Instituto Universitário de Lisboa

Department of Information Science and Technology

Control of Robot Swarms Through Natural Language Dialogue

A Case Study on Monitoring Fires

Nuno Miguel Amorim Mendonça

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Engineering

Supervisor

Sancho Moura Oliveira, Assistant Professor, Ph.D.

ISCTE-IUL

Co-Supervisor

Ricardo Daniel Santos Faro Marques Ribeiro, Assistant Professor, Ph.D.

ISCTE-IUL

October, 2019

“Have the courage to follow your heart and intuition.

They somehow know what you truly want to become.”

- Steve Jobs

Abstract

There are numerous environmental and non-environmental disasters happening throughout the world, representing a big danger to common people, community helpers, to the fauna and flora. Developing a program capable of controlling swarms of robots, using natural language processing (NLP) and further on, a speech to text system, will enable a more mobile solution, with no need for keyboard and mouse or a mobile device for operating with the robots. Using a well-developed NLP system will allow the program to understand natural language-based interactions, making this system able to be used in different contexts. In firefighting, the use of robots, more specifically drones, enables new ways to obtain reliable information that before was based on guesses or knowledge from someone who had long-time experience on field. Using a swarm of robots to monitor fire enables innumerous advantages, from the creation of a dynamic fire map, climate information inside the fire, to finding lost firefighters on field through the generated map. This work uses firefighting as a case-study, but other situations can be considered, like searching someone in the sea or searching for toxins in an open environmental area.

Keywords: Natural Language Processing, Swarm robotics, Natural Language Understanding, Multi-robot Cooperative Control

Resumo

Existem muitos desastres ambientais e não ambientais em todo o mundo, representando um grande perigo para pessoas comuns, ajudantes da comunidade e para a fauna e flora. O desenvolvimento de um programa capaz de controlar enxames de robôs, usando Processamento Computacional da Língua (PCL) e, posteriormente, um sistema de fala-para-texto, permitirá uma solução mais móvel, sem necessidade de teclado e rato ou dispositivos móveis para operar com os robôs. O uso de um sistema bem desenvolvido de PCL permitirá que o programa entenda interações baseadas em linguagem natural, tornando-o capaz de ser usado em diferentes contextos. O uso de robôs (mais especificamente drones) no combate a incêndios, permite novas maneiras de obter informações confiáveis que antes eram baseadas em suposições ou conhecimentos de pessoas com longa experiência em campo. O uso de um enxame de robôs para monitorizar o incêndio permite inúmeras vantagens, desde a criação de um mapa dinâmico do incêndio, informações climáticas dentro do mesmo, até encontrar bombeiros perdidos no campo, através do mapa gerado pelos robôs. Este trabalho usa o combate a incêndios como um estudo de caso, mas outras situações podem ser consideradas, como procurar alguém no mar ou procurar toxinas numa área ambiental aberta.

Palavras-chave: Processamento Computacional da Língua, Robótica de Enxame, Compreensão de Língua Natural, Controlo Cooperativo de Vários Robôs

Acknowledgements

I would like to thank my Dad, Mom, Brother, Godmother and Grandmother, for the unconditional support throughout all these years, not only from my bachelor's degree to the process of researching and writing this thesis, but for every moment since I was born. All my accomplishments wouldn't happen without you! Thank You! You will always have my very profound and eternal gratitude!

I would like to thank all my close friends Diogo Baptista, Rodrigo Gomes, Carina Abreu and João Pedro Borges for all the support you have given and still give me every day! I know I can always count on you, as you can always count on me! Also, to my big friends, that I met on my master's degree, André Oliveira, Pedro Camacho and Ruben Ribeiro, that always pushed me further to do more and gave me strength to make this work! And all my other friends who participated in this journey! Without you, it just wouldn't be the same!

I would like to thank my supervisor Professor Sancho Moura Oliveira of ISCTE-IUL and my co-supervisor Professor Ricardo Ribeiro of ISCTE-IUL, for their continuous support and counseling throughout the whole process of making this research, from the very beginning, where nothing existed, to the very end. I am gratefully indebted for your very valuable inputs and discussions on this thesis.

I would also like to express my gratitude to IT (Instituto de Telecomunicações), where they gave me a space to work, with excellent conditions, on this research and to my university ISCTE-IUL for all the amazing years, there is not a moment I want to erase from my memory! Thank you for everything!

Contents

List of Figures

List of Tables

Abbreviations

compound	Noun Compound Modifier. 36, 37
dobj	Direct Object. 35, 37, 39
GUI	Graphical User Interface. 4, 17
NLP	Natural Language Processing. 3, 4, 6, 17, 19, 33, 40, 42, 45, 54, 55
NLU	Natural Language Understanding. v, 33, 38, 39, 40, 41
nsubj	Nominal Subject. 35, 36, 37, 39, 41
nummod	Numeric Modifier. 36, 41

Chapter 1

Introduction

1.1 Motivation

Robots nowadays are essential, not only to perform tasks that are dangerous or nearly impossible for human beings, like working in deep ocean, but also to do simple day-to-day tasks such as vacuuming the floor. But tasks are not the focus of everything, controlling the robots brings another complexity to the table, specially swarms [Velagapudi et al., 2008]. Traditionally, the interaction between the user and the robots is accomplished either by a mobile touch device or a keyboard and mouse. Using voice commands allows more freedom to the user, like when it is mandatory to wear gloves at the work place, the user does not need to remove them every time an input is needed. They also make multiple tasks much easier to achieve, as long as the interaction and cognitive capabilities are well developed [Briggs and Scheutz, 2012]. Voice inputs, compared to manual inputs, will enable better performance [Draper et al., 2003], especially if multiple robots / agents are being used in the field [Velagapudi et al., 2008]. The whole system developed for this thesis was designed to be open enough to cover different situations and to be natural language friendly. Not only is it easy to add and remove features, but also to be adaptable to different scenarios. The robots are programmed to monitor the zone regardless of the mission, as long as it is on an environmental open area. For example, this can be used to search people in the sea, by changing the environmental nodes from fire to water, or one can use this system to search for toxins on a certain area, by changing the water to toxins.

1.2 Context

Due to the climate change and global warming, the weather has been changing its standards in a rapid pace and in a more aggressive way year by year [Mora et al., 2013][Dey et al., 2015]. Firefighting has been gaining more importance given the increasing occurrence and severity of fires and casualties over the years [Qian et al., 2008]. According to the European Commission in the annual report on forest fires in 2017 [Jesús San-Miguel-Ayanz, 2018], Portugal was the country with the most fires and the second largest area burned with 21,002 fires and 540,630 hectares, respectively. Regarding fatalities the report shows 114 casualties: 66 in the fires of June 2017 (65 civilians and a firefighter), 46 in October 2017, plus a helicopter pilot and bulldozer operator. In addition, firefighters who work directly in the field face immense dangers, from being surrounded in the fire to spatial disorientation. By using robots during these disasters, firefighters have more information about the area surrounding them due to the continuous monitoring of the location of the fires and their sizes.

1.3 Research Questions

There were three research questions designed to conduct this study and they are as follows:

- Is it possible to develop a System capable of controlling swarms of robots, using Natural Language and to adapt this system to different situations?
- Can the Natural Language Processing System understand different ways of saying the same thing, as long as, it is on the context of the system?

- Can the Natural Language Processing System perceive context?

1.4 Objectives

The main goal of this thesis is to develop a generic and open system to monitor something using robots, in a field on an open space. The way to control the robots working in the field is through natural language by written text, but changing to a speech based instance should not be a problem, as there are multiple libraries that can do this with ease. The system should be able to understand the user no matter how (s)he writes. This system must be ready to be used by non-technological experts, and by being so, it must be able to understand natural language sentences and take action on real time. Although the framework of this thesis is firefighting monitoring, the system is open enough to work for multiple purposes, from hard tasks such as firefighting monitoring to monitoring the ocean. To demonstrate the feasibility of such system, a software simulation must be developed with multiple robots communicating with each other, mapping the fire in a virtual map. The Natural Language Processing (NLP) System should understand whatever the user says and finally, it should be able to map people in the dynamic map, as well as cars. Finally, the whole system should be ready to be linked to real-life robots, making this whole experience possible in real life situations.

1.5 Research Methodology

The Research Methodology of this work is the Design Science Research Methodology Process Model. In this methodology it is necessary to identify six activities which are as follows:

- 1 - Problem identification and motivation

  Controlling multiple robots on a mission, at the same time can become chaotic and pointless since, with a lot of robots being operated, people tend to neglect some of them either entirely or after an initial movement [Velagapudi et al., 2008].

Using mouse and keyboard to operate can also be a problem, since there are a lot of things happening at the same time. Selecting a group of robots to do one thing, then selecting another can become exhausting with a keyboard and mouse, after repeating it a lot of times.

- 2 - Define the objectives for a solution

  The objectives for the solution developed were to design an open system, capable of adapting to different missions, from monitoring fire to monitor the sea. Everything needs to be simple enough to the user since the system can be used by non-tech experts. More about the objectives can be found on section 1.4.

- 3 - Design and development

  This artifact will have two main components: The Robotic System and the Natural Language Processing (NLP) System. The Robotic system will be the one responsible to represent all the robots and calculate the next position of each robot in the field, as well as their representation on the Graphical User Interface (GUI). The NLP System will be responsible for understanding what the user is writing on the chat system and take all the meaningful information out of it. The Graphical User Interface (GUI) must be really simple with three different compositions: the topology, meaning the map with every robot inside it, as well as other type of information, like people and cars. On the right of the topology there will be the chat, responsible to show the user what is happening on the field and allowing him or her to interact with the robots or the system in general.

- 4 - Demonstration

  For the purpose of this thesis, the demonstration will be made on a virtual basis, with the vision of bringing it to real life in future work.

- 5 - Evaluation

In order to evaluate this system, an inquiry was distributed to understand the different ways of communication considering people with different age, nationality, academic background, gender, etc... All tests made by the respondents were registered and tested on the system. There were some entries that result in errors or were ignored by the NLP System, meanwhile, they were solved and considered as successful. To evaluate the System different actions, formations and general commands were created. People were shown the different commands available and they were asked to try to say the command in the most natural way possible, like if the Commander was a person himself. People were also asked to mix actions and formations in the same sentence and to try to use the context system. The results were analyzed and conclusions were drawn.

- 6 - Communication

  After the submission of this thesis, an article will be prepared to present the achieved results to the scientific community.

Chapter 2

Related Work

To be able to understand what the scientific community has already done, a Related Work chapter has been made. Here, there will either be concepts that support the system architecture, later on section 3, or work that is related to this demonstration, from the robot system to the Natural Language Processing (NLP) System, to other projects that are similar to this one. Also, some basic concepts are written and they are not directly interconnected with the work, but, in their own way, they influenced the work.

2.1 Multi-Robot Communication

Nowadays, single robots are not enough to perform the daily dangerous tasks that is possible to find in our world. Compared with a single robot, multi-robot collaborative planning brings many advantages, specially when tasks can be broken, where each robot of the multi-robot squad performs a single task, speeding up the task execution efficiency. To achieve a successful interaction between robots [Guo et al., 2016], whenever a robot is going to perform a task, one robot needs to transmit the order to another robot, thus, making a multi-robot communication. This communication can be classified into two main categories: Point to point communication and point to multi-point communication [Eriksson E et al., 2015].

These two categories will be used to solve different problems on the mission, such as computer-robot control and teams of multiple robots information communication [Guo et al., 2016].

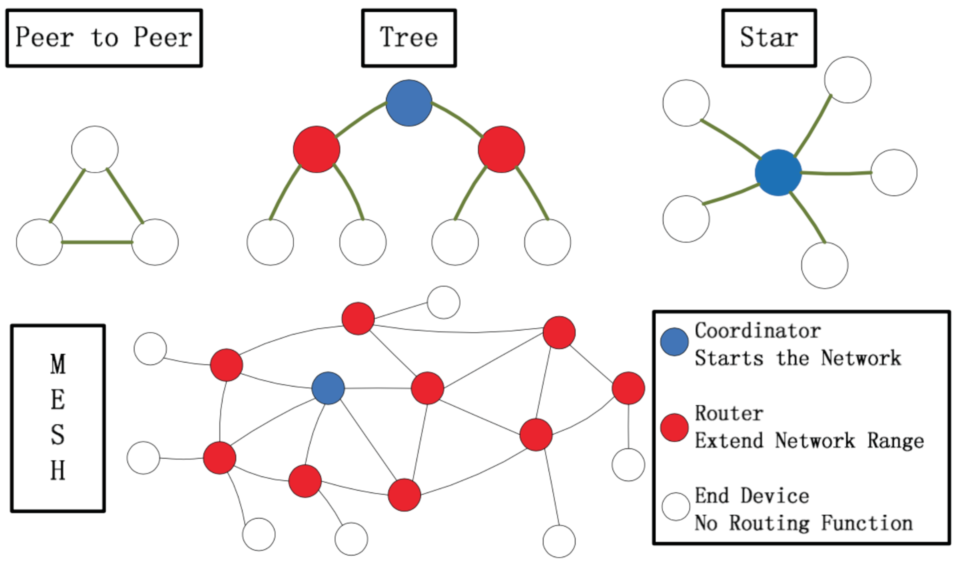


FIGURE 2.1: Different ways of spreading information for Multi-Robot cooperative control [Guo et al., 2016].

There are different ways of communication between devices, also known as nodes, as one can see in Fig. 2.1. Although this is about networking, the same applies to communication between robots [Guo et al., 2016]. Peer-to-peer communication, also known as, Point to Point, is where there is a direct communication between two nodes together, making the most simple communication since the data flows, unidirectionally or bidirectionally between two points [Ray, 2018]. The Tree topology network consists of having a central node, which is the root of the tree, having routers to extend the network coverage to all the end devices. These end devices can not possess children, only the router and the root can have them. The star topology consists of the root, also known as the Coordinator, the one that starts the network, having several end devices directly connected to him. This means that the end devices can only communicate with the coordinator. "The disadvantage of this topology is the operation of the network depends on the coordinator of the network, and because all packets between devices must go through the coordinator, the coordinator may become bottlenecked. Also, there is no alternative path from the source to the destination. The advantage of star topology is that it is simple and packets go through at most two hops to reach their destination" [Elahi and Gschwender, 2009]. Finally, one has the mesh network, which consists of one coordinator, having several routers, having several end devices. This network, besides being a resizable network, is a multi-hop network, this means that packets that travel through one robot to another need to pass through multiple nodes in order to reach their destination. If one path fails, the node will find an alternative to reach its destination. Although this topology brings a lot of advantages, it also brings the disadvantage of requiring greater overhead when compared to the previous one, the start topology, since mesh uses a more complex routing protocol than the star one [Elahi and Gschwender, 2009].

2.2 Belief Communication Between Agents

Natural language dialogue is the most used type of communication between humans if there is a task in common between them. Through perceived and communicated information, humans can predict the actions of their team mates based on their tasks or destinations. This same way of thinking can be applied nowadays to robot teams or swarms. With defined principles for belief modeling and updating for autonomous agents, spoken dialogue between humans and agents can be achieved. Different beliefs and intentions of other agents are used to create mental models that are rich enough to capture task-based aspects of other agents and their own beliefs. They can also update their beliefs on other agents by communication.

Explicit rules must be added to all this equation so it is possible to represent relationships among linguistic expressions as well as past and future beliefs. For task-based agents, there are rules that need to be implemented that allow agents to reason about the effects of perceptions, actions and past beliefs on new/updated ones and the effects of different utterance types [Briggs and Scheutz, 2012].

2.3 Interaction Between Humans and the Swarm

In order to establish a dialog platform to be able to cope with open domains considering the possible interactions between the embodied agent and humans, there must be a validation and interpretation of the natural language utterances provided to the system against the knowledge structures of an intelligent agent's mind. Although it requires high capability for describing language at multiple levels (morphosyntatic, syntactic, ontological), the algorithm described in [Ventura et al., 2012] can solve ambiguities and acquire knowledge about unknown entities.

The management of mobile service robot operating in different environments by different operators brings multiple conditions to the developed systems, such as [Drews and Fromm, 1997]:

- Fast, time deterministic reaction on commands of high priority;
- Constant interaction between the robot and the environment;
- Validation of vague geometrical information;
- Easy adaptation to the environment (especially concerning the objects, object activities and object status available within the operation area);
- Bi-directional communication between the user and the control module.

2.3.1 How Do They Interact?

As robots are used more and more throughout the years, users that are not experts are starting to find it easier to control them, through the different types of input, either by text, speech or mouse input. Unlike highly professional users that work with professional-related robots, the inexperienced users need an easy way of interaction. Since non-expert users are not familiar with Robot Command Language (RCL), it is necessary to build a system that translates natural language commands into RCL. Before translating the spoken sentences, it is necessary to understand them in the first place. This poses some difficulties like the acoustic recognition of the spoken word, also known as, word recognition and the extraction of the information contained within a frame of words, alias language processing [Drews and Fromm, 1997].

This is achieved by using a semantic parser. The system developed in the work [Thenmozhi et al., 2017], is able to get the natural language command from the user and convert it into RCL using tagging approach, implemented using a Hidden Markov Model approach. After tagging the command, the parser builds the RCL, which is then converted to configurations.

In the article [Rossi et al., 2017], it is studied how people interact with a group of robots, the vocabulary they use, and the multimodal interfaces for dynamic and interactive control with them. It is also possible to visualize how robots can deviate from certain obstacles, developing awareness of the space around them. With speech recordings of 41 people, 60% being males and 40% females aged between 24-60, with an average age of 28.9, interacting with a graphical interface showing different robots, spread throughout the map, people were asked to, through their native language (Italian), control the robots in the map. There were three settings shown to the participants with an increasing number of robots per setting: two, four and eight robots, and a varying spatial distributions generated randomly by the following Gestalt Principles:

- Proximity principle: elements are perceived as aggregated into groups if they are near each other;
- Similarity principle: elements tend to be integrated into groups if they are similar to each other;
- Continuity principle: oriented units tend to be integrated into perceptual wholes if they are aligned with each other;
- Closure principle: elements tend to be grouped together if they are parts of a known figure, even if the shape is not complete and closed.

Single Features	%	Combined Features	%
Name	2.6	Name/Categ + Physical	25.7
Category	7.7	Name/Category + Physical + Spatial	15.1
Physical	20.6	Name/Category + Spatial	10.1
Spatial	11.3	Physical + Spatial	2.4
Other	4.3		
Neg	0.2		

Table 2.1: Use percentage of single or combined features.

As shown in Table 2.1, the testers use the name of the robot 2.6% of the cases and the category, also known as robot class/type (drone, ground, robot), 7.7% of the cases. The most common preference was to call the robots by their physical characteristic, with 20.6% of cases. In 11.3% of the cases the participants used the spatial disposition as a way of communication, while 4.3% use some other ways of expressions that were not shown above. Finally 0.2% of the people used the negation to interact with the machines (for example: "all those that are not ..."). For those participants who decided to use combined features in the remaining sentences, the most used one was name/category with physical aspects, with 25.7% of the cases (for example: "the yellow drone"), while 15.1% used name/category + physical aspects and spatial disposition (for example: "The red Pioneer on the right"). 10.1% used name/category + spatial, and, finally, 2.4% used physical aspects alongside spatial disposition (for example: "the first red"). It is important to know how people tend to interact with individual robots or various groups of robots in order to properly prepare the system before its release.

Intelligent robots need to be able to understand natural language sentences in an efficient and accurate way, whether it is in a complex way or in an extremely simple way, like using keywords. Developing an agent capable of communicating with users through natural language while learning semantic meanings from conversations is the goal of this paper [Thomason et al., 2015]. The agent integrates a semantic parser, producing the logical form representations of user utterances with a dialog manager and maintaining a belief-state for the user's goal. When running the agent for the first time, a few training examples are run for the parser, inducing more during natural clarification dialogues with ordinary users. To understand new ways of saying things incrementally, whenever the agent understands the user's goal, it pairs the logical form representing that goal with previously misunderstood utterances in the conversation to form new training examples for the semantic parser.

2.3.2 Translation System for Natural Language to RCL

The system architecture presented in Fig. 2.2 is able to translate natural language commands into RCL [Thenmozhi et al., 2017]. This can be achieved by multiple components inside the whole system. "The trainer with the help of the Annotations.txt and the Commands.txt is able to produce the mapped lexicons and their frequencies with respect to the given command. These produce frequencies, lexicons and chunks that are passed on to the parser as inputs. We use production rules for reading the RCL from the Annotations.txt to determine the lexicons like colour, type, event etc. Using the grammar, the lexicons are mapped and tagged. That is, mapping the lexicons in the commands and the lexicons in the RCL. The chunker splits the commands into unigrams, bigrams, trigrams to find the probability of the occurrence in the commands.txt. Highest probability sequences are obtained by Hidden Markov Model (HMM) tagger. This tagger uses Viterbi algorithm to find the highest probability sequences." [Thenmozhi et al., 2017].

The scene manager uses the Configurations.txt in order to generate a random scene, giving it to the Parser. The spatial planner will generate a space with a size, for example $8 * 8 * 8$ with different objects. The Parser uses the outputs of the trainer and scene manager and generates a specific RCL command. The Move Validator validates if the generated RCL by the Parser is valid or not, using the random world generated by the Spatial Planner. "If the scene is valid, move validator returns the moves to provide the required scene and it converts the RCL commands to Configuration files with the help of the Configurations Generator"

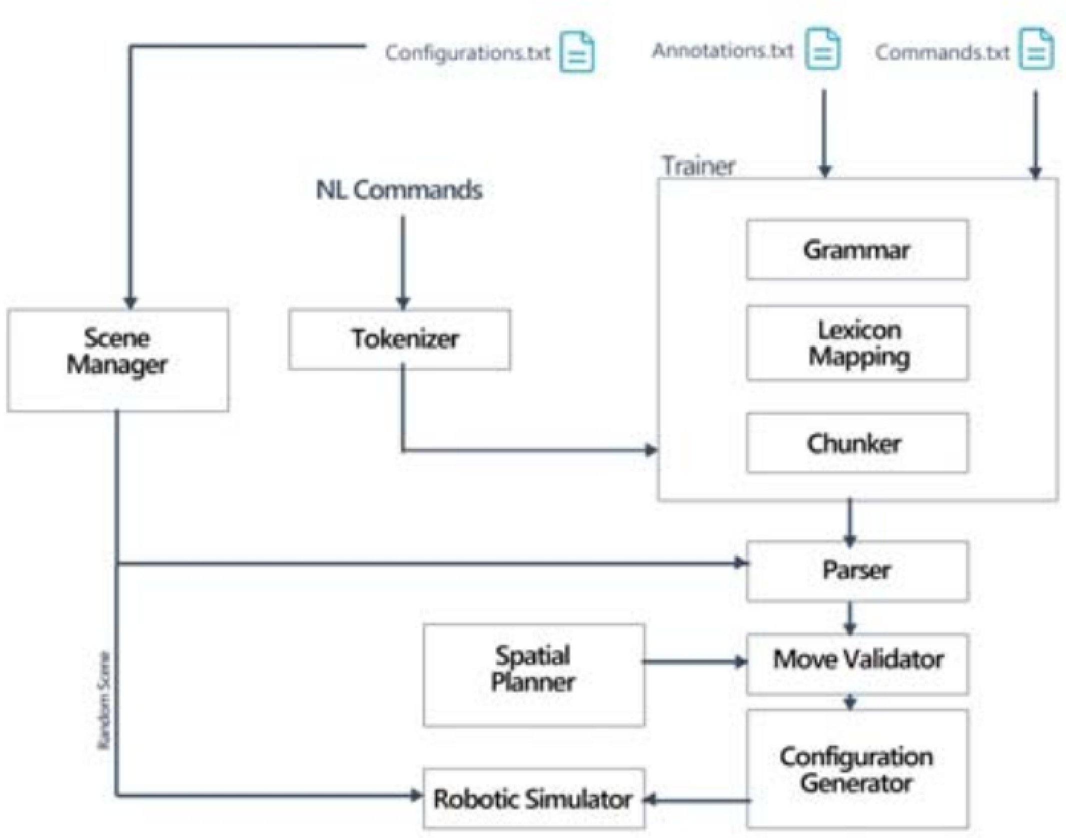


FIGURE 2.2: System Architecture for a robotic simulation [Thenmozhi et al., 2017].

[Thenmozhi et al., 2017]. Finally, the Robotic Simulator is able to generate a simulation onto the Graphical User Interface based on everything that was managed from the previous steps.

In order to test this system, the authors had a dataset used in SemEvel 2014 with all the .txt files to evaluate their system. The Commands.txt had 3409 commands including their annotated parse trees. Annotations.txt contained 3409 RCL for the 3409 commands, and, finally, Configurations.txt contained configurations for 1000 scenes (125 worlds).

As it can be observed in Table 2.2, the commands without the "and" connector reach a 96% accuracy, the commands with the "and" connector reach 50% accuracy, making the overall accuracy of 92.45%. The authors reached a conclusion out of this information that "once RCL is generated correctly for the command,

TABLE 2.2: Performance Evaluation

	Total Number of Statements	Number of Commands for which correct RCL was generated	Number of Commands for which RCL was not correct	Accuracy (%)
Statements	106	98	8	92,45%
Involving two commands Conjugated by "and"	10	5	5	50%
Commands without "and"	96	93	3	96%

the robotic simulator does the action accordingly" and that "Training with a large number of data may also increase the efficiency".

2.3.3 Commanding Multi-Robot Systems using Battle Management Language

Having multiple robots brings numerous new possibilities that would not otherwise be possible with only one robot. For example, using UAVs to produce aerial photos and UGV producing a 3D grid using laser scanners. In the work [Remmersmann et al., 2012], it is shown how it is possible for a single user to control a multi-robot system using Battle Management Language (BML), showing how quickly and efficiently the robots can be coordinated. A set of commands was defined and implemented to test their approach.

The communication between humans and robots is done in a way that one node, the master, receives the command from the user and breaks it down into sub-commands for all the slave robots. There are two different approaches for the Multi-Agent System (MAS) [Coppin and Legras, 2012], the first one is "control-by-behavior", the operator selects one of the agents. The problem of this first approach is that it is not scalable for larger groups, since there are more complex behaviors. The second approach is the "control-by-policy", where the operator defines constrains or advices in a limited natural language and the agent plans corresponding actions.

To understand what a person is saying in English, complex intelligent nodes are needed on the lead. From the Leader node to the slave nodes, BML is used since it is human readable and unambiguous. The BML can be used to express orders, reports and requests between command and control systems. The main problem of using BML is the translation between high level commands to basic orders to robots.

As a final product, the work [Remmersmann et al., 2012] presented a system capable of receiving commands in a restricted normal English to control a Multi-Robot System with BML. Giving the orders in natural language means having complex intelligent nodes, capable of understanding this type of language and translating higher level commands into basic orders to all the end nodes.

2.4 Robot Swarm to Generate a Dynamic Fire Map With Pheromone

Dealing with a swarm of robots makes it necessary to understand the scaling effects of multi-robot control. It is shown how the number of controlled robots in a realistic simulated environment can affect an urban search and rescue mission. The task performance increased in going from four to eight controlled robots but deteriorated in moving from eight to twelve. It is necessary to have a healthy ratio of robots and it is important to remember that more is not always better [Velagapudi et al., 2008].

In order to control a swarm of Unmanned Aerial Vehicles (UAVs) for the purpose of fire dynamic mapping, the paper [Howden, 2013] studies an algorithm based on distributed (inverted) pheromone onto grid maps to effectively track a moving fire front. The pheromone increases automatically during time by an amount that is proportional to the required survey frequency and is reset to zero when a robot visits a grid cell. The more pheromone available in a cell, the more attracted is the most nearby robot. Each UAV keeps its own internal map, and broadcasts it periodically when it resets a cell. Each cell on the grid map is initialized to the time when the mission began. Equation 2.1 shows the quantity of pheromone at each cell is the product of the cell's priority and time unobserved.

$$pheromone = priority \times \Delta t \tag{2.1}$$

$$f(x) = \frac{pheromone_x^2}{distance(self, x) + distance(A, x)} \tag{2.2}$$

By using Equation 2.2 it is possible to represent the distance between two points: the agent's self position and the attraction point A. This equation will allow an emergent swarm behaviour.

$$NS = \sqrt{2}(CR - AR) \tag{2.3}$$

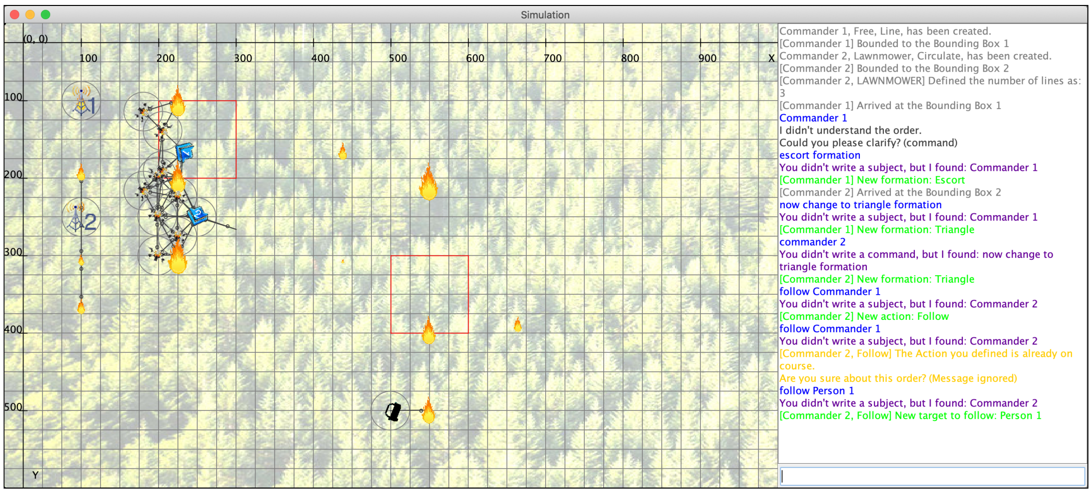
In order to calculate the optimal node separation between cells, Equation 2.3 ensures a coverage of the environment with minimum overlap. The separation (NS) needs to be matched to the UAV's physical specifications, where CR is the camera's footprint radius and AR is the Arrival radius.

The approach was tested with one and two fires simultaneously. It uses a single heuristic to maintain a persistent search for new fires while tracking the known ones. [Howden, 2013]

Chapter 3

System Architecture

To achieve the objectives it is necessary to draw the system architecture. Although this thesis has a case-study on monitoring fire, the program itself is designed and developed to be as open as possible to accept different types of missions with either virtual or real-life robots. As there are two different areas covered in this project, meaning a robotic area and an Natural Language Processing (NLP) area, the project needs to be divided by two systems: The Robotic System and the NLP System. The Robotic System is the representation of the robots and other types of nodes in the software, simulating their actions virtually, and later on physically in the field. This system also includes the Graphical User Interface (GUI). The NLP System is responsible for capturing, understanding and translating what the user



```
Commander 1, Free, Line, has been created.
[Commander 1] Bounded to the Bounding Box 1
Commander 2, Lawnmower, Circulate, has been created.
[Commander 2] Bounded to the Bounding Box 2
[Commander 2, LAWNMOWER] Defined the number of lines as:
3
[Commander 1] Arrived at the Bounding Box 1
Commander 1
I didn't understand the order.
Could you please clarify? (command)
escort formation
You didn't write a subject, but I found: Commander 1
[Commander 1] New formation: Escort
[Commander 2] Arrived at the Bounding Box 2
now change to triangle formation
You didn't write a subject, but I found: Commander 1
[Commander 1] New formation: Triangle
commander 2
You didn't write a command, but I found: now change to
triangle formation
[Commander 2] New formation: Triangle
follow Commander 1
You didn't write a subject, but I found: Commander 2
[Commander 2] New action: Follow
follow Commander 1
You didn't write a subject, but I found: Commander 2
[Commander 2, Follow] The Action you defined is already on
course.
Are you sure about this order? (Message ignored)
follow Person 1
You didn't write a subject, but I found: Commander 2
[Commander 2, Follow] New target to follow: Person 1
```

Figure 3.1: The developed working simulator.

said/wrote or is saying/writing to the nodes and move the treated information to the Robotic System in the form of commands. The goal of this architecture is to create a functional system that understands what the user is writing and linking it to a functional robotic system where it can accept actions, formations and general commands and reproduce them in real life. The first step is to build a robust Robotic System with different options on the formations, actions and general commands. Also adding different intervenients like towers, people and cars will help developing a more robust experience because it allows more interaction between all the things inside the mission. Secondly, to develop a good Natural Language Processing (NLP) system that is able to understand what the user is saying. It is fundamental to make it easy to add new commands to the system. For example, if a user needs a new type of formation, it only requires three steps to do so, more on this can be found on sections 3.1.3.1 and 3.1.3.2. Finally, linking both systems will result in the expected user experience with interaction between the nodes and the user, making them both understand each other.

3.1 Robotic System

The Robotic System is one of the two systems available on this program. Its purpose is to bring all the robotic part into the whole program, meaning, bringing the nodes, which represent an element of the network, and everything associated with them to the program. This element is not necessarily a machine, it can be a person, a drone, a virtual robot, anything! It is responsible to make all the machines move to a certain point in a certain period of time, make them stop in place and, most importantly, to make the nodes communicate with each other. This part of the units uses the JBotSim 1.0.0 library [Casteigts, 2015].

The Master-Slave communication architecture was chosen for the communication between robots. Where the "Master bot decides on the path to be taken and also supplies the slave bots with the coordinates to be reached" and "the advantage of using numerous robots are several, one being reduced cost since the robots are physically simple others being more robust and highly scalable system" [Anand et al., 2014].

3.1.1 Nodes

The nodes represent an element of the graph/network [Casteigts, 2015]. They represent different types of elements in a simulation, which will be described in the following sub-subsections as different extensions of the node class, such as: Commander Node, Robot Node, Environmental Node, Person Node, Car Node and Tower Node. These extensions allow the system to be organized in a way that if someone needs to use this system as an API, it will be easy to create new types of nodes or adapt the existing ones into different missions or purposes. For example, the environmental node can either be fire or water, if the mission is monitoring fire or monitoring the sea, respectively.

Commander Node

A Commander Node is the leader of the squad of robots. It is the one responsible for receiving the information given by the NLP System and translate it into the command itself and this node may or may not exist in real life. This is one of the many settings available before the launch of the program. If reliability is key on a mission, then making the Commander virtual is preferable as it avoids various problems that may or may not happen, with the disadvantage of having one less node searching in the field. Different types of information are assigned to the Commander Node, such as a Bounding Box, which is responsible to restrain where the commander can go on the map, explained on subsection 3.1.2, a name, which is given automatically by the system to avoid user errors, a Tower Node, more about it further ahead on subsection 3.1.1, an action (for himself), a formation (for all the robots associated with it), described on sections 3.1.3.1 and 3.1.3.2, respectively, and finally the number of robots that the Commander must have in order to achieve its purpose. If no action or formation is attributed on the creation of a commander node, it will use the action or formation defined on the variables default_action or default_formation, in the configuration file.

The number of robots that the Commander has is defined by the user, but it is as well limited by the number of robots available in the Commander Node's Tower. If the user asks for more robots than the ones that exist, then an error message will be prompted up asking for a lower number of robots for that Commander.

The Actions are assigned to the Commander Node and they are responsible to tell how the Commander Node will move in the field, inside its Bounding Box. To change the formation of the Commander's Squad, the user needs to speak to the commander.

The communication between the Commander Node and the Robot nodes is done in a star topology network way, where there is one coordinator, the Commander Node, and all end devices are connected to it, the Robot Nodes, as seen on section 2.1. To solve the disadvantage of this typology, the virtual Commander method is used, which, as previously explained, is a Commander that does not exist and is merely representative on the map. This means that the Commander does not have to process real-life movements or other concerns. This way, it is looser and can handle all robots that are linked to it and exchange messages between them when needed.

Robot Node

The responsibility of a Robot Node depends on the purpose of the mission, while focusing on following the commander with a certain formation with its robot squad companions. Unlike a Commander Node, a Robot Node cannot be virtual. In this project, they are drones, looking for fires in the field. Drones where chosen because they can rotate on themselves without executing long curves, making it easier to maneuver and to calculate their next position in time. When called they need to have a formation and an index, attributed by the Commander. With this information they can know their next position in the next pulse of a clock. In case one Robot Node fails, the index is automatically changed, making the whole squad change their position to accommodate the new changes.

Tower Node

The Tower Node is responsible to deploy all the nodes that are machines, meaning, where the Tower Node is, it is where all the Commander and Robot Nodes, who are bounded to the Tower Node in question, are going to be released into the field. The Tower Node is a mere representation of a station with all the machinery available. It can be a tower, a car, whatever the user wants to set it as. For the purpose of this case, the Tower Node will be represented as a Tower. It possesses all the information about the machinery available on it and contains an ID to differ from other available Tower Nodes. A Commander can also interact with a Tower Node, for example, to return to it or to follow the Tower, in case it moves.

Environmental Node

The Environmental Node can be considered the most abstract node out of all of them and it is done this way so the user can program it to the most adequate situation depending on the mission/purpose. For the case that the system is dealing right now, monitoring fire, the Environmental Node is considered as a fire with automatic and different temperatures. To represent how big the fire is, a scale from 0 to 20. 0 means no fire, 1 not a big fire on 50° Celsius, and 20 a fire with at least 100° Celsius, is represented by the following equation $y = \frac{2}{5}x - 20$. This equation is used so that only fires with temperature above 50° Celsius are actually considered fires and are shown on the map. These temperatures are mere supposed values and are far away from the real-life values, which tend to be around 726.85° Celsius [Qian et al., 2008]. They are used in order to better explain how it works. The temperature also varies every pulse of a clock and, in order to maintain a realistic simulation, the information about a fire is updated whenever a robot has a new link with it. The maximum and minimum temperature and at what temperature is considered a fire can be configured.

Person Node and Car Node

A Person Node and a Car Node are representations of a person and a car, respectively, in the simulation or in the real-life mission. It allows the user to know where they are in the field. If there is a need to escort or follow that person or car, the interaction is easier by looking at the map instead of guessing where the person or car is. It also allows the user to have a general awareness of who is on the field. A commander can also interact with both of them.

3.1.2 Bounding Box

Bounding Boxes are the areas, defined by the user, to the Commander Nodes. These are the limitations to where the Commander Node can possibly go, unless the Commander Node has an action that works outside the Bounding Box, then, it will be completely ignored, like the Follow Action (in section 3.1.3.1).

The Bounding Boxes do not possess any kind of influence on any other type of node except for the Commander Node. More on this can be found in the sub-section of the Formations (3.1.3.2).

3.1.3 Actions, Formations and General Commands

To aid on the missions, formations, actions and general commands were created to control the Robot Nodes, the Commander Nodes and general things, respectively. A formation is responsible to calculate where the whole squad of the Commander Node will be on the map on the next pulse of clock and the actions will inform the Commander Node how it will scan the Bounding Box linked to himself. The general commands are responsible for the most generic tasks like creating and removing Commander Nodes.

3.1.3.1 Actions

Actions are patterns given to the Commander Node in order to scan the Bounding Box, or to get out of it, in a certain and specific way. They can be split into four different commands, which are lawn mower, free, follow, and return. Most of them do not need an object in a phrase in order to be executed, with the exception of the Follow action because the Commander Node needs to know who he is going to follow. In order to avoid errors in the calculations of the routes, once a new action is given to the Commander Node, it will always go to the beginning of the Bounding Box and then it will execute the action, with the exception of the Follow Action.

It is possible to add new Actions according to the user's need and there are three steps to do it properly: First, the user needs to add the new action into the Enumeration on the *Enum_Action.java* file. Second, create a new class with the action name implementing the Action class and set it up according to user needs. Finally, the Commander Node needs to accept this new action and this can be done on the *CommanderNode.java* file on the *method onClock()*, in the *switch case* for each action.

Free Action

The Free action consists on the Commander Node moving freely, with no rules, inside its bounding. Whenever the Commander Node touches the Y-axis walls of the Bounding Box, it will turn around on himself, making it go on the opposite direction he was heading to. Unlike the behavior of the Y-axis walls, the X-axis walls were not programmed to make the Commander Node turn around on himself. This was done this way so it creates a better random feeling to the whole action, so, if the Commander Node hits the X-axis walls then it will sum up 98^o to the angle of the direction he was heading to.

Lawn Mower Action

The Lawn Mower Action consists of the Commander Node walking in a zig zag way inside its Bounding Box. When a Commander Node receives a Bounding Box, the number of lines that the Commander has to go through is calculated with the Equation 3.1.

$$numberOfLines = (int)(\frac{BoundingBox_{Height}}{DefaultNode_{SensingRange}}) \tag{3.1}$$

The $DefaultNode_{SensingRange}$ is the range of all nodes that are not virtual or those that do not have a custom range. The Sensing Range consists in the range to sense other nodes around the node in cause. The Sensing Range of the Commander Node can not be used since it can be virtual making this value equal to zero, so, in order to keep the same Sensing Range throughout all the squad, the default one is used. Next, the Commander Node needs to reach the Bounding Box starting point with a safe value, so, if the Bounding Box is set at $x_1 = 200$ and $y_1 = 100$ and goes to $x_2 = 400$ and $y_2 = 500$ and the $safe\_value = 10$ then the starting point of the Bounding Box is $StartingPoint_{BB_x} = 200 + 10 = 210$ and $StartingPoint_{BB_y} = 100 + 10 = 110$. The safe value is to ensure that the Commander Node does not get out of the Bounding Box and it can be set on the General Configuration file.

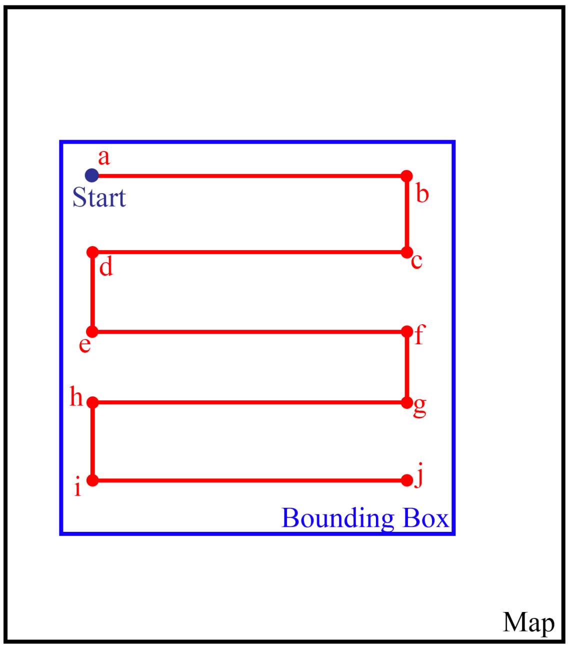


FIGURE 3.2: Example of a Lawn Mower path by a Commander Node with five lines.

As it can be seen on Figure 3.2, the number of lines needed to scan the whole Bounding Box is five lines. Upon the arrival of the $StartingPoint$ the direction of the Commander Node is given by the $Width_{BoundingBox} - safe\_value$. If there is a new line to be scanned, then the Commander Node will go up or down depending on the direction it is heading to. The distance of the vertical lines is given by the following equation:

$$VerticalLength = \frac{BoundingBox_{Height}}{numberOfLines} \tag{3.2}$$

Return Action

When a Commander is released into the field, it has to be released on a Tower Node, so the tower is considered the base location. The Return action has the goal to simplify the process of asking a Commander Node to return to its base. Instead of giving the coordinates, the Commander Node possesses all the information about its tower, including its location and ID number. When a user asks a Commander Node to return, the Commander Node will automatically ask the Tower its position and return to it immediately. This action is almost identical to the Retreat Formation. There is this difference so the user can send the Commander to return to the base or, instead, retreat the whole squad to the linked Tower Node, without the Commander Node.

Follow Action

The Follow Action was developed in order to allow a user to ask a Commander Node to follow another node. It is different from the rest of the actions, because when it is requested it needs an object, meaning, a node name to follow. It can be a Person Node to make sure that the person arrives safely to a certain area or to know if the person is being surrounded by an environmental node, a Car Node, a Tower or a Commander Node. So in order to work, once the Commander Node knows who he is going to follow, at every pulse of a clock, he will ask the position of the node to be followed and set his direction to it.

3.1.3.2 Formations

Formations are shapes are requests by the user to the Commander Node and from the Commander Node to its Robot Node squad. These formations will allow to make the robots search for whatever they are meant to search in a certain shape, alongside their robot companions from their squad, always accordingly with the Commander's location. Without being linked to the Bounding Box, the Robot Nodes do not have to worry about maintaining inside it, making it possible to create a whole different type of shapes or configurations without concerning to much about spacing. While creating a shape or a configuration, the squad always has the concern to be following their Commander Node, so all the shapes and configurations are done considering one thing: its location. It is possible to have different formations for different squads, as long as the squads do not share the same Commander Node. There are five different formations on this system:

circulate, escort, line, retreat, and triangulate. All of them do not need an object on the natural language command, meaning that it is only necessary to say the Commander Node number and the formation itself.

Circulate Formation and Escort Formation

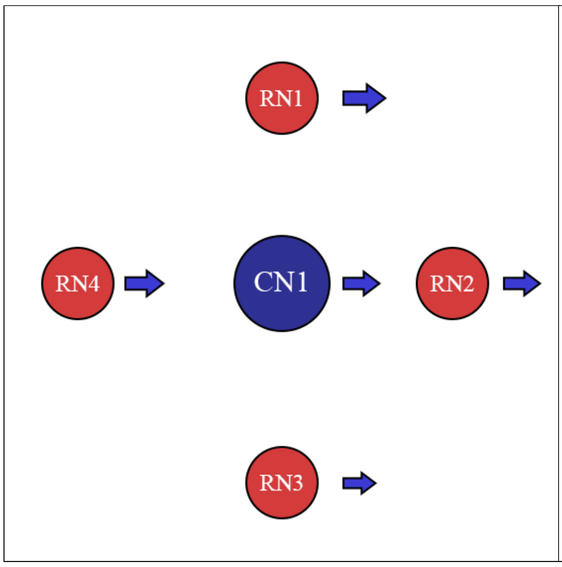


FIGURE 3.3: Escort Formation

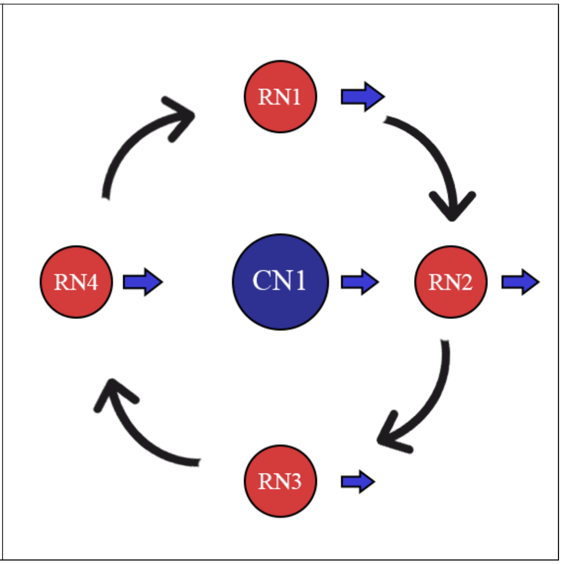


FIGURE 3.4: Circulate Formation

The Circulate Formation and the Escort Formation are very similar as shown on Figures 3.3 and 3.4. The escort formation follows the Commander Node without rotating around him, unlike the Circulate, which makes the whole squad rotate around their Commander Node. The position of each robot is calculated based on their own index and the number of robots in the squad. The following equations will determine the position in the field of each robot:

$$radius = Communication_{Range} - Sensing_{Range} \tag{3.3}$$

$$angle = \frac{360}{SquadSize} * Robot_{Index} + Circulate_{Extra} \tag{3.4}$$

$$x = Commander_X + \cos(angle * \frac{180}{\pi}) * radius \tag{3.5}$$

$$y = Commander_Y + \sin(angle * \frac{180}{\pi}) * radius \tag{3.6}$$

As shown on Equation 3.3, in order to calculate the *radius* it is necessary to have Communication Range, which is the maximum distance within which a node can reach other nodes and the Sensing Range, previously explained on section 3.1.3.1. If the circulate formation is being used, then the $Circulate_{Extra}$ will be used on Equation 3.4 and at every pulse of a clock, this value will be incremented, otherwise it will always be equal to 0. Finally, the Equation 3.5 will determine the next x, as the Equation 3.6 will determine the next y coordinate.

Line Formation

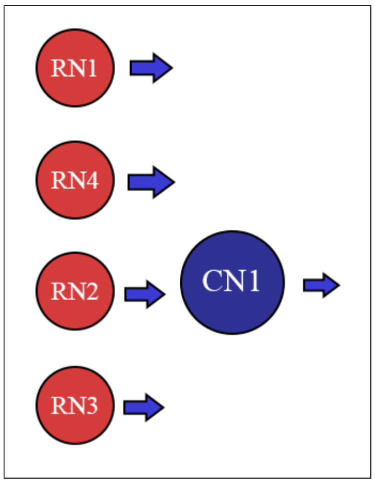


FIGURE 3.5: Example of the line formation with a Commander Node with a squad of four Robot Nodes.

As shown on Figure 3.5, the Line Formation consists on creating a line of robots behind the Commander Node and the next x and y position can be determined by the following equations:

$$x = Commander_X - Sensing_{Range} \tag{3.7}$$

$$\begin{aligned} y = Commander_Y - Sensing_{Range} * Robot_{Index} + \\ (Sensing_{Range} * \frac{SquadSize}{2}) \end{aligned} \tag{3.8}$$

Triangle Formation

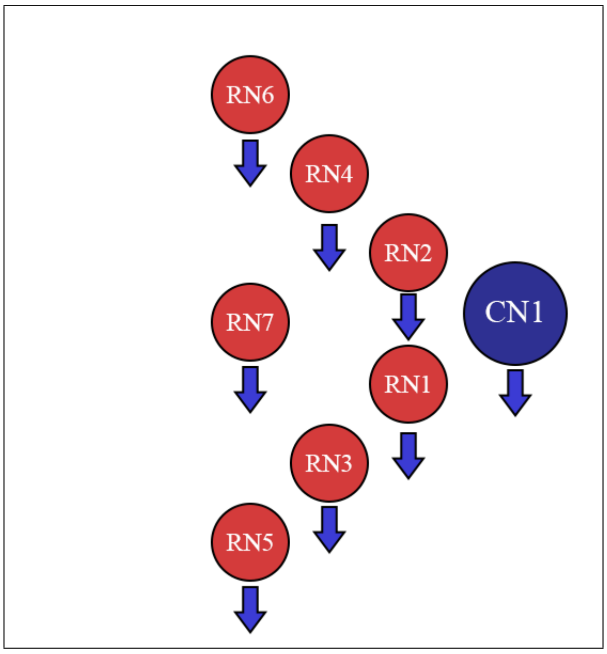


Figure 3.6: Example of the triangle formation with a Commander Node with a squad of seven Robot Nodes.

The Triangle Formation is responsible to make a triangle out of the squad available. If the squad size is less or equal than two, then it is not possible to make this formation. If the user asks to perform a triangle formation with the previous squad size, an error message is shown, and the line formation is chosen instead. If the squad size is bigger than two, then the squad is divided into three different groups, namely: the even, the odds and the middle. As shown on the Figure 3.6, the odd numbers will be the lower robots on the triangle, the even robots will be the upper robots on the triangle, and, if the squad size is an odd number, the last robot will belong in the middle end part of the triangle.

In order to know if there is going to be a middle robot on the triangle, a condition has to be verified: if the number of robots is an odd number and the robot is the last one on the squad list. If this condition is verified then the robot in question is, in fact, the middle robot on the triangle. To calculate its position, the Equation 3.9 shows how to get the x coordinate. The squad size is divided by two to make it in the middle of the triangle instead of the very end. The y coordinate is given by the $CommanderNode_y$. On the Figure 3.6, this robot would be the RN7.

$$x = Commander_X - (\frac{SquadSize}{2}) * Sensing_{Range} \tag{3.9}$$

To know the x position of the robots, one just needs to add to the index 1 and divide it by 2, so when transformed into an integer the robot will be positioned on the next vertical column on the triangle, for example $\frac{2+1}{2} = 1.5$ transforming into an integer will result in 1, so, the robot number two will belong on the first column of the triangle, as well as his companion robot number 1, with $\frac{1+1}{2} = 1$. After this, multiply everything by the sensing range. An example can be found on the Equation 3.10 for the x coordinate for either the odd number and the even number.

$$x_{odd/even} = (int)(Commander_X - (\frac{Robot_{Index} + 1}{2}) * Sensing_{Range}) \tag{3.10}$$

About y coordinate, all the odd robot nodes were positioned on a lower diagonal position to the Commander Node and the pair robot nodes were positioned on the opposite side, at the higher diagonal position to the Commander Node. Both equations 3.11 and 3.12 represent the way to get the y coordinate for the odds and for the even, respectively.

$$y_{odd} = Commander_Y + (\frac{Robot_{Index} + 1}{2}) * Sensing_{Range} \tag{3.11}$$

$$y_{even} = Commander_Y - (\frac{Robot_{Index} + 1}{2}) * Sensing_{Range} \tag{3.12}$$

Retreat Formation

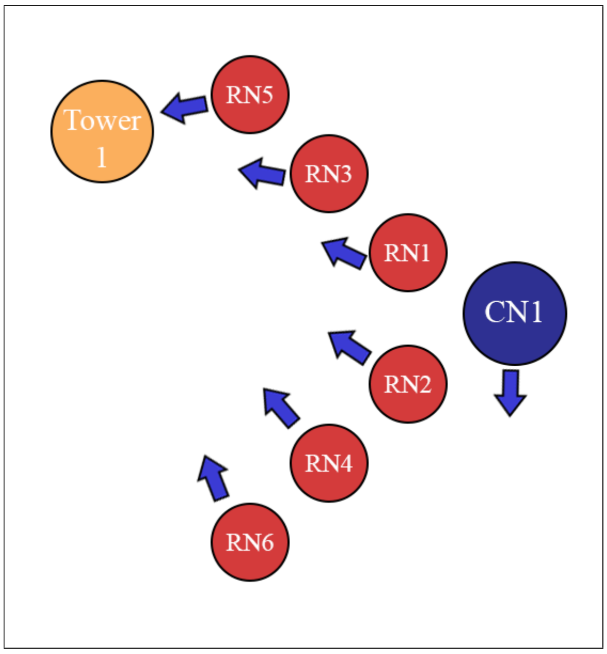


Figure 3.7: Example of the retreat formation with a Commander Node with a squad of six Robot Nodes and a Tower Node with the ID 1.

The Retreat Formation will make all the robots in the Commander Node's squad retreat to the Commander's Tower Node, while the Commander is staying inside his Bounding Box. Although there is no way to add more robots to a Commander right now, this might be more useful to the future to send all the robots back to base and bringing more or less on the next conversation with this specific Commander. When the user asks for this formation, the Commander Node will ask the Tower Node its current position, and it will inform the whole squad of the location, making it their destination. Upon the arrival, if the user configured the variable *retreat_ upon_ arrival_ remove_ robot* to true in the configurations file, then upon the arrival of each robot, they will be stored in the tower and removed from the field.

3.1.3.3 General Commands

General Commands are general possibilities available to the user in order to aid the amount of Commander Nodes and Robot Nodes on the field. These are very different from the Actions and Formations since the natural language commands can or cannot use a subject or an object for the same command. Since these commands are more general and it is dangerous to misspell some information, context is blocked on these phrases so if a user wants to add a new commander, (s)he must use a single natural language command in order to do it.

Add new Commander

Whenever a user wants to add a new Commander to the field, the user is prompted with six questions and these are always the same, carefully protected against misleading or wrong information. To begin with, it is necessary to know where the user wants the Commander Node to act on, so, there are four questions just to obtain the new commander's Bounding Box:

1. X-axis where the Bounding Box should start;
2. Y-axis where the Bounding should start;
3. X-axis where the Bounding Box should end;
4. Y-axis where the Bounding should end.

If the user tries to enter an X-axis or Y-axis smaller than the starting ones, an error is shown, and a new input is requested. If the user enters successfully all the requested valid data, then it is asked the Tower ID number where the Commander Node and Robot Nodes will spawn. If the user enters a wrong Tower ID then an error pops up and it is requested new information, once again. Finally, it is requested to the user the number of Robots the Commander should have. If the user enters a bigger number of robots than there actually is, it is requested a new possible number.

Remove Commander

In order to remove Commanders or Robots from the field, the generic remove command must be used. It is not possible to remove Robots directly, only by removing Commander Nodes associated with the Robots the user wants to retrieve. This is only possible if the following variable is set to true on the configurations file: *remove_commander_removes_robots*.

3.2 NLP System

The Natural Language Processing (NLP) System is the second system and it is necessary to achieve all the remaining objectives proposed on section 1.4. This system will not only be responsible for the understanding of what the user is saying, but it will also be responsible for translating it into commands and transmitting them to the Commanders or to the System itself. There are a lot of ways to say the same thing and it is not enough to create "if" conditions on a program, otherwise there would be a million conditions just to treat some sentences... It is necessary to deeply understand the written sentence and all its grammar and structure to obtain good results. In order to achieve this deep understanding, the NLP System uses the Stanford CoreNLP library [Manning et al., 2014] in order to make the parsing, through the dependency parser. With this library it is possible to obtain the base forms of words, their parts of speech, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate sentiment, extract particular or open-class relations between entity mentions, etc.

3.2.1 Natural Language Understanding (NLU) System

As the base of the whole NLP System, the Natural Language Understanding (NLU) is responsible for the phrase treatment. Whenever a user speaks or writes a new sentence there is a need to treat it in order to get useful information out of it and take some conclusions.

To treat every natural language command by the user, for this system, there is the need to retrieve the grammatical tag of each word as well as the relationship between the *"head"* words of the sentence and the words. For this purpose, the dependency parsing from the Stanford CoreNLP library is being used. "A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between *"head"* words and words which modify those heads. The figure below" 3.8 "shows a dependency parse of a short sentence. The arrow from the word moving to the word faster indicates that faster modifies moving, and the label *advmod* assigned to the arrow describes the exact nature of the dependency" [Manning et al., 2014].

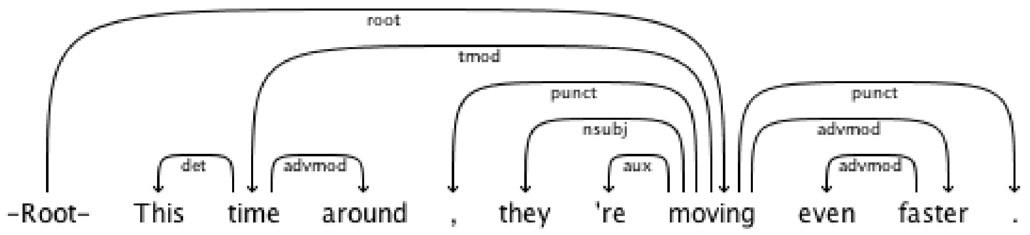


FIGURE 3.8: Example of a dependency parse of a short sentence on the Stanford. Image downloaded from https://nlp.stanford.edu/software/nndep.html in October 2019.

The general structure of a sentence, for this system, is S-V-O, meaning Subject-Verb-Object sentence. It is more than enough to make the whole system understand what the user wants of the Commanders or the general system itself. To use this structure, the Subject, the Object and the Command, represented by the verb, need to be rebuilt out of the whole sentence, even if the subject or the object are optional. The verb is represented by the Command since it is possible to capture more than one command in a single phrase, as shown on the natural language command example 5 (page 38). The Subject and the Object can be optional because there are phrases like:

Commander 1 escort formation. (1)

The natural language command number 1 represents a request that can be used in the system and does not require any object since with the subject and the command the Escort Formation can be executed.

Add new Commander. (2)

The natural language command number 2 shows a request where there is no need for a subject but only an object because the user is speaking to the whole system, not a specific node.

Commander 1 follow Person 4. (3)

Finally, on the natural language command number 3 there is a need for both since the subject "Commander 1" needs to know who it is going to follow, in this example: "Person 4".

Depending on the action, formation or general command this will affect the calculation when evaluating the right action, formation or general command. More of this calculation can be found later on section 3.2.3.

3.2.1.1 Building the Subject and the Object

Since there are multiple Class-types of subjects and objects, there are different ways to detect who is the subject and who is the object. For example, on the natural language command 2 the object does not represent an existing type of Node, so the object is a String "new Commander". On the other hand, on the natural language command number 3 the object "Person 4" is a type of node so the result is the node itself.

To generate the Subject or the Object, the system tries to get the Indexed Word, meaning the word inside the dependency parse tree, with the tag Nominal Subject (nsubj) or Direct Object (dobj), respectively. If there is an Indexed Word with the tags previously mentioned, then it is necessary to complete them (if there is something to complete).

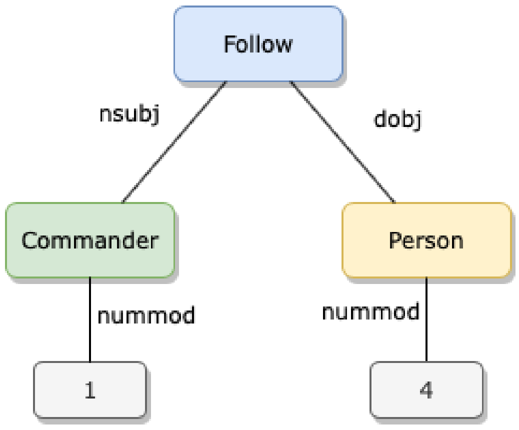


FIGURE 3.9: Dependency Parsing Tree of the sentence: Commander 1 follow Person 4.

For example, on the natural language command 3, the subject is **Commander 1** and the dependency parser tree can be seen on Figure 3.9. When trying to grab the Indexed Word of the tag nsubj, the result is **Commander**. Then, there is a check if there are any children to that same Indexed Word, in this example, there is a Numeric Modifier (nummod) which its value is going to be **1**. Merging this two will result in **Commander 1**. After the merging there is going to be a check on the entire node list if there is any node type: **Commander** with the number **1**. If there is then the result of the method *Node getSubjAsNode()* or *Node getDobjAsNode()* is going to be the **Commander 1** if not, before returning *null*, the method will try to get another result by searching for a Noun Compound Modifier (compound) instead of a nsubj.

The subject or the object can be labeled as a compound because it is a noun that serves to modify the head noun. In this case there is no *head* noun to be modified so it is considered a wrong interpretation from the library, since **Commander** can be considered a description of a *head* noun. Whenever there is a subject or an object labeled as compound, the nummod that identifies who is the user speaking to is not directly connected to the compound itself, but to the ROOT. So, every time there is a compound, the method needs to check for the closest nummod available near it. If the merge results in a node then this node is returned by the same method previously mentioned, if not, it will return *null* if the subject is not considered as a root, more on this on section 3.2.2.

Commander 1 change to triangle. (4)

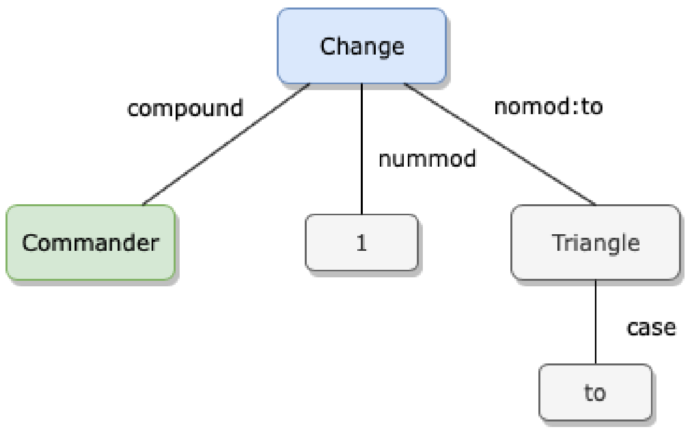


FIGURE 3.10: Dependency Parsing Tree of the sentence: Commander 1 change to triangle.

An example of a classification with a compound can be found on the natural language command number 4 followed by its dependency parsing tree on Figure 3.10.

When there is the need to check for a string subject or a string object instead of a node, the method *String getSubjAsString()* or *String getDobjAsString()* is run and if the Node is *null* but the string has something, then it means that there is a subject on the request but it is not a Node type subject. This situation can be found on the example natural language command number 2, where **new Commander** is the object. The previous methods simply get the Indexed Word with either the tag nsubj or dobj and complete it with all its children and transform the overall merge into a String. It can also return *null* if there is no subject or object, respectively.

The possible types of results for the Subject or the Object are Commander Node, Tower Node, Person Node, Car Node, and String. The Robot Node does not need to be included in this since there is no interaction between the user and the robot directly, but rather from the Commander Node.

3.2.1.2 Building the Command

In a first try of developing this system, the NLU System was trying to build the subject, the command and the object at the same time without the aid of each other. Although it was a successful try, it brought a lot of problems when trying to build the Command part. Without the help of the subject and the object, there is a need to build it from the root (if the root is not the subject) to all its last children on the Dependency Parsing Tree. This means that there needs to be a lot of exceptions to cover all situations and since it is possible to say the same thing in a lot of different ways, this idea is not optimal.

So, to build the current system, the logic of the passwords was used: "instead of creating exceptions for all the bad characters it is much safer and easier to only allow certain characters". So, instead of building multiple and countless exceptions, it is much easier and accurate to grab the Subject and the Object and remove it from the general sentence resulting in the Command.

$$\textit{Commander 1 change to escort formation and follow Commander 2} \tag{5}$$

By using the system as it is, getting the Command it is pretty simple. It just needs to obtain the Subject and the Object and remove it from the phrase, so the NLU System can conclude the following information of the natural language command number 5:

- Subject: Commander 1
- Command: change to escort formation and follow
- Object: Commander 2

This information will be presented and used on the Form, explained on the next section.

3.2.2 Form

The Form is not only used to obtain all the conclusions from the NLU System in a much cleaner way, with the information more organized and centralized, but also to bring context to the conversation between Human-Machine, like the figure 3.11 shows. Like the NLU System, a unique Form is generated at every new sentence. A Form only receives a NLU System and it uses all its functionalities to fill itself up. So, whenever a Form is generated not only it automatically asks the NLU System for the Subject, in the form of Node, String and the nsubj Indexed Word word, but also the Object with the same types as the subject needs (Node, String and dobj Indexed Word word) and finally the Command itself. Before finishing the initial setup, the Form will also generate a new and unique Evaluator for this sentence, more about it on section 3.2.3.

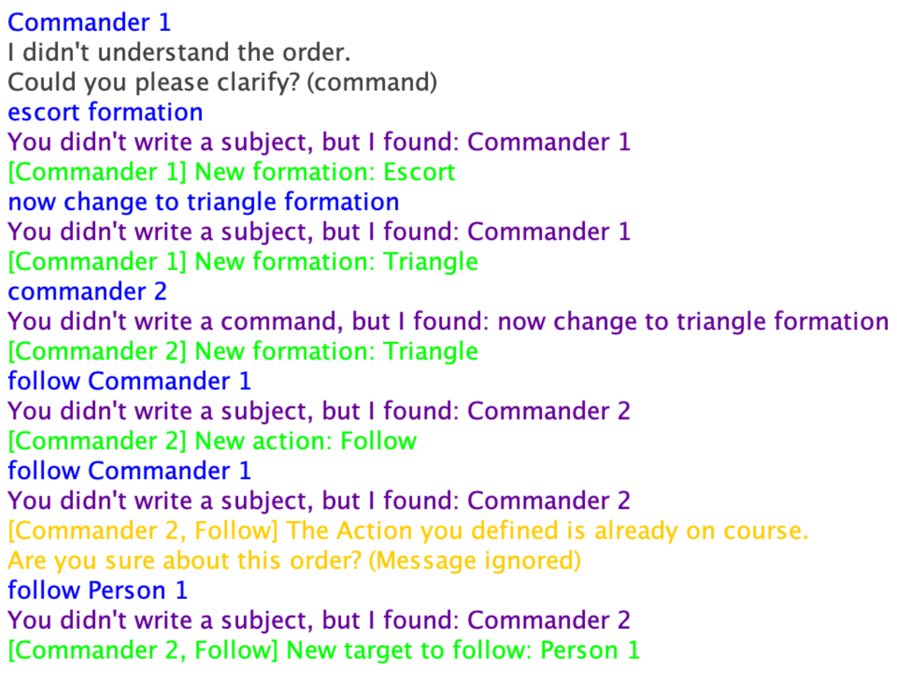

```
Commander 1
I didn't understand the order.
Could you please clarify? (command)
escort formation
You didn't write a subject, but I found: Commander 1
[Commander 1] New formation: Escort
now change to triangle formation
You didn't write a subject, but I found: Commander 1
[Commander 1] New formation: Triangle
commander 2
You didn't write a command, but I found: now change to triangle formation
[Commander 2] New formation: Triangle
follow Commander 1
You didn't write a subject, but I found: Commander 2
[Commander 2] New action: Follow
follow Commander 1
You didn't write a subject, but I found: Commander 2
[Commander 2, Follow] The Action you defined is already on course.
Are you sure about this order? (Message ignored)
follow Person 1
You didn't write a subject, but I found: Commander 2
[Commander 2, Follow] New target to follow: Person 1
```

FIGURE 3.11: Example of the Form Context System working in different situations.

On the Figure 3.11, the blue represents what the user wrote, the dark gray represents a question, the context is represented by the purple color, the green color is the successful command and finally the yellow message with a warning message. The context was introduced in this system not only to simplify the conversation between the user and the machine, but also to make the interaction more natural avoiding the exhausting experience always saying the same thing like: "Commander 1 change to ...", "Commander 1 now change to ...", "Commander 1 return". Instead, the user just needs to say the subject once, and then, all the next phrases will be contextually compromised by the subject from the older form or forms. It also works for the command, where the user can say the command and then the subject. The search context is limited, since the information can become inadequate or old. This limit can be set on the variable *number_ of_forms_ till_ old* on the configuration file (by default it is six).

Bringing context to the NLP System means breaking a little bit the S-V-O structure previously introduced in section 3.2.1 and, for this reason, the way the NLU System works needs to be a little different as well: from only detecting who is the subject, what is the command and finally who or what is the object, to accept the subject, the command and the object as a possible ROOT. To make this possible, every form has a group of flags, which can be translated to a group of boolean variables, that will trigger a new question and expectation of answer upon a new Form. So, every form, except for the very first one, will ask the previous form before it if there is a flag that needs to be satisfied.

Change to escort Formation. (6)

Commander 1. (7)

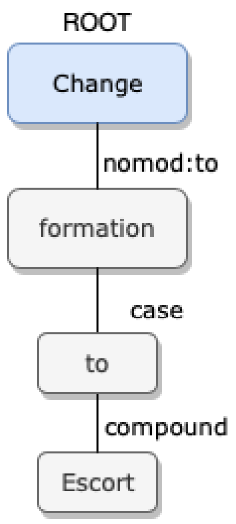


FIGURE 3.12: Dependency Parsing Tree of the natural language commands number 6

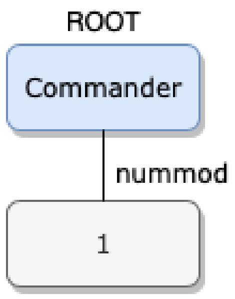


FIGURE 3.13: Dependency Parsing Tree of the natural language commands number 7

An example is shown on the natural language commands 6 and 7 with their dependency parsing trees on figures 3.12 and 3.13, respectively. When the user writes "Change to escort Formation.", the NLU System does not recognize any subject, because there is in fact no subject. When there is no subject, the flag **needSubj** is activated on the current form and the next Form will know that probably the message that the user wrote is, in fact, the expected subject. If the user writes something other than the subject, then the system will ignore the previous command and consider it canceled.

To make the NLU System accept a subject as a ROOT there is a condition verified right before detecting a subject: if there is no nsubj tag on the phrase but there is a nummod connected to the root and the size of the dependency parser is two, then the root is considered the subject or the object, depending on what is asked. This is shown on the natural language command number 7 with the dependency parsing tree on the Figure 3.13. If the system just needs the command then if the subject and the object are *null* and there is nothing to take out of the whole sentence, it means that the user wrote the command.

There are multiple flags available to use, namely: *needSubj, needCommand, needDobj, needBBFirstX, needBBLastX, needBBFirstY, needBBLastY, needTower, needRobots* and finally *needName.* Each flag can be used to ask the user whatever information is needed to the system execute the desired order. The flags *needSubj*, *needCommand* and *needDobj* are used whenever a subject, a command or an object is needed, respectively. The flags *needBBFirstX*, *needBBLastX*, *needBBFirstY* and *needBBLastY* are to be used whenever the system needs to have the Bounding Box Starting point: with the needBBFirstX and needBBFirst Y and the Bounding Box end point: with the needBBLastX and the needBBLastY. Finally, the last three flags, *needTower*, *needRobots* and *needName*, are used when the system needs to know a Tower, how many robots a commander should have and what name should be given to something, respectively. The needName is not used in the project since all the nodes have an automatic name, which is the ID, but it was developed for future work.

3.2.3 Evaluator

The Evaluator is the last part of the NLP System and it is necessary to determine which action, formation or general command the user wants. It can also detect if the user instructed one or two orders in one sentence, although the general commands need to be isolated commands, due to their impact on the operation and to avoid a removal of a whole squad, by mistake.

Commander 1 change to Escort Formation and to Lawnmower action. (8)

The natural language command 8 shows an accepted and tested phrase by the NLP System containing one action and one formation. This is only possible due to correlation between the Evaluator and all the abstract factories that make up all the actions and formations. Abstract factories were chosen because of different benefits: firstly, it is possible to instantiate different objects out of a global category, making all the instances have the same common methods. Secondly, it is possible to update all the actions, formations and general command in a much more accurate way, since changing the category implies a change in all its instances [Jia Li et al., 2012].

After the Factory Provider, responsible for generating each new factory, creates all three factories, namely the Formation, Action and General Command Factories, and after each factory creates an instance of every possible type of order from the Robotic System, each instance will calculate the probability of matching the phrase by the user with itself.

To link a command to a sentence, the matching probability needs to be calculated and it needs to be greater than 0.5. Each command has three variables that attribute weight to the S-V-O, in this case, to the Subject, the Command and to the Object. This weight changes for every type of command. For example, the Follow Action needs to have a Subject, who will follow, a command, the follow itself and an Object, who will be followed. So, for the follow, the weights were distributed like 0.25 to the subject, 0.5 to the command and finally 0.25 to the object. But, for the Escort Formation, the weights were distributed as 0.3, 0.7 and 0.0, respectively. The object is 0.0 because it is not needed. There are some commands that need a negative weight, for example, the Follow Command, if there is no Object, then, to ensure that the follow command is not executed, there is a negative value of 0.4. So, if a user commands a commander to follow no one, the probability will be: $0.25 + 0.5 - 0.4 = 0.35 < 0.5$. Since it is lower than 0.5, then the command cannot be chosen and executed.

After calculating each probability, the Evaluator gets the biggest probability, greater than 0.5, of each factory's instances, resulting in the most likely order of each factory. After choosing the biggest probability, the Evaluator will then compare who has the biggest probability out of all factories. The one with the value closer to one will win and it will be the selected action/formation/general command.

To bring the double commands on a sentence to this Evaluator, each action/formation has a method that verifies if the verb that is on its lexicon is being used.

If it is being used and the probability of the natural language command has values above 0.5, then the Evaluator will consider the command as a valid one. For example, on the natural language command 8, one can deconstruct the natural language command into two sections:

- Subject: Commander 1
- Command: change to Escort Formation and to Lawnmower action.
- Object: *null*

Inside the command the Escort and the Lawnmower will result in a matching probability of 1.0, since there is a subject, no object and the verb is inside the lexicon for the Escort and the Lawnmower.

Chapter 4

Results and Discussion

To test the entire program with the previously explained architecture, 30 people were willing to participate in a short experiment, which can be seen on the attached script on the appendix A. There were two ways of making the test, one was personally, where they got the script as a paper and had to run through it and saw their commands happening in real commands, and the other was through a Google Form test. The google form test was equal to the paper one, the only difference is that people could not see the commands happening to the robots / Commander. In order to obtain the most diversified results, the Google Form test was put on reddit[1] on the sub-reddits: r/NLP, r/technology, r/SampleSize and r/robotics. The test consisted of people writing as naturally as possible, as if they were talking to another person, to the Commanders and to the System itself. They tested all the possible commands, meaning, all the four actions, five formations and two general commands, then they tried twice to mix one formation and one action to see if the NLP System managed to understand, in one sentence, both commands and finally they were asked to test out the context with five context phrases.

[1] https://www.reddit.com.

4.1 Evaluators Characterization

Some data was collected from the participants so that a characterization of who participated could be made.

Gender	Amount	%
Male	21	70
Female	6	20
N/A	3	10
Total:	30	100

TABLE 4.1: Gender of participants.

Like it is shown on Table 4.1, there were 30 people participating, 21 were male, representing 70% of the respondents, six were female (20% of the respondents) and three chose not to identify themselves (10% of the respondents). Ages range from 16 years old to 40 years old and the average age of the participants was 23 years old, with 22 being the mode.

Academic Level			Field Of Study		
Level	Amount	%	Field	Amount	%
High School	3	10	Computer Science	21	70
Bachelor	16	53	Nursing/Medicine	3	10
Master's	8	27	Economy/Management	2	7
PhD	1	3	Sociology	1	3
N/A	2	7	Physics / Mathematics	1	3
			N/A	2	7
Total:	30	100	Total:	30	100

TABLE 4.2: Academic Level and Field of Study of the participants

As for the academic level, as shown on Table 4.2, the participants were asked to fill in their finished academic level or if they are taking a course to write down the level of that course. It is important to enroll more people from other areas outside technology, since they will probably interact in a different way with the program. Sixteen out of 30 are taking or finished their bachelor degree, representing more than half, with a value of 53%, eight are taking or already took a master's degree, representing 27% of the respondents, one is taking or already took a PhD, representing 3%, three are in high school, representing 10%, and, finally, two did not reply representing 7%. Most courses are related to Computer Science with 21 people, representing 70% of the fields of study, next is nursing or medicine with three people, representing 10%, followed by Economy/Management with two people, representing 7%, next Sociology and Physics/Mathematics with one person each representing 3% each. Finally, the two that did not reply represent 7% out of all the fields of study.

Participant work?			Field of Work		
Work?	Amount	%	Field	Amount	%
Yes	14	47	Computer Science	11	79
No	14	47	Management	1	7
N/A	2	7	Research	2	14
Total:	30	≈100	Total:	14	100

TABLE 4.3: Information about the participants work

It is also interesting to understand if people are working and in what field, as this might as well influence the results. On Table 4.3 it is possible to see that 47% are working, meaning 14 people and 47% are not working, while two people did not reply, making it 7%. Inside those 14 people that are working, 11 are working on a Computer Science Field, representing 79% of the participants, two are in Research, representing 14% and 1 is working in Management, representing 7%.

Country Information		
Country	Amount	%
Portugal	10	33
USA	6	20
India	2	7
UK	2	7
Holand	1	3
Australia	1	3
Germany	1	3
Poland	1	3
Ireland	1	3
Canada	1	3
New Zealand	1	3
N/A	3	10
Total:	30	≈100

Table 4.4: Amount of replies sorted by Country

Finally, the last data is about people's nationality. Because some tests were made through the Internet, there are people from different countries, eleven to be specific. On Table 4.4 it is understood that ten participants (33%) have Portuguese nationality, while eight people are English and American, representing 27% of the surveyed population. There were also two Indian people that replied to this survey, representing 7%, while the remaining countries, meaning: Holland, Australia, Germany, Poland, Ireland, Canada and New Zealand were represented by one person (3% each). Three people opted to not reply to this question, representing 10% of the surveyed population.

4.2 Experiments

Overall there were five experiments made, per person. The first experiment consisted of four exercises, each to say, as naturally as possible, a different action from the four existing ones. The second experiment, very similar to the first and third, consisted of five exercises, where in each exercise was required to change Commander 2 squad formation in a unique way, meaning, one single formation per exercise. The third experiment, such as the first and second, consisted of testing the different General Commands, namely Add and Remove, in two different exercises. In the fourth experiment, the participant was asked to merge an action and a formation into one natural phrase, twice, so that there could be two mixed phrases per person. Finally, the last exercise was to test the context capability, where each person had to enter five different commands in separate sentences, for example:

- Sentence one: Commander 1;
- Sentence two: Line up;
- Sentence three: Commander 2;
- Sentence four: retreat;
- Sentence five: walk freely.

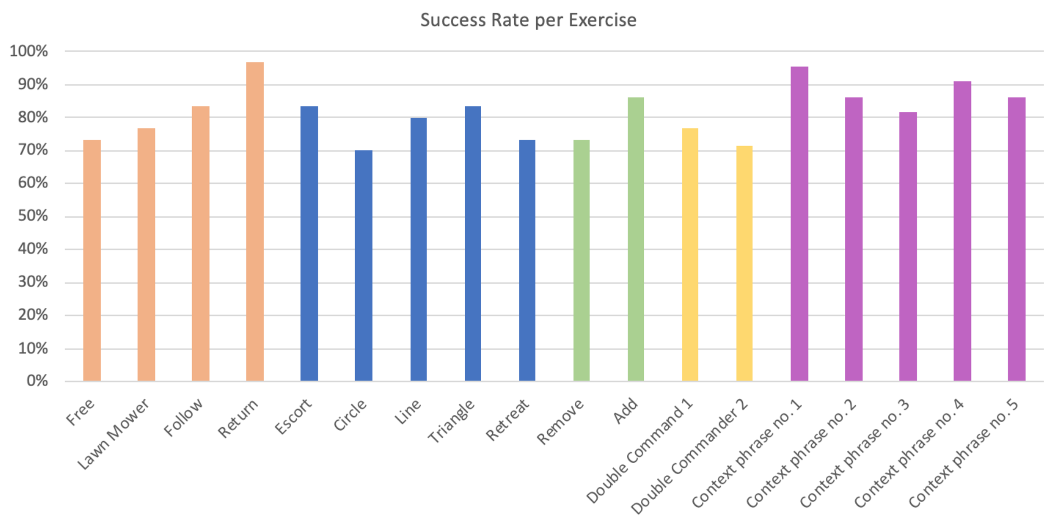


FIGURE 4.1: Success rate after the 30 tests of each command.

On Figure 4.1 and on the tables of the appendix B, it is possible to see the different results for the success rate of each exercise. The global success rate of all the 30 people's experiments together was about 82%.

About the actions, 22 out of 30 people succeeded in commanding the Commander 2 to be free, making the success rate 73%. The Lawn Mower action had a success rate of 77% (23 successes out of 30), the Follow action had 83% (25/30) and finally, the highest value of all exercises, the return action with 97% (29/30) of success rate. The average Action success rate is 83%.

On the formations, 83% of the people succeeded to make the squad escort (25/30), the circulate formation had 70% success rate with 21 people succeeding out of 30, making it the less successful command out of all of them, due to the way people construct the sentence and the failure of the system on building the correct subject or object, to then extract the command. When requested to make a line, 80% (24/30) of the surveyed population succeeded, while 83% (25/30) and 73% (22/30) managed to make the Commander 2 triangulate and retreat, respectively. The average Formation success rate is 78%.

The General Commands had an average success rate of 80% with 73% (22/30) and 86% (25/29) for Remove and Add commands, respectively.

For the double commands there was an average success of 76% and on the first sentence with double commands there were 20 successes out of 25, making it a 80%. For the second Double Command sentence there were 15 out of 21 successes, making the success rate go down to 71%, compared to the first double command sentence.

Finally, the context phrases had an overall average success rate of 88%, with the following success rate for each phrase: 95% (21/22),86% (19/22),82% (18/22), 91% (20/22) and 86% (19/22).

4.3 Discussion of the Results

When analyzing the different commands made by the 30 people who tested the program, there were some that stood out for their originality. For example, one person decided to call the triangle formation as “Mighty Duck” Formation and one decided to be more technical and call it “delta formation”, which is a formation flying maneuvre in the shape of a "V". There were also user inputs that did not make much sense such as calling line formation: “Single file”. This may be because it is a direct translation from the person’s original language. The person that wrote this last formation did not fill in the personal information, so it will be impossible to say where this person is originally from. The results were also influenced by the different images that were placed in the script A, where for example some people called the “line formation” as a “wall formation” or the “lawn mower action” as a “zig-zag action”.

Crossing the results of the tests with the personal information given by the users themselves it is possible to retrieve some interesting information. All those that did not reply on the personal data will be excluded of this discussion, since there is no data to cross with the results.

Due to the low number of female respondents, it was not possible to draw conclusions as to, whether or not, gender influences the results, as there are only six female entries and 21 male entries, which as a percentage, means a difference from ≈22% to ≈77%. Also, as previously seen on Table 4.2, Computer Science is the field of 70% of the participants, making it difficult to compare the results between academic fields, as well as the working field, with 79% being on Computer Science field, as shown on Table 4.3.

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	12	11	12	14	13	9	12	12	11	10	12	10	6	10	10	9	10	9
Fails	3	4	3	1	2	6	3	3	4	5	2	3	4	1	1	2	1	2
Absents	0	0	0	0	0	0	0	0	0	0	1	2	5	4	4	4	4	4
Success Rate	80%	73%	80%	93%	87%	60%	80%	80%	73%	67%	86%	77%	60%	91%	91%	82%	91%	82%
										Average Success Rate:			80%		Number of people with 22 years old or less:			15

TABLE 4.5: Success Rate of each exercise by people with 22 years old or less.

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	8	12	12	13	10	10	11	11	9	10	11	9	8	10	8	8	9	10
Fails	5	1	1	0	3	3	2	2	4	3	2	2	2	0	2	2	1	0
Absents	0	0	0	0	0	0	0	0	0	0	0	2	3	3	3	3	3	3
Success Rate	62%	92%	92%	100%	77%	77%	85%	85%	69%	77%	85%	82%	80%	100%	80%	80%	90%	100%
										Average Success Rate:			84%		Number of people with 23+ years old:			13

TABLE 4.6: Success Rate of each exercise by people with 23 years old and more.

From Table 4.5 and Table 4.6, it is possible to conclude that age is not a decisive factor for greater success when using natural language commands, since the average general success rate consists in 80% and 84%, respectively, although, people with 23 years old and more did get more success per exercise than the ones that were 22 years old and less, with 11 exercises with superior percentage of success rate, versus seven exercises, respectively.

	Ex1	Ex2	Ex3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	8	9	10	11	10	8	9	8	8	9	8	7	7	4	4	4	4	4
Fails	4	3	2	1	2	4	3	4	4	3	4	2	2	1	1	1	1	1
Absents	0	0	0	0	0	0	0	0	0	0	0	3	3	7	7	7	7	7
Success Rate	67%	75%	83%	92%	83%	67%	75%	67%	67%	75%	67%	78%	78%	80%	80%	80%	80%	80%
										Average Success Rate:			76%		Total # English Natives:			12

TABLE 4.7: Success Rate of each exercise by native-English people.

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	11	14	13	15	12	10	13	14	11	11	15	11	6	15	13	12	14	14
Fails	4	1	2	0	3	5	2	1	4	4	0	3	4	0	2	3	1	1
Absents	0	0	0	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0
Success Rate	73%	93%	87%	100%	80%	67%	87%	93%	73%	73%	100%	79%	60%	100%	87%	80%	93%	93%
										Average Success Rate:			84%		Total # non-English Natives:			15

TABLE 4.8: Success Rate of each exercise by nonnative-English people.

People from the native-English countries (Table 4.7) like Australia, Canada, Ireland, New Zealand, UK and USA had worse success rate, meaning, commands that were successfully achieved, with an average of global success of 76% and only three exercises had a superior percentage of success than the nonnative-English countries (table 4.8) like Portugal, Poland, Germany, Holland and India, which had average global success rate out of all exercises of 84% with 13 exercises with superior percentage of success than the people from native-English Countries. Two of the exercises were tied. The result might be worse for the native-English speakers as they have a more complex way to say things, while those who are not native might say the commands in a more basic way. There were 12 English native people and 15 nonnative-English people out of all participants.

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	10	15	14	16	14	10	13	13	11	12	15	11	9	10	10	8	9	9
Fails	6	1	2	0	2	6	3	3	5	4	1	3	2	0	0	2	1	1
Absents	0	0	0	0	0	0	0	0	0	0	0	2	5	6	6	6	6	6
Success Rate	63%	94%	88%	100%	88%	63%	81%	81%	69%	75%	94%	79%	82%	100%	100%	80%	90%	90%
										Average Success Rate:			84%	Number of people with/taking bachelor's degrees:				16

TABLE 4.9: Success Rate of each exercise by people with or taking a bachelor's degree.

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	Ex 11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17	Ex 18
Successes	6	7	7	8	7	6	7	7	6	6	6	5	4	7	6	7	7	7
Fails	2	1	1	0	1	2	1	1	2	2	2	1	2	0	1	0	0	0
Absents	0	0	0	0	0	0	0	0	0	0	0	2	2	1	1	1	1	1
Success Rate	75%	88%	88%	100%	88%	75%	88%	88%	75%	75%	75%	83%	67%	100%	86%	100%	100%	100%
										Average Success Rate:			86%	Number of people with/taking master's degree:				8

TABLE 4.10: Success Rate of each exercise by people with or taking a master's degree.

On Tables 4.9 and 4.10 there is not a big difference between people with/taking a Bachelor Degree and people with/taking a Master's Degree, with an average success rate difference of 2% only, making it 84% and 86%, respectively. The difference between the number of exercises with biggest percentage on both academic levels is also not significance, while people that are taking/with bachelor's degree have four exercises with bigger percentage than the masters, where they have seven exercises compared to the bachelor ones.

Chapter 5

Conclusion

There are numerous dangerous scenarios throughout this world and developing a system to monitor and adapt to them, brings value to the whole world, specially to those in the field. With the developed work, it is possible to control swarms through the NLP System. It is competent enough to understand the most basic sentences to more complex sentences with 2 commands, regardless of one's knowledge, as long as he or she can speak or write English. Although the case-study of this thesis is monitoring fire, it is possible to adapt it into different environmental and non-environmental scenarios like monitor fire or to check damage of an earthquake, through a building analysis. While the robots are monitoring, everything that is happening on the field is shown on a dynamic map with people, cars, towers and, of course, the robots themselves. All of this was possible to accomplish with the Java program that was developed alongside this thesis.

The robotic system was developed to accept four actions, five formations and two general commands. It was also developed to represent the virtual Commanders, Robots, Cars, Towers, People and Environmental Nodes, while taking care of the movement of all of them, at the same time. All in the scope of bringing this to real robots in real life scenarios. The NLP system brings the easier interaction between the user and the robotic system just by writing in a natural way, as if he or she is talking to a normal person, but instead multiple robots. It is also capable of bringing context to the dialogue, in order to make the whole experience less exhausting to the user, as well as multiple commands in just one sentence.

5.1 Answering the Research Questions

This thesis aimed to answer three research questions that motivated all this work: if the developed system is capable enough to be adaptable to different environmental or non-environmental situations, if the user says the same thing in a different way, can the NLP System understand and finally if the previous system can work in a context way making the experience less exhausting. The answer to this questions can be found on the following three paragraphs.

Is it possible to develop a System capable of controlling swarms of robots, using Natural Language and to adapt this system to different situations?

Based on the way the system was originally made, from root, to be as open as possible to environmental or non-environmental situations and because the creation of nodes like the Environmental Node, the system is more than adaptable to different situations and so are all the drones, as long as, they are meant to monitor, not to act.

Can the Natural Language Processing System understand different ways of saying the same thing, as long as, it is on the context of the system?

The way that the NLP System was designed was to try to understand, as naturally as possible, all the users that interact with it. As the results show, there is a global average of success of the written commands in natural language of 82%, which means that, all the 30 people that participated in this test, around 25 could interact well with the program, while being completely unaware before about what the program was and what it did. With everyday usage, this percentage could potentially grow, specially while learning new ways of saying the same thing in a different way. In order to add these new things, a lexicon file was developed, so, to add it, one just needs to add to the right list, the new matching word.

Can the Natural Language Processing System perceive context?

Finally, the context work of this system works, as the results show with a success rate of 88%, making the experience less exhausting by not repeating every time who is the Commander that needs to receive the order, or repeating the same command for every Commander.

5.2 Future Work

Future work will address the speech-to-text functionality, as well as, getting more information out of a natural language sentence. Also bringing more actions, formations and general commands to expand the possibilities of the robotic system, like adding robots to a commander and, finally, test the program as a whole in a real scenario with real drones, people and different situations, like fire monitoring or sea monitoring.

Appendices

Appendix A

Human-Machines Communication Experience

A.1 Introduction to the Script

Hello! Thank you for helping me out on my Master Thesis on using Natural Language Processing to control a swarm of robots with the master-slave technique.

For this form you should reply to all the questions on a natural way. As natural possible. As if you are having a conversation with the system!

This system works with robots (drones) and, because there are a lot of drones operating, you need to talk to their master, the Commander. The Commander is responsible to hear your command and to redirect your command to all his robot-slaves. This way, you can control 20,100,2000 robots at the same time, just by speaking to one Commander!

There are three types of Commands a user can give to the Master Robot (Commander) or to the General system itself:

Actions - Which are directly related to the Master (also known as Commander or Group) on how he will walk inside the area he is limited to.

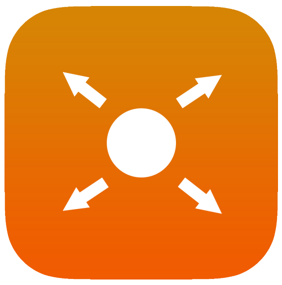
Formations - How the slaves will position themselves around the Leader.

General Commands - to add and remove commanders and, consequently, their robot slaves.

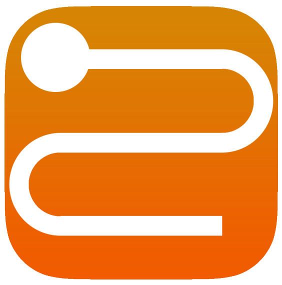
A.2 Actions - Related to the Commander

There are four possible actions. These actions are going to tell the Commander (also known as the Master/Leader) how he is going to walk inside his area.

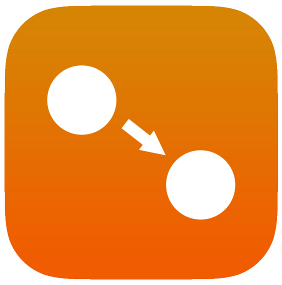
How would you ask the Commander 1 to walk freely on his area?



How would you ask the Commander 1 to walk on a lawn-mower way on his area?



How would you tell the Commander 1 to follow Commander 2?



How would you tell the Commander 1 to return to his base?

A.3 Formations - Related to the Robot Slaves

There are five possible formations and they are going to tell how the robots around the Commander will position themselves around him.

How would you tell the robots of the Commander 2 to be on an escort formation.

How would you tell the Commander 2 to make the robots circulate on himself?

How would you tell Commander 2 to make his robots make a line behind him?

How would you tell Commander 2 to make his robots triangulate?

How would tell Commander 2 to make his robots retreat?

A.4 General Commands - Add and Remove Commanders

The goal of this section is to tell the system to add a new commander and remove an existing one.

How would you naturally tell the system to add a new Commander?

How would you naturally tell the system to remove the Commander 2 from the map?

A.5 Try to mix the Actions and the Formations in one phrase!

Formations: Escort, Circulate, Line, Triangle and Retreat. Actions: Free, Lawn Mower, Follow and Return.

Commanders: Commander 1, Commander 2.

Mix actions and formations in one sentence to the Commander 1

Mix actions and formations in one sentence to the Commander 2

A.6 Context Exercise

The context is in this project to enrich the experience of the user. Instead of having the same sentence format: subject, followed by the verb, followed by the object, the user is allowed to say just one of the three grammatical forms, all by themselves. The system will then try to connect everything in the end!

So, in this exercise you can do for example:

Context phrase 1: Commander 1

Context phrase 2: line up

[The Commander 1 will change to line formation]

Context phrase 3: follow Commander 2

[The Commander 1 will follow Commander 2]

Context phrase 4: lawnmower action

[The Commander 1 will change to lawnmower position]

Context phrase 5: Commander 2

[The Commander 2 will change to lawnmower position]

Possible choices: Formations: Escort, Circulate, Line, Triangle and Retreat. Actions: Free, Lawn Mower, Follow and Return.

Commanders: Commander 1, Commander 2.

Context phrase 1

Context phrase 2

Context phrase 3

Context phrase 4

Context phrase 5

A.7 That's a wrap! Please provide us with anonymous information about yourself!

What's your nationality?

What's your age?

Gender

☐ Male

☐ Female

☐ Rather Not Say

Field of Study?

Education Level - high school, bachelor, masters, etc... (currently taking or concluded)

Are you working? If yes in which field of work?

Appendix B

Success Rate of the Experiments

	Actions			
	Free	Lawn Mower	Follow	Return
Successes	22	23	25	29
Fails	8	7	5	1
N/A	0	0	0	0
Success Rate	73%	77%	83%	97%

Mean of the Success Rate
83%

Table B.1: Complete Success Rate Table for Actions.

	Formations				
	Escort	Circle	Line	Triangle	Retreat
Successes	25	21	24	25	22
Fails	5	9	6	5	8
N/A	0	0	0	0	0
Success Rate	83%	70%	80%	83%	73%

Mean of the Success Rate
78%

Table B.2: Complete Success Rate table for Formations.

	General Commands	
	Remove	Add
Successes	22	25
Fails	8	4
N/A	0	1
Success Rate	73%	86%

Mean of the Success Rate
80%

Table B.3: Complete Success Rate table for General Commands.

	Double Command Sentences	
	Double Command 1	Double command 2
Successes	20	15
Fails	5	6
N/A	5	9
Success Rate	80%	71%

Mean of the Success Rate
76%

Table B.4: Complete Success Rate table for the Double Command sentences.

	Context Phrases				
	Phrase 1	Phrase 2	Phrase 3	Phrase 4	Phrase 5
Successes	21	19	18	20	19
Fails	1	3	4	2	3
N/A	8	8	8	8	8
Success Rate	95%	86%	82%	91%	86%

Mean of the Success Rate
88%

Table B.5: Complete Success Rate table for the Context phrases.

Bibliography

[Anand et al., 2014] Anand, A., Nithya, M., and Sudarshan, T. (2014). Coordination of mobile robots with master-slave architecture for a service application. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pages 539–543.

[Briggs and Scheutz, 2012] Briggs, G. and Scheutz, M. (2012). Multi-modal belief updates in multi-robot human-robot dialogue interactions. In *Multi-modal Belief Updates in Multi-Robot Human-Robot Dialogue Interactions.*

[Casteigts, 2015] Casteigts, A. (2015). JBotSim: a tool for fast prototyping of distributed algorithms in dynamic networks. In *Proceedings of the 8nd international ICST conference on simulation tools and techniques, SIMUTools'15, Athens, Greece.*

[Coppin and Legras, 2012] Coppin, G. and Legras, F. (2012). Autonomy spectrum and performance perception issues in swarm supervisory control. *Proceedings of the IEEE*, 100(3):590–603.

[Dey et al., 2015] Dey, K. C., Mishra, A., and Chowdhury, M. (2015). Potential of intelligent transportation systems in mitigating adverse weather impacts on road mobility: A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1107–1119.

[Draper et al., 2003] Draper, M., Calhoun, G., Ruff, H., Williamson, D., and Barry, T. (2003). Manual versus speech input for unmanned aerial vehicle control station operations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 47(1):109–113.

[Drews and Fromm, 1997] Drews, P. and Fromm, P. (1997). A natural language processing approach for mobile service robot control. In *Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control, and Instrumentation (Cat. No.97CH36066)*, volume 3, pages 1275–1277 vol.3.

[Elahi and Gschwender, 2009] Elahi, A. and Gschwender, A. (2009). Introduction to the ZigBee Wireless Sensor and Control Network. In *ZigBee Wireless Sensor and Control Network*, chapter 2.3 ZigBee.

[Eriksson E et al., 2015] Eriksson E, A., Ohlman, B., Persson, K., Malik, A. M., Ihlar, M., and Sunde, L. (2015). Scalable point-to-multipoint communication for cloud networking using information-centric networking. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 654–662.

[Guo et al., 2016] Guo, S., Li, X., and Guo, J. (2016). Study on a multi-robot cooperative wireless communication control system for the spherical amphibious robot. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 1143–1148.

[Howden, 2013] Howden, D. J. (2013). Fire tracking with collective intelligence using dynamic priority maps. In *2013 IEEE Congress on Evolutionary Computation*, pages 2610–2617.

[Jesús San-Miguel-Ayanz, 2018] Jesús San-Miguel-Ayanz, Tracy Durrant, R. B. G. L. A. B. D. d. R. D. F. P. M. T. A. V. H. C. F. L. P. L. D. N. A. C. A. T. L. (2018). Forest Fires in Europe, Middle East and North Africa 2017.

[Jia Li et al., 2012] Jia Li, He Xiao, and Dong Yi (2012). Designing universal template for database application system based on abstract factory. In *2012 International Conference on Computer Science and Information Processing (CSIP)*, pages 1167–1170.

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Mora et al., 2013] Mora, C., Frazier, A. G., Longman, R. J., Dacks, R. S., Walton, M. M., Tong, E. J., Sanchez, J. J., Kaiser, L. R., Stender, Y. O., Anderson, J. M., Ambrosino, C. M., Fernandez-Silva, I., Giuseffi, L. M., and Giambelluca, T. W. (2013). The projected timing of climate departure from recent variability. *Nature*, 502:183 EP –.

[Qian et al., 2008] Qian, Y., Yan, G., Li, Z., Duan, S., Zhang, R., and Kong, X. (2008). Retrieval of subpixel fire temperature and fire area using simulated hj-1b data. In *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages III – 836–III – 839.

[Ray, 2018] Ray, B. (2018). Comparing Mesh, Star & Point-To-Point Topology In IoT Networking.

[Remmersmann et al., 2012] Remmersmann, T., Tiderko, A., Langerwisch, M., Thamke, S., and Ax, M. (2012). Commanding multi-robot systems with robot operating system using battle management language. In *2012 Military Communications and Information Systems Conference (MCC)*, pages 1–6.

[Rossi et al., 2017] Rossi, A., Staffa, M., and Rossi, S. (2017). Supervisory control of multiple robots through group communication. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1):56–67.

[Thenmozhi et al., 2017] Thenmozhi, D., Seshathiri, R., Revanth, K., and Ruban, B. (2017). Robotic simulation using natural language commands. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–4.

[Thomason et al., 2015] Thomason, J., Zhang, S., Mooney, R., and Stone, P. (2015). Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1923–1929. AAAI Press.

[Velagapudi et al., 2008] Velagapudi, P., Scerri, P., Sycara, K., Lee, W.-H., , and J, W. (2008). Scaling effects in multi-robot control. In *International Conference on Intelligent Robots and Systems (IROS'08).*

[Ventura et al., 2012] Ventura, A., Diegues, N., and de Matos, D. M. (2012). Frame interpretation and validation in a open domain dialogue system. *CoRR*, abs/1207.4307.