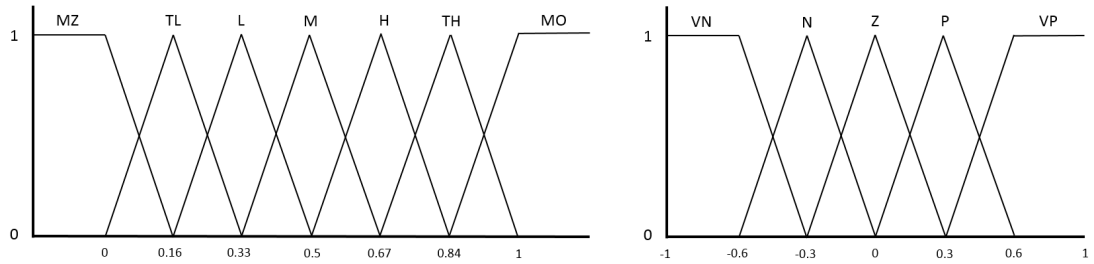
$$\begin{aligned} adjustValue+ &= (0.1 * \text{diff}_{\text{Ahead}}) + (0.3 * \text{diff}_{\text{WayAhead}}) \\ adjustValue- &= (0.1 * \text{diff}_{\text{Behind}}) + (0.3 * \text{diff}_{\text{WayBehind}}) \end{aligned} \tag{3.7}$$

3.4.3 Anticipated Message Estimator Module

This module runs only on the Follower Robot and its main goal is to activate when the robot detects that a message from the Leader Robot is missing; basically it acts as an aid to the communication between the Leader and the Follower robots. The main inputs of this module are defined by the last two messages from the Leader Robot that are constantly updated every time a new message is received. Old messages that do not fit in this list anymore are dropped. Basically, the list of messages works as a queue with a fixed size of two elements. This configuration has been already proven in the research for my Masters thesis and it is not necessary to use a list bigger than two previous messages. The anticipation module is supposed to return a message with the same format that the Message Generator module creates (Figure 3.7b). In total, five different models were created to be run as anticipation modules: one Fuzzy Logic model and four Neural Network models.

5 different anticipation models are used in the Anticipated Message Estimator Module during these experiments:

- One Fuzzy Logic model (FL)
- A Neural Network model with an ARX configuration trained with Backpropagation (A_NNBP)
- A Neural Network model with an ARX configuration trained with a Genetic Algorithm (A_NNGA)
- A Neural Network model with an ARXMAX configuration trained with Backpropagation (AM_NNBP)



(a) Fuzzy Membership Input (b) Fuzzy Difference Input

Figure 3.9: Fuzzy Sets that generates/anticipates the next Fuzzy membership based on the Fuzzy membership of the current message and the difference with the Fuzzy membership of the previous message

- A Neural Network model with an ARXMAX configuration trained with a Genetic Algorithm (AM_NNGA)

3.4.3.1 Fuzzy Logic

To generate an anticipated message, the Fuzzy Logic [45] module unwraps the content of the last two messages and evaluates the values of the one membership in both messages (Figure 3.8). Then, it builds the new message after comparing all five membership values in a progress message. The Fuzzy Logic module uses two input variables: a membership value of the last Leader message and the error on the same membership value for the last two messages. The memberships for the first inputs are displayed in Figure 3.9a and these memberships are defined as: Zero (MZ), Too Low (TL), Low (L), Medium (M), High (H), Too High (TH) and One (MO). For the second input, the memberships are displayed in Figure 3.9b and the memberships are defined as: Very Negative (VN), Negative (N), Zero (EZ), Positive (P), Very Positive (VP). In addition, the rules that are implemented for these inputs are defined in Table 3.4.

An additional small set of rules was added to the fuzzy logic module to allow it to create a crossed relationship between the values in a message. The additional fuzzy rules associate the anticipated value of a membership value m_i in the message

Table 3.4: General fuzzy rules for anticipation

Error \ **Message**	*MZ*	*TL*	*L*	*M*	*H*	*TH*	*MO*
VN	MZ	MZ	TL	TL	L	M	MO
N	MZ	TL	L	L	M	H	MO
EZ	MZ	TL	L	M	H	TH	MO
P	MZ	L	M	H	H	MO	MO
VP	MZ	M	H	TH	TH	MO	MO

with its neighboring membership values. These fuzzy rules are used to determine if a membership value in a message has a current value of 0 and if one of its neighbors (membership values m_{i-1} or m_{i+1}) are close to their medium value and increasing. If so, it can be anticipated that the membership value m_i will be about to change. If this condition happens, when the value of m_{i-1} or m_{i+1} gets closer to one, the value of m_i should have started changing already. These rules can be defined with the following equation:

$$\begin{aligned} \mathrm{TL}_{m_i} &= \mathrm{MZ}_{m_i} \&\& M_{m_{i+1}} \&\& (N_{Ei+1} || \mathrm{VN}_{Ei+1}) \\ \mathrm{TL}_{m_i} &= \mathrm{MZ}_{m_i} \&\& M_{m_{i-1}} \&\& (N_{Ei-1} || \mathrm{VN}_{Ei-1}) \end{aligned} \tag{3.8}$$

After evaluating all five membership values in the message with the Fuzzy Logic module, the outputs are put back together following the same format that was displayed in Figure 3.7b. When this model is picked as the active anticipation model, the result of this process is used as the output for the anticipation module which is used as a replacement for the missing message.

3.4.3.2 Neural Networks

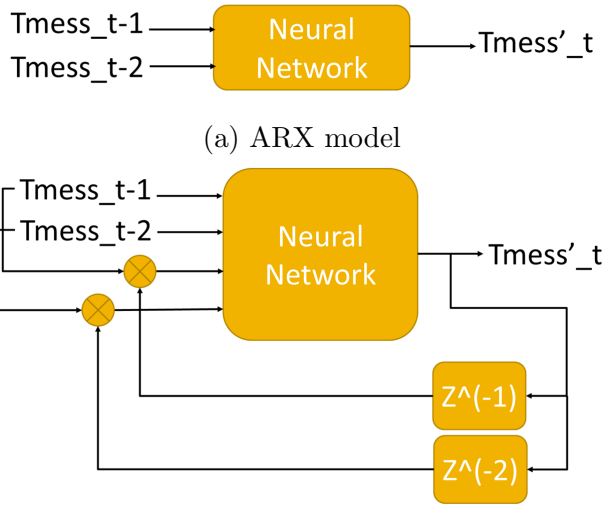
Based on the results that were obtained in my Masters thesis, which tested a number of different Neural Network structures, we picked the best general Neural Network structure for all models to use in this research. The Neural Network structure uses

Figure 3.10: Mission that was used to obtain the Leader messages for the training data using two waypoints. It includes a (simulated) distance of 60 meters to each waypoint and a right turn of 45 degrees

two previous messages as inputs and 10 neurons in the hidden layer. Building on this basic structure we defined four specific Neural Network models. For this part of the research, two of these models were trained using Backpropagation [45] and the other two models were trained using a Genetic algorithm [46]. The training data for the Neural Networks was based on the messages of the Leader Robot during a mission with two waypoints (Figure 3.10). During this mission, the Leader Robot ran multiple, separate test using 5 different speeds that produce high values in each of its five message variables representing the different behaviors: Way Behind Schedule, Behind Schedule, On Schedule, Ahead of Schedule and Way Ahead of Schedule. In total, the training data contained 805 samples from all of these five behaviors.

For Backpropagation training, this data was used by passing messages *n-1* and *n-2* as inputs while using message n as output. The Neural Network was trained to anticipate the next message received based on the previous two messages that were received.

For the training using a Genetic Algorithm, our approach was to evolve the weights in the neurons of the Neural Network. We used a standard configuration for the Genetic Algorithm: a tournament selection, single point crossover, and a mutation



(a) ARX model

(b) ARMAX Model

Figure 3.11: Input configurations for the anticipation models that were used on the Neural Networks. These model are based on the models for system identification

strategy where we picked a small group of neurons to add a small random value to their current value. The fitness function that we selected evaluates the Neural Network by passing samples in the training data in groups where the messages *n-1* and *n-2* are selected as inputs and the message n is selected as the output. The output of the Neural Network is compared with message n and we calculate the RMSE using all of the samples. This value becomes the score for the fitness function.

In addition, we created two additional configuration for the Neural Network models that are based on polynomial models defined by the theory of system identification models [37]. These two models are the *AutoRegressive with eXogeneous input* (ARX) and the *AutoRegressive Moving Average with eXogenous input* models (Figure 3.11). These Neural Network models have redesigned input configurations. Normally, when trying to identify a system based on the information of its input X and output Y, ARX and ARMAX use a time-series combination of these two set as its input to generate an estimated output Y'. For this research, our X and Y data come from the same source (the messages from the Leader) by trying to imitate a message in a time T while using messages in times T-1 and T-2. The ARX model is one of the

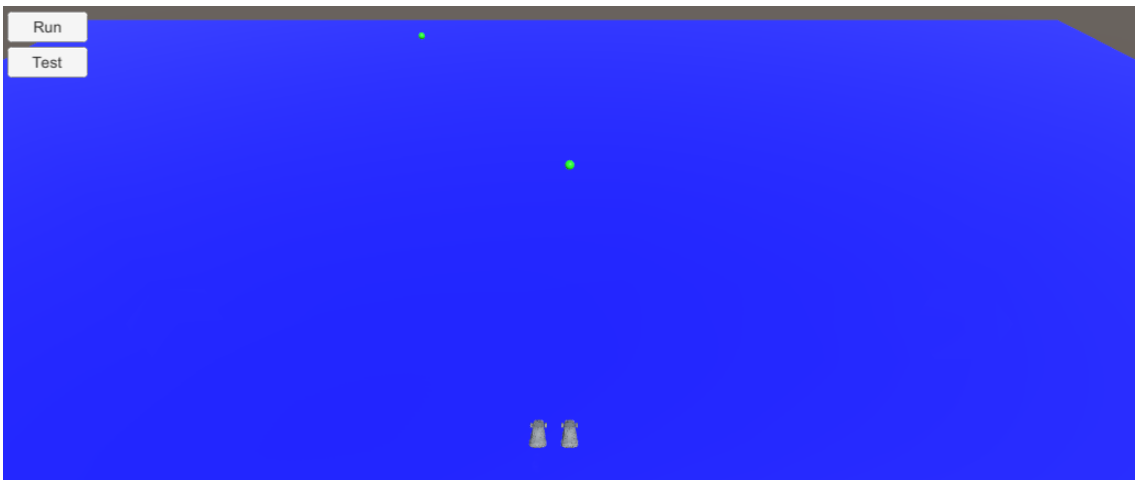


Figure 3.12: The mission that was used to test all the anticipation modules. It includes a distance of 50 meters to each waypoint and a left turn of 35 degrees

simplest polynomial models and it is how we can also defined the configuration that we have used in previous models. The ARMAX model includes feedback from the previous generated messages which are compared with original messages that have been received previously. By keeping track of this error, the ARMAX model may be able to reduce disturbances in the trained model [37, 47].

3.4.4 Experiments

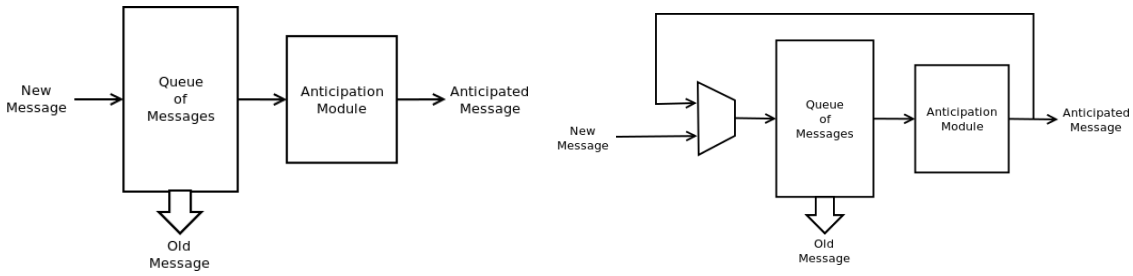
Figure 3.12 shows the starting position of the robots and the location of the waypoints that the robots need to reach. The objective of the robots is to drive in parallel through all waypoints. The Leader Robot broadcasts its progress to the Follower Robot to help the follower stay synchronized with the leader. Previously, for the indoor testing, the distance difference between the two robots to the waypoint when they reach the meeting point was used as a metric to measure the effectiveness of anticipation while running different robot behaviors. We will use the same metric for these experiments. This gives us a position to compare the results of these experiments with past results, as well. Similarly, in these experiments the Leader Robot will run different behaviors (slowing down, speeding up, etc.). These behaviors will

generate variations in the Leader's progress and therefore in the content of progress messages. These variations will create a response from the Follower Robot by trying to adjust its own speed by comparing its progress messages with the messages sent by the Leader robot. Under optimal conditions, the Follower Robot should react immediately to any changes in the Leader behavior. The distance difference at the end of the missions with optimal conditions will be used as a reference for statistical analysis with other result data that uses missing messages or anticipation messages. When missing messages are introduced in the simulation, the Follower Robot is in charged of detecting this event and run the selected anticipation model which will generate an progress message that will be used as a reference while filling in for the missing message. In this case, successful missions will be determined by the distance difference that the robots will have when they reach each waypoint and the error of this distance when it is compared with the distance difference under optimal conditions. These results can give a good description if the Follower Robot was able to anticipate/predict with its generated anticipation messages how the Leader Robot was behaving and stay synchronized with the Leader Robot during the mission.

Three testing missions were selected for these experiments: one mission with the Leader Robot driving On Schedule (LSpeed=5.2) was used as a default mission, one mission using a slower (LSpeed=4.25), and one mission with a faster (LSpeed=5.95) speed. These two values were selected because they are not included in the training data and it is important to test how the Anticipated Message Estimator Module behaves in scenarios that it was not trained on.

For these experiments, we picked three different message gap frequencies that the Leader Robot can use to simulate the loss of messages while broadcasting its progress messages: one message every time step (1S/M), one message every two time steps (2S/M), and one message every six time steps (6S/M).

Additionally, the Follower Robot can also use two different strategies for the An-



(a) Message queue without Feedback

(b) Message queue with Feedback

Figure 3.13: The Follower Robot saves the previous two Leader progress messages as inputs to the anticipation module. This list of messages is used as a queue and every time a new Leader message is received, the oldest message is dropped and the new message is inserted. We can simulate a case where the list of messages is updated only with Leader messages and another case where the list can be updated with the messages generated from the Anticipated Message Estimator Module.[1]

ticipated Message Estimator Module that are related on how the message inputs are saved and organized before being used:

- Without Feedback: only the original messages from the Leader Robot are used as inputs to the Anticipated Message Estimator Module. (Figure 3.13a)
- With Feedback: the messages generated by the Anticipated Message Estimator Module are 'recycled' and used as input messages when the Follower Robot does not receive a message from the Leader Robot. (Figure 3.13b)

We have show in previous work [1], that an Anticipated Message Estimator Module without Feedback generates more accurate messages than using Feedback. That research used a configuration similar to the ARX model used in this research. We are retesting the system using a configuration with and without Feedback to compare this feedback the new ARMAX approach, which is another form of feedback. Based on the underlying ARMAX theory, the new model should work better with inputs that have disturbances in their signal [47]. The Follower Robot might be able to use this in its advantage to create more accurate messages.

For every combination of the parameters that were mentioned previously, ten trials were run. We collected data at the moment the robots cross a waypoint. With a test

mission containing two waypoints (Figure 3.12) and by collecting samples from both robots, the total number of samples that are collected for this experiment is 2760. Note that these experiments do not include tests with message frequency of 1S/M while running anticipation because it is not necessary to run anticipation when the Follower Robot is receiving a message every time step from the Leader Robot.

3.5 Results

This section presents the results for two robots driving in parallel in an arena that contains multiple waypoints. The tests consist of letting the robots drive through all the waypoints and collecting data on their positions at the moment the first robot reaches each waypoint. Based on these results, we calculate the distance between the robots when each waypoint is reached, which determines how well the robots maintained synchronization while they navigated through the map. We tested the robots using a mission that is divided into two waypoints and the results are split in two tables (Table 3.5 and Table 3.6); one for each waypoint.

The Leader behaviors that were picked for these tests represent two groups of behaviors: one (On schedule) that is part of the training data and two others (Slow and Fast) that are outside of the training data. This will allow us to analyze the anticipation behaviors of the robots under expected conditions and as a generalization behavior; both behaviors that we previously presented in [1] and [26] respectively.

For the first waypoint (Table 3.5), the robots follow a synchronized behavior similar to what was observed with the ground robots and indoor experiments in previous research. For the majority of tests, the Follower Robot was able to reduce the error with the Leader Robot along the way to the first waypoint.

For the On Schedule (OnS) behavior, the Leader robot tends to finish slightly behind the Follower robot. This means that, based on what was observed during the experiments, the Leader robot ends with less that half of its front body ahead of the

Table 3.5: Average and standard deviation of the meeting point error of two robots for the first out of two waypoints in one mission while the Leader Robot runs three behaviors (OnS - On Schedule, Slow and Fast). The results represent simulations using No Anticipation (NA) and five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value). A positive average value means that, when the robots reached the waypoint, their final formation tends to show the Leader Robot tailing the Follower robot. If average value is negative, the Follower Robot tails the Leader Robot. For example, for the Slow behavior with 6 time steps per message (6S/M) and without anticipation, the average distance difference between the Leader Robot and the follower Robot is -9.3106 when they reach the first target point; with anticipation using no feedback and FL, the difference is -6.7681; and with anticipation using feedback and FL, the difference is -6.7063.

Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	1.2995(0.3564)	N/A	N/A	N/A	N/A	N/A
	2 S/M	-1.7420(0.3905)	1.2014(0.5095)	0.1098(0.7433)	-0.3247(0.4922)	0.3856(0.3841)	0.0626(0.5520)
			1.1423(0.6882)	-0.2846(0.6501)	0.0358(0.3562)	0.3945(0.9051)	0.2093(0.5710)
	6 S/M	-1.9721(0.4083)	-0.8304(0.3804)	0.7018(0.5089)	0.3843(0.8125)	0.3718(0.3470)	0.2356(0.4273)
			-0.1557(0.9549)	0.1207(0.4529)	0.5272(0.4095)	0.5800(0.5997)	0.4869(0.4764)
Slow	1 S/M	-3.2402(0.2737)	N/A	N/A	N/A	N/A	N/A
	2 S/M	-7.1082(0.6666)	-4.1797(0.4695)	-6.0415(0.5879)	-6.0202(0.6440)	-6.7089(0.6797)	-6.5994(0.4402)
			-4.6054(0.2342)	-5.7045(0.5583)	-5.8885(0.4827)	-8.3597(0.6851)	-6.9049(0.9100)
	6 S/M	-9.3106(0.3216)	-6.7681(0.3484)	-6.5373(0.3057)	-6.6480(0.5711)	-7.1377(0.4948)	-6.9766(0.4914)
			-6.7063(0.9340)	-6.9405(0.5767)	-6.7859(0.5288)	-6.6850(1.3071)	-7.3746(1.0228)
Fast	1 S/M	1.7191(0.3051)	N/A	N/A	N/A	N/A	N/A
	2 S/M	2.0793(1.0494)	1.7203(0.9742)	1.3229(0.6412)	1.5220(0.9413)	1.6590(0.6138)	2.0282(0.9805)
			2.5045(0.8466)	1.8244(0.6340)	1.7846(0.8640)	1.6042(0.9393)	1.6250(0.7490)
	6 S/M	2.4275(0.7074)	1.8208(0.26)	1.8612(0.8717)	2.4621(1.0233)	2.1022(0.8170)	1.5614(0.8402)
			1.5291(1.0765)	2.8217(0.5937)	2.0255(0.6277)	1.8669(1.0047)	2.6903(0.7040)

Follower on average or almost head-to-head. We believe that his offset can be caused by the noise that is always part of the Leader message. When messages get lost, the Leader Robot typically gets ahead and finishes first. We believe that the gap between the progress messages makes the Follower Robot less reactive to this extra space that is generated. The results with all 5 anticipation models show that the Follower robot could anticipate the behavior of the Leader Robot and reduce the distance between them. Also, in almost all of the cases the Follower Robot was able to recover and get back into formations. The AM_NNGA had the best performance out of all the 5

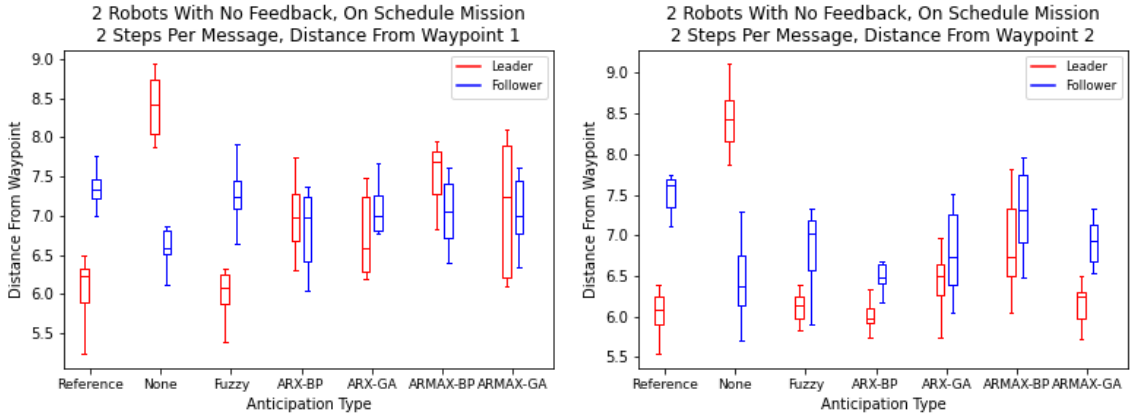
Table 3.6: Average and standard deviation of the meeting point error of two robots for the second and last waypoint of a mission while the Leader Robot runs three behaviors (OnS - On Schedule, Slow and Fast). The results represent simulations using No Anticipation (NA) and five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value).

Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	1.4286(0.3458)	N/A	N/A	N/A	N/A	N/A
	2 S/M	-2.0174(0.3350)	0.6938(0.5142) 1.7100(0.8729)	-0.4481(0.7369) -0.5104(0.5316)	-0.3765(0.4126) -0.5548(0.7634)	-0.4314(0.6854) -0.7495(0.7775)	-0.6077(0.6875) -0.5006(0.6446)
	6 S/M	-2.4877(0.8210)	0.8134(0.5942) 1.1621(1.8327)	0.6341(1.0608) -0.6341(0.6336)	-0.1063(0.6147) -0.1280(0.8549)	-0.5075(0.7801) -0.1955(0.6738)	-0.2441(0.3756) -0.2587(0.8187)
Slow	1 S/M	-3.3358(0.7268)	N/A	N/A	N/A	N/A	N/A
	2 S/M	-9.6168(0.4280)	-3.6165(0.9883) -4.1561(0.4680)	-9.3639(0.4243) -7.8613(0.8521)	-7.7475(1.3752) -6.7734(0.9241)	-13.0587(3.4973) -13.4298(1.6313)	-7.8415(0.8810) -5.7715(1.8476)
	6 S/M	-20.5537(1.2311)	-6.6901(1.7528) -2.1581(2.6411)	-12.7476(0.3541) -13.0646(1.1421)	-7.7418(1.2570) -11.0883(0.6383)	-7.4490(2.2722) -3.5177(6.1784)	-8.6531(0.8468) -7.2815(2.3198)
Fast	1 S/M	1.0382(0.7555)	N/A	N/A	N/A	N/A	N/A
	2 S/M	3.1035(1.1618)	2.4269(1.0601) 3.1760(0.5268)	2.4180(0.7343) 2.4954(1.8759)	2.1110(1.3765) 3.4846(1.6135)	6.9292(1.9357) 6.6883(2.0139)	7.1848(1.2366) 6.5167(2.0524)
	6 S/M	6.6104(1.6138)	2.9694(1.3348) 4.9119(0.9498)	3.9547(1.3877) 7.3007(1.2587)	4.7944(1.1563) 6.6626(2.9832)	6.2116(3.2404) 6.7004(1.4225)	7.3403(1.4964) 6.5274(1.6473)

anticipation models while trying to solve the On Schedule behavior.

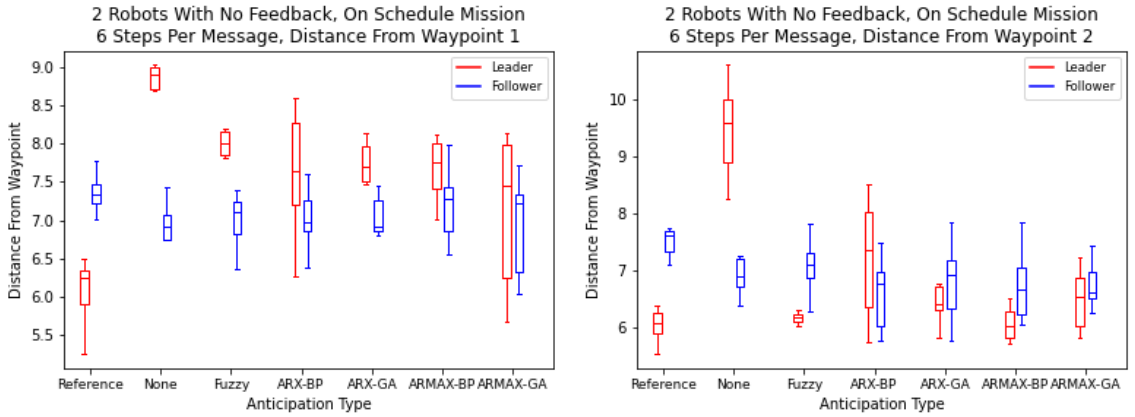
The other two missions were slightly more challenging for the anticipation models but, in general, the Follower Robot could reduce the error between the Leader Robot. The Fuzzy Logic model had the best results out of the 5 models for the Slow and Fast behaviors combined but, in general, it was more challenging for the anticipation models to solve the Fast behavior than the Slow behavior. For all the tests where the Follower Robot reduced the distance error considerably, the results show that the Follower Robot was able to recover the formation that the robots had on optimal conditions, as well.

For the second waypoint (Table 3.6), it was easy to notice that, when the Follower Robot was not using anticipation, there was an accumulated error that was carried from the results of the previous waypoint. By increasing the gap between



(a) Waypoint 1 (b) Waypoint 2

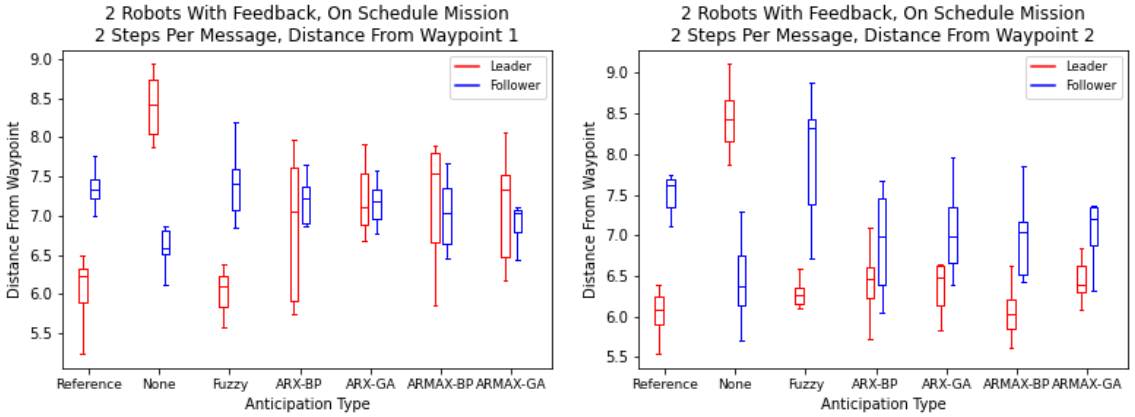
Figure 3.14: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the On Schedule behavior, a message frequency of 2 steps per message and anticipation with no feedback



(a) Waypoint 1 (b) Waypoint 2

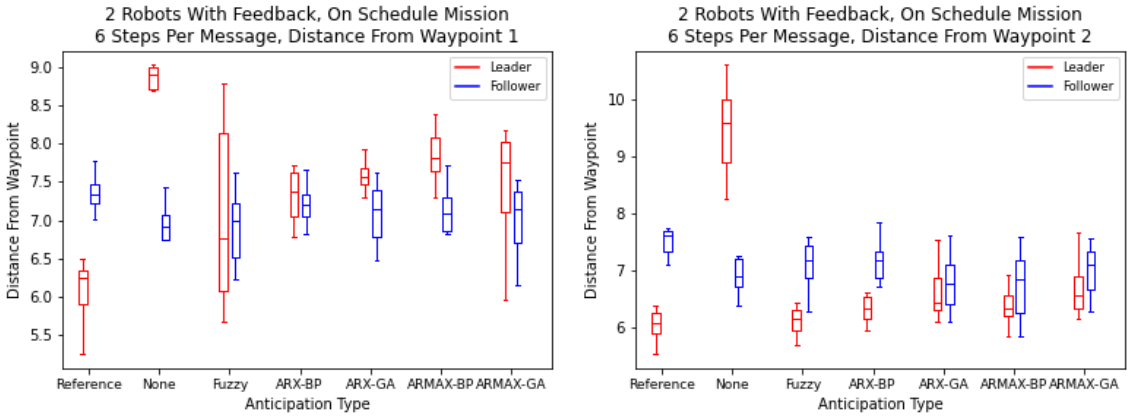
Figure 3.15: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the On Schedule behavior, a message frequency of 6 steps per message and anticipation with no feedback

messages with No Anticipation, the average distance error increases considerably once the Robots reach the final waypoint. During the On Schedule tests for the second waypoint, all the anticipation models were able to infer the behavior coming from the Leader Robot and the Follower Robot was able to stay very close to the Leader Robot. Compared to the On Schedule results on Table 3.5, the Follower and the Leader Robots drove side by side in average until they reached that second waypoint while the follower was always slightly ahead during the previous section of the path.



(a) Waypoint 1 (b) Waypoint 2

Figure 3.16: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the On Schedule behavior, a message frequency of 2 steps per message and anticipation with feedback enabled

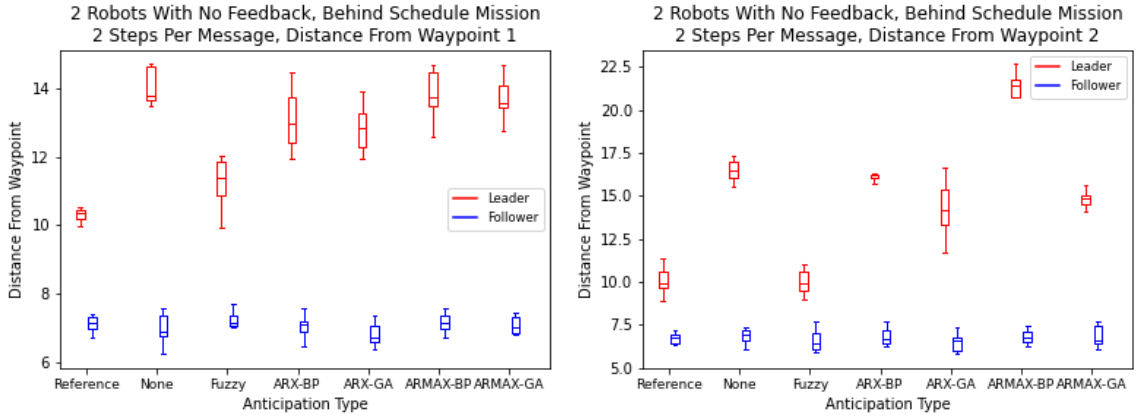


(a) Waypoint 1 (b) Waypoint 2

Figure 3.17: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the On Schedule behavior, a message frequency of 6 steps per message and anticipation with feedback enabled

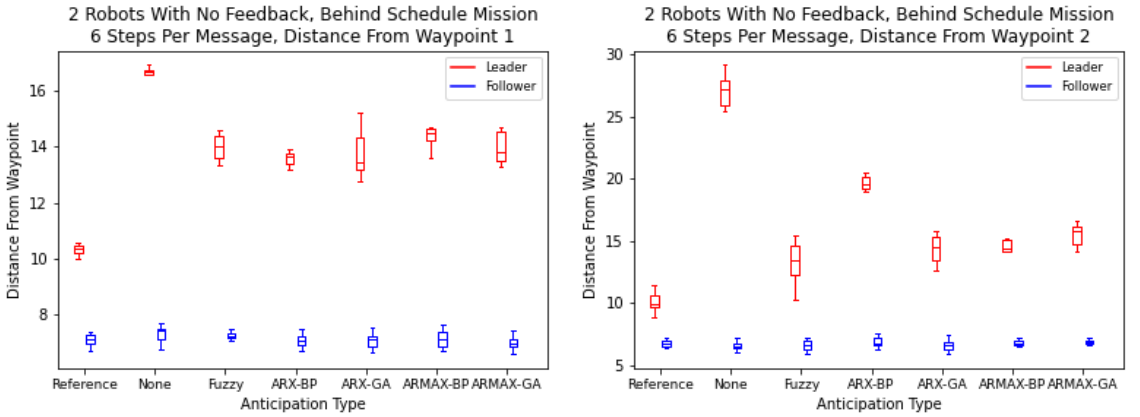
For this behavior, the A_NNGA was definitely the most successful anticipation model by having the smallest error an almost every test.

By analyzing the results for the Slow and Fast behaviors in Table 3.6, it is clear that it was easier for the Follower Robot to solve the problem when the Leader was driving slowly towards the waypoint. In fact, Table 3.5 and 3.6 show a consistent behavior that, for this system, the follower cannot react as fast as necessary when the Leader Robot uses fast behaviors outside the training data. One of the main reasons



(a) Waypoint 1 (b) Waypoint 2

Figure 3.18: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Slow behavior, a message frequency of 2 steps per message and anticipation with no feedback

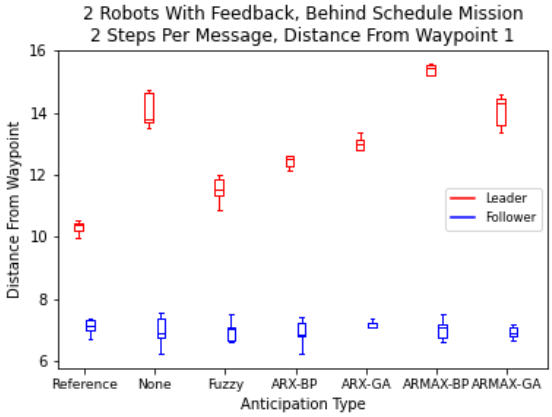


(a) Waypoint 1 (b) Waypoint 2

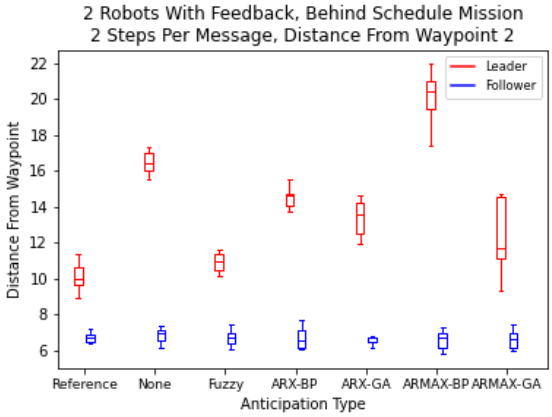
Figure 3.19: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Slow behavior, a message frequency of 6 steps per message and anticipation with no feedback

that causes this limitation might be due to the preexisting noise that the Leader messages contain which was also included in cases for the training data. On the other hand, the results for Slow behaviors were a lot more successful. All the anticipation models were able to reduce the distance error between the Follower and the Leader Robots; especially the Fuzzy Logic model which had the smallest error results.

The figures in this section represent represent the the distance of the robots to the immediate waypoint when the first robot reaches this waypoint. We present 24 figures

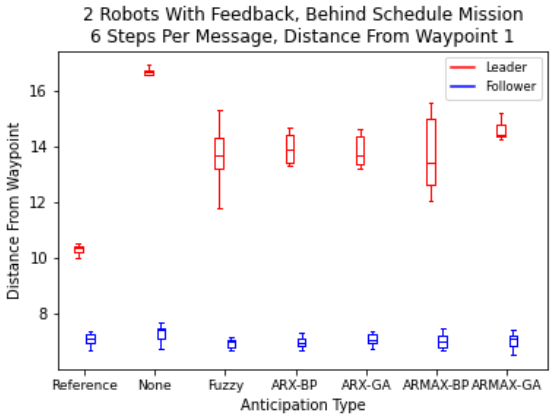


(a) Waypoint 1

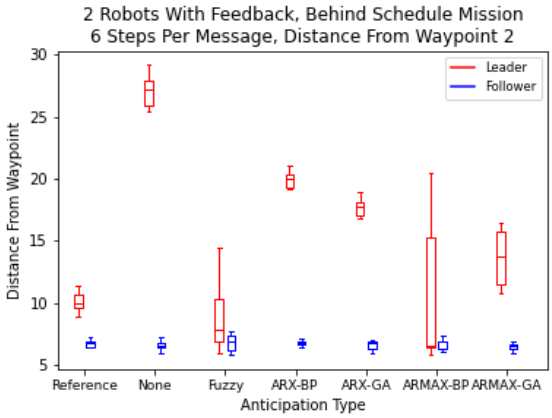


(b) Waypoint 2

Figure 3.20: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Slow behavior, a message frequency of 2 steps per message and anticipation with feedback enabled



(a) Waypoint 1



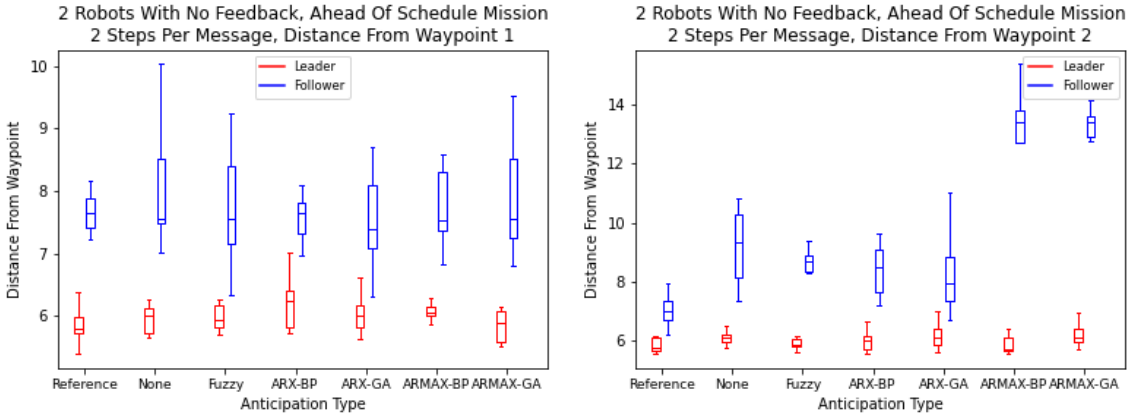
(b) Waypoint 2

Figure 3.21: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Slow behavior, a message frequency of 6 steps per message and anticipation with feedback enabled

that show all the test configurations that were presented in Tables 3.5 and Table 3.6.

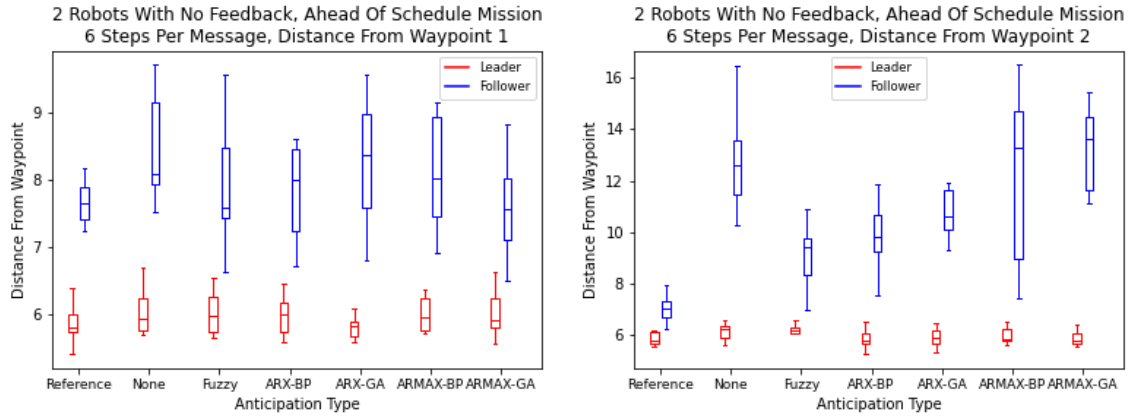
These figures are organized in small groups of 4 figures based on the behavior of the Leader Robot and the configuration of the message queue:

- On schedule and No feedback (Figures 3.14 and 3.15)
- On schedule and With feedback (Figures 3.16 and 3.17)
- Slow (Behind Schedule) and No feedback (Figures 3.18 and 3.19)



(a) Waypoint 1 (b) Waypoint 2

Figure 3.22: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Fast behavior, a message frequency of 2 steps per message and anticipation with no feedback

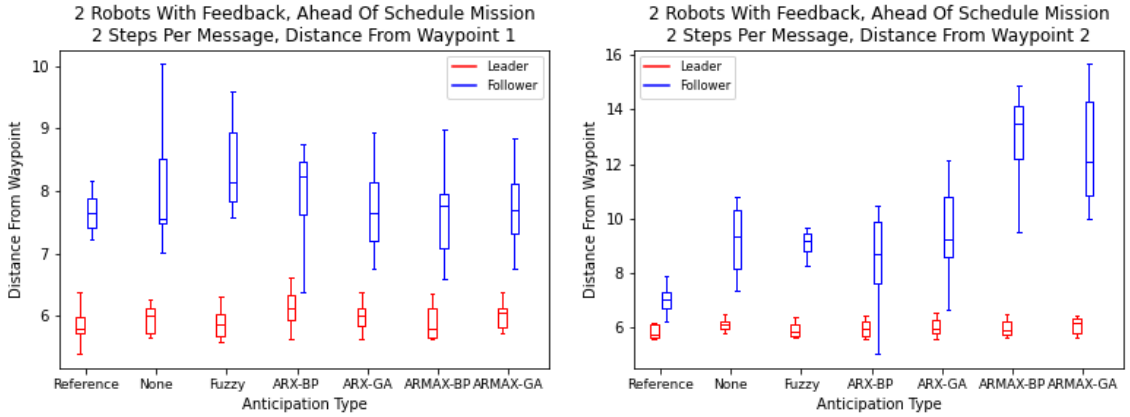


(a) Waypoint 1 (b) Waypoint 2

Figure 3.23: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Fast behavior, a message frequency of 6 steps per message and anticipation with no feedback

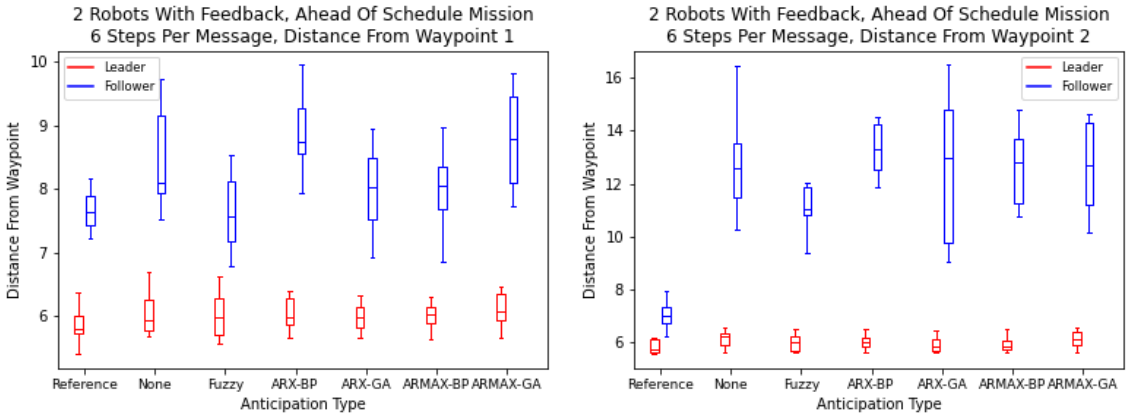
- Slow (Behind Schedule) and With feedback (Figures 3.20 and 3.21)
- Fast (Ahead of Schedule) and No feedback (Figures 3.22 and 3.23)
- Fast (Ahead of Schedule) and With feedback (Figures 3.24 and 3.25)

In addition, each figure comes with 2 subfigures that shows the results for the two waypoints that were used in the mission for these experiments. Each subfigure shows 7 sets of data where each group shows the results for both the Leader Robot and the



(a) Waypoint 1 (b) Waypoint 2

Figure 3.24: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Fast behavior, a message frequency of 2 steps per message and anticipation with feedback enabled



(a) Waypoint 1 (b) Waypoint 2

Figure 3.25: Distance error at the moment a waypoint is reached by the Leader Robot and the Follower Robot for the Fast behavior, a message frequency of 2 steps per message and anticipation with feedback enabled

Follower Robot. The first set is defined as the reference and it shows the results of the robots driving on optimal conditions for that configuration; this means using a message frequency of 1 Step per Message (1S/M). The other sets in the figure show the results at a different frequency (2S/M or 6S/M). The second set shows the results for the configuration when no anticipation is used at the specified frequency. And, the next 5 sets show the results for the all 5 anticipation models that we are studying; FL, A_NNBP, A_NNGA, AM_NNBP and AM_NNGA respectively.

3.6 Conclusions

In the previous section, the results were presented in two separate groups: the first group focuses on the results obtained during the first waypoint (Table 3.5) while the second group of data showed the results for the second waypoint (Table 3.6). We noticed that, even though this new environment offered more difficult navigation challenges for the robots and the anticipation modules in general, it was easier for the Follower Robot to synchronize with the Leader Robot during the section before the first waypoint than during the section between the first and second waypoints. Normally, after reaching the first waypoint, the robots experienced their first mutual collisions. This clearly depends on how close they are driving from each other. This adds another level of difficulty to the anticipation models because the Leader gets affected by these collisions as well, which makes the Leader robot change the information in its progress messages. Most of the times, the anticipation models recognized the change in the Leader behavior and create the correct substitute for the missing Leader message which is reflected in the Follower behavior. In a very few cases, most of them observed while using the AM_NNBP model and Leader behaviors outside the training data, the anticipation model could not infer the right Leader messages after passing the first waypoint and hitting those first collisions.

Based on the results that were presented in this chapter, the anticipation models, that we tested, helped the Follower Robot on improving its behavior and keeping it close to the Leader Robot when messages from the Leader Robot are missing. This improvement is represented by the reduction of the distance error when the results are compared to the No Anticipation case. In the best cases, this improvement shows a reduction of 15 units in distance if we use the scale used in Unity. Some models were successful than others for specific test configurations; especially for Leader Robot behaviors. The Leader behaviors that were picked for testing can be divided in two groups: part of the training dataset, like the On Schedule behavior, and not part

of the training dataset, like the Slow and Fast behaviors. We observed that the two ARMAX models are very effective while the Leader Robot uses behavior that are part of the training data. These results include mostly small distance errors with a small standard deviation which means less disperse data. On the other hand, the ARMAX models performed poorly with the behaviors outside the training data, which means that we could not identify generalization as an emerging behavior from these tests in particular.

The ARX models showed a much better performance with the Slow and Fast behaviors. The FL, A_NNBP and A_NNGA models could reduced the distance error better than the other two models for this particular group of Leader behavior. This also means, that we could identify some generalization behaviors from the Follower Robot by being able to infer the right Leader messages during these tests in particular. The generalization behavior, which we studied and presented in [26], was defined as an emergent behavior that shows up during successful cases that does not involve missions with Leader messages related to the training data.

In a general overview, the FL and the A_NNGA models were the models with best results from all of the combined tests by getting the smallest distance error values for 10 and 5 cases respectively. Both A_NNBP and AM_NNGA models had 4 cases with the best smallest distance error while the least successful anticipation model was the AM_NNBP with just 1 case.

Chapter 4: Natural Language Processing (NLP) and Fuzzy Logic Parsing for messages to a group of Robots in a Human-Robot interaction

4.1 Introduction

In addition to the study on anticipation using autonomous robots, a study involving NLP tools and fuzzy Logic is also included in this research. The goal of this study is to create an alternate application and approach for anticipation for communication between two agents where messages may be corrupted. To test this new approach, we moved from a robot-robot interaction model to a human-robot interaction model.

This approach was used to create a module that can fulfill the 3 following tasks:

- Recognizing the content of a voice message
- Deciding if the content of that message is valid
- Detecting if an homophone word of any of the valid commands is part of the message and replacing it with the right word.
- Reconstructing a corrupted message into a valid command for the robot.

The goal is not to create a system that uses tree-based search to evaluate each possible branch for every input until it finds a plausible answer. Instead, the system uses a rule-based technique by implementing an AI method that avoids a full search. This can be described as a weight-based system that can identify the right command; including the correct order of words and similar sounds based on the context from an audio input. Once identified the valid command will be sent to the correct robot in order to be executed immediately.

This chapter is organized as follows: Section 4.2 presents a brief state of the art relevant to this part of the research and to the work on human-robot interaction. Section 4.3 describes the approach that was used to solve this problem and the reasons why some strategies were picked over other options before running the final experiment. Section 4.4 presents the results of the experiment that was set between one human user and a group of robots. Finally, section 4.5 lists the conclusions that came up after analysing the previous results.

4.2 Background

When someone speaks in a specific language, for example in English, different features can be extracted just by listening to the person, for example differentiating between native and second-language speakers, approximate age, accent, or race. The authors in [48] suggest to use phonetic algorithms to numerical representation of words based on how they sound and, this way, it is easier to create a map of the language that can not only identify the type of speaker but also identify homophone words or a word that sounds different because of the features of the speakers. For this project, the use of phonetic algorithms will help to identify keywords based on how they sound and not how they are spelled. This approach will help us identify the right command that will be passed to a fleet of robots without any concerns of what type of accent or other features that the speaker has that can modify the original content of the command.

A simple approach to robot control by voice command is presented in [49]. Here, the process for voice recognition is done by implementing signal filters and calculating the Mel-Frequency Campestral Coefficients (MFCC) that are commonly used in this kind of process. This paper presents a robot that can identify voice commands containing one or two words within a small dictionary. In this dissertation, we implemented an approach that uses a set of Fuzzy Logic modules and is capable of

handling more complex and longer command with sentences up to 6 words. Also, we will describe how this approach can be modified easily in order to increase the length of the phrase for longer commands.

A closer approach to the solution that is introduced in this chapter was presented by Microsoft in its patent [50]. Here, the authors described a voice recognition model and a text-to-speech module that include additional processing for recognizing homonyms and it can be used later to generate a desired command for a dial application. This approach uses a pre-configured database of homonyms and runs an analysis based on a Context Free Grammar model to decide what homonym the program must use. Our approach uses phonetic search libraries and a Fuzzy Logic module to identify the phonetic distance between words and to identify homonyms without storing a large list of possibilities. Each word receives a value and a second Fuzzy Logic module validates the incoming command.

The authors in [51] propose an approach to indoor navigation using voice commands on a robot. The authors refer to this as behavioral navigation because the commands that are passed to the robot are similar to how human would give directions to someone else. These directions are normally based on descriptions of rooms and hallways. The goal was to make the robot analyze the semantics of the phrase and identify the different commands that are mentioned in it and the sequence in the path that the robot must follow. For the research in this chapter, the robot assumes that every command is received with no problem; it does not considers background noise or the way the user speaks. We propose a way to solve this additional problem which is typical in a real-world environment.

Humans can adapt to perform a group task and show high level of coordination and adaptation once they get familiar with the task and the group [52]. Research of human-robot interaction has been studying the way robots can be part of a group task and mimic a similar levels of cognition to solve a task synchronously. This behavior

requires a constant two-way communication between both ends but it can still turn into an intermittent exchange of actions. The implementation of anticipation has been described in [52] as a way to improve coordination to solve a group task by adapting to actions, which is also another subject of discussion later in Chapter 5, or by adapting to language, which is close to the point of discussion in this chapter.

In this chapter, the approach for the solution in this part of the research was develop using Android programming and a set of robots that were programmed using Arduino boards. The program that was built in Android also included the implementation of a Fuzzy Logic library and Soundex algorithms, which are phonetic algorithms that can encode words by their sound based on the the English language. The experiments included a small group with two robots but, later in this chapter, we also show how this approach can be scaled up easily.

4.3 Methods

The main program is run as an Android application over a Samsung Galaxy S5 device. Android Studio comes with a Speech library that enables the use of the same voice recognition engine that Google uses in its other services. The only main requirement to run this library is to have a constant access to Internet because this library was built with the intention of being run on mobile devices, which means that it should not take a lot of disk space. For this the reason, the app will try to establish a connection to the online service that runs the main voice recognition engine. When the speech library is activated and receives an audio input via the microphone, this module returns a String variable with all the detected words.

Next, the string is analyzed String by taking each word and passing them through a set of phonetic encoders available in the *org.apache.commons.codec.language* package. In general, each phonetic encoder in this package can assign a value to any word based on the way each one of the letters affects the sound the word in the English

language. In general, each one of these encoders can generate a unique map that contains every word possible. Words with similar sounds are expected to have similar or close values after they are encoded. Using one phonetic encoder showed that the Fuzzy Logic was not succesful in identifying all the possible homophone cases related to the words in the robot dictionary. For this reason, we used two phonetic encoders to solve this problem. Based on the results that we found during the experimental process, we concluded that it was more effective to solve this problem like a "triangulation"approach but with a Fuzzy Logic module trying to converge to the right answer based on the information of two functions. The Fuzzy Logic module uses the values of the encoders *Daitch–Mokotoff Soundex* (DMS) and *DoubleMetaphone* (DM) in parallel for each word to solve the homophone problem with more accuracy. As an additional effect, it also can identify if the input word is not part of the dictionary nor homophone. The results that we obtained from these tests are analyzed and compared in the next section (Table 4.1) as part of the results of these experiments.

The set of memberships for each input variable in the Fuzzy Logic module are set as triangular functions and they are created based on the values that are generated by the DMS and the DM functions (Table 4.1). In total, the DMS variable is created with 20 memberships and the DM variable is crated with 19 because this function returns the same value for the words *to* and *two*. This is one of the reasons why we decided to implement the "triangulation"approach using two functions in order to make the Fuzzy Logic module come up with the right answer when one of the functions fails. The output of this fuzzy Logic module is an evenly distributed set of memberships that contain every word in the robot dictionary, which means, if the input word was correctly identified as the right word or as an homophone, the right output membership should have a value close to one (1).

After converting every word of the input String into its fuzzified representation, the value is passed to another Fuzzy Logic module to determine if the words match a

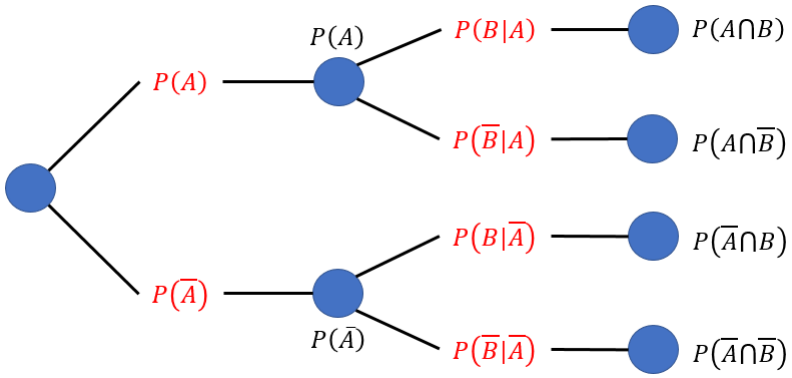


Figure 4.1: Example of a probability tree

specific command. Here, the inputs to this Fuzzy Logic module represent a position in the command and every input contains every word in the dictionary as a membership. The robot uses commands that contains up to 6 words but it also includes shorter commands, which means, that the robot do not necessarily uses every input in every fuzzy rule that it is defined. The output of the module indicates the command that the robot should perform. This strategy might look similar to a search-tree method where going down through one branch represents a robot command and a tree node represents a word in the robot command. The right combination of words would generate a value representing a branch by matching the maximum number of words in terms of ordering and phonetics. The difference between this method and a tree-search algorithm is that the time complexity would remain constant close to $O(1)$ or $O(n)$ in the worst-case scenario because the Fuzzy rules are treated similar to IF-ELSE statements. On the other hand, if the number of rules and the number of words in a command increases, the time complexity of a tree-search method, like a Depth First Search, would increase in the order of $O(|V| + |E|)$, where V is the number of the already-visited edges or nodes and E is number of vertices or branches.

This method creates an approach similar to a probability tree 4.1 but, by using fuzzy rules to analyze every branch, finding the correct end node will be faster than a basic tree-search method; like depth-first search (DFS) or breadth-first search (BFS).

By having a set of rules that covers the commands that the robot can use, a fuzzy logic based approach can identify the right command/branch by selecting the membership with the highest fuzzyfied value in its output.

To test this module, a set of sentences will be used on the robot. Some of these sentences will be correct commands while the others will be similar sentences that will try to fool/break the analysis and decision of the robot. Each sentence will also be intended to be used by different users. The type of user will cover features like first-language and second-language English speakers. This cases will test the robustness of the module through different accents. A successful result will be able to send a accurate command to the robot; otherwise the command will be deleted/dropped by the module itself.

4.4 Results

The first part of the experiment involves defining the commands and the words inside each command which will be used in the dictionary of the robots. Initially, we want to set up commands that will allow the robots to follow some basic navigation commands like moving forward or backwards, turning left or right, and speeding up or slowing down.

In addition, we add the option to identify which robot the message is intended to by giving a robot a unique ID and making this ID as an important/required part of the message. We also used this ID as part of a header to identify when the user is trying to initialize a valid command. For example, the header of every command should include the words *Robot* and the ID number of the robot in order to consider the input as a valid command. For the experiments, we enabled up to four robots to received messages from the user. While running some test with the voice engine, we noticed that, sometimes, the ID value is returned as a numeric value and other times, is returned as a word. For example, $ID = 1$ can appear as '1' or as 'one'. By

following all these previous basic rules, the only user commands that will be accepted by the robot are listed as follows:

- *Robot # move to waypoint #*
- *Robot # move backwards*
- *Robot # move forward*
- *Robot # move left*
- *Robot # move right*
- *Robot # set speed faster*
- *Robot # set speed slower*
- *Robot # set speed to #*
- *Robot # stop*

Based on the content of each command, we selected 19 words in total for the robot dictionary. These words are listed in Table 4.1. This table also shows the respective values for the DaitchMokotoffSoundex (DMS) and DoubleMetaphone (DM) phonetic functions. Each phonetic function has its own method to calculate a score for English words. This means, that words can be sorted in different ways just by using different phonetic functions. Table 4.1 shows the dictionary words sorted based on the values returned by the DMS phonetic function. The next column shows the values for the DM phonetic function and it shows that the order of these values are not close to the values in the previous column. The idea of using two phonetic functions is to pinpoint the right word for every mismatch case because the tests, that were performed using just one function, did not solve all the homophone cases. the goal is to solve all possible cases because this application is intended to be used by the AUV project in a future phase.

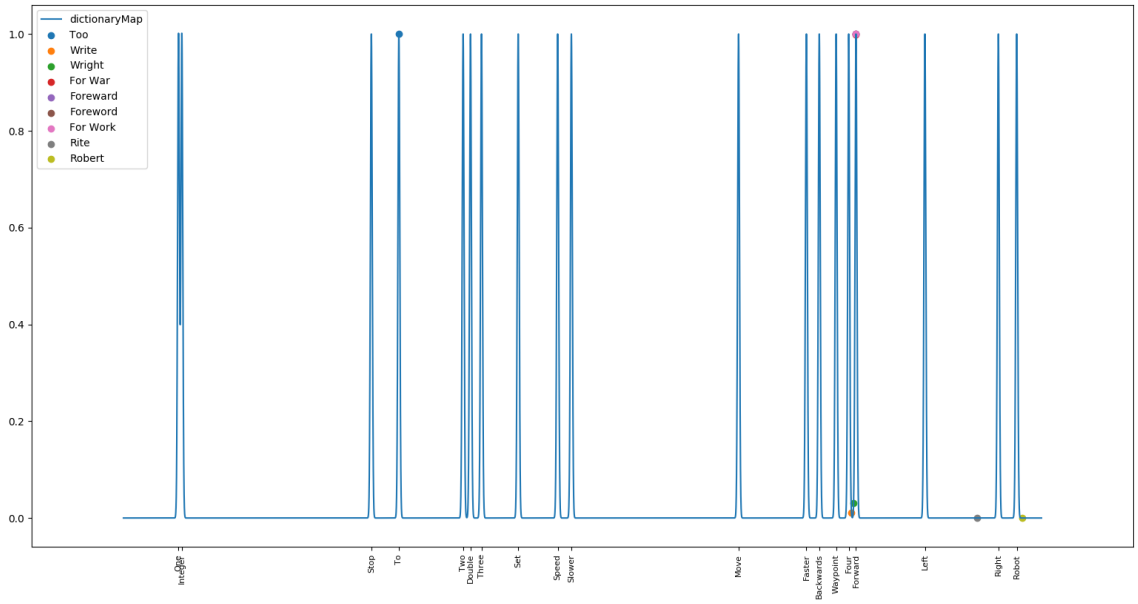


Figure 4.2: Gaussian Map of the dictionary words based on the DaitchMokotoff-Soundex phonetic function. Variance=0.01. This figure also displays the location of the common homophone words that were encountered during the experiments

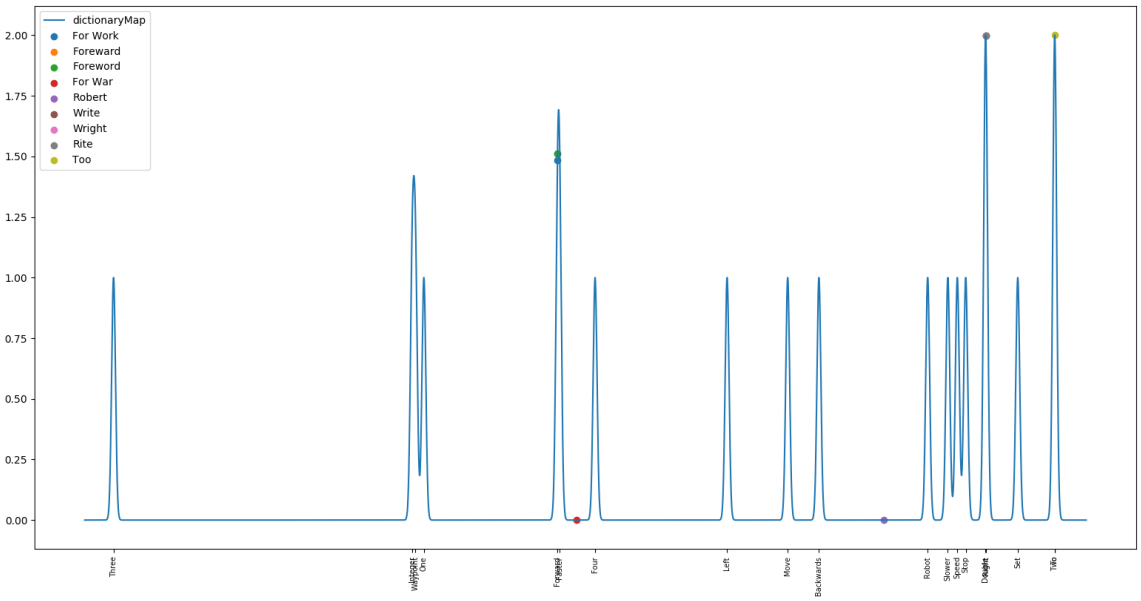


Figure 4.3: Gaussian Map of the dictionary words based on the DoubleMetaphone phonetic function. Variance=0.005. This figure also displays the location of the common homophone words that were encountered during the experiments

Table 4.1: Values for the dictionary words generated by the DaitchMokotoffSoundex and DoubleMetaphone phonetic functions. Each function organizes the words in a different order crating a different mapping for the dictionary.

Function / **Words**	*DaitchMokotoffSoundex*	*DoubleMetaphone*
One	0.6	0.18055
Integer	0.6359	0.15041
Stop	2.7	1.58716
To	3.0	1.81818
Two	3.7	1.81818
Double	3.78	1.6386
Three	3.9	-0.625
Set	4.3	1.72222
Speed	4.73	1.56532
Slower	4.879	1.54074
Move	6.7	1.125
Faster	7.439	0.53327
Backwards	7.5794	1.20606
Waypoint	7.763	0.15868
Four	7.9	0.625
Forward	7.9793	0.52749
Left	8.73	0.96768
Right	9.53	1.63889
Robot	9.73	1.48839

Figures 4.2 and 4.3 show an approach to the distribution of the center for each dictionary word by using their DMS and DM phonetic values as a position for the membership. The distribution of the words in each Figures shows how some dictionary words are very close to each other for one function but not for the other function. This shows that it would be more difficult to identify some words around those areas accurately and the combined use of two phonetic functions and a Fuzzy Logic model helps to improve the accuracy of the phonetic analysis for homophone words.

Based on the phonetic values in Table 4.1, we can design a Fuzzy Logic model to evaluate the input words and to identify homophone words and/or mismatched cases. Figure 4.4 shows the structure of the phonetic analyzer using a Fuzzy Logic model. After receiving the command phrase from the voice-to-text module, the phonetic

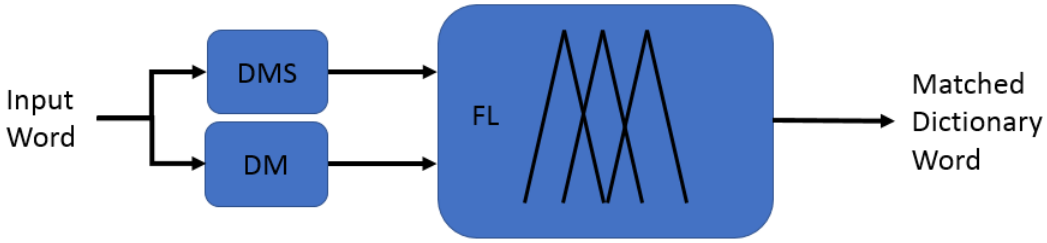


Figure 4.4: Model for the phonetic analyzer using a Fuzzy Logic model and two phonetic functions

analyzer module takes each word of the input command and applies two different scores using the two selected phonetic libraries (DMS and DM).

Next, the inputs for the fuzzy logic model are configured by using triangular memberships for each word in the dictionary and by using the values in Table 4.1 as the center for each one of the triangular memberships. The Fuzzy Logic model just follows a simple format for the rules and checks if both inputs are referring to the same dictionary word.

$$IF\ \text{input}_{\text{DMS}} == \text{DictWord}_{\text{i}}\ \&\&\ \text{input}_{\text{DM}} == \text{DictWord}_{\text{i}}\ THEN\ \text{DictWordID}_{\text{i}} \tag{4.1}$$

The output of the Fuzzy Logic model classifies the analyzed word by assigning a ID value that represents a dictionary word. Input words that match the dictionary words are assigned with the exact ID value. Homophone words and mismatched cases will be assigned an ID value based on the neighboring memberships of each inputs. Table 4.2 shows a list of 9 homophone words and mismatched cases that the speech engine returned during the experiments. In addition, this table also shows the values that the DMS and DM phonetic functions assigned to these words. These words were created while trying to identify words like *To*, *Two*, *Right*, *Forward* and *Robert*. Figures 4.2 and 4.3 show the scores that the phonetic functions assigned to homophone words and the mismatched cases and their relative position to the dictionary words.

Table 4.2: Values for homophone words and/or mismatched cases generated by the speech engine and evaluated by the DaitchMokotoffSoundex and DoubleMetaphone phonetic functions.

Function / **Words**	*DaitchMokotoffSoundex*	*DoubleMetaphone*
Too	3.0	1.81818
Write	7.93	1.63889
Wright	7.953	1.63889
For War	7.979	0.57624
Foreward	7.9793	0.52749
Foreword	7.9793	0.52749
For Work	7.9795	0.52725
Rite	9.3	1.63889
Robert	9.793	1.37443

In general, the figures show that most of these values are close to their homophone words in the dictionary. For example, *Robert* is very closer to *Robot* for both phonetic functions than any other dictionary word. Also, if we take the mismatched cases for the word Forward (For War, Foreward, Foreword, For Work) and check their position for the DM phonetic function, they are close to the center of *Forward* except for the case *For War* (Figure 4.3). But this is solved by implementing the DMS phonetic function. For this case, the mismatched cases, including *For War*, are close to center of *Forward*.

The job of the phonetic analyzer is to identify and to classify the words in the input command. A second Fuzzy Logic module is to take the command as a whole and evaluate its syntax (Figure 4.5). First, we define the number of inputs that this Fuzzy Logic model requires and this is simply based on the maximum number of words that can be found in the longest command. In this case, we must consider a model with 6 inputs and each one of them was defined with the same number of memberships. These memberships are triangular-shape and they are defined based on the category IDs that were created by the previous module. In this case, we considered 19 categories that represent all 19 words in the robot dictionary. Next,

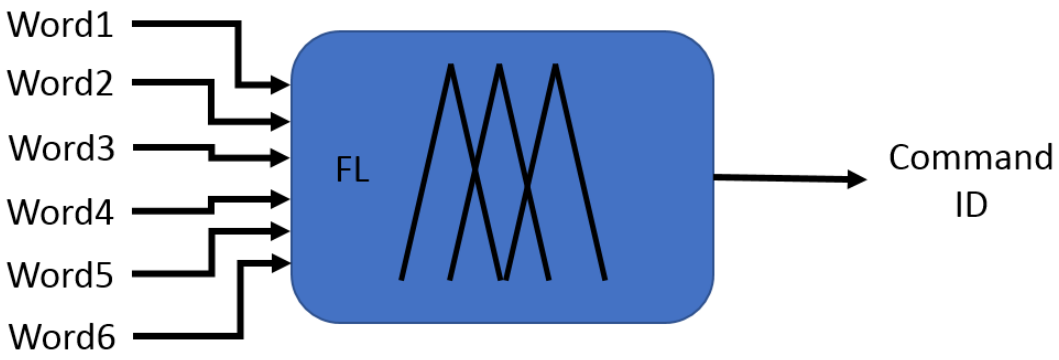


Figure 4.5: Model for the command identification method using a Fuzzy Logic model

the Fuzzy Logic rules of this model consist on the user commands that we defined at the beginning of this section. These rules will check if the input words have a sequence of IDs in the right order. If the previous module failed on identifying one of the input words by sound, this module will detect that there is an ID in the wrong place and return an invalid command (output value equal or close to zero - 0). If every input word matches one of the rules, this Fuzzy Logic module will return a value that represent one of the nine available robot commands. $Output \in [1 - 9]$. This output range follows the idea of the probability tree that was mentioned previously (Figure 4.1) and the output value of the fuzzy Logic module determines which branch of the tree is the closest (or robot command) matches the input command.

Based on the rule syntax that was used in the last Fuzzy Logic module, it is still possible to encounter some cases that the module will tag as invalid which will result in the system dropping the entire message. This is a short list of examples related to invalid commands:

- *move forward* (*No Header)
- *# Robot move backwards* (*inverted Header)
- *Robot # move* (*Incomplete information)
- *Robot # return* (*Non-dictionary words)

- *Robot # move to* right (*mixed commands)

The approach of using a sequence of Fuzzy Logic models, that was presented in this chapter, could identify and restore all the homophone words that were mistakenly returned by the speech engine during the experiments.

4.5 Conclusions

The experiment that was run in this chapter show that the approach of using a sequence of Fuzzy Logic models was successful as a tool to recover corrupted voice commands. Initially, the Fuzzy Logic model was not successful at solving every possible mismatched case with just one phonetic function. This happened because every phonetic function has its own way to evaluate a word phonetically and give it a score. To solve this problem, we use two phonetic functions on an input word as a way to pinpoint and identify the right dictionary word. The score that is applied by these functions on the input word is used by the Fuzzy Logic model which improved the results of the experiments. After implementing this new approach, every mismatched case was corrected successfully.

In addition, the way that the commands are defined as Fuzzy Logic rules makes it simpler for this approach to integrate more commands to the current command library of the robot. Although these are simple rules, this approach would allow us to increase the number of rules and the number of words inside the command easily, without any big change in the code for the Fuzzy Logic model. This also means, that this model is not dependant of the number robots that are connected to the user device. This can be considered as a module that is just in charge of evaluating the message that are being shared; without considering if it is running in a centralized or decentralized system.

Chapter 5: Anticipation Strategy and Formation Control for Multiple Ground Robots in an Outdoor Environment

5.1 Introduction

In this chapter, we move from the system that was described in Chapter 3 to a system that involves a larger group of robots. For this case, we move to a group with 3 robots. This setup can help us test the movement limitations of the robots; especially for the Leader Robot. By placing another Follower Robot on the right side of the Leader, the Leader will have less freedom to move around and its navigation decision will also affect the progress message that is broadcasted. The general goal of a Multiple Robot System (MRS) is to make robots work together to solve a given task. According to the authors in [53], there are two more definitions in Robotics that should not be confused with MRS: Multi-Agent System (MAS) and Distributed Artificial Intelligence (DAI). A MAS refers to a distributed computer system where the individual nodes are stationary. DAI focuses on problems that involves software agents. The main feature, that these systems share, is that a collective team is more powerful and effective than a single unit of the team [53].

A MRS has commonly been labeled as *Swarm Robotics* or *Collective Robotics*. But MRS is a term that researchers use carefully when they mention it because focuses on synchronization in general rather than the actual number of robots or size of the swarm. MRS are generally considered to be scalable and by definition, can be replicated with few units or with thousands units [54].

Another feature of MRS is the implementation of a decentralized system that does not follow a hierarchical structure and allows every unit to make its own decision, but it is shared information between the robots that helps to create a collective behavior [55]. Self-organization and tolerance to individual failure are some of the

emergent behaviors that swarm robotics can display. Self-organization is described as a cooperative behavior that is performed by a group of agents while they perform their own simple tasks and share constant and reliable information [56]. Individual failure is identified when an robot loses the ability to maintain formation due to aspects like collisions with other agents or objects [57].

This part of the research tests whether anticipation can be used to improve self-organization and tolerance to individual failures (in the form of collisions between robots).

This chapter is organized as follows: Section 5.2 presents the current state of the art that is relevant to this research that helps tying the work that we also presented Chapter 3. Section 5.3 complements the methods that were described in Section 3.4 previously in order to scale up the simulation to a larger number of robots. Section 5.4 presents the results of the simulation that was run for multiple robots while using anticipation. Finally, section 5.5 presents the conclusions after analyzing the previous results.

5.2 Background

According to the author in [55], one formation control strategy is defined using a reference path that is passed to the robots beforehand or by inferring it from the leader. Each agent compares this path with its relative position and the desired position to obtain the longitudinal and lateral error that are used to correct the current path. This approach is presented as a non-AI solution and the paper shows a detailed mathematical model to run the formation controller on the group of robots.

In one recent article [58], the author presents a multi-robot system inside a multi-target environment. The paper describes two navigation strategies: path planning with intermediate way points that are chosen based on the sensor range of the robot and a combination of bee colony for path planning and collision avoidance method

using evolutionary programming to smooth and optimize the intermediate positions. The author concludes that this merged strategy outperformed other methods for path planning like Particle Swarm Optimization, A* Algorithm, and Genetic Algorithm that were also tested during these experiments.

On a similar note, the authors in [59] present a robotic system that implements a somewhat different approach to anticipation for an indoor environment representing a supermarket with multiple aisles. Four types of strategies are used on the robots for navigation: *random*, that indicates that every time that an obstacle is detected, the robot receives a new command with a random direction; *reactive*, that states that the priority of robots is to take always the shortest path to the goal; *planning*, where it uses an A* based navigation to drive around the map and reach the goal "optimally"; and *anticipation*, that is an expansion of the planning method and a robot considers the other robots as stationary objects to move around. There is a central system that uses a camera over the environment to detect all robots. Based on this information, the system calculates the location of each of the robots and a suggested path. Then a command is sent to each robot that allows them to move through the environment. Although the experiment in this paper is described using two robots, the authors mention that this model can be easily expanded to multiple robots because of the capacity of the central system to detect more robots and transmit more commands. This is a good example of a central system creating anticipation for multiple agents while using global information. In contrast our approach is to create a decentralized system where independent agent can create their own decisions while using local information.

The authors in [60] present a flexible simulator for robot navigation that also includes crowd simulation. They state that even for simulation environments, it is necessary to implement accurate strategies to avoid collisions. This means that a robot must be aware, at least, of its immediate surroundings. For their particular

research, the accuracy of the robot location can also be flexible, which means that a strategy for obstacle avoidance and a strategy for robot navigation can run in parallel and independently. The accuracy that they refer to in [60] is more related to the decision of when to turn to avoid other robots or people.

The approach that is presented in this chapter continues with the idea that was presented in 3. Previously, the simulation was run using only two robots. In this chapter, we increase the number of robots around the Leader Robot and the goal of the Follower Robots is to stay in formation, i.e. close to the leader, based on the information that the lead robot is sharing. The robots use the same navigation strategy that was defined in Section 3.4. This strategy is a combination of controllers for long and short distance navigation that use GPS, compass, and a camera for blob detection as sensors. The anticipation behaviors are used on the Follower Robots as an aide to help them stay synchronized with the behavior of the Leader Robot. There are five anticipation models that were tested in this chapter. We perform a similar analysis of the results but we also considered the number of collisions that the robots have between each other under poor conditions with and without anticipation. This additional data on collisions helps us to evaluate the success of the robots during all the missions.

5.3 Methods

The idea of expanding the model described in Chapter 3 to a model that includes multiple followers is based on the statement in [53] that indicates that a Multiple Robot System (MRS) would have more advantages than a Single Robot System while trying to solve a complex problem. Some of these advantages are:

- Better spatial distribution.
- Better overall system performance (it can be defined in terms of energy con-

sumption or total time to complete the task).

- Robustness and fault-tolerance is easily enforced by data fusion and information sharing.

- Lower cost: simple and cheaper robots can be better than one complex and expensive one.

- Reliability, flexibility, scalability, and versatility through cooperation where robots can be categorized as heterogeneous (robots are different and can be specialized in different tasks) or homogeneous (capabilities of the robots are identical).

A group of robots can display a collective behavior, which is defined as a response of a group of individuals to a common influence [53]. Collective behavior includes what is known as cooperative and competitive behaviors. While cooperative behavior refers robots communicating between each other to work together for a common goal, competitive behavior refers to a robot trying to win over the others to fulfill it own goal. In this research the goal is to accomplish cooperative behaviors.

To make an MRS work cooperatively one of the key features that must be addressed is coordination, which can be defined as either *static* or *dynamic*. Static coordination normally involves some preliminary convention or rules that are set on the robots before the task starts. Dynamic coordination is a behavior that occurs while the task is being executed and it is defined based on the type of communication that is established between the robots. This communication can be defined as *explicit* or *implicit*. Explicit communication refers to a direct link that is established between two or more robots as a means to share information directly. On the other hand, implicit communication refers to the way robots share information by using the environment. For implicit communication robots are normally equipped with different types of sensors that allow them extract this information from the environment

correctly. [53]

An MRS can also be defined based on the decision-making method, which is classified as *centralized* or *decentralized*. A centralized decision-making system consists of one main robot that has a global view of the environment and makes a decision for every robot that is connected to the system. In a decentralized decision-making system, there is no central robot with a global view. Instead, every robot in the system generates its own decision using the local information that is available; including the information it receives from other robots. [53]

Based on these definitions, the MRS that is used in this chapter will contain the following features: *cooperative behavior*, *dynamic coordination*, *explicit communication*, and *decentralized decision-making*.

The main objective is to build a fleet of ground robots that will use the GPS and compass sensors that were described in Chapter 3 to help them move long distances and a camera with a blob detection method running on the robot for more accurate movement over short distances. In addition, the robots will use their current location and the influence of the group to calculate a new goal location. This group behavior can be described as a simple approach to swarm behavior for navigation where an agent is aware of the position of its neighbour only. But just by using and sharing local information, it is possible to generate a group formation as an emergent behavior at the same time the robots try to finish the task.

The experiments were run under the similar conditions that were described for the first part of this research in Section 3.4. In this case, the data that will be used to analyze the experiments are mainly the location data and total number of collisions that the Follower Robots have during the entire mission. The combination of these two variable can show additional information about the effectiveness of the navigation and anticipation strategies for each robot. Not only the path that a robot can follow is important, we are also interested in comparing the behavior of a robot during

different attempts of the same mission. For this reason, we are not including a direct strategy for collision avoidance. We want to let the robots have the chance to bump into each other and determine if anticipation can help the robots to reorganize on time before reaching their destination. We hypothesize that an emergent collective behavior might exist after implementing anticipation in a group of robots and we will analyze this by looking at the data in the next section. The number of collision will be tracked under optimal conditions and we will compare this value with the number of collisions under noisy conditions as an additional metric (along with measuring coordination in reaching the goal) to compare the performance of the robots with and without anticipation.

In order to have comparable data, we will use the same distribution of waypoints as in Chapter 3 (Figure 3.12). For this simulation, we followed the same combination of parameters that were described in Section 3.4 that involves the type of mission for Leader Robot, the message frequency of the Leader Robot, the anticipation model selected for the follower Robots and the use of Feedback on the anticipation module. We ran ten trails for every combination of these parameters and we collected data at the moment the robots cross a waypoint. In total, we collected 2760 samples after running all possible combination of these input parameters.

5.4 Results

In order to compare the data with research done in Chapter 3, we take a similar approach for these experiments and we add an additional robot on the right side of the leader. The idea is that the groups of robot must drive in parallel trough a set of waypoints. When the first robot reaches on of the waypoints, we collect the information of the current remaining distance of of all robot to that waypoint. Because they are driving in parallel, these values can give us an idea of how close or separate they are from each other. In an environment where the Leader Robot can

Table 5.1: Average and standard deviation of the absolute value of the meeting point error of three robots for the first of two waypoints of one mission while while the Leader Robot runs three behaviors (OnS - On Schedule, Slow, and Fast). The results represent simulations using No Anticipation (NA) and the five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value). The absolute value of the error represents the average distance of the robots as a group. For example, for the Slow behavior with 6 time steps per message (6S/M) and without anticipation, the average absolute value of distance difference between the Leader Robot and the Follower Robots is 8.8987 when they reach the first target point; with anticipation using no feedback and FL, the average is 4.8174; and with anticipation using feedback and FL, the difference is 6.2859. Best values are highlighted.

Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	1.6113(0.9936)	N/A	N/A	N/A	N/A	N/A
	2 S/M	3.0289(1.4346)	1.1372(0.9405)	1.1857(0.6398)	0.8169(0.6701)	1.4241(0.8419)	2.6824(1.1293)
			1.0396(0.7139)	0.9961(0.7935)	2.5153(0.7761)	1.7791(0.5243)	2.7290(1.2202)
	6 S/M	4.0870(1.5675)	1.8824(1.1205)	2.0402(0.9804)	1.4302(1.0862)	1.6977(1.1380)	1.8871(0.9244)
			1.9000(0.9649)	2.1266(1.0211)	2.7577(1.1541)	1.9995(1.2189)	2.9423(0.7296)
Slow	1 S/M	3.2305(0.3786)	N/A	N/A	N/A	N/A	N/A
	2 S/M	7.2620(0.6587)	4.3314(0.5309)	5.8688(0.7411)	5.4242(0.7657)	6.9318(0.6775)	6.2594(0.8219)
			2.4276(0.7144)	5.8436(0.6629)	4.9388(1.3878)	7.5492(1.0161)	6.7889(0.8543)
	6 S/M	8.8987(0.7904)	4.8174(1.3280)	6.7465(0.9345)	7.1897(0.7947)	6.5634(1.0836)	6.6680(1.0434)
			6.2859(0.9739)	6.9524(1.0656)	6.2260(1.2159)	8.1532(0.8354)	7.4292(0.6091)
Fast	1 S/M	1.8966(0.8791)	N/A	N/A	N/A	N/A	N/A
	2 S/M	2.4101(1.1022)	2.2487(0.8403)	2.2664(0.8183)	2.2574(0.2844)	1.9149(1.1325)	1.9619(0.9856)
			2.1400(0.8396)	2.3025(0.4443)	2.2867(0.9355)	2.5261(1.0108)	2.2965(0.8119)
	6 S/M	2.8735(0.7759)	2.3747(0.9156)	2.6135(0.77153)	2.3010(0.9737)	2.2779(1.0837)	2.2286(1.3096)
			1.5304(0.6562)	2.6431(0.8289)	2.5798(1.1703)	2.5070(1.1680)	2.6924(0.6591)

change its behavior but messages are getting lost, if the Follower Robots can navigate through the map and stay close to the Leader by using their own assumptions, we can consider that the Robots kept their synchronization by using anticipation. Keeping a formation with more robots restrains the freedom of some of the robots which will affect their behavior. We are also testing and observing how well the robots can recover from collisions between each other. Similar to the experiments in 3, we tested the robots in a mission that includes two waypoints and the results are split in two tables (Table 3.5 and Table 5.2); one for each waypoint. We also kept track the

Table 5.2: Average and standard deviation of the absolute value of the meeting point error of three robots for the second and last waypoint for a mission while the Leader Robot runs three behaviors (OnS - On Schedule, Slow and Fast). The results represent simulations using No Anticipation (NA) and five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value). Best values are highlighted.

Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	1.7845(1.0238)	N/A	N/A	N/A	N/A	N/A
	2 S/M	3.9885(2.6279)	1.7808(1.0774)	1.6828(1.0244)	1.6023(1.2096)	3.4929(1.6470)	3.1307(1.4421)
			2.2957(1.5753)	1.5782(0.8910)	2.7577(0.9617)	2.0456(1.2972)	1.8740(1.2482)
	6 S/M	4.4347(2.9216)	3.7961(1.3429)	1.6233(1.2575)	1.7823(1.1752)	4.2203(2.0902)	2.5565(1.5891)
			3.9949(1.3967)	2.4441(1.3430)	1.4438(1.1784)	1.5240(1.0894)	1.9331(1.5809)
Slow	1 S/M	1.0839(0.7437)	N/A	N/A	N/A	N/A	N/A
	2 S/M	8.1649(4.7548)	2.6944(1.7213)	7.7480(2.5256)	7.3725(1.2897)	12.1196(1.9120)	3.5873(1.4416)
			2.3105(1.0779)	6.4803(1.5195)	4.4307(1.5919)	8.7832(2.5488)	2.0289(1.4318)
	6 S/M	18.6021(2.0593)	1.9815(1.0005)	10.3927(3.7102)	2.5128(1.4656)	5.1654(1.8338)	4.0878(2.4461)
			2.3982(1.4426)	10.4327(1.0754)	6.5289(1.5282)	9.4779(5.3510)	7.3773(1.3240)
Fast	1 S/M	1.9104(1.2293)	N/A	N/A	N/A	N/A	N/A
	2 S/M	3.7720(1.7592)	2.8757(1.3325)	3.5706(1.1964)	3.4956(1.1354)	9.5391(1.4602)	8.6450(1.7501)
			2.7181(1.6956)	3.6440(1.1593)	3.5456(1.0840)	9.3079(2.6282)	9.3437(1.5777)
	6 S/M	9.3441(1.9015)	3.5104(1.9994)	8.7695(1.7425)	8.7883(0.9530)	9.6384(2.7591)	9.7643(2.4281)
			5.6833(1.8390)	8.7705(1.8256)	8.9498(0.9367)	10.7761(2.7483)	11.3682(1.4551)

average and standard deviation of the number of collisions that the robots had, as a group, during each experiment. These results are collected in Table 5.3. Furthermore, we noticed that there were some tests configurations where there was a high number of cases with zero collision against a few cases with a high number of collisions. For this reason, it was important to create an additional table to have a better view of how the data is distributed for every test configuration. Table 5.4 shows the results for the total number of collision and the number of cases in which these collisions occurred for all the trials that were run using a specific test configuration. It is important tho highlight that we run 10 trials for each configuration but we have two robots keeping track of the number of collisions. For this reason, we count 20 combined samples for every test configurations.

The data in Table 5.1 show that, for the first waypoint, all the anticipation models

could reduce the distance error between the Follower Robots and the Leader Robot. These results show a similar behavior that we saw in Chapter 3 for a pair of robots driving to the first waypoint. The first part of the mission start with the robots driving exactly in parallel without an external influence that could cause them to divert from that straight-line path. No collision were present and we observed that this helped to keep the robots close to each other for all mission. This is a similar behavior that we observed on the experiments with the indoor robots for my Masters research. For this first waypoint, the anticipation model using Fuzzy Logic showed the best results from all 5 models by obtaining 6 out of 12 of the lowest average values for the first part of the mission; especially with the Leader behaviors outside the training data. Also, the ARX Neural Network model trained with a Genetic Algorithm (A_NNGA) performed better than the other models while running the behavior that was included in the training data.

Table 5.2 shows the average and standard deviation of the absolute value of the distance error between the Leader Robot and the Follower Robots for the second part of the mission before the reach the second waypoint and complete the mission. In this case, some anticipation models found more difficult to reduce the distance error considerably but it was noticeable the effect of the anticipation models on reducing the distance error between the Leader and Follower Robots. Just like the results in Chapter 3, the ARMAX models were not as successful as the other anticipation model for the cases where the Leader Robot was running behaviors that were not part of the training data. This shows a more consistent behavior that these configuration of ARMAX models show a much stable and successful behavior when the information is closely related to the training data. In general, the anticipation model using Fuzzy Logic had more cases with the lowest values (7 out of 12) while compared to the results from the other anticipation models.

Also, we kept track of the number collisions that occurred between the robots

Table 5.3: Average and standard deviation of the number of collisions between the three robots for the second and last waypoint for a mission while the Leader Robot runs three behaviors (OnS - On Schedule, Slow and Fast). The results represent simulations using No Anticipation (NA) and five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value). The highlighted values in grey color represent the anticipation model with the lowest standard deviation for the number of collisions for a particular test configuration (row). The highlighted values in green color represents the test configurations with the best results in Table 5.2 as an approach to correlate the data between both tables. A blue color represents a case that matches both conditions.

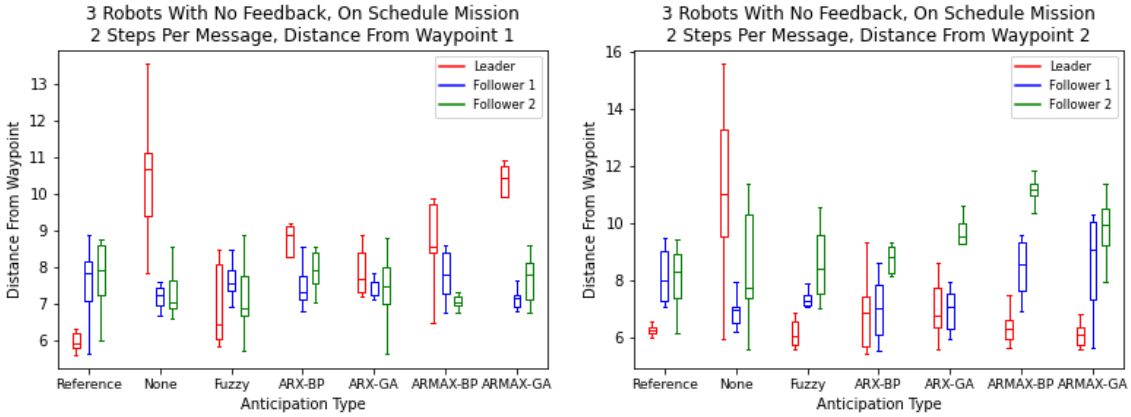
Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	1.900(2.6338)	N/A	N/A	N/A	N/A	N/A
	2 S/M	1.2500(2.4682)	0.9444(1.4337)	2.2222(2.3403)	1.2778(1.6735)	1.1667(1.6539)	1.5555(1.7896)
			1.6667(2.7008)	1.0556(1.3048)	1.6500(2.1831)	0.7778(1.5925)	0.7778(1.0603)
	6 S/M	1.3000(2.1545)	1.4500(1.9324)	1.4000(2.3930)	1.5500(2.1392)	1.9000(1.8890)	2.2500(1.6819)
			1.1000(1.2937)	2.0000(2.0520)	1.4000(1.7592)	0.7500(1.2513)	1.1500(1.4244)
Slow	1 S/M	1.4500(1.7614)	N/A	N/A	N/A	N/A	N/A
	2 S/M	1.5000(2.9155)	0.3000(0.8013)	0.6000(1.8468)	0.0500(0.2236)	0(0)	0(0)
			3.6500(3.0826)	0.6000(1.3534)	0.1000(0.4472)	0.4000(1.2312)	0(0)
	6 S/M	0(0)	0.5000(1.1002)	0.5000(1.5390)	0.5000(1.5728)	0(0)	0.3500(1.0894)
			0.9500(1.9050)	0.3500(1.0894)	0(0)	1.3500(2.3232)	0(0)
Fast	1 S/M	0.6000(1.6983)	N/A	N/A	N/A	N/A	N/A
	2 S/M	1.2778(2.4925)	0.9500(1.8202)	0.5000(1.1471)	1.2500(2.4894)	1.9500(2.1145)	0.5000(1.15045)
			0.1667(0.7071)	0.6111(1.5005)	0.5000(1.5435)	0(0)	0.7222(1.4061)
	6 S/M	0.5500(1.2763)	1.8000(2.3530)	0.6500(1.6631)	0.4500(1.1459)	0.7500(1.9433)	0.3000(0.8013)
			0.1500(0.4894)	0.3000(1.3416)	0.2500(0.7163)	0.3000(0.9234)	0.4500(1.1459)

on each trial. The robots always started facing at the first waypoint and this initial position and orientation make the robots to avoid collisions at least for the first section of the mission. For Tables 5.3 and 5.4, the data represents the collisions that the robots encountered once they reached the first waypoint. This means that these two new tables are directly related to Table 5.2. Table 5.3 shows the average and standard deviation of the collisions that occurred while the robots were driving in-between the first and second waypoints. Before calculating these values, we run 10 trials for every test configuration but we collected the collision data from two robots which gives us 20 samples for every test configuration. At first sight, it is easier to

Table 5.4: Total numbers of collisions and sum of the cases when the collisions occur between the three robots for the second and last waypoint for a mission while the Leader Robot runs three behaviors (OnS - On Schedule, Slow and Fast). The results represent simulations using No Anticipation (NA) and five anticipation models: Fuzzy Logic (FL), ARX Neural Network trained with Backpropagation (A_NNBP), ARX Neural Network with trained with a Genetic Algorithm (A_NNGA), ARMAX Neural Network trained with Backpropagation (AM_NNBP) and ARMAX Neural Network with trained with a Genetic Algorithm (AM_NNGA). Each mission is run with three different message frequencies: 1 message every time step (1S/M), one message every two time steps (2S/M) and one message every 6 time steps (6S/M). In addition, each anticipation model is run with two different input structures: without feedback (top value) and with feedback (bottom value). The highlighted values in grey color represent the anticipation model with the highest number of collisions for that particular test configuration (row). The highlighted values in green color represents the test configurations with the best results in Table 5.2 as an approach to correlate the data between both tables. A blue color represents a case that matches both conditions

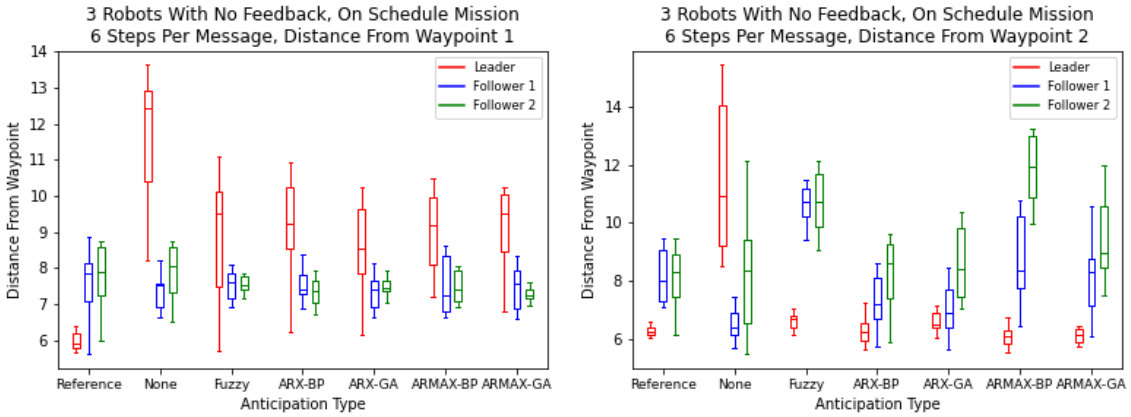
Test \ Model		***NA***	***FL***	***A_NNBP***	***A_NNGA***	***AM_NNBP***	***AM_NNGA***
OnS	1 S/M	38(8)	N/A	N/A	N/A	N/A	N/A
	2 S/M	25(5)	17(6)	40(10)	23(8)	21(7)	28(9)
			30(6)	19(8)	33(9)	14(5)	14(7)
	6 S/M	26(7)	29(9)	28(6)	31(9)	38(12)	45(14)
			22(9)	40(12)	28(9)	15(6)	23(9)
Slow	1 S/M	29(9)	N/A	N/A	N/A	N/A	N/A
	2 S/M	27(4)	6(3)	12(2)	1(1)	0(0)	0(0)
			64(14)	12(4)	2(1)	8(2)	0(0)
	6 S/M	0(0)	10(4)	10(2)	10(2)	0(0)	7(2)
			19(5)	7(2)	0(0)	23(6)	0(0)
Fast	1 S/M	12(3)	N/A	N/A	N/A	N/A	N/A
	2 S/M	23(5)	19(5)	10(4)	25(6)	39(10)	9(4)
			3(1)	11(3)	9(2)	0(0)	13(4)
	6 S/M	11(4)	36(9)	13(3)	9(3)	15(3)	6(3)
			3(2)	6(1)	5(3)	6(2)	9(3)

notice that the cases that have an average value of zero (0) match the cases in Table 5.2 where the cases were the least successful in reducing the distance error. This happened mainly in a few cases while the Leader used a Slow speed behavior and the Follower Robots used the ARMAX models. For these few cases, the Follower Robots drove always right behind the Leader Robot which avoid causing collisions between them. The gray highlighted cells in Table 5.3 represent the anticipation models with the smallest standard deviation for a specific test configuration which is defined by



(a) Waypoint 1 (b) Waypoint 2

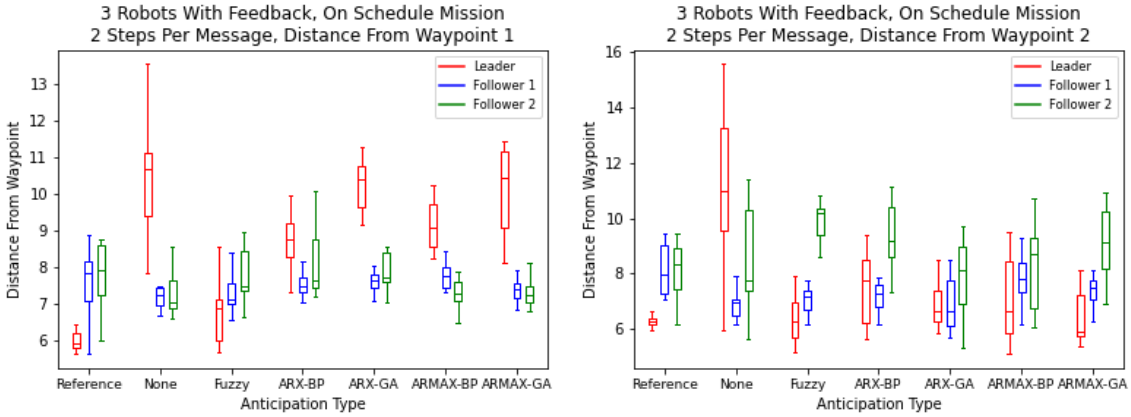
Figure 5.1: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the On Schedule behavior, a message frequency of 2 steps per message and anticipation with no feedback



(a) Waypoint 1 (b) Waypoint 2

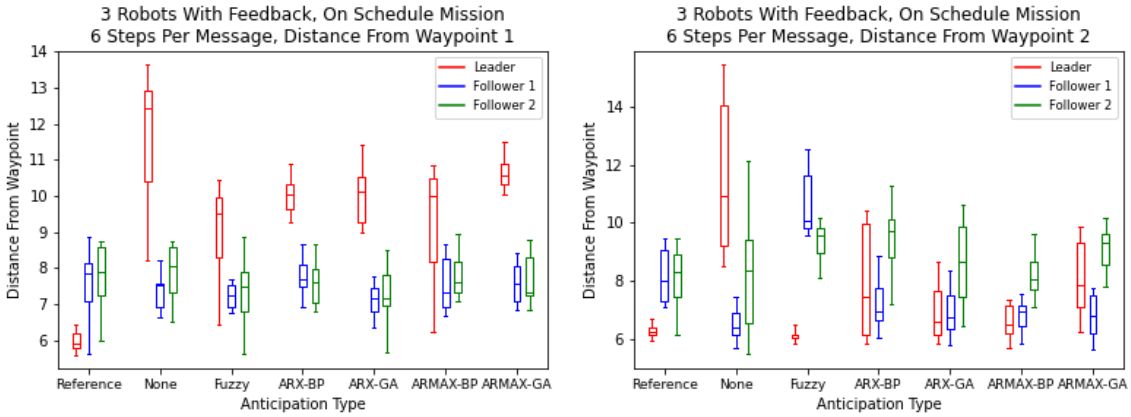
Figure 5.2: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the On Schedule behavior, a message frequency of 6 steps per message and anticipation with no feedback

every row in the table. For this table, a high standard deviation means that the collisions were not evenly distributed through all the test samples. For this particular situation, we noticed that some test configurations had some few cases with a high number of collision while the majority of cases did not present a collision. In contrast, small standard deviation values mean that the number collisions were more constant and evenly distributed throughout every test of that test configuration case. The cases with smallest values are highlighted in green in Table 5.3.



(a) Waypoint 1 (b) Waypoint 2

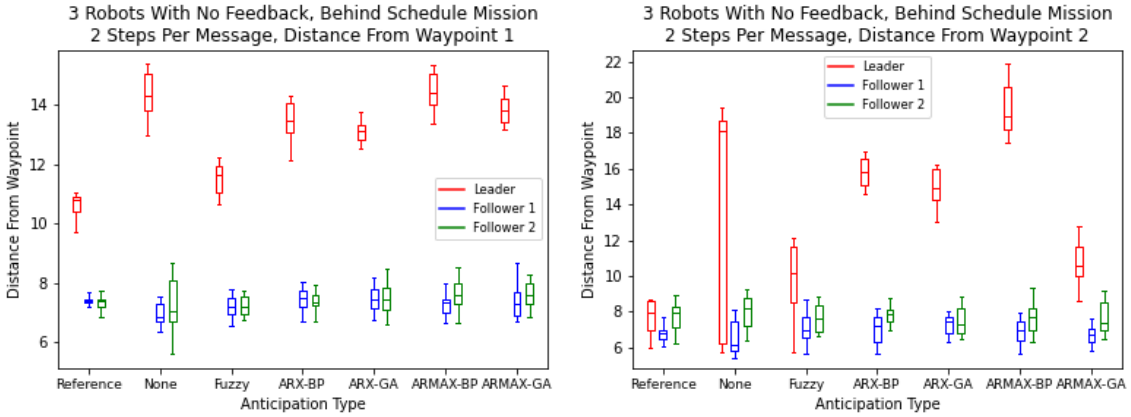
Figure 5.3: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the On Schedule behavior, a message frequency of 2 steps per message and anticipation with feedback enabled



(a) Waypoint 1 (b) Waypoint 2

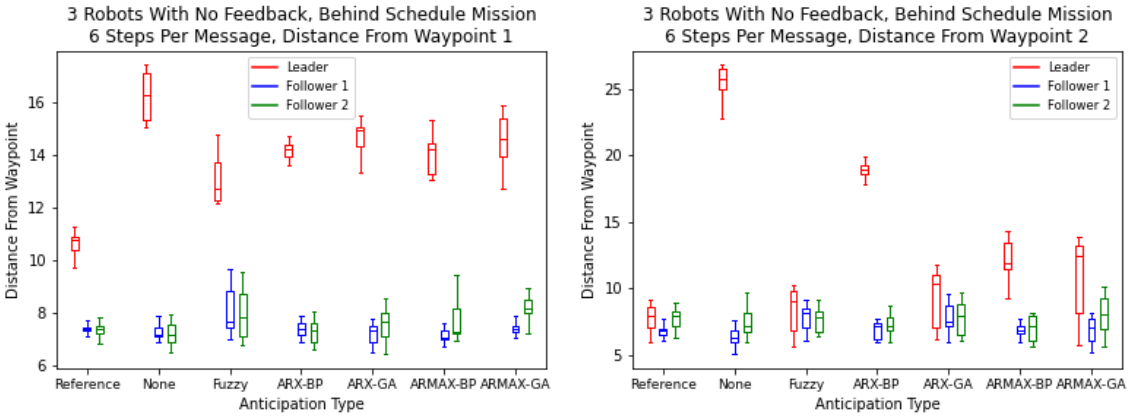
Figure 5.4: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the On Schedule behavior, a message frequency of 6 steps per message and anticipation with feedback enabled

In addition, the results in Table 5.4 shows the total number of collision that the robots had during each trial for every test configuration and, inside the parenthesis, it shows the number of combined cases between the Follower Robots and for a specific test configuration where collision were identified. Just like in Table 5.3, we highlighted with green color the cells that showed the best results for the distance error in Table 5.2. while analyzing these data, we can identify that the best results for the distance error are not related to high or small number of collisions; the best results are found



(a) Waypoint 1 (b) Waypoint 2

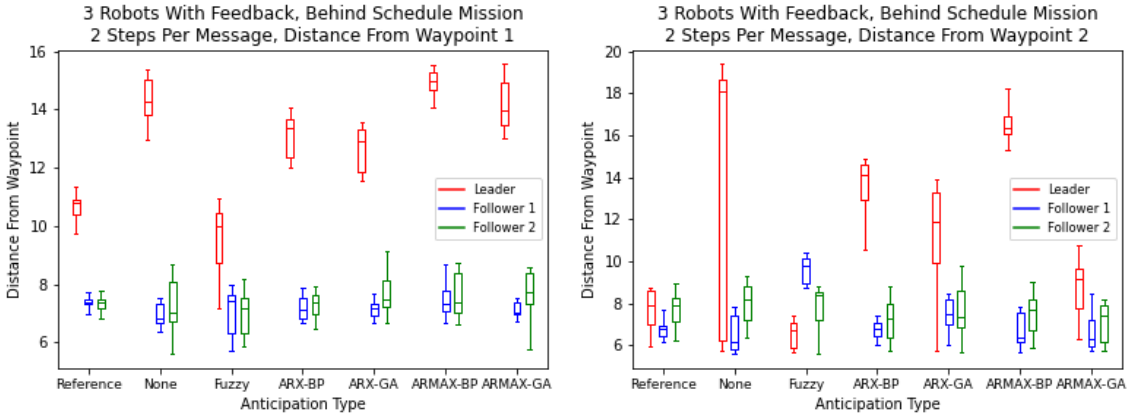
Figure 5.5: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Slow behavior, a message frequency of 2 steps per message and anticipation with no feedback



(a) Waypoint 1 (b) Waypoint 2

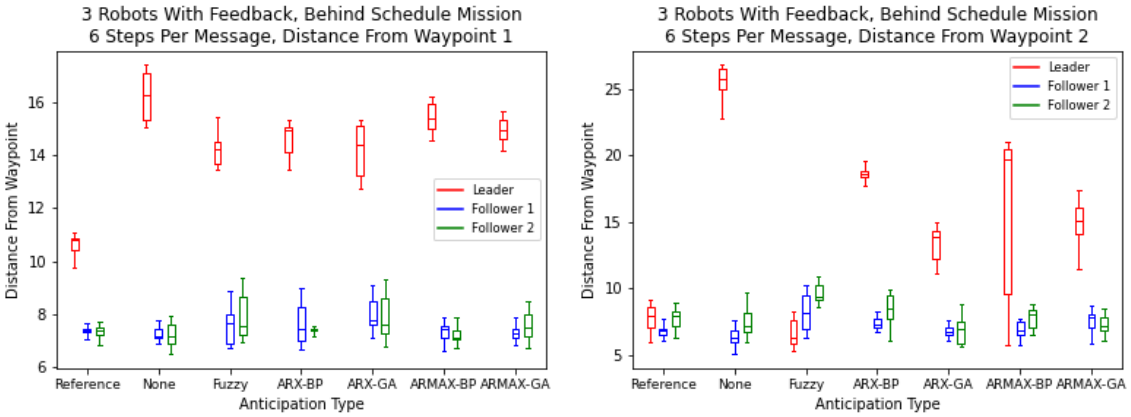
Figure 5.6: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Slow behavior, a message frequency of 6 steps per message and anticipation with no feedback

in the middle of these values for almost every case. This observation helps to explain that given no collision, it is most likely that the Follower Robots were not driving fully in parallel with the Leader Robot and were not close enough to generate a collision. Also, a high number of collision can affect the Robots in a way that their Physics reaction is to slow down every time and prevents the Robots to get back in formation even with the constant help of the anticipation models. Based on this analysis, we noticed that best results for the distance error in Table 5.2 are related



(a) Waypoint 1 (b) Waypoint 2

Figure 5.7: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Slow behavior, a message frequency of 2 steps per message and anticipation with feedback enabled

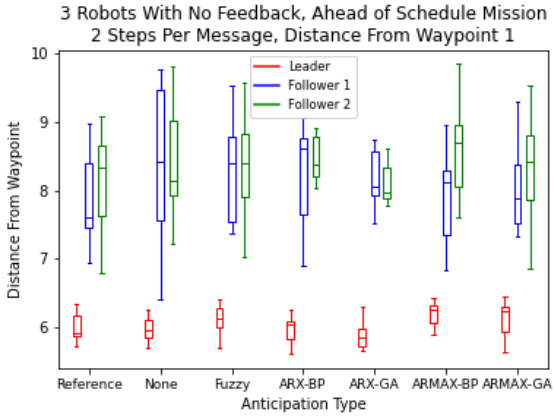


(a) Waypoint 1 (b) Waypoint 2

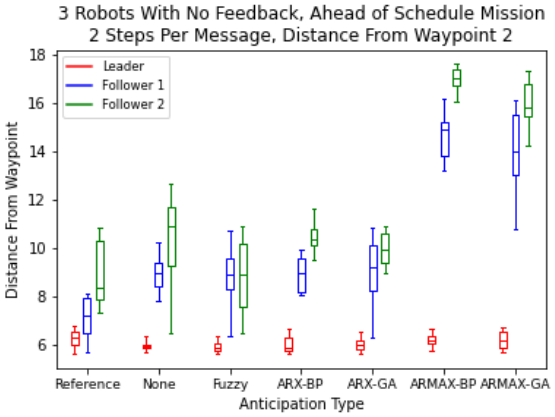
Figure 5.8: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Slow behavior, a message frequency of 6 steps per message and anticipation with feedback enabled

to the data located in the middle of the distribution of the collision values. We must remember that the priority is no to implement a collision controller running in parallel of the navigation controller. Our goal is to study the effects of this collisions on the effectiveness of the anticipation models over the Follower navigation controller.

just how presented the data in the figures in Chapter 3, the figures in this section represent the the distance of the robots to the immediate waypoint when the first robot reaches this waypoint. We present 24 figures that show all the test configura-

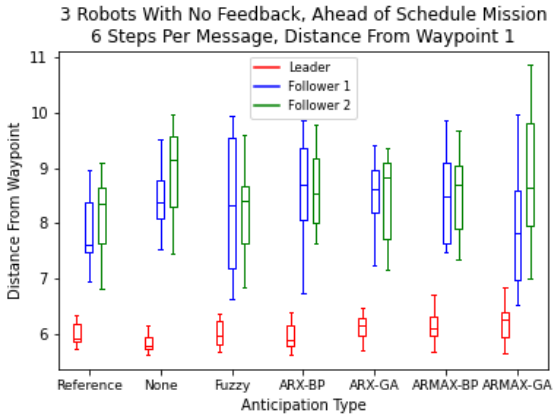


(a) Waypoint 1

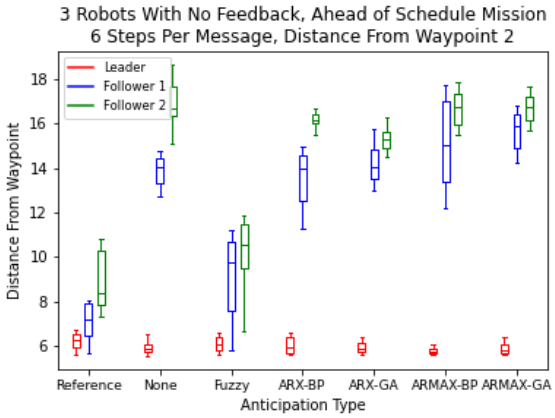


(b) Waypoint 2

Figure 5.9: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Fast behavior, a message frequency of 2 steps per message and anticipation with no feedback



(a) Waypoint 1

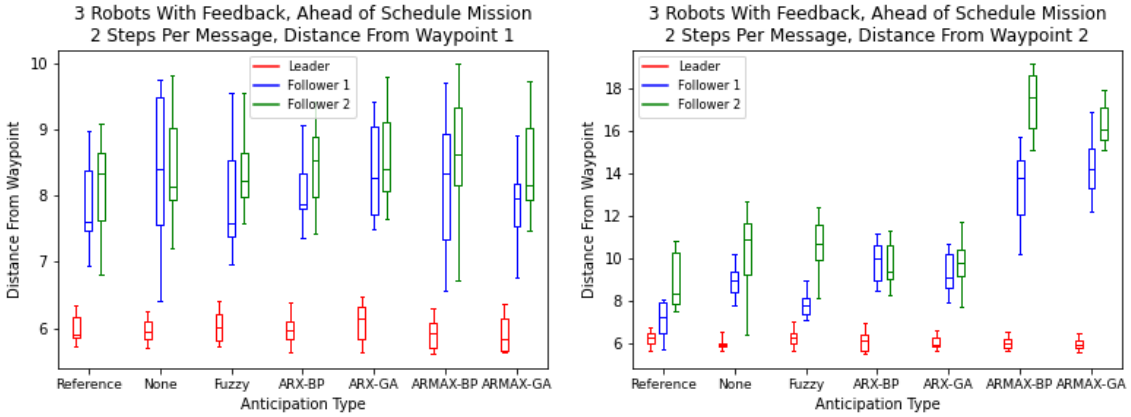


(b) Waypoint 2

Figure 5.10: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Fast behavior, a message frequency of 6 steps per message and anticipation with no feedback

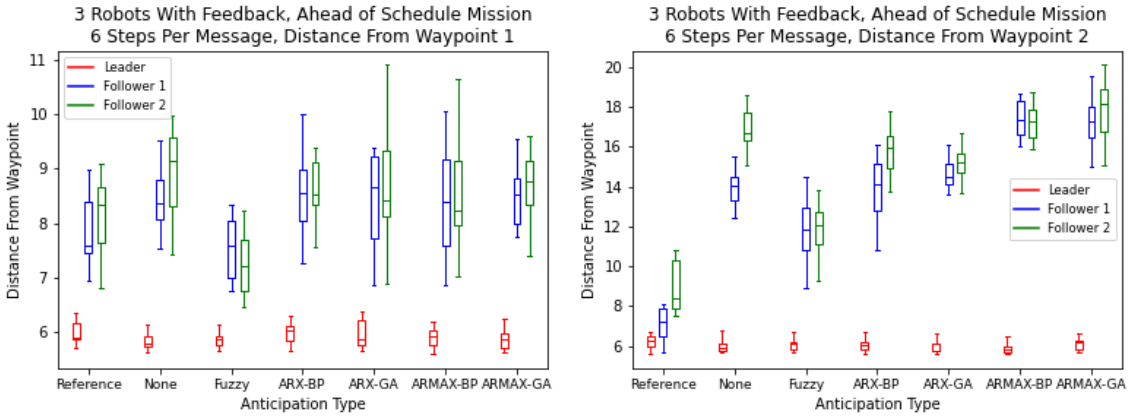
tions that were presented in Tables 3.5 and Table 3.6. These figures are organized in small groups of 4 figures based on the behavior of the Leader Robot and the configuration of the message queue:

- On schedule and No feedback (Figures 5.1 and 5.2)
- On schedule and With feedback (Figures 5.3 and 5.4)
- Slow (Behind Schedule) and No feedback (Figures 5.5 and 5.6)



(a) Waypoint 1 (b) Waypoint 2

Figure 5.11: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Fast behavior, a message frequency of 2 steps per message and anticipation with feedback enabled



(a) Waypoint 1 (b) Waypoint 2

Figure 5.12: Distance error at the moment a waypoint is reached by the Leader Robot and two Follower Robots for the Fast behavior, a message frequency of 2 steps per message and anticipation with feedback enabled

- Slow (Behind Schedule) and With feedback (Figures 5.7 and 5.8)
- Fast (Ahead of Schedule) and No feedback (Figures 5.9 and 5.10)
- Fast (Ahead of Schedule) and With feedback (Figures 5.11 and 5.12)

Each figure comes with 2 subfigures that shows the results for the two waypoints that were used in the mission for these experiments. Each subfigure shows 7 sets of data where each group shows the results for both the Leader Robot and the Follower

Robot. The first set is defined as the reference and it shows the results of the robots driving on optimal conditions for that configuration; this means using a message frequency of 1 Step per Message (1S/M). The other sets in the figure show the results at a different frequency (2S/M or 6S/M). The second set shows the results for the configuration when no anticipation is used at the specified frequency. And, the next 5 sets show the results for the all 5 anticipation models that we are studying; FL, A_NNBP, A_NNGA, AM_NNBP and AM_NNGA respectively.

5.5 Conclusions

For this section we analyzed the data in a similar way as it was presented in Chapter 3. The difference is that the data is presented as the average of the absolute value of the distance error between both Follower Robots and the Leader Robot. This approach allows to have a better idea of their performance as a group. First, we focused on the behavior of the robots as a group during the first waypoint. Because of their initial position and orientation, the behavior was fairly similar to the results for two robots in Chapter 3. No collision were detected during this first part of the mission and the all the anticipation model reduced the distance error between the Follower Robots and the Leader Robot without noticeable problems.

For the second section of the mission, some anticipation models showed some struggle while trying to infer the missing information from the Leader Robot. In a similar way, the ARMAX and the ARX models showed good results for the On Schedule behavior by reducing the distance error between the Follower Robots and the Leader Robot. For the Slow and Fast behaviors, The ARX models were able to solve all the test cases but it was clear that, as a group, the distance error showed a higher standard deviation than the results with only two robots. We also identify some generalization behaviors from the ARX models by comparing these results with the No Anticipation data where the ARX models could improve/reduce the error

in the cases that were not included in the training data. In addition, the ARMAX models struggled on more cases while trying to solve the Slow and Fast behaviors. In fact, in a few of these cases, the distance error was not improved/reduced which brings the conclusion of ARMAX not being a good model for generalization.

Also, we analyzed the influence of the collisions on the decisions of the anticipation models and on the behavior of the robots. This unit of measure gave us a good idea of how close the robots were from each other and how well they can react to additional noise before they reached the final waypoint. By comparing the numbers collected in Table 5.3 and Table 5.4 with the behavior that was observed during the experiments, cases with zero collisions normally represent the Leader Robot or The group of Follower Robots staying behind from the other which give them enough horizontal distance to avoid collisions but it also means that they could not synchronize completely and aligned themselves to continuing driving in parallel. On the other side of the spectrum, cases with a very high number of collisions performed poorly as well. The analysis of these data showed that a lot of collisions affected the speed of the robot(s) staying behind. The constant collisions did not allow the robots to recover on time before reaching the final waypoint. The best successful cases were found close to the average value of the collisions. These value varies depending on the speed of the Leader Robot; slow speeds will create higher average values than higher speed values.

In a general overview, the FL was the models with best results from all of the combined tests by getting the smallest distance error values for 13. The A_NNGA, A_NNBP and AM_NNGA models had 5, 3 and 2 cases with the best smallest distance error respectively. And the least successful anticipation model was the AM_NNBP model with just 1 case.

Chapter 6: Conclusions

We showed in this research that anticipation strategies can be a very powerful tool when multiple agents try to communicate between each other in very noisy environment. We used anticipation as a tool to aid a group of robots to keep their synchronization under control when their communication fluctuates and messages get lost. When the receiver agents detect that a message never arrived at the designated timestamp, they decide to run their anticipation modules independently which creates an estimated progress message that contains an approximate information from the original message. Then, this new message is used to replace the missing message in the regular process of the navigation controller. This means that the anticipation module is not used as a replacement of the navigation system but it is used an aide for the navigation controller in order to keep its input information updated when necessary.

By creating the environment for Robot navigation in Unity for Chapter 3 and Chapter 5, we were able to simulate some behaviors that we could study with the real ground robots during field testing thanks to the physics libraries that are part of this software. This noisy environment was used to create the Leader messages in the training data for the neural network models. As we saw in Table 3.3, the messages that are generated by the Leader Robot are also affected by the environment and some membership values inside the message present clear evidence of noise. Clearly, this was the biggest challenge for the neural networks to solve. In contrast, the Fuzzy Logic model showed the most amount of cases with the smallest error. I hypothesise that this happens because this model did not go through a preliminary training process that included noisy training data; like the Neural Networks models. Instead, it used a set of fixed rules to analyze the input messages. If we consider that we always keep the same five of memberships inside message and the values in these memberships just shift based on the changes of speed of the Leader agent, it might be possible

that the anticipation models can be trained with data that was generated in a non-noisy environment to obtained models with a better fitness. Then, the generated anticipation model can be imported to the noisy and compared if they can perform better. We leave for a next stage of this research in the future.

We presented, in Chapter 4, an approach to be used as a tool to recover commands from corrupted voice messages that are sent to a group of robots. The goal of this approach is to work as a complement for the anticipation module because the experiments with the anticipation module, normally, assume corrupted messages as another missing message. The anticipation module does not analyzes the context of a corrupted message; it focuses on creating an estimated message by keeping track of the timing of the incoming messages and the immediate previous messages. For this reason, we were interested in studying and creating a system capable of detecting cases when a corrupted message is received and letting an AI module try to analyze it by its phonetic content. This means that it checks if the input words sounds familiar to the words in its dictionary and then, it checks if these words are being used in the right order based on the knowledge it has about the user command list. From all the corrupted messages where homophone cases were encountered, this approach was successful in fixing all the corrupted cases and identifying the right user command in the message.

In total, five anticipation models were designed and tested in order to analyze and compare their performance by using a group of ground robots. One Fuzzy Logic model and Four different Neural Network models and were used as anticipation models. The Neural Network models used the same internal configuration for the hidden layer. This was based on the results that we obtained the previous work for my Masters thesis where the best results for anticipation came from a system using two previous messages and 10 neurons in the hidden layer. We use this as a base configuration to test a combination of two different training methods (Backpropagation and Genetic

Algorithms) and 2 different input structures (ARX and ARMAX). Additionally, we kept the idea that we proposed for the message update method that was presented before: anticipation without feedback and anticipation with feedback. The difference between these two methods is how the history of messages is updated for the inputs. A No-Feedback method uses the real Leader messages only while the Feedback method uses the previous anticipated message to updated the history of messages and uses it as inputs. We also generated three different behaviors for Leader robot to test response of the anticipation module by selecting all possible combinations of these methods and models. These Leader behaviors represent changes in speed that and some of these behaviors were selected to train the neural networks and others for testing only. Finally, we also introduce a variable to change the frequency the messages are sent to the Follower Robots. This method is used to simulate the messages that get lost while the robots are navigating which alters the response and behavior of the Follower Robots.

The results in Chapter 3 and Chapter 5 showed that it was very difficult for some anticipation model to match the reference formation at 1 step per message; especially for the mission using behaviors outside the training data. This is mainly because of the nature of the Leader messages that are being transmitted. This simulation is using physics libraries that imitate how friction and inertia would affect the robots. This something that was reflected in the messages that the Leader Robot transmitted. As Table 3.3, the messages from the Leader Robot includes notable noise but this is something that we should expect in a real environment. Based on these results, we believe that the next step for this research is to explore more complex structures of Neural Networks that improve the behavior of the Robots but, at the same time, calculate the nee message in the same amount of time available. Recurrent Neural Networks can be the next strategy to try because of their bidirectional connectivity which might help in a time-series problem.

The model that was presented in Chapter 4 introduces an AI approach to message correction based on phonetic analysis that can solve homophone problems. This approach makes it easier to add new commands without changing the base of the code because these commands are interpreted as Fuzzy rules; we would need to add an additional rule to the list of Fuzzy rules defined for the Fuzzy Logic model. A future study for this work would be to find an way to make these rules more flexible, for example, based on possible swapped positions of the in the sentence and how far they are from their valid position. Sometimes people would say phrases where some words are note properly used in the right position of the sentence but other people receiving this information can infer the context of the original message by identifying some words that become key for the structure of the phrase.

References

[1] J. Marulanda, D. Edwards, R. Heckendorn, and T. Soule, "Learned anticipation strategy for speed control in an auv fleet," in *2013 OCEANS-San Diego*. IEEE, 2013, pp. 1–9.

[2] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Theraulaz *et al.*, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.

[3] K. M. Passino, *Biomimicry for optimization, control, and automation*. Springer Science & Business Media, 2005.

[4] M. J. Krieger, J.-B. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, vol. 406, no. 6799, p. 992, 2000.

[5] J. Marulanda, W. Alfonso, and E. Caicedo, "Competitive multi-swarm system in adaptive resource allocation for a multi-process system," *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 66, pp. 168–180, 2013.

[6] N. J. Hallin, B. Johnson, H. Egbo, M. O'Rourke, and D. Edwards, "Using language-centered intelligence to optimize mine-like object inspections for a fleet of autonomous underwater vehicles," *UUST'09*, 2009.

[7] N. Hallin, B. Johnson, M. O'Rourke, and D. Edwards, "A fuzzy logic resource optimizer for a fleet of autonomous vehicles in low-bandwidth conditions," in *OCEANS 2009-EUROPE*. IEEE, 2009, pp. 1–8.

[8] W. Christensen and C. Hooker, "Anticipation in autonomous systems: foundations for a theory of embodied agents," *Int J Comput Anticip Syst*, vol. 5, pp. 135–154, 2000.

[9] A. Gorbenko and V. Popov, "The force law design of artificial physics optimization for robot anticipation of motion," *Advanced Studies in Theoretical Physics*, vol. 6, no. 13, pp. 625–628, 2012.

[10] ——, "Anticipation in simple robot navigation and learning of effects of robot's actions and changes of the environment," *International Journal of Mathematical Analysis*, vol. 6, no. 55, pp. 2747–2751, 2012.

[11] M. M. Waldrop, "Autonomous vehicles: No drivers required," *Nature News*, vol. 518, no. 7537, p. 20, 2015.

[12] R. B. Hughes, "The autonomous vehicle revolution and the global commons," *SAIS Review of International Affairs*, vol. 36, no. 2, pp. 41–56, 2016.

[13] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. Mutz *et al.*, "Self-driving cars: A survey," *arXiv preprint arXiv:1901.04407*, 2019.

[14] M. Maurette, "Mars rover autonomous navigation," *Autonomous Robots*, vol. 14, no. 2-3, pp. 199–208, 2003.

[15] M. Santora, J. Alberts, and D. Edwards, "Control of underwater autonomous vehicles using neural networks," in *OCEANS 2006*. IEEE, 2006, pp. 1–5.

[16] J. Stilgoe, "Machine learning, social learning and the governance of self-driving cars," *Social studies of science*, vol. 48, no. 1, pp. 25–56, 2018.

[17] E. North, J. Georgy, M. Tarbouchi, U. Iqbal, and A. Noureldin, "Enhanced mobile robot outdoor localization using ins/gps integration," in *Computer Engineering & Systems, 2009. ICCES 2009. International Conference on*. IEEE, 2009, pp. 127–132.

[18] N. Hallin, H. Egbo, P. Ray, T. Soule, M. O'Rourke, and D. Edwards, "Enabling autonomous underwater vehicles to reason hypothetically," in *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*. IEEE, 2009, pp. 1–7.

[19] N. Hallin, B. Johnson, H. Egbo, M. O'Rourke, T. Soule, and D. Edwards, "Simulating human reasoning in mine-like object inspection assignments for a formation of unmanned underwater vehicles," in *Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL'09.* IEEE, 2009, pp. 158–162.

[20] A. Rajala, M. O'Rourke, and D. Edwards, "Auvish: An application-based language for cooperating auvs," in *OCEANS 2006*. IEEE, 2006, pp. 1–6.

[21] A. Rajala and D. Edwards, "Allocating auvs for mine map development in mcm," in *OCEANS 2006 - Asia Pacific*. IEEE, 2007.

[22] B. Johnson, N. Hallin, H. Leidenfrost, M. O'Rourke, and D. Edwards, "Collaborative mapping with autonomous underwater vehicles in low-bandwidth conditions," in *OCEANS 2009-EUROPE*. IEEE, 2009, pp. 1–7.

[23] B. Gill, J. F. M. Arias, D. Edwards, and T. Soule, "Natural language-based correction method for autonomous underwater vehicle communications," in *2013 OCEANS-San Diego*. IEEE, 2013, pp. 1–10.

[24] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization," *Communications Magazine, IEEE*, vol. 47, no. 1, pp. 84–89, 2009.

[25] A. K. Morozov, T. W. Altshuler, C. P. Jones, L. E. Freitag, P. A. Koski, and S. Singh, "Underwater acoustic technologies for long-range navigation and communications in the arctic," in *Proc. 4th International Conf. Underwater Acoustic Measurements: Technologies and Results*, 2011, pp. 1119–1126.

[26] J. F. Marulanda, D. B. Edwards, R. B. Heckendorn, and T. Soule, "Learned anticipation strategy on complex behaviors and as an approach to generalization behavior for the coordination of an auv fleet," in *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018, pp. 1–9.

[27] N. Hallin, J. Horn, H. Taheri, M. O'Rourke, and D. Edwards, "Message anticipation applied to collaborating unmanned underwater vehicles," in *OCEANS 2011*. IEEE, 2011, pp. 1–10.

[28] H. Brument, L. Podkosova, H. Kaufmann, A. H. Olivier, and F. Argelaguet, "Virtual vs. physical navigation in vr: Study of gaze and body segments temporal reorientation behaviour," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2019, pp. 680–689.

[29] J. Hyun, M. J. Sliwinski, and J. M. Smyth, "Waking up on the wrong side of the bed: The effects of stress anticipation on working memory in daily life," *The Journals of Gerontology: Series B*, vol. 74, no. 1, pp. 38–46, 2019.

[30] E. Mastrantuono, D. Saldaña, and I. R. Rodríguez-Ortiz, "Inferencing in deaf adolescents during sign-supported speech comprehension," *Discourse Processes*, vol. 56, no. 4, pp. 363–383, 2019.

[31] S. Satake, T. Kanda, D. F. Glas, M. Imai, H. Ishiguro, and N. Hagita, "A robot that approaches pedestrians," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 508–524, 2012.

[32] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2015.

[33] T. Vintr, Z. Yan, T. Duckett, and T. Krajník, "Spatio-temporal representation for long-term anticipation of human presence in service robotics," in *2019 In-*

ternational Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 2620–2626.

[34] A. F. Winfield, V. V. Hafner, and R. Poli, "Anticipation in robotics," *Handbook of Anticipation: Theoretical and Applied Aspects of the Use of Future in Decision Making*, pp. 1–30, 2018.

[35] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy *et al.*, "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 60–71, 2013.

[36] J. Dedek, M. Golembiovsky, and Z. Slanina, "Sensoric system for navigation of swarm robotics platform," in *2017 18th International Carpathian Control Conference (ICCC)*. IEEE, 2017, pp. 429–433.

[37] F. Piltan, S. TayebiHaghighi, and N. B. Sulaiman, "Comparative study between arx and armax system identification," *International Journal of Intelligent Systems and Applications (IJISA)*, vol. 9, no. 2, pp. 25–34, 2017.

[38] P. Urcola, M.-T. Lorente, J. L. Villarroel, and L. Montano, "Robust navigation and seamless localization for carlike robots in indoor-outdoor environments," *Journal of Field Robotics*, vol. 34, no. 4, pp. 704–735, 2017.

[39] C. Toth and G. Jóźków, "Remote sensing platforms and sensors: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 22–36, 2016.

[40] C. Gomez, A. Hernandez, J. Crespo, and R. Barber, "Uncertainty-based localization in a topological robot navigation system," in *Autonomous Robot Systems and Competitions (ICARSC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 67–72.

[41] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[42] V. Schmuck and D. Meredith, "Training networks separately on static and dynamic obstacles improves collision avoidance during indoor robot navigation," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine LearningEuropean Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Ciaco-i6doc. com, 2019, pp. 655–660.

[43] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," *Journal of robotic systems*, vol. 14, no. 4, pp. 231–249, 1997.

[44] F. Fell and M. Tanenbaum, "Preliminary comparisons of the wgs84 (egm 96) geoid with national vertical datums," in *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No. 01CH37295)*, vol. 1. IEEE, 2001, pp. 571–574.

[45] P. R. Norvig and S. A. Intelligence, *A modern approach*. Prentice Hall, 2002.

[46] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[47] J. F. Marulanda, M. G. Barco, and J. A. López, "Intelligent control of a chemical reactor (spanish)," *Grupo de percepción y sistemas inteligentes. Universidad del Valle. Cali, Valle, Colombia*, 2007.

[48] C. Smiley and S. Kübler, "Native language identification using phonetic algorithms," in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 2017, pp. 405–412.

[49] X. Lv, M. Zhang, and H. Li, "Robot control based on voice command," in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*. IEEE, 2008, pp. 2490–2494.

[50] Y.-C. Ju, D. Ollason, and S. Bhatia, "Homonym processing in the context of voice-activated command systems," Feb. 20 2007, uS Patent 7,181,387.

[51] X. Zang, M. Vázquez, J. C. Niebles, A. Soto, and S. Savarese, "Behavioral indoor navigation with natural language directions," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 283–284.

[52] G. Hoffman, "Anticipation in human-robot interaction," in *AAAI Spring Symposium: It's All in the Timing*, 2010.

[53] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[54] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*. Springer, 2004, pp. 1–9.

[55] E. Sahin and A. F. Winfield, "Special issue on swarm robotics." *Swarm Intelligence*, vol. 2, no. 2-4, pp. 69–72, 2008.

[56] A. Guillet, R. Lenain, B. Thuilot, and P. Martinet, "Adaptable robot formation control: adaptive and predictive formation control of autonomous vehicles," *IEEE Robotics & Automation Magazine*, vol. 21, no. 1, pp. 28–39, 2014.

[57] J. Gregoire, M. Čáp, and E. Frazzoli, "Locally-optimal multi-robot navigation under delaying disturbances using homotopy constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 895–907, 2018.

[58] A. Q. Faridi, S. Sharma, A. Shukla, R. Tiwari, and J. Dhar, "Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment," *Intelligent Service Robotics*, vol. 11, no. 2, pp. 171–186, 2018.

[59] B. Johansson and C. Balkenius, "An experimental study of anticipation in simple robot navigation," in *Workshop on Anticipatory Behavior in Adaptive Learning Systems*. Springer, 2006, pp. 365–378.

[60] A. Aroor, S. L. Esptein, and R. Korpan, "Mengeros: A crowd simulation tool for autonomous robot navigation," in *2017 AAAI Fall Symposium Series*, 2017.