Simulating Swarm Behavior for Surrounding a Target

Presented by:
Sindiso Mkhatshwa

Prepared for:
Jarryd Son
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering

November 11, 2020

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:........[illegible]............
S. Mkhatshwa

Date:......11 November 2020........

Acknowledgments

- Firstly, I would like to thank my supervisor, Jarryd Son, for his invaluable guidance through this project.
- To my father, Samuel, thank you for believing in my potential and always pushing me to be the best version of myself. You are a great man kind sir.
- To my best friend, Bonolo, I do not see how I would have made it through the last four years without you man. You went beyond your call of duty as a friend in many ways than I can remember. Thank you my brother.
- To the rest of my family, thank you for the support you showed me through out this project. Love and blessings la familia.
- Last but not least, a special thanks to Dr Jordan Peterson for the daily words of encouragement and wisdom.

Abstract

Swarm robotics is a sub-field of multi-agent robotics that employs simple rules to coordinate populations of mobile robots to perform tasks cooperatively. The individual robots generally have limited capabilities and exhibit very simple behavior while complex behavior emerges at the population level from interactions within the swarm and between the swarm and the environment. These interactions enable the swarm to perform tasks that would be impossible to achieve using an individual robot. This project explores this swarm capability by simulating swarm behavior for surrounding a target. The chosen use case for this project is the rapid containment of an oil spill in the open seas.

Marine oil spills occur very often during the production and transportation of crude liquid petroleum oil and other derived oils. These incidents result in heavy water pollution and threaten the survival of endangered species in marine ecosystems. They therefore require fast and eco-friendly response actions in order to reduce environmental damage. However, current response actions, such as in-situ burning, can be very inefficient because they rely almost entirely on human supervision and favourable weather conditions.

Swarm robotics, on the other hand, posseses unique characteristics that promise to offer an efficient, fast and eco-friendly response action in order to prevent cataclysmic impacts on marine ecosystems. As such, this project implements and evaluates a swarm robotics system for the purpose of containing an oil spill in the open seas. Experiments with simulated robotic swarms of up to six robots show that behavior-based cooperative intelligence can be used to coordinate swarms to obtain robust and scalable surrounding behavior using an algorithm that is based on a local communication strategy.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Background to the study

Swarm robotics draws inspiration from the self-organized behaviors of social animals [1] such as ant colonies [2], bee colonies [3], bird crowds [4] and fish schools [5]. There is a rich literature that shows that self organisation in social animals is achieved through distributed communication. However, the strategies for implementing distributed communication differ from one animal group to another.

Although the communication strategies differ, the emerging collective behavior predominantly exhibits robustness, scalability and adaptability across all the social animals [1, 6, 7, 8, 9]. These qualities make swarm robotics attractive across a wide range of application domains. One application domain which is of interest in the field is that of tasks which are too dangerous for human beings.

This work seeks to build on previous attempts at advancing the practical applications of swarm robotics in this domain by evaluating the effectiveness of target surrounding swarm algorithms. Target surrounding behavior has many practical use cases such as search and rescue, and the containment of hazards such as oil spills in the open seas. This work focuses on the latter.

Marine oil spills occur often during the production and transportation of crude liquid petroleum oil and other derived oils. These incidents can have cataclysmic impact on marine ecosystems and result in the loss of aquatic biodiversity, as seen in recent years [10], and they consequently require fast, effecient and eco-friendly response actions in order to reduce environmental damage (see figure 1.1). Current response actions, such as in situ burning, can be harmful to the environment and rely almost entirely on human oversight which can be very inefficient because when sea conditions (e.g. a sea storm) become too dangerous the operation is halted for a later time when it is safe again for humans.

With robot swarms however, once deployed they can fulfill their mission objectives with little to no human intervention at all. This removes the human in the loop. Moreover, swarms are inherently robust and therefore capable of operating reliably well (although performance may decrease after losing many individuals) even during extremely dangerous conditions. Therefore robot swarms have the potential to do well in this task.

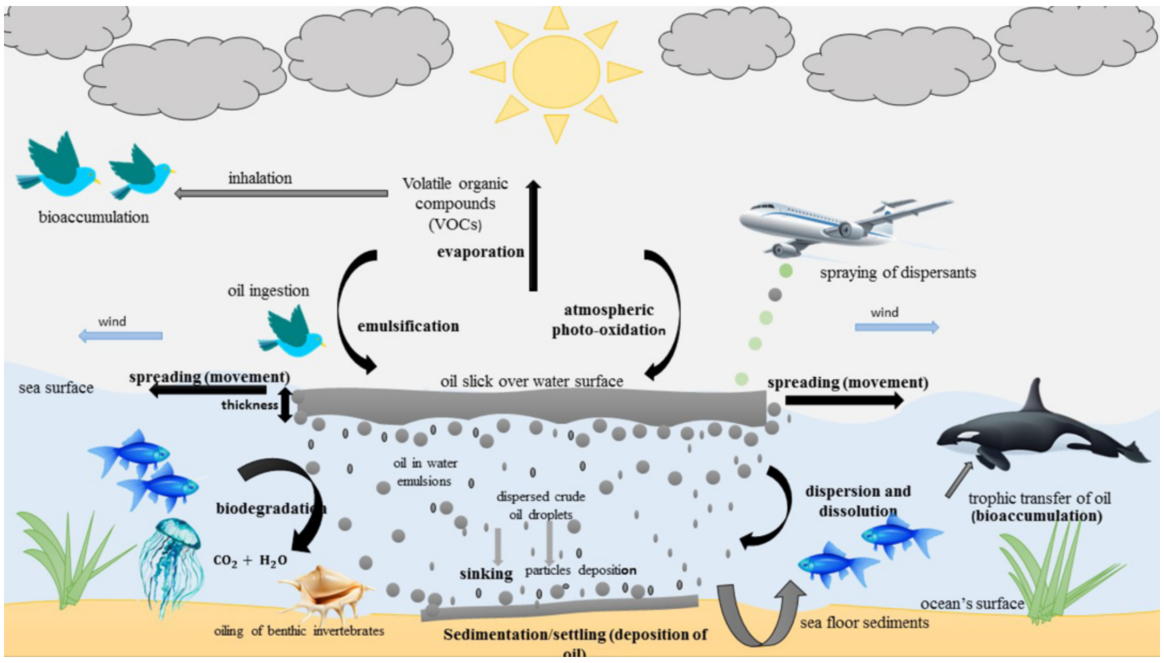


Figure 1.1: An illustration of oil weathering processes (spreading, dispersion, evaporation, dissolution, emulsification, sinking, sedimentation and biodegradation) as well as the visible effects of these processes on the marine ecosystem [10].

1.2 Objectives of this study

1.2.1 Problems to be investigated

This research seeks to address the following question, *what are the core methodological requirements for implementing a swarm exhibiting efficient surrounding behavior for the purpose of containing an oil spill in unknown or changing sea environments?*

Answering this question will require tackling the following sub-questions:

- How can continuous adaptation to the loss of individuals (possibly because of changing environmental conditions) be obtained in a swarm?
- How can multiple robots be coordinated if communication is limited?
- What is an effective method of implementing distributed communication in a swarm?
- How does varying the population size of a swarm affect performance?
- How does the target sensing and communication range of the individual robots affect performance at the swarm level?

1.2.2 Purpose of the study

This study seeks to simulate a swarm of robots that exhibit surrounding behavior, and to investigate how this behavior can be used to provide a robust and scalable response action to contain an oil spill in changing or dynamic sea environments. This will involve investigating an effective method of implementing distributed control of multiple agents of a swarm when communication is limited as well as an effective method of implementing distributed communication. A method of self-stabilizing the swarm will also be investigated. The aim is to identify simple behaviors at the robot level that, through local interactions, will lead to the emergence of surrounding behavior at the swarm level. An investigation will then be made of how varying the population size and the loss of agents in the middle of an operation affect performance. Lastly, an investigation will be made of how the target sensing and communication range of individual robots affect the overall system performance.

1.3 Scope and Limitations

This work focuses on the design and implementation of a swarm robotics system with the specific goal of surrounding a static oil spill. It is assumed that the target occupies a single coordinate and that communication between the robots is instantaneous. The robot model used in the simulation is designed from scratch; however, the project does not discuss swarm robotics hardware. Instead great focus is given to the design and order of integration of the various local behaviors of the individual robots such that surrounding behavior emerges at the swarm level from local interactions within the swarm and between the swarm and the environment.

In terms of limitations, the project is expected to be completed within 13 weeks with a budget of R1500. Lastly, this project is undertaken in the remote learning context due to COVID-19.

1.4 Plan of development

This report will progress as follows, section 2 formulates the user and technical requirements of the system. This is followed by section 3, which presents a brief background theory of swarm robotics. Section 4 presents an in depth review of four target surrounding swarm algorithms. Section 5 begins with a technical comparison of the four algorithms, from which one is selected to implement the system. This is followed by a brief review of the most widely used swarm robotics development platforms and lastly, the subsystems of the system are identified and designed. Section 6 describes how the various subsystems of the system are implemented. Section 7 presents the design of subsystem and system level tests and experiments. In section 8 the results of the various test and experiment designs are presented. A brief discussion of these results is given in section 9. Thereafter, conclusions of this work are presented in section 10 and lastly, recommendations for further work are laid out in section 11.

Chapter 2

Requirements Analysis

2.1 System Description

2.1.1 High Level Description

This project aims to simulate swarm behavior for surrounding a target. The chosen use case for this behavior is the rapid containment of an oil spill in open sea environments. As such, the swarm robotics system for this purpose needs to be able to operate with little to no human intervention at all, i.e. autonomous, the system must also be able to operate realibly in the costantly changing environments of the oceans.

The rest of this section is dedicated to formulating system requirements, both at the user level and at the technical level, to realise swarm behavior to solve the stated problem. In order to be able to ascertain that the final system behavior meets the requirements, a set of acceptance test procedures (ATP) is also formulated and presented towards the end of the chapter.

Each user requirement will be given a unique identifier of this format[Swarm Robotics System-User Requirement-Number]; SRS-URM-XX. In a similar fashion, each functional requirement will be assigned a unique identifier of this format [Swarm Robotics System-Functional Requirement-Number]; SRS-FRM-XX. Lastly, each ATP will be assigned a unique identifier of this format [Swarm Robotics System-Acceptance Test Procedure-Number]; SRS-ATP-XX. This makes referencing and linking the requirements with the ATPs easy and systematic.

The user requirements are presented first followed by the functional requirements and then the ATPs.

2.1.2 User Requirements

From the system description given above, the user requirements of the system can be formulated. The user requirements are presented in table 2.1 below.

URM ID	**Refined By**	**URM Description**
SRS-URM-01	SRS-FRM-01	The robotic swarm must be able to locate an oil spill.
SRS-URM-02	SRS-FRM-02	The robotic swarm must be able to surround a static oil spill. The oil spill only needs to be surrounded as if it were a 2D problem.
SRS-URM-03	SRS-FRM-03	The swarm robotics system must be self-stabilizing (self healing).
SRS-URM-04	SRS-FRM-04	The robotic swarm must operate reliably in changing environments (robust), even if an individual malfunctions, this should not result in catastrophic failure of the whole swarm.
SRS-URM-05	SRS-FRM-05	The robotic swarm must be scalable.

Table 2.1: System user requirements.

2.1.3 Functional Requirements

From the user requirements given above, the functional requirements of the system can be formulated. The functional requirements are presented in table 2.2 below.

FRM ID	Refines	FRM Description
SRS-FRM-01	SRS-URM-01	The robotic swarm must have a method by which the position of an oil spill can be determined, i.e. sense target within range.
SRS-FRM-02	SRS-URM-02	Individuals of the swarm must be autonomous, i.e. be capable of interacting and moving in their environment.
SRS-FRM-03	SRS-URM-03	The algorithm controlling the robotic swarm must be oblivious, i.e. move computations must only depend on the current status of individuals in the system.
SRS-FRM-04	SRS-URM-04	The control of the swarm must be distributed, i.e.each robot must execute the same algorithm independently of other robots. Robots must only receive information as input to their control algorithm from neigboring robots.
SRS-FRM-05	SRS-URM-05	Communication within the robotic swarm must be distributed to ensure that there is never a communication bottleneck for large populations. The robots must only share information with members within their communication range.

Table 2.2: System functional requirements.

2.1.4 Acceptance Test Procedures

From analysing the system description, user requirements and functional requirements of the system above, the acceptance test procedures given in table 2.3 below are formulated.

ATP ID	**Refines**	**Acceptance Condition**
SRS-ATP-01	SRS-FRM-01	The individuals must be able to sense the target when it is within a predetermined sensing range.
SRS-ATP-02	SRS-FRM-02	The individuals of the swarm must be capable of moving around the environment autonomously, and avoid collisions.
SRS-ATP-03	SRS-FRM-03	The individuals of the swarm must be able to exchange information with neighboring robots when within communication range.
SRS-ATP-04	SRS-FRM-04	When individuals suddenly quit (e.g. malfunction) or join in the middle of an operation, this should not result in catastrophic failure of the system.
SRS-ATP-05	SRS-FRM-05	The swarm robotics system should perform relatively well with varying population sizes.

Table 2.3: System acceptance test procedures.

Chapter 3

Theory

This section presents the basic theory of swarm robotics; from characteristics to collective behaviors to real world applications. It will progress as follows; section 2.1 gives a concise definition of swarm robotics, section 2.2 deliberates on the characteristics of swarm robotics systems, section 2.3 gives some predominant basic collective behaviors of robotic swarms which can be combined to obtain complex behaviors that can be used to tackle real world tasks and lastly, section 2.4 presents four real world applications of swarm robotics which utilise the surrounding behavior.

3.1 What is Swarm Robotics?

Many researchers in the field have attempted to come up with a formal definition of swarm robotics. In fact according to [7], the term swarm robotics has turned into a 'buzzword' and consequently, it is imperative that we also distinguish and clearly state what swarm robotics is and what it is not and most importantly, clearly identify the definition that we will use in this project.

In [1], swarm robotics was defined as a "new approach to the coordination of multi-robot systems which consists of large numbers of mostly simple physical robots", and an almost similar definition was given in [9] as, "a novel approach to the coordination of large numbers of robots."

A slightly different definition was given in [7] as the study of "how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and the environment." This definition has been widely used in most predominant literature. Although widely accepted, it is not without its own flaws. For instance, how large does the number have to be to constitute a swarm, i.e. what is the threshold? Secondly, some works [6, 1] argue that the individuals do not necessarily have to be simple; modern and future swarms are likely to be sophisticated and highly capable. Although this is true, the last argument is however not as strong as the first because the underlying issue vis-a-vis simplicity is that the intelligence of the individuals is quite simple compared to the emerging group intelligence.

All the definitions presented above share the same inherent flaws or ambiguity. However, a completely different definition (in terms of clarity and specifics) is given in [6], "A robot swarm is a group of three or more robots that perform tasks cooperatively while receiving limited or no control from human operators." Since this work aims to simulate swarm behavior for surrounding a target using at least three robots, this definition is inherently in accord with the work presented in this report. And consequently, this is the definition that we adopt in this report.

To further solidify the understanding of swarm robotics and its concepts, the next section presents predominantly accepted characteristics that distinguish swarming robots from individual robots and other multi-agent systems, such as sensor networks.

3.2 Characteristics of a Swarm Robotics System

Some literature uses the terms 'multi-robot systems' and 'swarm robotics systems' interchangeably. However, swarm robotics is a very specific subdomain of multi-robot systems. The main differences between the two are population size, control, homogeneity and functional extension. This section does not do a direct comparison of swarm robotics with the other multi-agent systems (such as sensor networks and multi-agent systems), but instead it presents the above features as they apply to swarms. A direct comparison of that nature is beyond the scope of this report.

It is important to be cognisant of the fact that swarm robotics draws inspiration from social animals, and consequently the group of robots exhibits special features which characterise social animals [11]. These features are robustness, flexibility and scalability at the swarm level. Furthermore, swarm robotics systems are energy efficient, economical and homogeneous, as elaborated below:

Robustness: Robustness is achieved through decentralized control, distributed sensing and redundancy [7]. Decentralized control means that the control of the system does not depend on one specific part or individual of the system, i.e. each robot executes its own control algorithm.

Redundancy means that the system can compensate for malfunctioning individuals, i.e. the robots are unsophisticated [6] and highly replaceable. This means that individuals can quit and join at any time and the objective is still achieved. As such, robustness capacitates the system to maintain relatively constant performance despite loss of individuals and disturbances in the environment (this makes swarm robotics systems suitable for solving tasks in dangerous and dynamic environments). However, performance degrades [1] inevitably with fewer robots.

Flexibility: This is achieved through redundancy, simple individual behaviors and mechanisms - such as task allocation [12]. These qualities allow the swarm to create complex behavioral patterns according to the demands placed by the environment and the nature of the task at hand. Consequently, flexibility allows the system to perform a wide variety of tasks in different environments.

Scalability: This is the ability of the system to cope relatively well with varying population sizes, and this is achieved through distributed sensing and communication. This removes the bottleneck that could be introduced by a centralised communication or sensing component in the system when large numbers of individuals are introduced to the system. Furthermore, distributed sensing results in an increased signal-to-noise ratio [7].

Energy efficient: The individuals of the swarm are relatively small and simple, with a much smaller energy cost than a complex single robot designed for a specific application. And consequently, in the absence of a recharge station this results in a longer life time of the swarm.

Economical: Since the individuals have limited capabilities, the costs of design, manufacture and maintenance are quite low compared to a complex single robot. The whole swarm population is generally cheaper than a single robot.

Homogeneous: This means that all the individuals execute the same program with very few or without any role divisions.

Autonomous: The robots of the swarm are capable of interacting with the environment autonomously. This is why sensor networks can not be classified as a swarm [7] because sensor networks are not capable of interacting with the environment autonomously.

Relatively simple: The individual robots are not unwise as such [1], nonetheless they are relatively simple when compared to the global intelligence emerging from the local interactions. Moreover, these individuals are normally small and cheap.

Limited sensing and communication range: Due to the low hardware costs, the robots generally have a limited sensing and communication range. This is, however, desirable because it enhances the scalability and flexibility of the system, unlike global communication which results in a reduction of these features.

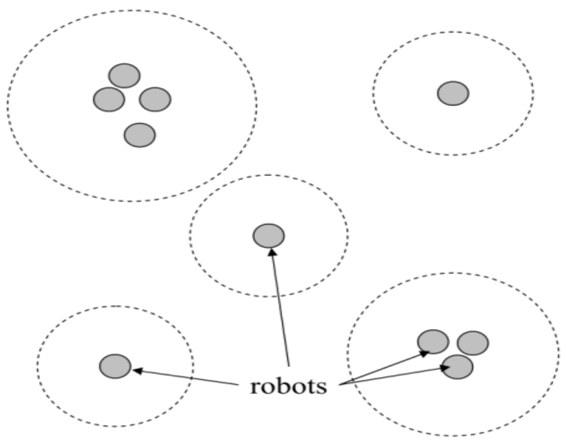
3.3 Collective Behaviors

This section reviews some common basic swarm collective behaviors which when combined could give rise to real world applications. The behaviors are presented according to the classification given in [9].

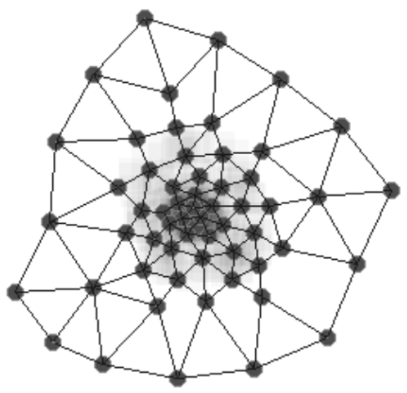
Spatially-organizing behaviors

Spatially-organizing collective behaviors are concerned with the organization and distribution of a swarm of robots in the environment in which they operate (see figure 3.1). These include the following subdomains:

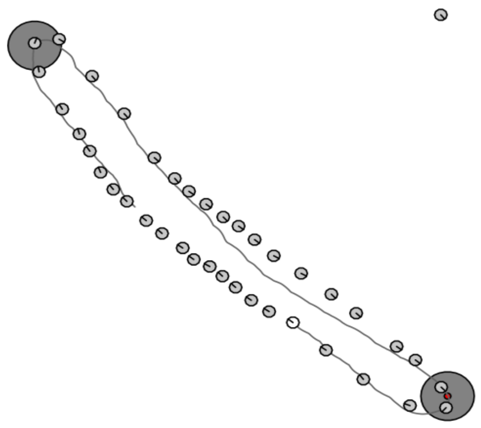
- **Aggregation:** allows the individual robots to come together to form clusters, possibly to allow more complicated interactions to take place(subfigure 3.1(a)).
- **Pattern formation:** arranges the robots such that they form a specific shape (subfigure 3.1(b)).
- **Chain formation:** a special case of pattern formation in which the robots form a line (subfigure 3.1(c)).
- **Self-assembly and morphogenesis:** robots are connected together to form structures (this can either be physical or virtual structures, e.g. communication links [13])(subfigure 3.1(d))



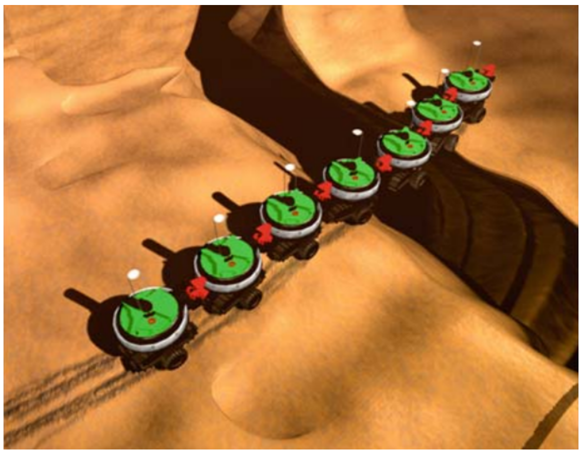
(a) An example of aggregation behavior obtained using probabilistic finite state machines (PFSM) design [14].



(b) An example of pattern formation behavior obtained via virtual springs interactions between robots [15].



(c) An example of chaining behavior obtained using artificial evolution [16].



(d) Robot collectives crossing a trough using self-assembly behavior [17].

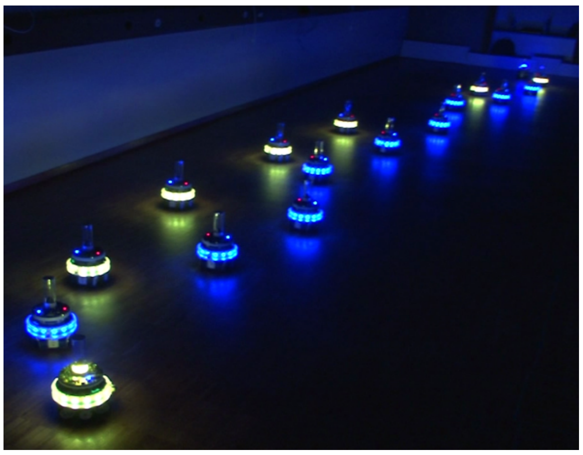
Figure 3.1: Spatially-organizing collective behaviors

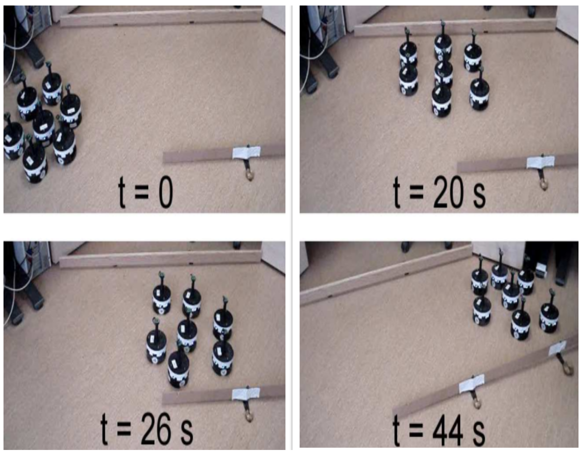
Collective navigation behaviors

Collective navigation behaviors deal with movement coordination of a robotic swarm (see figure 3.2).

- **Collective exploration:** includes area coverage and swarm-guided navigation. Virtual physics-based design approaches are usually used to implement area coverage while distributed communication is the preferred design approach for swarm-guided navigation (subfigure 3.2(a)), often taking inspiration from nature [9].

- **Coordinated motion:** ensures a swarm of autonomous robots can move around in specific formations without colliding with each other (an example of this behavior is shown in subfigure 3.2(b)).
- **Collective transport:** allows the swarm to move prey or objects that are too big to be moved by an individual robot.


(a) A dynamic chain as an example of swarm-guided navigation behavior [18].



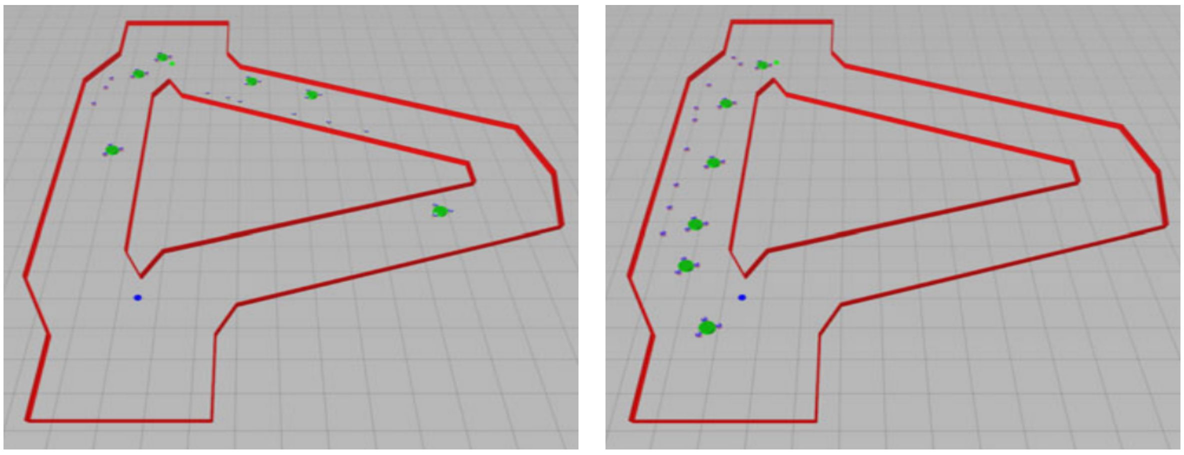
(b) A swarm of 7 self-organised flocking Kobots as an example of coordinated motion [19].

Figure 3.2: Collective navigation behaviors

Collective decision-making behaviors

Collective decision-making behaviors allow individuals of the swarm to settle for one common choice in any given situation.

- **Consensus achievement:** lets the individuals of the swarm agree on one given choice given many different options. Figure 3.3 shows a swarm of robots with the goal of transporting an object from a start position to a target position (blue dot on the diagram) in groups of three. The swarm has to choose between two different paths in order to reach the target. The one path is shorter than the other and consequently results in the swarm reaching the target faster. This is the path that the swarm eventually agrees on.
- **Specialization:** allows the swarm to dynamically partition itself to tackle different tasks in the environment.



(a) In progress. (b) Consensus achieved.

Figure 3.3: An example of consensus achievement in a swarm [20].

3.4 Real world applications

By combining the collective behaviors of the previous subsection, swarm robotics can be applied in a wide range of application domains such as tasks that require cheap designs, like agricultural foraging and mining tasks [21], to tasks that are dangerous to human beings [6], like target searching and post-disaster relief. However, despite immense research in the field, literature on swarm robotics applications remains very light. Real world swarm applications usually just use certain parts of swarm algorithms owing to the fact that the performance of swarm behavior emerging from local interactions can be difficult to predict, quantify or evaluate [13]. In this section we review aquatic and aerial applications of swarm robotics that utilise surrounding behavior.

3.4.1 Aquatic

Oil spill containment

Seaswarm is the code name for a swarm of robots developed at MIT for cooperative sea skimming and oil collection. Robots use a GPS to communicate their positions with the rest of the swarm. Each individual of the swarm is 5 meters long and 2 meters wide and consists of a nanowire conveyor belt to collect crude oil on the water surface (see figure

3.4). The robots use solar panels for propulsion and can clean continuously for a week with just 100 watts of power [22].

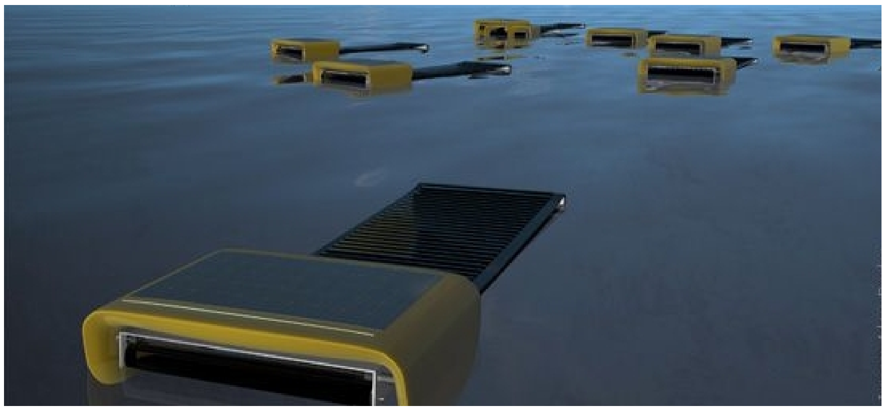


Figure 3.4: MIT's Seaswarm for cleaning oil spills in the sea [22].

The individuals spread over the oil spill contours and suck the crude oil over the water surface while making their way into the spill. In one implementation the individuals burn the oil on the spot while in another implementation they take breaks to deposit the oil elsewhere. Futhermore, the conveyor belt is stable, i.e. adheres to the water surface, and therefore does not capsize unlike traditional boat skimmers which tend to capsize very often because of the lack of adherance to the water surface. Importantly, the individuals can suck in oil up to 20 times their weight.

Military applications

The US Navy has successfully implemented and tested escort and swarming attack behavior on a swarm of 5 unmanned 2 meter by 4 meter boats. The swarm was controlled through the Control Architecture for Robotic Agent Command and Sensing (CARACaS) system. CARACaS was originally developed for NASA's Mars Rover program. This system enables the boats to plan their individual paths to reach their destinations while avoiding collisions with obstacles and other boats.

The individuals react to their environment autonomously and share their individual radar views with other boats to increase perception accuracy and situational awareness. In one of the tests conducted by the US Navy, the swarm was assigned a mission of esccorting a manned ship of high value from attackers. In the absence of intruders, the boats simply formed defence lines around the ship to guard it from attacks. However, upon detection of an intruder, the vessels left the high value ship and surrounded the unknown possibly enemy vessel just like how a whale encircles prey (see figure 3.5).

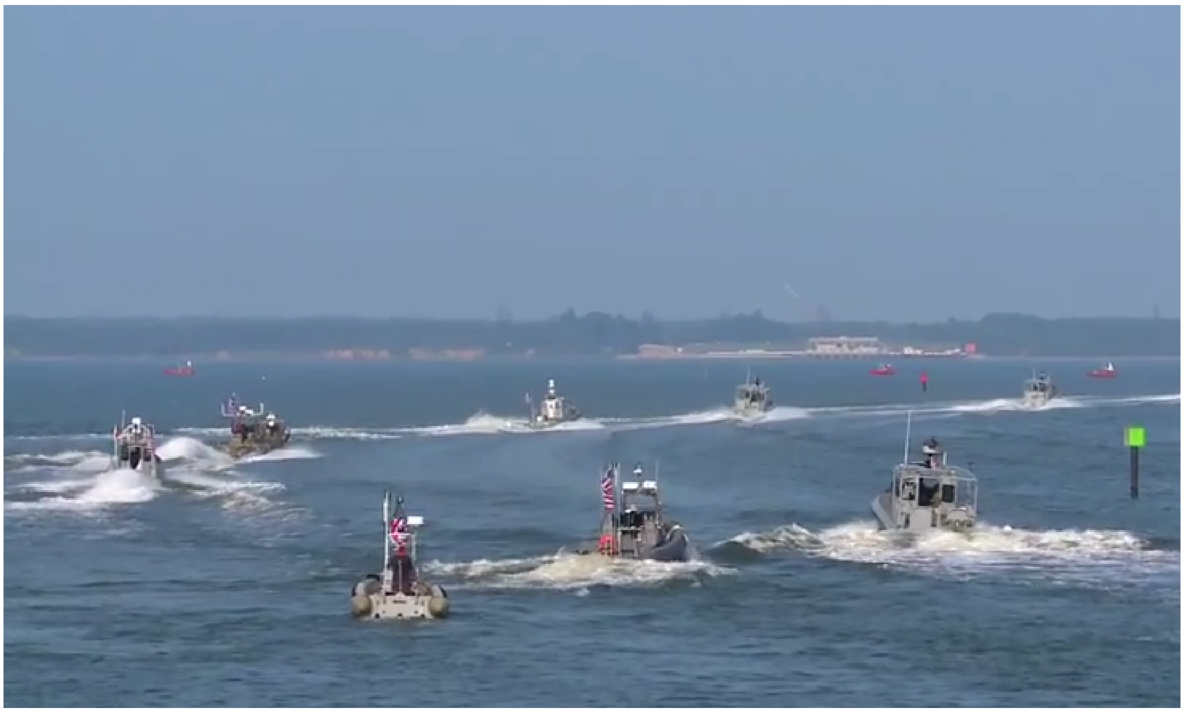


Figure 3.5: Unmanned boat swarm encircling an intruder vessel [23].

3.4.2 Aerial

Search and rescue

An autonomous swarm of flying robots was implemented in [24], through simulation, with the goal of gathering situational awareness data in the first couple of hours following a major natural disaster.

The swarm was controlled through behavior-based cooperative intelligence which integrated the following set of reactive behaviors; collisions avoidance, battery recharge, formation control, altitude maintenance and search methods.

The simulation implementation consisted of swarm sizes of 1 to 20 robots. It utilised pragmatic location data, post-disaster satellite imagery and pragmatic locations of buildings and victims based on personal interviews and encounters. The swarm successfully located 90% of the survivors within an hour (see figure 4.1).

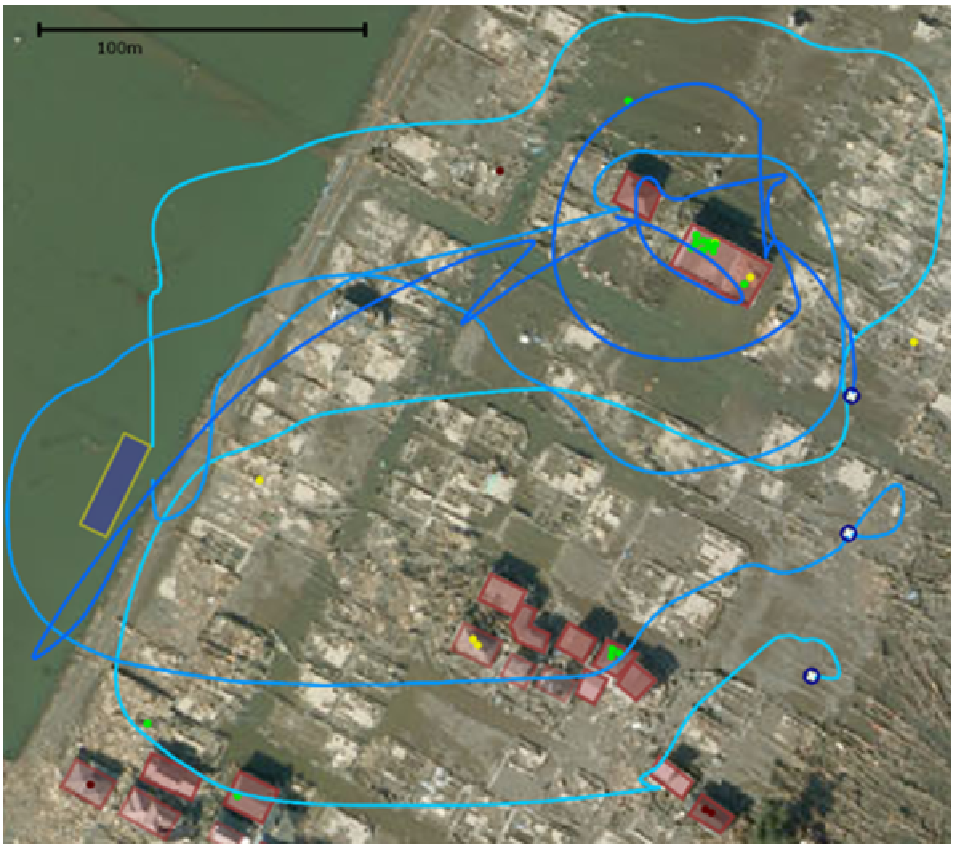


Figure 3.6: The paths traced out by three UAVs which were all launched from the blue rectangle. Importantly, note the circular pattern on the top right corner following a UAV successfully locating the survivors (green dots) on top of the building [24].

Perdix autonomous swarm of the Pentagon

Perdix is a bird sized, inexpensive drone that was developed by students at MIT, funded by the Pentagon [25]. It is inspired by the commercial smartphone industry. Its wings are made of carbon fiber and the fuselage is a composite of kevlar. It has one rear-facing push propeller which is powered by a lithium polymer battery pack.

The drones work in swarms of 20 or more cooperatively to achieve a goal. During one particular test the swarm was released from two fighter jets with two missions, hovering over a target and forming a 100-meter-wide circle in the sky. This test sought to display three collective behaviors of the UAV swarm; collective decision-making, adaptive formation flying and self-healing.

Chapter 4

Literature Review

This section reviews four swarm algorithms that have been used to tackle the target surrounding problem. Each algorithm is identified, and then its key concepts are presented, followed by a detailed description of the algorithm and its results and performance analysis, and lastly a critical analysis of the applicability of the algorithm to the chosen use case of this work concludes the discussion of the algorithm.

4.1 Potential field based approach

In [26], a distributed potential field based target surrounding algorithm for robot swarms was proposed. The algorithm is based on a two dimensional potential field and can surround a stationary and moving target. The robots maintain dedicated formations while circulating around the target in order to ensure the target does not escape. The key concepts of this algorithm are described below.

4.1.1 Key Concepts

This algorithm uses the concept of multiple-orbit surrounding in order to ensure the target being surrounded does not escape. Firstly, this approach defines multiple orbital trajectories around a target on which robots can move. It then defines a heading direction on each trajectory which serves to minimize inter agent collisions, to enable pincer movements and to speed up the surrounding process. The movement of robots in the desired direction is enforced by using gradient vectors of the potential field.

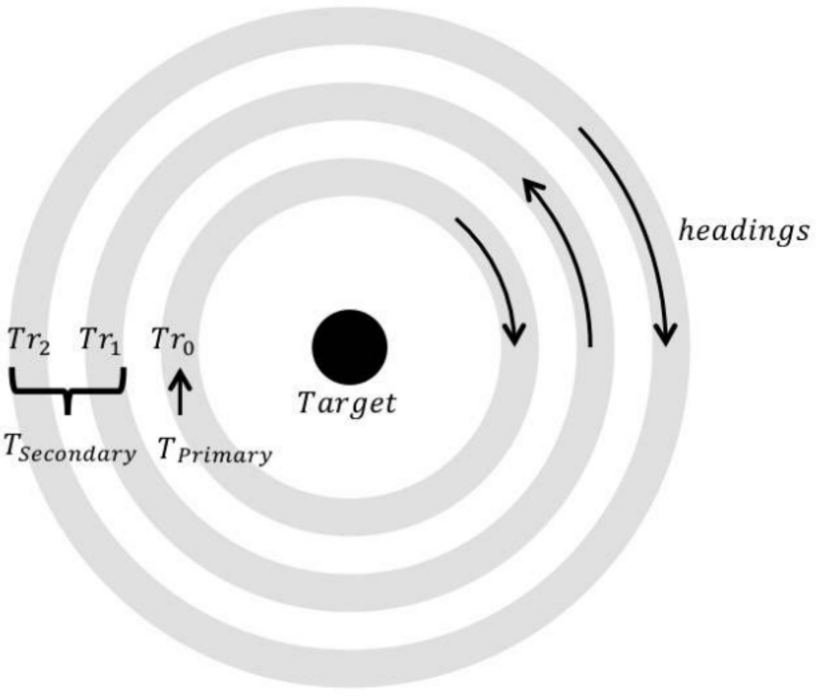


Figure 4.1: A target and three orbital trajectories around it [26].

There are two types of trajectories; primary and secondary (see figure 4.1). Radial movements are possible between any two trajectories (only towards the target). The potential of robots decreases as they approach the target. Robots that do not fall in either the primary or secondary trajectories do not make tangential movements around the target.

Primary trajectory: this is the goal trajectory for all robots. It is the closest trajectory to the target. Once robots reach this trajectory, they can no longer make radial movements, but instead they constantly circulate around the target in the given heading direction. Importantly, there is only one primary trajectory.

Secondary trajectory: this is any other trajectory other than the primary trajectory. It is further from the target than the primary trajectory. If a robot is making a radial movement (i.e. passing through a secondary trajectory) and suddenly encounters another robot infront of it, then it just circulates the target on the current trajectory. This prevents collisions while also speeding up the surrounding process. Robots in the secondary trajectory are always attracted to holes in the primary trajectory, i.e. in the absence of a blocking robot in the immediate inner trajectory, robots will always move towards the primary trajectory.

Secondly, a two dimensional potential function is used to generate the potential field. The gradient of the potential function is used as a pure attraction function to attract the robots towards inner trajectories, and the repulsion function, which is used to keep the robots from getting too close to the target, is obtained by inverting the direction of the gradients.

Lastly, perpendicular gradients of the potential function are used to keep the robots in circulation on the defined trajectories.

4.1.2 Description of algorithm

Each robot checks if its current position is on the primary trajectory or if there is a neighbor in front of it with a higher priority that prohibits a step towards the target. If any of the two checks evaluate to true then the robot circulates around its current trajectory according to the given trajectory heading direction. Otherwise if both of the checks are false the agent takes a step towards the target. This is repeated until convergence. Convergence occurs when an agent reaches the primary trajectory

4.1.3 Results and performance analysis

Experiments for this approach were carried out on the Virtual Robot Experimentation Platform (V-REP). The experimentations were carried out in 2D and the robots were equipped with infrared distance sensors.

Two implementation approaches of the algorithm were investigated, one where the robots could only circulate the target in one direction and one in which different headings were allowed. Results show that the former takes longer to converge than the latter.

Lastly, a proof was given that the robots never get stuck in a deadlock situation as well as a proof that the target surrounding always converges. However, we find that the deadlock proof is not as watertight as claimed by the authors - this is explained below.

4.1.4 Analysis of the algorithm

The algorithm assigns identifiers to individuals of the swarm, i.e. robots are not anonymous. The identifiers are shared via a communication interface to neighboring robots who are within a predetermined communication range. The assignment of identifiers to robots enables the algorithm to assign a priority status to individual robots. Priority statuses are used to break move ties.

The problem with this approach is that if a robot with a higher priority status suddenly malfunctions or quits in the middle of a move tie, neighboring robots with lower priority statuses could possibly get stuck in a deadlock. This could result in a catastrophic failure of the system and consequently, this means that the algorithm is not robust.

Another problem with this approach is that it assumes that the target location is known, which is not realistic.

4.2 Virtual viscoelastic forces approach

In [27], a physics inspired approach that can be used by robot swarms to self-organise into a circle formation of a predetermined radius is proposed. The approach employs virtual viscoelastic links between neighbors and also makes use of the circumscribed circle theory to coordinate a group of robots to form the required circle formation. Although this approach was not directly proposed for the target surrounding problem, it is still considered because of its relevance to the problem.

4.2.1 Key Concepts

The viscoelastic links between neighbors are used as attractive primitives and repulsive primitives to avoid collisions between robots and obstacles in the environment. A Voigt model is used to model the viscoelastic links. This model consists of a Hookean elastic spring and a Newtonian damper connected in a parallel configuration.

Attractive primitive: This primitive keeps and arranges robots together by applying virtual viscoelastic links (VVL) between neighbors. Each VVL exerts a uniform virtual force, F_{VVL} , on both the linked robots.

Repulsive primitive: This primitive serves to avoid collisions between robots and obstacles in the environment. A repulsive potential force is used to implement this primitive; the potential force is generated by each robot and acts only in the surroundings of the generating robot. The total influence on one robot from multiple robots is simply the resultant force, $\mathrm{F}_{\mathrm{VRP}}$, of the repulsive forces acting on that robot.

Circumscribed circle theory: A circumcircle of a polygon is a circle that connects all the vertices of that polygon. The circumcircle shares the same center as the regular polygon and its radius is equal to the radius of the polygon.

4.2.2 Description of algorithm

The circle is formed by adding the forces due to the primitives defined above. The force due to the attractive primitive, $\mathrm{F}_{\mathrm{VVL}}$, is the most dominant because the force due to the repulsive primitive, $\mathrm{F}_{\mathrm{VRP}}$, only contributes towards avoiding obstacles. The system is guaranteed to stabilize when all forces exerted on the robots are balanced. Thereafter, the system converges to a stable state - where robot velocities become zero. This is guaranteed for any viscoelastic mesh for as long as a damping coefficient with a magnitude greater than zero is used [28].

4.2.3 Results and performance analysis

The ARGoS simulator was used to evaluate the performance and robustness of the algorithm. The following performance evaluation metrics were used:

- *Group speed (GS)*: this refers to the magnitude of velocity of the center of gravity of the swarm.
- *Mean Distance Error (MDER)*: this metric indicates the inter-robots distance error as an average of all robots and neighbors.

The evaluation of robustness was performed by adding Gaussian noise to the range and bearing sensors. The results presented showed that even in the existence of noise the robots could keep a constant MDER and a tiny variable GS.

4.2.4 Analysis of the algorithm

This algorithm was shown to be very effective and robust for a small number of robots. However, when the number of robots was increased, it took very long for the system to stabilize due to hard computations of the distances between neighbors. Swarm robotics tend to have limited battery life, therefore this situation is not desirable in real world applications.

Furthermore, this algorithm may require a robot to be positioned at the center of the circle being formed which may not be desirable nor pragmatic when surrounding a target in the real world.

4.3 Communication-based approach

4.3.1 Sources of inspiration

The Suzuki model: The Individual Broadcast of Coordinates (IBC) algorithm was first proposed in [8] as an adaptation of another model by Suzuki [29], the Suzuki model. The Suzuki model is sensor based, and assumes unlimited visibility which makes it not scalable, i.e. large numbers of robots cannot not be used in practice.

Batch Broadcast of Coordinates (BBC) model: A transitional model from the Suzuki model to the IBC model was proposed in [8]. The proposed model is called a Batch Broadcast of Coordinates (BBC) and employs explicit wireless communication. Each robot can broadcast and receive coordinates from other robots over a limited range. This makes it a bit more realistic than the Suzuki model.

However, this model is memory intensive and computationally expensive because every time a robot moves it broadcasts its new and previous coordinates and a list of new and previous coordinates of other robots received by it to its neighbors. For large swarms this can quickly become problematic, making the model not scalable. It also suffers from significant delays in propagation of information through the system because robots only transmit updates when they move.

Individual Broadcast of Coordinates (IBC) model: The IBC model was proposed to solve all these shortcomings presented by the BBC model. In this model a robot does not have to broadcast any previous coordinates; not its own nor its neighbors'. This solves the memory issue. Moreover, each robot broadcasts updated coordinates to its neighbors immediately after updates are performed, therefore mitigating the propagation delay problem.

However, this model does not provide a way to control where the circle to be formed will be positioned in the environment. Consequently, circle formation can occur anywhere. This means that this model in its current form cannot be used to simulate swarm behavior for surrounding a target. In [30] an algorithm based on the IBC model is proposed to tackle the shortcomings of the IBC model regarding surrounding a target. The following subsections are dedicated to the analysis of this novel approach.

4.3.2 Key Concepts

This model employs explicit wireless communication within a group of homogeneous, anonymous, oblivious robots with limited sensing and communication abilities placed in a 2D environment. Anonymous means that the robots do not have any identifiers. Oblivious means that the robots make decisions based on the current situation with no reference to past situations. Limited sensing and communication means that each robot can only sense and communicate within a specific distance.

Robots follow the same coordinate system (global). To achieve this, robots need to agree on the origin as well as the x-y axis orientation. Each robot knows its own position in the global coordinate system and uses this to estimate a sensed target's position. The robot then broadcasts its own coordinates as well as the target coordinates. The target is surrounded as though it occupies a single coordinate as shown in figure 4.2

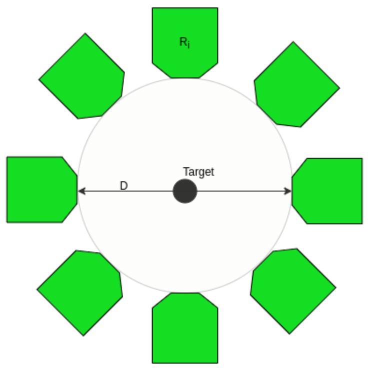


Figure 4.2: A robot swarm surrounding a target.

Each robot has two states, an awake state and an asleep state. A robot in the awake state can move and also broadcast its new position coordinates as well as the target's position. On the other hand a robot in the asleep state remains stationary but can still hear incoming information and can transmit the information.

4.3.3 Description of algorithm

Awake state: A robot in the awake state broadcasts both its coordinates and the coordinates of the target. It also receives coordinates of other robots as well as the coordinates of the target from the robots in its communication range. The stored nearest and farthest coordinates are only updated if closer or much further coordinates than stored values are received respectively. The robot then uses these received coordinates to update its distances to the nearest and farthest robots respectively, likewise it updates the coordinates of the target.

Each robot moves according to three possible current conditions. Firstly, if the target coordinates are known and the distance to the position of the target is less than the circle radius, then the robot takes a step away from the target. On the other hand if the distance to the target position is greater than the circle radius it takes a step towards the position of the target.

Secondly, if the coordinates of the target are unknown then the robot calculates the midpoint between its nearest and farthest robots. Again, if this distance is less than the radius of the circle then the robot takes a step away from the midpoint position. However, if this distance is greater than the radius of the circle then it takes a step towards the position of the midpoint. Lastly, if a robot has no information about both the position of the target and other robots then it takes a random step.

Sleep state: A robot in the sleep state receives other robots' coordinates and evaluates its distance to these coordinates as in the awake state. The farthest and nearest robot coordinates are then updated accordingly. Although the robot does not move it still broadcasts the received robot and target coordinates.

4.3.4 Performance analysis

A circularity error metric, first used in [8], was used to evaluate the performance of the algorithm (see equation 7.1). This is a sum of squares of the distances between the robot positions and the required circle formation. A low value denotes an accurate circle formation by the swarm. So the aim is to minimise this value. However, this metric does not capture whether robots are evenly distributed around the target.

$$C^{err} = 1/N \sum_{i=1}^{N} X_i({D^t}_i - D/2)^2 \tag{4.1}$$

Where:

- N is the population size of the swarm
- X_i has two possible values: 1 if $\mathrm{R_i}$ is aware of the target's location and 0 if $\mathrm{R_i}$ has no target information
- ${D_i}^t$ is the distance between robot $\mathrm{R_i}$ and the target location ($\mathrm{x^t}$, $\mathrm{y^t}$)
- D is the required diameter of the circle being formed

In [8], further performance investigations were made, including the effect of the diameter of the circle as well as the effect of the radius of communication.

It was found that a small diameter results in poor performance due to the isolation of robots. Isolated robots introduce a large magnitude of the circularity error. It was also found that large communication radii improve overall circle formation. This however introduces the possibility of wireless traffic congestion. All these investigations made in [8] still hold in this adaptation of the IBC algorithm.

4.3.5 Analysis of the algorithm

This algorithm was shown to be scalable, robust and most certainly designed for surrounding a target. All these qualities of the algorithm are desirable.

However, the predetermined assignment of the diameter of the circle to be formed is not entirely pragmatic in real world applications. For instance, if the swarm is tasked with surrounding an *ongoing* oil leakage the actual required circle diameter may change before the swarm can complete the surrounding behavior. Ultimately, the predetermined circle diameter assignment limits the adaptability of the swarm and inevitably makes the swarm dependent on human oversight.

Furthermore, since the algorithm is predicated on the assumption that the target occupies a single coordinate it means that in order to surround an oil spill the coordinates of the center of the oil spill would be required. This, however, becomes problematic when dealing with large oil spills, and this might result in undefined pattern formations by the swarm, because there is no obvious way for the swarm to get the exact center of the oil spill. Moreover, forming a circle formation as a means of surrounding a large oil spill might require an unnecessarily large swarm as compared to just mapping the oil contour, see figure 4.3.

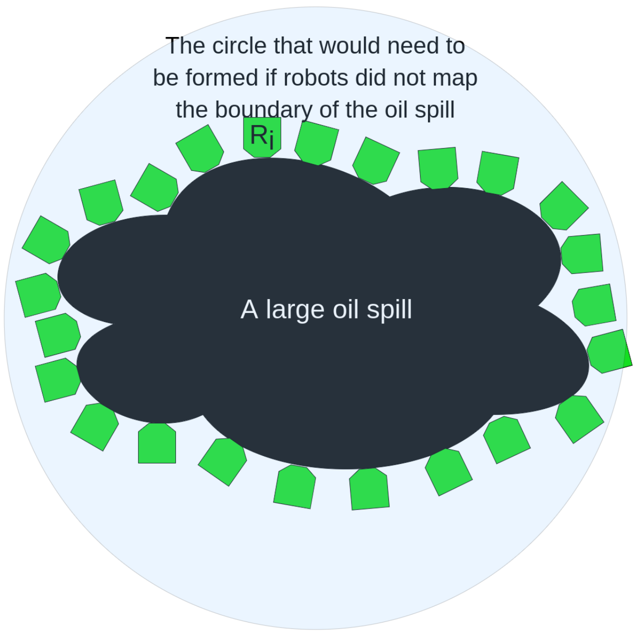

The circle that would need to be formed if robots did not map the boundary of the oil spill
R_i
A large oil spill

Figure 4.3: A robot swarm surrounding a large spill by mapping the boundary of the spill.

4.4 Leader-follower approach

This approach uses the *leader-follower* concept, in which a designated robot (chosen according to some criterion) acts as a superior to the other group members and possibly directs how the other robots should act.

4.4.1 Key concepts

This approach involves two major stages:

- **identification:** firstly, the robots aggregate in order to recognise their kin, and then the robot that is able to detect all the individuals of a given grid becomes a leader
- **referencing:** the leader then establishes a positioning mechanism that is used by all the members

4.4.2 Description of algorithm

The grouping algorithm is force based, and robots move in the direction of the resultant force, and are allowed to approach each other until they reach a certain allowable minimum distance. Leadership is assumed by the robot that is able to detect all other robots in the same quadrant (see figure 4.4).

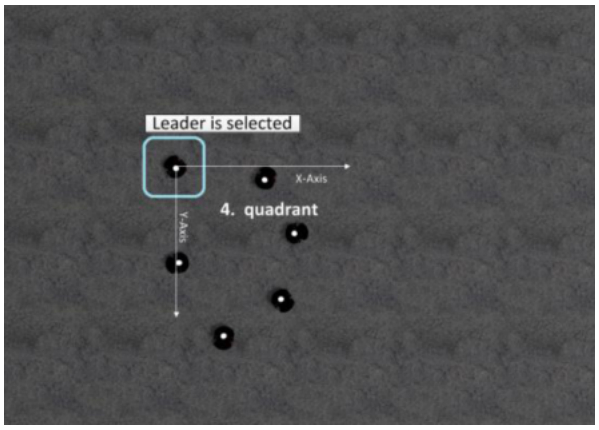


Figure 4.4: An illustration of the leader selection behavior [31].

The leader then determines the positions of all member robots using its ranger sensor and accordingly calculates the position of the center of gravity *(COG)* point. The leader uses an artificial vector, C_L, drawn from itself to the *COG* point to estimate the positions of the other member robots ($\mathrm{R_1}$, $\mathrm{R_2}$, ..., $\mathrm{R_N}$) with respect to the *COG* point. Member robots are then sorted according to the angles their artificial vectors ($\mathrm{C_{R1}}$, $\mathrm{C_{R2}}$, ..., $\mathrm{C_{RN}}$) make with the COG point using the C_L vector as reference and moving in the counterclockwise direction.

The leader communicates the COG point, formation role number, as well as the population size of the group to each member. The authors do not mention anything regarding the range of communication. However, it seems that the leader can send this information to all robots in its field of view, i.e. its group members. This information allows the member robots to calculate their possible position in the required circle formation.

Each robot then moves towards its calculated position in the circle formation according to a potential function which is given by the sum of attractive and repulsive forces. Attractive potential is assigned to the desired final position of the member robot while the repulsive potential is calculated and assigned to obstacles in the environment.

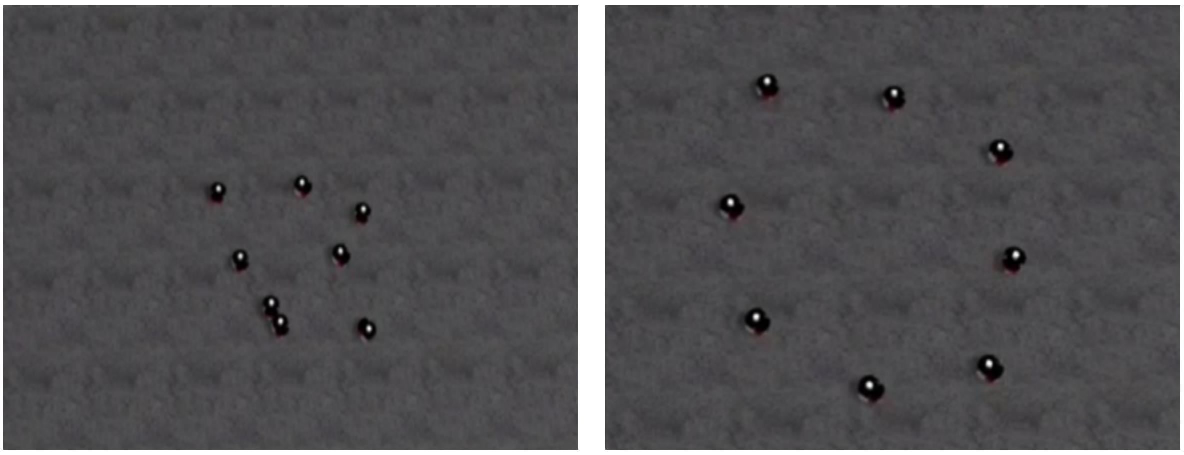
This process is repeated until the desired formation emerges. The iterative nature of the algorithm makes it adaptive because if a member robot breaks the formation and therefore changes the COG point of the group new robot positions as well as a COG point will be recalculated on the next iteration of the algorithm.

4.4.3 Performance analysis

The performance of the algorithm is evaluated using the following parameters:

- **Total formation time:** this parameter measures the total time taken to form the desired circle formation.
- **Total travel distance:** this parameter measures the total distance travelled by the swarm to form the desired circle formation.

It was shown that the total formation time as well as the distance travelled increase as the population of the swarm increases. Nonetheless the algorithm was able to consistently form the required circle formation (see figure 4.5) under different population sizes.



(a) At time t=21s the group is executing grouping behavior.

(b) At time t=31s the group has managed to distribute itself on the desired circle formation.

Figure 4.5: A swarm of a population size of 8 robots distributing itself to form the required circle formation. [31]

4.4.4 Analysis of the algorithm

Firstly, there is no way of specifying where the desired circle formation should be formed. Instead the leader of the group just finds the center of gravity for the group at random locations in the environment. Secondly, the leader calculates the relative angles of each and every member in the group with respect to the *COG* point which does not scale well for large population sizes. In fact, the time taken by the group to form the required circle tripled when the swarm size was doubled, and this is a direct consequence of having the leader acting as a *central manager*. Lastly, when a member malfunctions new positions have to be calculated for the whole group. Depending on how often members malfunction or encounter obstacles preventing them from assuming their predetermined position in the desired circle formation, this could really be computationally expensive for large population sizes.

Chapter 5

Design

This section is dedicated to the design of the system according to the system requirements formulated in section 2. It will progress as follows; section 5.1 selects an algorithm that will be used to implement the system, section 5.2 gives a brief review of swarm robotics simulators and selects one that will be used to perform system tests and experiments, section 5.3 identifies the various subsystems of the system and lastly, section 5.4 is dedicated to the design of each of the subsystems of the system.

5.1 Algorithm selection

Four target surrounding algorithms were reviewed in the literature review section (section 4). The main concepts, implementation description, performance and general analysis of each algorithm with regard to the goal of surrounding an oil spill were all presented as part of the review. This section now seeks to compare these four algorithms according to their capabilities to meet the user requirements and technical requirements of the system, which were given in section 2.

5.1.1 Algorithm 1 (Potential field based approach)

This algorithm (section 4.1) defines a primary trajectory as well as secondary trajectories along which robots circulate the target. However, the constant circulation of the robots around the target is not desirable in the containment of an oil spill as this action could further accelerate the rate of spread of the spill in the hostile sea environments.

Furthermore, concerns were raised about the deadlock proof presented for this algorithm in the literature review. As discussed in the literature review, it would seem that if a robot with a higher potential malfunctioned, the neighboring robots would get stuck in a deadlock. This directly contradicts user requirement SRS-URM-04 of this system, which states that the failure (malfunction) of an individual within the swarm should not result in a catastrophic failure of the system. Consequently this algorithm cannot be used to implement the system.

5.1.2 Algorithm 2 (Virtual Viscoelastic Forces Approach)

This algorithm (section 4.2) shows very effective and robust behavior for a small number of robots. However, this approach may require that a robot be positioned at the center of the circle being formed and this could be problematic if the center robot malfunctioned. This would make the system not self-stabilizing, which directly contradicts SRS-URM-03.

Moreover when the number of robots is increased it takes longer for the system to stabilize due to the expensive computations of distances between neighbors. This means that user requirement SRS-URM-05 can not be attained using this approach because the algorithm is not scalable. Therefore this algorithm cannot be used to implement the system.

5.1.3 Algorithm 3 (Leader-follower approach)

This algorithm (section 4.4) was shown to be adaptive. However, the assignment of specific roles (leader or follower) makes this algorithm not to scale well because the leader acts as a *central manager* and calculates the positions that the follower robots have to assume on the required circle formation. This drastically increases the time taken by the swarm to self-organize to form the circle formation for large population sizes. Furthermore, if the leader malfunctioned in the middle of an operation the whole operation would have to be halted. This directly contradicts user requirement SRS-URM-04, which requires reliable swarm performance even if some individuals malfunction.

Furthermore, the leader directs its followers to form the required circle formation at random locations in the environment according to where the group's center of gravity point happens to be located. Due to the scalability and robustness issue, and not having control over where the surrounding behavior occurs, this algorithm can not be used to implement the system.

5.1.4 Algorithm 4 (Communication - Based Approach)

This algorithm was shown (section 4.3) to be both robust and scalable, however it also has its own flaws. The main issue identified with this algorithm was the lack of flexibility, i.e swarms only formed a circle of which the diameter had to be predetermined. Moreover, the algorithm as is right now only assumes that the target occupies a single coordinate, of which this coordinate is then used as the center of the circle being formed.

However, for all intents and purposes this algorithm meets all the user requirements and functional requirements of the system. As such, this algorithm will be used to implement the system. The following assumptions are made:

- The target occupies a single coordinate and is static
- Communication between agents is instantaneous
- Individuals are anonymous, i.e. have no predefined roles
- Individuals can determine their positions in the global coordinate system of the simulation
- Movements are asynchronous

5.2 Simulation

Simulations will be used to perform experiments designed to answer the main questions of this work. This section presents a brief review of four of the most commonly used plartforms for developing robotic swarms and/or investigating the behavior of a population of robots.

5.2.1 Player/Stage

Player

Player is a device server that is divided into two halves [32]:

- **The player core library:** the core system offers the core API and functionality of the whole Player system. It consists of device and driver classes, dynamic library loading code, configuration file parsing and driver registry.
- **The transport layer:** the transport layer is based on the Transmission Control Protocol (TCP) and consists of two libraries; a TCP library and an eXternal Data Representation (XDR) library - which is a data marshaling library.

Clients communicate with player through a TCP-based protocol and can be implemented in C, C++, Tcl, Java and Python.

Stage

Stage is a C plus-plus software library that can simulate multiple robots. It is particularly suitable for swarm robotics or any other research that investigates the behavior of a population of robots. Its runtime scales linearly with population size up to at least 100 000 simple robots [33]. It is compatible with Player, realistic enough for most pragmatic purposes and provides models for most of the widely used robot sensors.

5.2.2 Gazebo

Gazebo is an extension of the Player/Stage project. It can simulate a population of robots in a 3D environment and it is also compatible with Player, which means that it has access to a vast range of devices.

Gazebo attempts to create a realistic world for robots however, this makes it resource extensive because of the computation needs of simulating rigid body dynamics as well as a 3D environment. Furthermore this simulator lacks distributed computing [34].

5.2.3 Webots

Webots is an open source simulator that offers local and global communication facilities for multi-agent systems [35]. It uses the Open Dynamics Engine for rigid body dynamics simulation and collision detection. Robot controllers can be programmed in C, C++ and Java. Nonetheless this can also be achieved using third party software using TCP/IP.

5.2.4 Autonomous Robots Go Swarming (ARGoS)

ARGoS is an open-source simulator for large heterogeneous swarms of robots. It allows the use of a variety of different physics engines which can be assigned to different parts of the environment, allowing it to achieve optimal scalability and parallelism in simulations [36].

However, this simulator's architecture, i.e. multiple engines, can make setting up experiments cumbersome and particularly confusing to unfamiliar users. Furthermore, it makes the simulator extremely resource extensive. Lastly, the number of device drivers readily available to the simulator are limited compared to Player/Stage/Gazebo.

5.2.5 Player/Stage control architecture

The Player/Stage project is the most widely used 2D simulator, as discussed above. This simulator is very lighweight, and comes pre-installed with drivers for most widely used devices. Its runtime scales linearly with the population size and more importantly the features of this simulator are enough to evaluate the functional requirements of the system formulated in section 2.1.3, and consequently the system is simulated on this simulator.

In order to understand how the various subsystems of our swarm robotics system implementation are going to interact together, it is important to firstly consider how Player works. This section deliberates on the various interactions of the system with the simulator. Figure 5.1 below shows the Player/Stage control architecture:

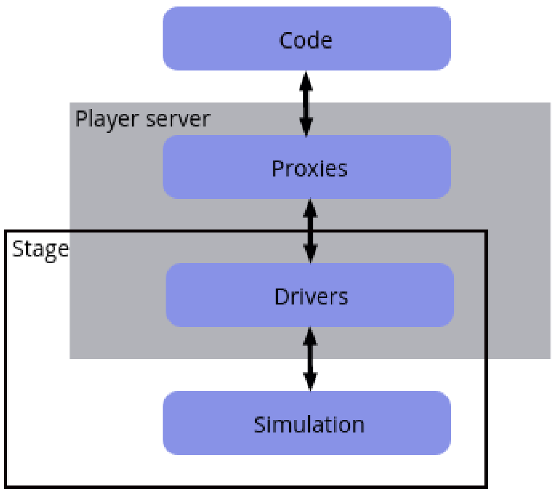


Figure 5.1: The Player/Stage control architecture

Code

This is the controller code. Player is compatible with C, C++ and Python controllers. The code connects to Player through a ‘proxy’. Player provides these proxy-classes. The controller for this project is implemented in C++.

Player

As alluded above, Player is a server and robots connect to Player as clients. Player uses a client/server architecture to pass data and instructions between the code and the robots; this can be real world robotics hardware or simulated robots. Player comes pre-installed with device drivers for widely used robotics hardware and proxy classes to connect to code.

Stage

Stage provides a simulation of the robot devices, such as sensors, and the world in which the robot operates. In this context, a device is understood to be a piece of hardware that interacts with a driver which conforms to a specific interface. The interface defines how the driver should return data from a device.

5.2.6 Building the simulation environment

Player/Stage uses three different file types to create a simulation; a configuration file (with a *.cfg* extension), a world file (with a *.world* extension) and an include file (with a *.inc* extension). The complete file structure is shown in figure 5.2 below.

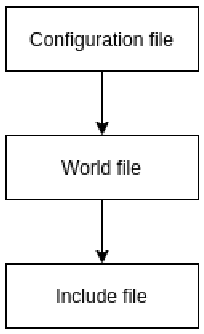


Figure 5.2: The Player/Stage file hierarchy

The configuration file

In order to create a new simulation (world), one needs to tell the Player server which drivers to use as well as which interfaces the respective drivers will make use of. This information is specified in a file called a configuration file.

In this file, one will typically specify a driver for each model in the simulation or a device on a robot if such a model or device is meant to be interactive. Once declared in this file, such a model can be called or referenced in code.

The world file

All the items (models) that are part of the simulation, including the environment and robots, are defined in a world file (with a *.world* extension). An important entity of the world file is the 'window' entity which defines the width and height of the actual simulation window in pixels.

The include file

Files with the *.inc* file extension also define models, except they can be included in other world files, this is good practice because it allows code reuse.

5.3 Subsystems identification

To identify the various subsystems of the system a use case diagram of the system is illustrated in figure 5.3.

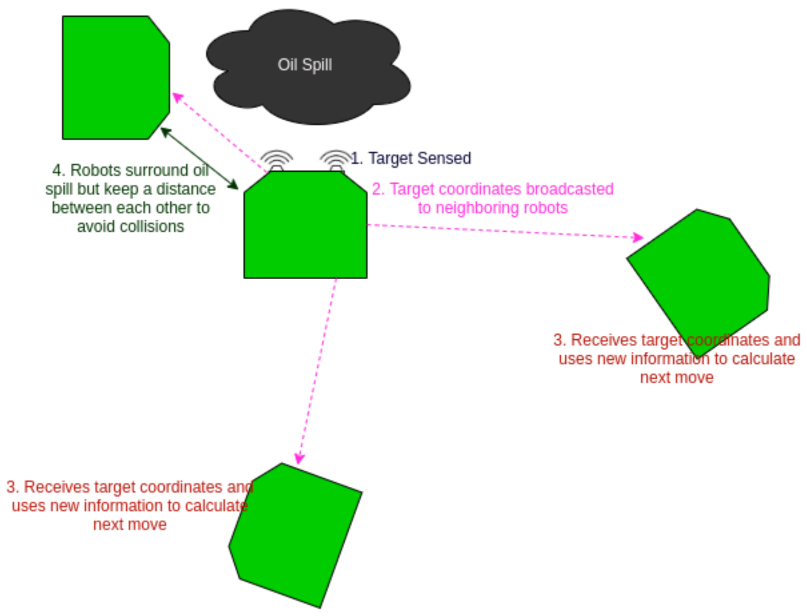

Oil Spill
1. Target Sensed
2. Target coordinates broadcasted to neighboring robots
3. Receives target coordinates and uses new information to calculate next move
3. Receives target coordinates and uses new information to calculate next move
4. Robots surround oil spill but keep a distance between each other to avoid collisions

Figure 5.3: Swarm robotics system use case diagram

A state diagram is also shown in figure 5.4. The state diagram captures the various states a robot of the swarm could possibly occupy at any given time.

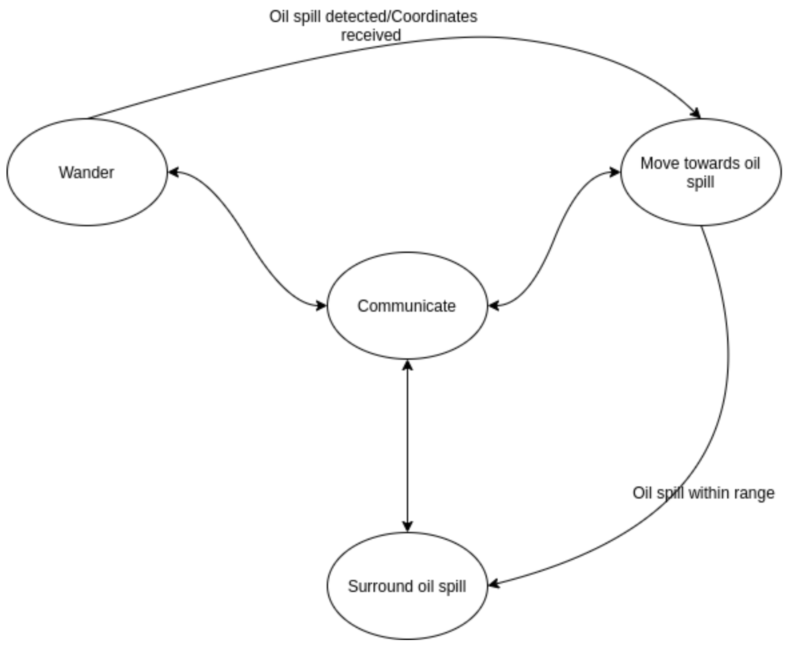


Figure 5.4: Swarm robotics system state diagram

From the state diagram above and the user and technical requirements deliberated earlier (section 2), the subsystems of the system are identified as:

1. Robot model (RMO)
2. Communication (COMMS)
3. Movement coordination (MOVCO)

Figure 5.5 illustrates how these subsystems interact with each other.

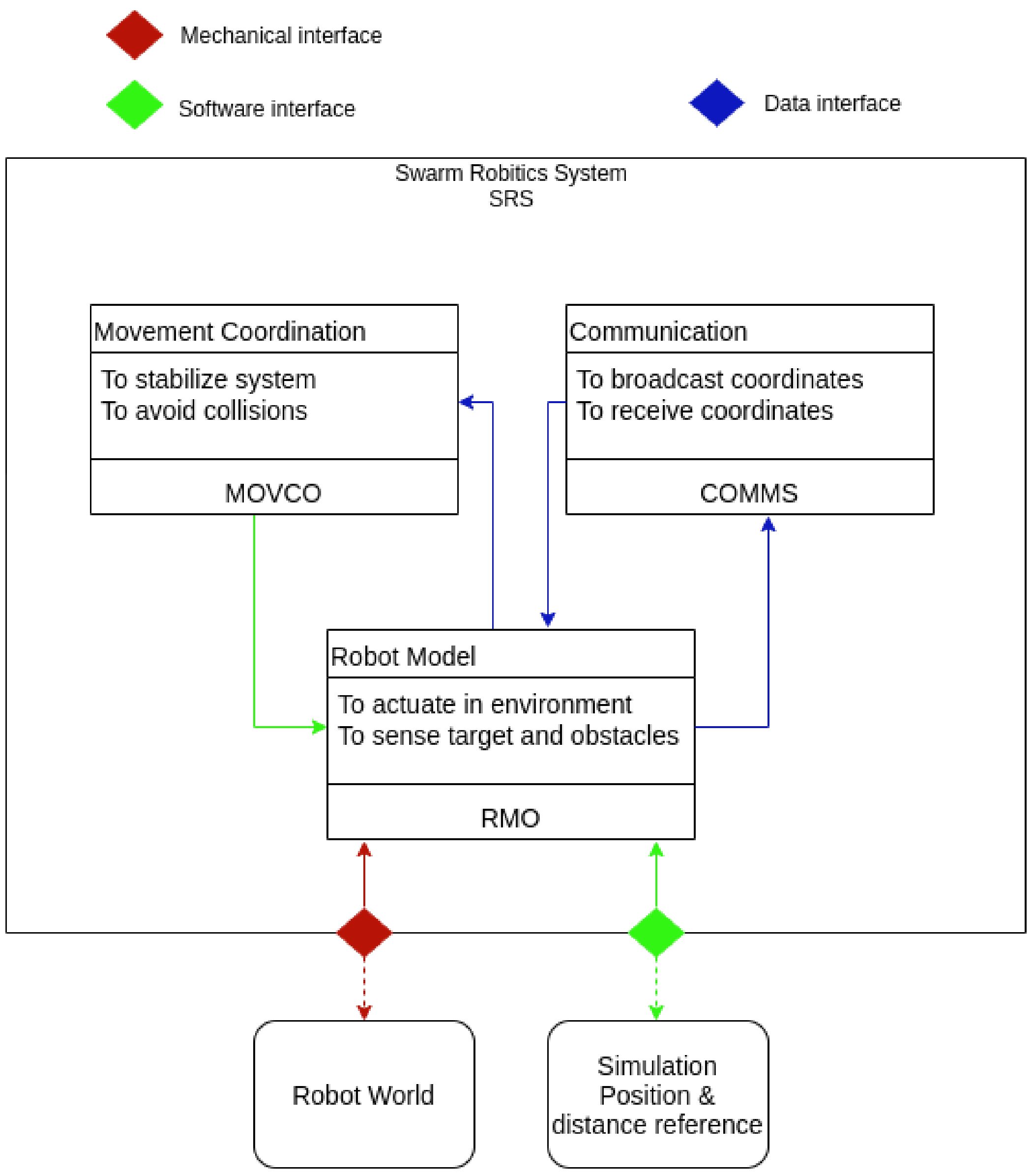


Figure 5.5: Swarm robotics system top-level functional flow diagram

External Interface	**Type**	**Description**
Robot world	Mechanical	This interface represents the interaction of the robots with the simulation environment
Simulation position and distance reference	Software	This interface represents the global coordinates system of the simulator

Table 5.1: Identification of external interfaces.

The subsystems are further broken down into their consituent atomic elements. This is illustrated in figure 5.6, and then each of these elements are described.

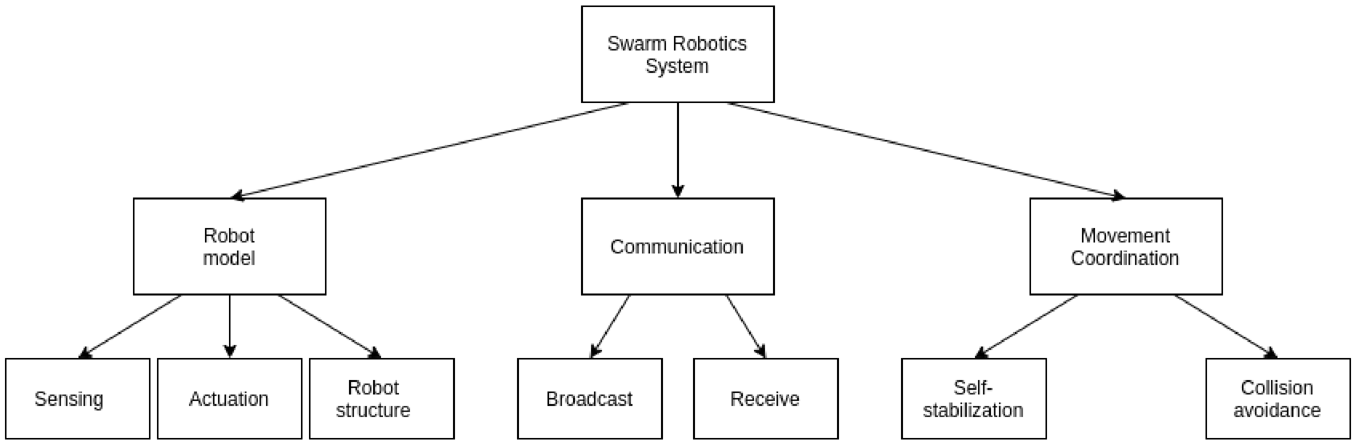


Figure 5.6: Swarm robotics system hierarchy

5.3.1 Robot model

Multiple autonomous, physically embodied robots are required. The following modules are identified for this subsystem:

Sensing

Agents will require a mechanism to sense/detect the oil spill as well as other objects within their sensing range.

Actuation

Agents will require a mechanism to actuate in their environment.

Robot structure

The physical structure of robots, the chassis that houses all the various components of the robot.

5.3.2 Communication

An effective distributed communication method is required to enable agents of the swarm to exchange information with other agents.

5.3.3 Movement coordination

Agents require a mechanism to time and control movements. This subsystem consists of the following modules:

Collision avoidance

The individuals require a mechanism to ensure they do not collide with each other, i.e. agents must not attempt to occupy the same location at the same time.

Self-stabilization

The swarm system needs to recover timeously after transient failure, i.e. an agent malfunctions, an agent quitting or an agent joining the swarm.

5.4 Subsystems design

5.4.1 Robot model(RMO)

This model seeks to implement requirements SRS-FRM-01 and SRS-FRM-02. The former requires the robotic swarm to have a method to determine the position of an oil spill. To meet this requirement, the robot model is equipped with a Sensing module which will not only sense the target but obstacles (i.e. walls in the simulation environment) as well. Functional requirement SRS-FRM-02 requires the members of the swarm to be autonomous. To this end, the robot model is equipped with an Actuation module. Figure 5.7 illustrates a conceptual design of how these modules interact together.

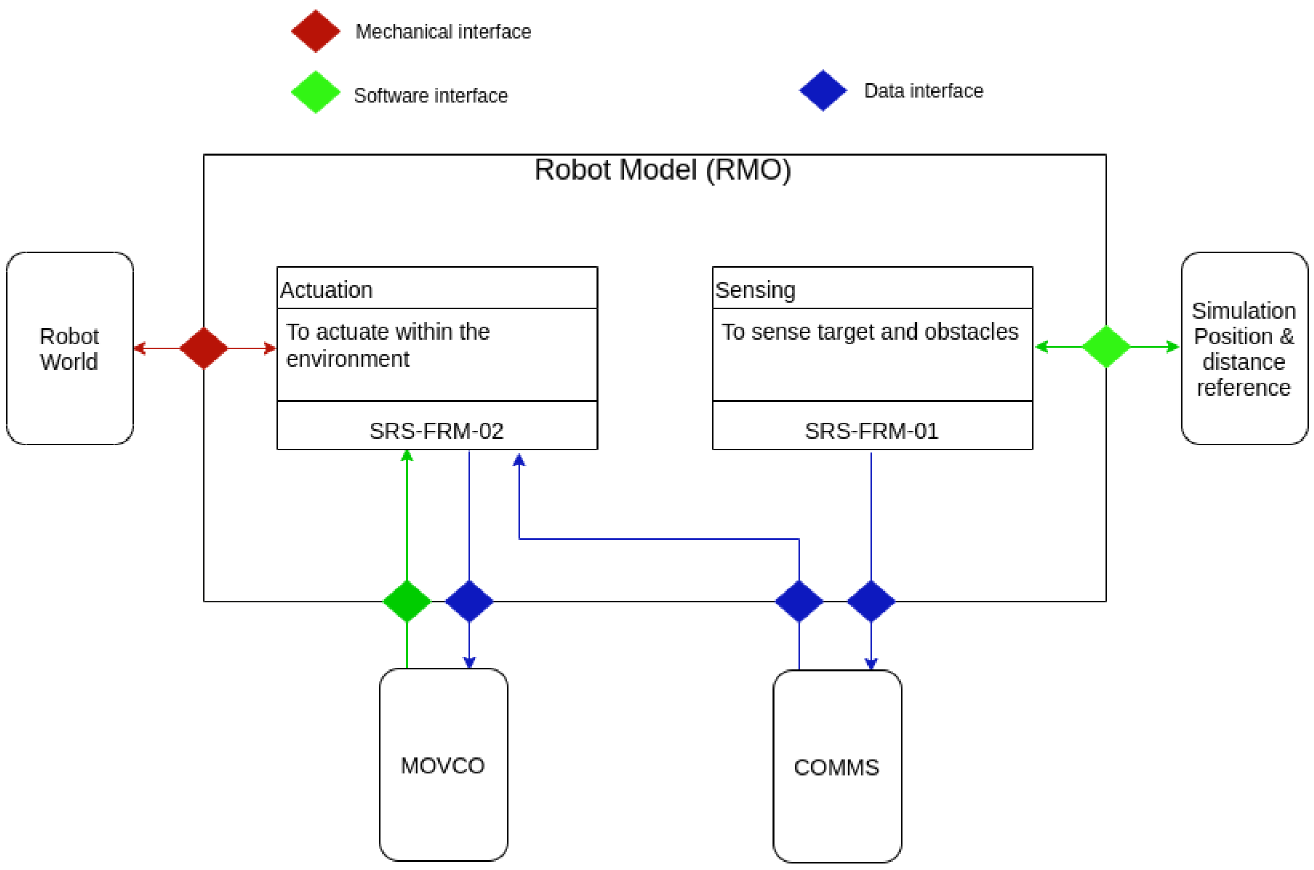


Figure 5.7: A functional flow diagram showing how the various modules of the Robot model subsystem interact with each other and external interfaces.

Furthermore, the diagram shows four external interfaces which interact with the subsystem. The first one, Robot World, is a mechanical interface which represents the interaction of the robot model with its environment. The operation of the robot in its environment is mechanical, hence the mechanical interface.

Moving anti-clockwise, the second external interaction, MOVCO (Movement Coordination) subsystem, is dual; this interaction utilises a data interface and a software interface. Through software (controller), the subsystem MOVCO is able to control the robot's movements. In return, the robot sends its own positional data as well as received neighboring robots' information to the MOVCO module via a data interface.

The third external interaction, COMMS (Communication) subsystem, is based on a data interface. The COMMS subsystem receives target coordinates as well as the location of sensed obstacles from the sensing module of the robot model and broadcasts this information for robots within communication range. It also receives target information and neighboring robots' locations from the robots within communication range and sends this information to the Actuation module, via a data interface, where it is sent to the MOVCO module to be used to calculate next steps.

Lastly, the subsystem interacts with the simulation to obtain position and distance references using a software interface. This represents the global coordinate system of the simulation which is used to position all the objects of the simulation. This interface is bidirectional because for the postion of a robot to be determined, certain software models of the robot are utilised (provided by Player as proxies).

The rest of this section is dedicated to a detailed design (which will include identification of key model parameters in Player and the assignment of specific values to the identified model parameters) of each of the modules just identified above.

Sensing

1. **Sensors**
   We desire to use five range sensors to detect obstacles in the environment as well as the oil spill. Below we define some of the key parameters of this model, as well as the values we have opted to assign to each of them. The sensor capabilities are limited as per the characteristics of swarm robots.

   (a) **Size**: specifies how big the sensors are. The following values are used (in meters):

      - *x size*: 0.01 m
      - *y size*: 0.05 m
      - *z size*: 0.01 m

   (b) **Range**: specifies the minimum and maximum distances that can be sensed. The following values are used (all values are in meters):

      - *min*: 0.01 m
      - *max*: 2.0 m

   (c) **Field of view**: specifies the field of view that can be covered by the sensors:

      - *fov*: 10 degrees

   (d) **Samples**: specifies the number of rays in a beam:

      - *samples*: 180

2. **Blobfinder model**

   The blobfinder model simulates color detection software. This model will be used to determine the type of objects detected based on the color of the objects, i.e. if it is a black object chances are it is a wall and if it is amber then it is most likely an oil spill. The number as well as the names of colors that the blobfinder can detect are specified in its definition. Other parameters include range and field of view. The following parameter values are chosen for our robot model:

   - *colors count*: 2
   - *colors*: black and amber
   - *image resolution*: 160 x 120
   - *range*: 9 meters (this value was not kept constant, see experiments later on.)
   - *fov*: 60 degrees

Actuation

The position model in Player/Stage allows the robot to keep track of where it is by recording the angles it turns and the number of times its wheels spin. One can also think of this model as allowing physical embodiment of the robot in its world, i.e. robot can collide with objects. It uses the *position2d* interface which tells Player where the robot is in the environment. Translational and rotational motion will suffice for the robots, as such we will implement the robot motor to only support these two forms of motion.

Robot structure

- **Shape and size:** The shape of the robot is given in figure 5.8. The sides of the robot is where booms could be connected to make the containment more effective. As such there are no sensors on the sides of the robot to create space for the booms. The following parameter values are chosen to define the robot's physical structure(in meters):
  - *x size*: 0.5 m
  - *y size*: 0.5 m
  - *z size*: 0.5 m
- **Location of sensors:** Each robot has 5 ranger sensors: two are located at the front (to cover the front perimeter), one other is on the left corner, the forth is

on the right corner (the two range sensors were placed on the corners in order to increase the overall field of view of the robot) and the fifth is at the back (in order to detect the enclosing walls in the environment when moving backwards).

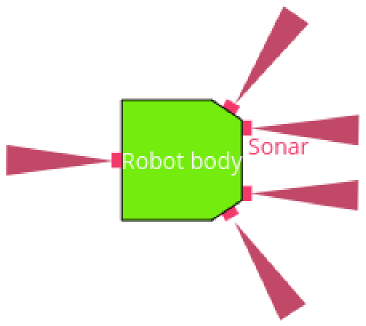


Figure 5.8: The physical structure of robots

Robot wrapper class

The modules presented above are defined in the *robot.inc* file mentioned in section 5.2.6. As deliberated earlier, this file, through the configuration file, will tell Player how to create the various models we have defined. However, for the created devices to be used in code, i.e. controller implementation, they need to be linked with the proxy classies provided by Player (please refer to figure 5.1 to make sense of this).

This class is used to wrap the proxy code provided by Player. The robot and each device of the robot are accessed through network sockets in code. The host IP address, i.e. the IP address of the Player server or robotic hardware, also needs to be specified. The class diagram in figure 5.9 shows some of the various fields and functions that characterise this class.

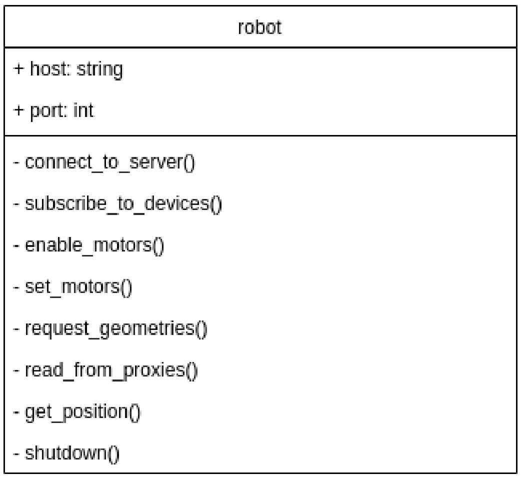


Figure 5.9: The robot wrapper class

- *connect_to_server()*: this function creates a Player client which is used to interact with Player on the given IP address and port number. We will be running the Player server and the controller code on the same computer, so we will use 'localhost' for the host address. However, different port numbers are assigned to each robot.
- *subscribe_to_devices()*: this function connects the Player client we established to the proxies associated with the devices we told Player to create in the configuration file so that the devices can be accessed in code. This function works by creating instances of each of the devices before connecting to them.
- *enable_motors()*: this function sets the robot motors to enabled, it wraps a function which takes boolean input to tell Player whether to enable the motors or not. Initially the motors are not enabled and the robot does not move or respond to commands.
- *set_motors(.)*: this function sets the translational speed of the robot in meters per second and the rotational speed in radians per second.
- *read_from_proxies()*: this function updates the Player client with new data from the sensors
- *get_position(.)*: this function gets the *x, y, z* and *yaw* position of the robot. It takes a simulation proxy as input.
- *shutdown()*: this function unsubscribes the Player client from the devices and then disconnects it from the server.

5.4.2 Communication (COMMS)

This subsystem seeks to implement the requirement of SRS-FRM-05 (section 2.1.3), which is that the communication of the system must be decentralized to eradicate the possibility of a communication bottleneck that could occur if a centralised approach is used for extremely large population sizes, i.e. scalability.

To this end, a local communication strategy is adopted to implement effective distributed communication. This approach allows agents to only exchange information with agents within their communication range. This type of communication is called explicit (or intentional) communication because robots directly (intentionally) exchange relevant information through radio [37]. Figure 5.10 illustrates how the two modules of this subsystem interact together as well as the subsystem's interactions with external interfaces. It is clear from the diagram that the functioning of this system is closely coupled with subsystem RMO, and all interactions are through data interfaces.

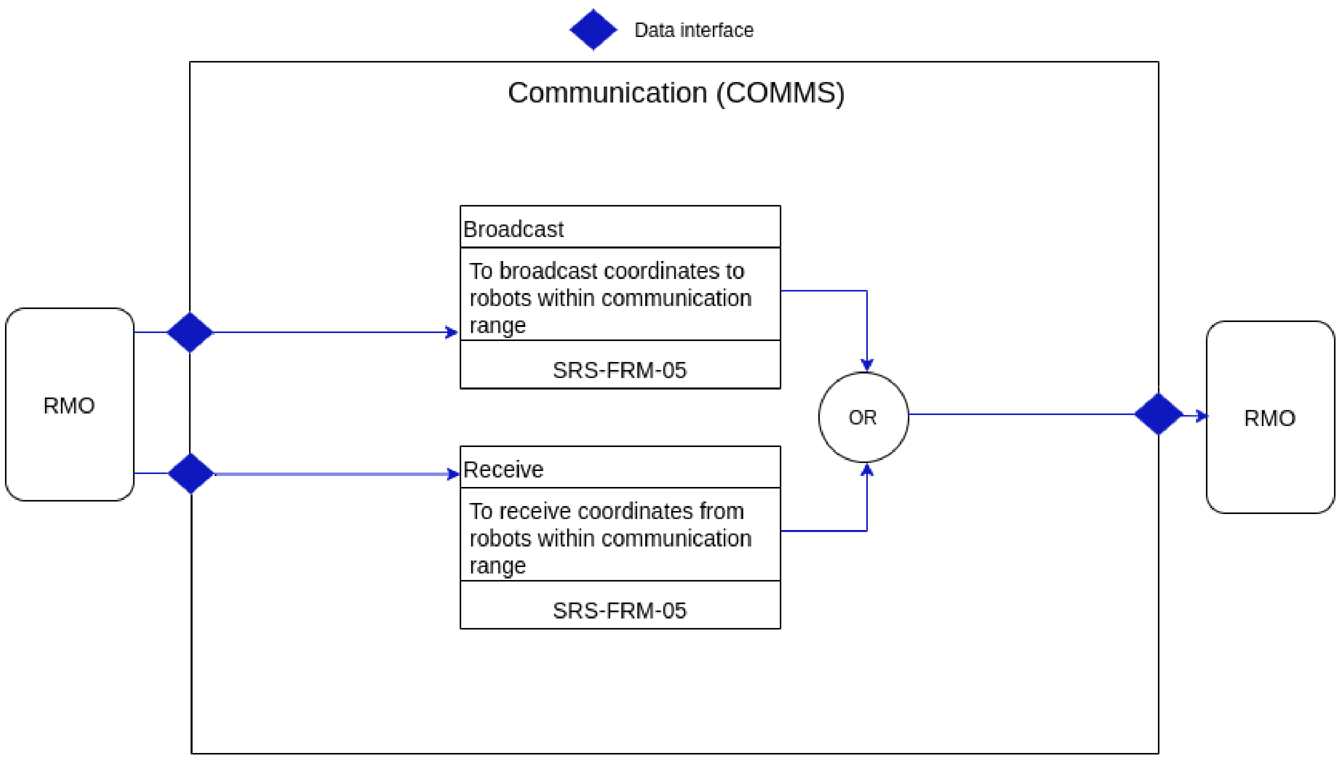

Data interface
Communication (COMMS)
RMO
Broadcast
To broadcast coordinates to robots within communication range
SRS-FRM-05
Receive
To receive coordinates from robots within communication range
SRS-FRM-05
OR
RMO

Figure 5.10: A functional flow diagram showing how the modules of the Communication subsystem interact with each other and external interfaces.

According to the IBC algorithm (section 4.3), each robot of the swarm will need to store coordinates of the farthest and nearest robots as well as coordinates corresponding to the target (location coordinates of the oil spill). Communication entails broadcasting and receiving coordinates broadcasted by other robots within communication range. The class diagram of the communication class is shown in figure 5.11.

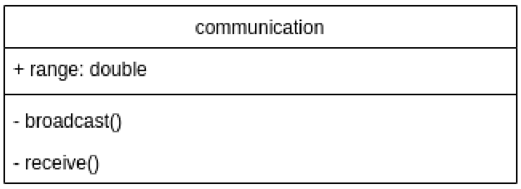


Figure 5.11: Communication class diagram

Broadcast

Each robot will broadcast its own coordinates as well as the coordinates of the target (if known) to its neighboring robots. Receivers will have to be within communication range. This process is illustrated by the flowchart in figure 5.12

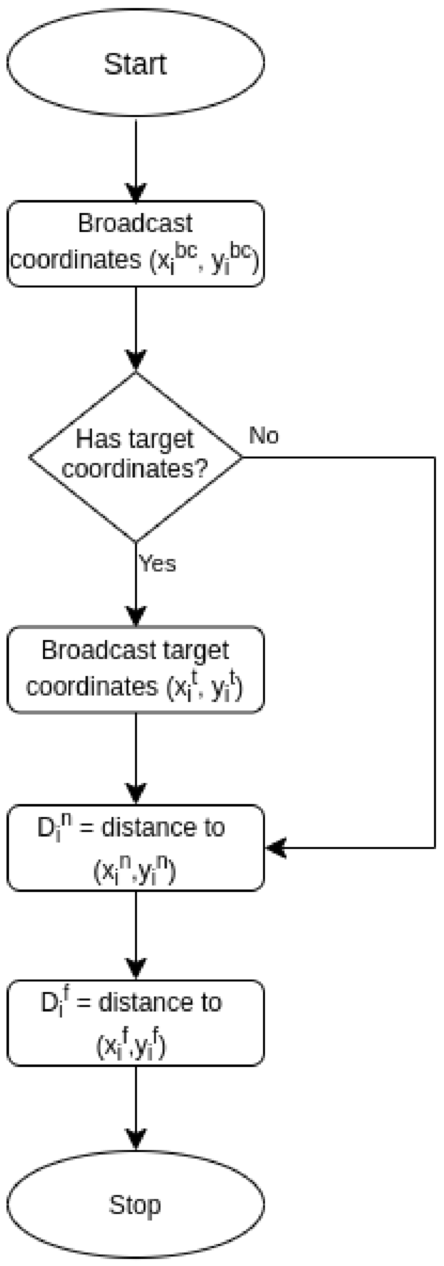

Start
Broadcast coordinates (x_i^{bc}, y_i^{bc})
Has target coordinates?
No
Yes
Broadcast target coordinates (x_i^t, y_i^t)
D_i^n = distance to (x_i^n,y_i^n)
D_i^f = distance to (x_i^f,y_i^f)
Stop

Figure 5.12: Flowchart diagram showing how messages are broadcasted by robots to neighboring robots.

Where:

- *($x_i{}^{bc}$, $y_i{}^{bc}$)* are the broadcast coordinates
- *($x_i{}^t$, $y_i{}^t$)* are the target coordinates
- *($x_i{}^n$, $y_i{}^n$)* are the coordinates of the nearest robot
- *(x_i^f, y_i^f)* are the coordinates of the farthest robot

- $D_i{}^n$ is the distance to the nearest robot
- D_i^f is the distance to the farthest robot

Receive

Each robot, R_i, will receive the coordinates of the target, *($x_j{}^t$, $y_j{}^t$)*, as well as the broadcasting robot's coordinates, *($x_j{}^{bc}$, $y_j{}^{bc}$)*. Robot R_i has to be within the communication range of robot R_j, the broadcasting robot. Upon receiving this information R_i will perform an update (if needed) of its nearest and farthest coordinates. The updates occur each time new coordinates are received, to reduce propagation delay in the swarm. This process is illustrated by the flowchart in figure 5.13.

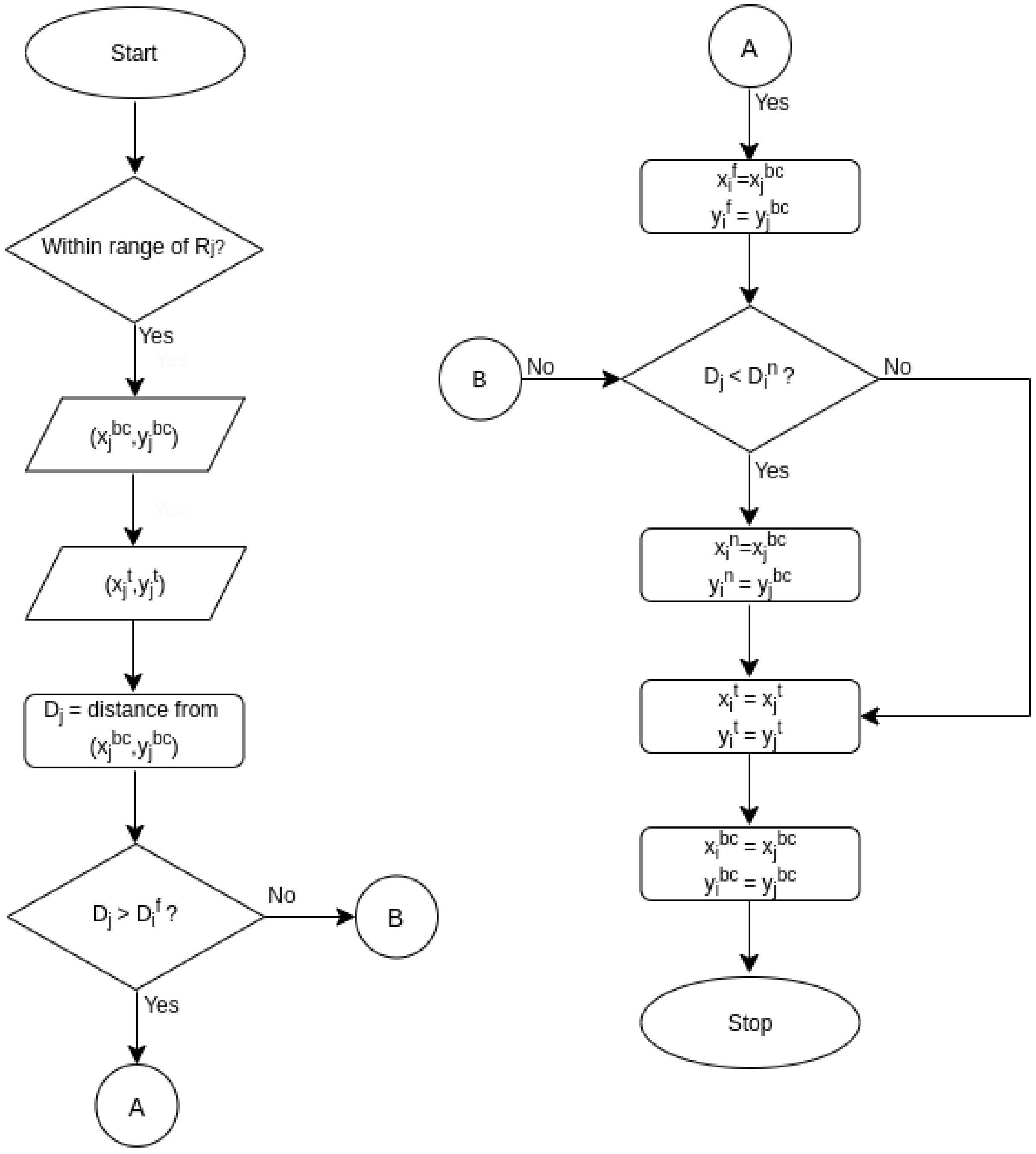


Figure 5.13: Flowchart diagram showing how information is received by robots.

5.4.3 Movement coordination (MOVCO)

This subsystem seeks to address the requirements of SRS-FRM-03 and SRS-FRM-04 (section 2.1.3) both of which specify how control for the robots has to be implemented.

The former states that the algorithm controlling the swarm must be oblivious which means that all robot steps have to be determined in real time based on the system state at the time, while the latter requirement reiterates the main feature of swarm robotics, which is that control has to be distributed.

There are four robot control classes [38]; deliberative control, reactive control, hybrid control and behavior-based control. Most swarm algorithms are predicated on the notion of local interaction between individuals until global behavior emerges [25], including the IBC algorithm this system seeks to implement. This is behavior-based control. However, survival behaviors, such as collision avoidance, are typically reactive, i.e. no intermediate thinking step, sensory inputs map directly to actuators to provide real-time responses. This is the approach adopted here.

Figure 5.14 shows a functional flow diagram which illustrates the various modules of this subsystem and how they interact with each other.

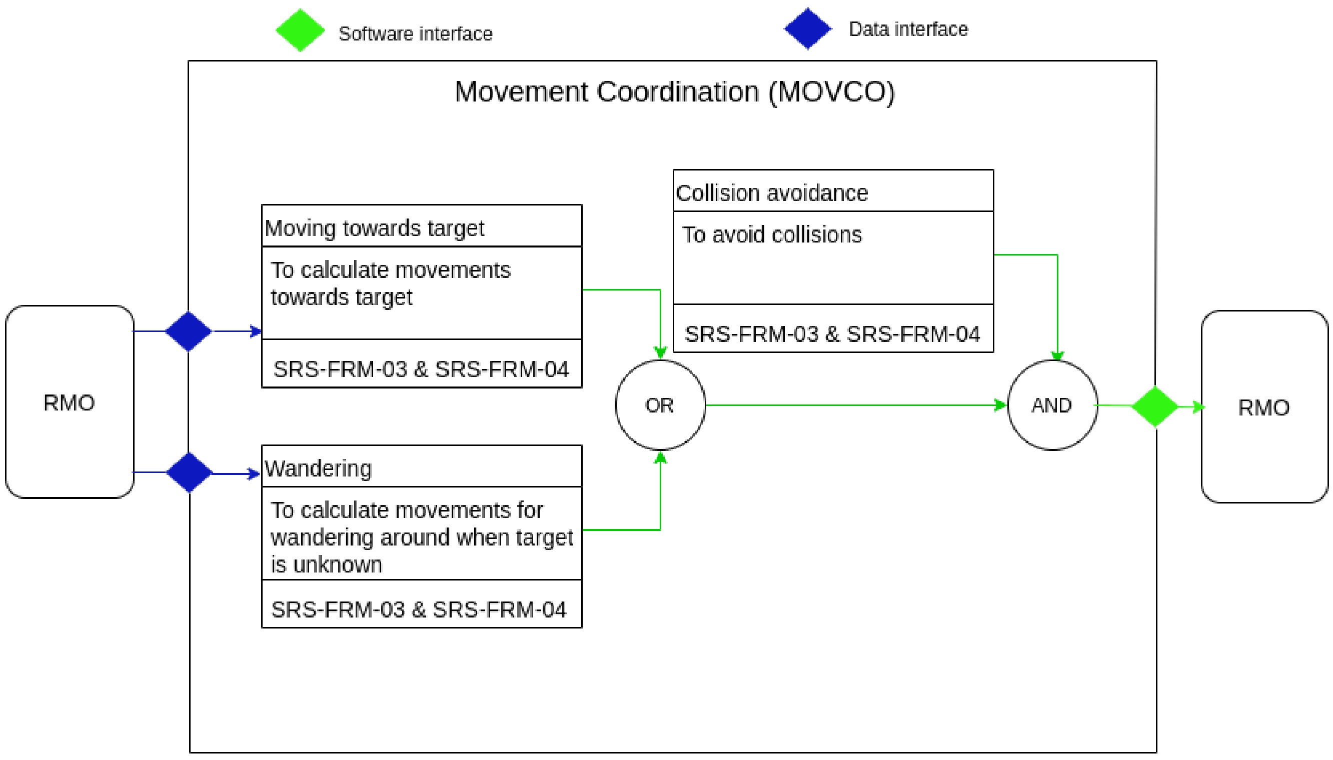


Figure 5.14: Functional flow diagram showing the different modules of this subsystem and how they interact together.

The key points from figure 5.14 is that the MOVCO subsystem only interacts with the RMO subsystem, and the interaction is such that RMO provides MOVCO with positional data which is used to determine the robot's next move. Using a software interface with the robot model, the controller is then able to enforce the required robot's 'next step'.

Each robot can be in one of a total of four states. A robot moves by taking one step or making a turn depending on the state it is in and the observations made in its local environment. All movements, regardless of the state, involve self-stabilization in the sense that each move is calculated based on the current system status. The class diagram of the movement class is shown in figure 5.15.

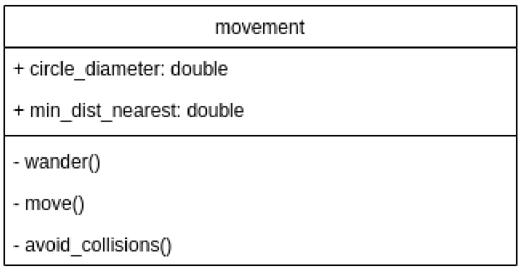


Figure 5.15: Class diagram for movement class

Self-stabalization

The swarm system needs to recover timeously after transient failure, i.e. an agent malfunctions, an agent quitting or an agent joining the swarm in the middle of an operation. To ensure the swarm is self-stabilizing (self-healing), the move for each robot is calculated based on the present state of the system. Each module (or behavior) takes input from the robot sensors or other modules and sends output to the robot motors or other modules. Sensing and action are tightly linked through the behaviors.

Initially, the whole population wanders around until a few individuals locate the target. Once the target is located, its coordinates are then broadcasted and consequently, percolate throughout the swarm. An individual which has received the target coordinates will first calculate its distance to the target, which it uses to determine whether to take a step towards or away from the target. These ideas are illustrated below with flowchart diagrams.

1. **Wandering**

   A wandering individual uses the coordinates of its nearest and farthest robots to calculate its steps. In the case where there are no known neighbors, the individual takes a random step. The flowchart diagram illustrated in figure 5.16 summarises this process.

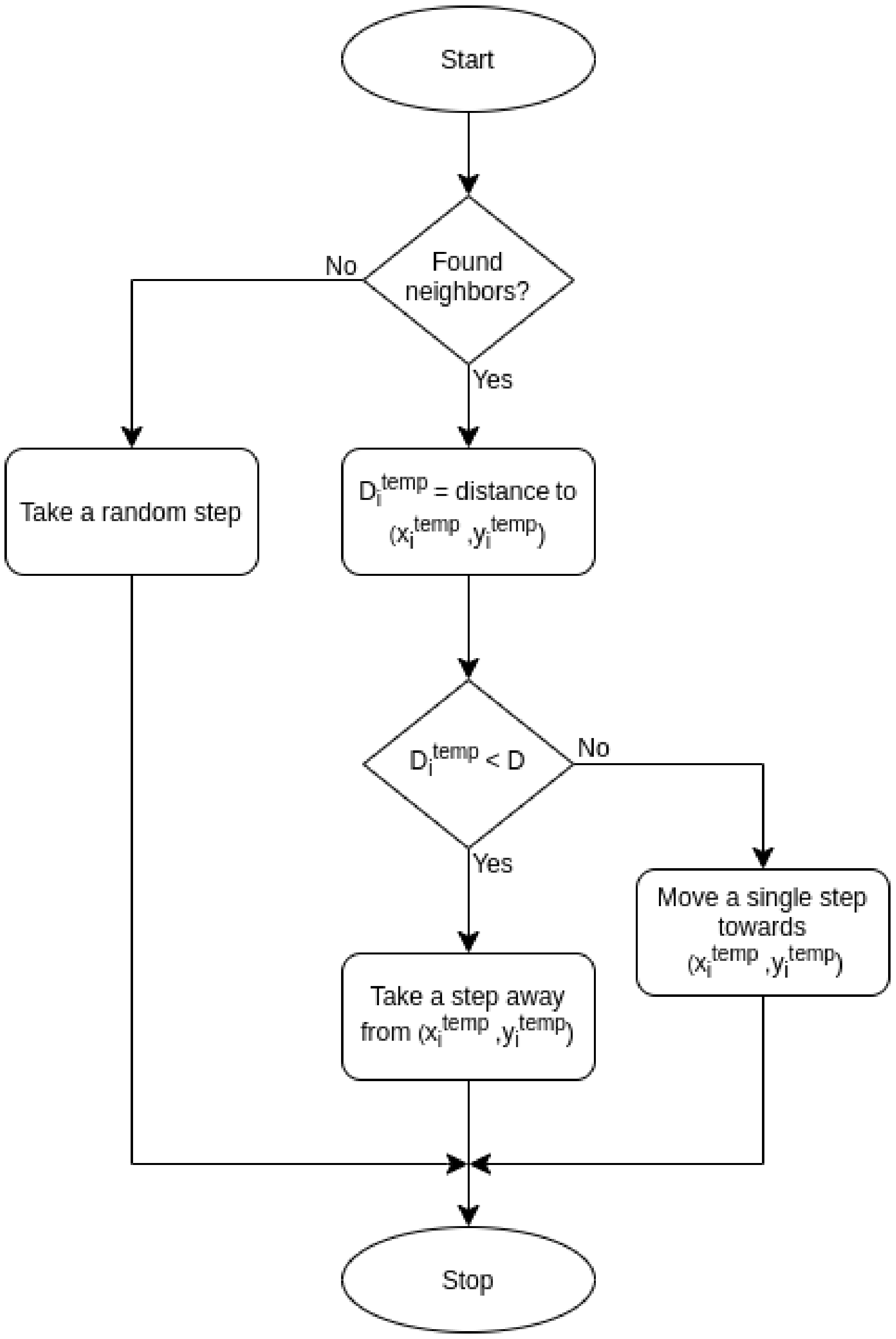

Start
Found neighbors?
No
Yes
Take a random step
D_i^{temp} = distance to (x_i^{temp}, y_i^{temp})
$D_i^{temp} < D$
No
Yes
Move a single step towards (x_i^{temp}, y_i^{temp})
Take a step away from (x_i^{temp}, y_i^{temp})
Stop

Figure 5.16: Flowchart diagram showing how robots wander in their environment when the target location is unknown.

Where:

- *(x_i^{temp}, y_i^{temp})* are the coordinates of the midpoint of the line joining the nearest and farthest robots
- D_i^{temp} is the distance to the midpoint of the line joining the nearest and the farthest robots
- D is the radius of the circle being formed

2. **Moving towards target**

Figure 5.17 shows a flowchart diagram which illustrates how robots move towards a known spill location to form the required surrounding behavior.

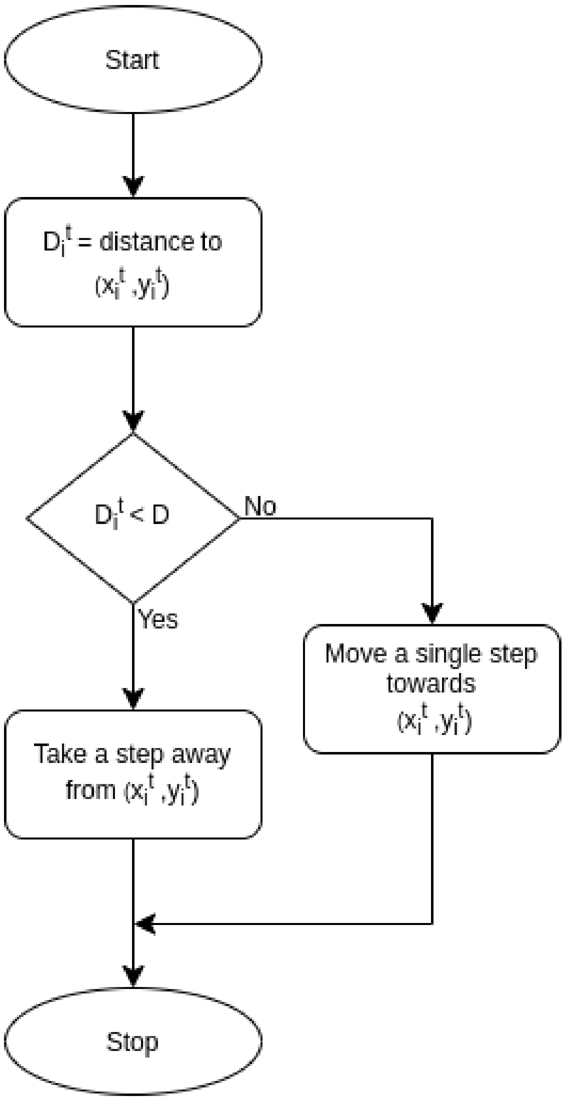

Start
D$_i^t$ = distance to (x$_i^t$,y$_i^t$)
D$_i^t$ < D
No
Yes
Move a single step towards (x$_i^t$,y$_i^t$)
Take a step away from (x$_i^t$,y$_i^t$)
Stop

Figure 5.17: Flowchart diagram showing how robots move towards a spill in order to surround it.

3. **Collision avoidance**

   The individuals require a mechanism to ensure that they do not collide with each other, i.e. agents must not attempt to occupy the same location at the same time. The robots can only sense objects in the environment, i.e. they can not sense each other. Robots rely on communication to determine the position of neighboring robots. Consequently, to avoid collisions with each other, the robots use received coordinates to calculate their distances to the nearest neighbor. The robot will then take a step away if the nearest neighbor is closer than a minimum allowable distance. The same approach applies when avoiding collisions with obstacles except sensors are used to determine the distance to the obstacles.

   Figure 5.18 shows a flowchart diagram which illustrates how robots are able to keep clear of each other in the environment.

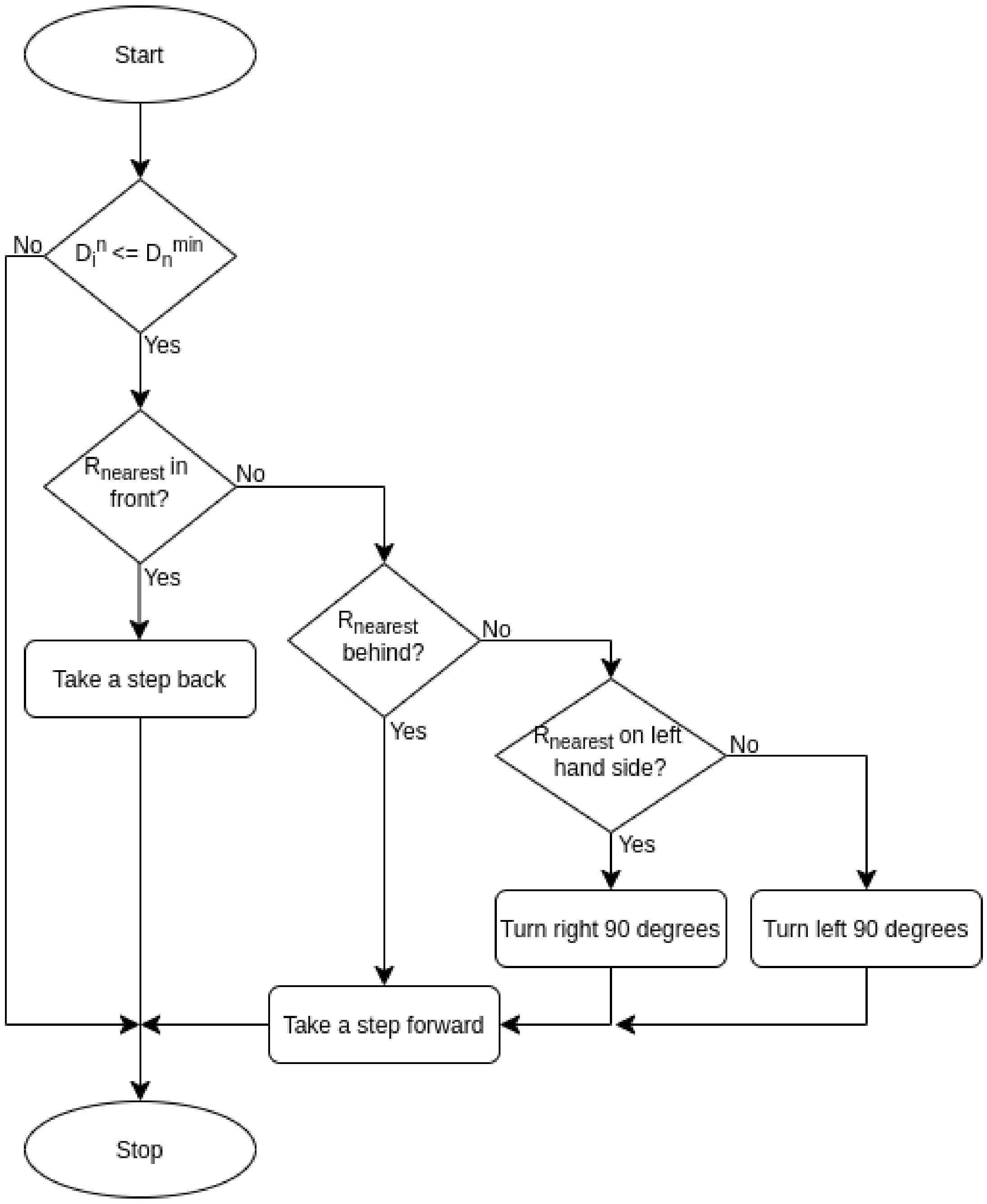


Figure 5.18: Flowchart diagram showing how robots are able to avoid colliding with each other.

Chapter 6

Implementation

This chapter describes the implementation of the various subsystems identified in the previous section (section 5).

6.1 Robot model

The environment of the simulation, which is described in a *map.inc* file, is kept fairly simple in order to focus on simulating the surrounding behavior. The robot devices, i.e. the sensors and actuators and its physical structure are implemented in a *robot.inc* file (the Player file structure was explained in section 5.2.6).

The driver for each robot is defined in a configuration file (see section 5.2.6). The driver specifies the devices which are part of the robot. The configuration file gets passed to Player whenever a new simulation is created. Most importantly, the driver for the Stage simulator is also defined in the configuration file. Parameters of this driver include a world file, which was described in section 5.2.6. Code listing 6.1 shows the content of a configuration file that defines a swarm of six robots in order to make clear how a simulation gets created.

```
 driver
(
  name "stage"
  plugin "stageplugin"
  provides ["simulation:0"]
  worldfile "robot_6.world"
)
```

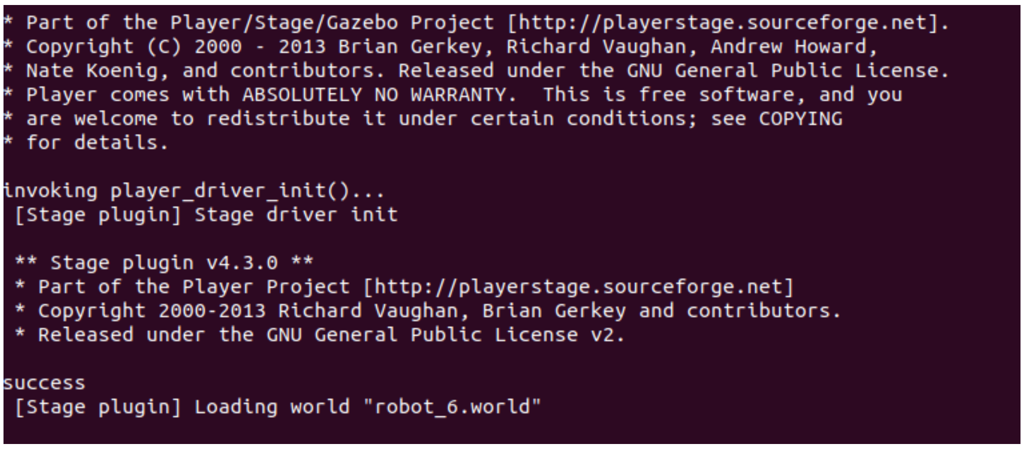
```
 8 driver
 9 (
10   name "stage"
11   provides ["6665:position2d:0" "6665:ranger:0" "6665:blobfinder:0"]
12   model "robot1"
13 )
14 driver
15 (
16   name "stage"
17   provides ["6666:position2d:0" "6666:ranger:0" "6666:blobfinder:0"]
18   model "robot2"
19 )
20 driver (...)
21 driver (...)
22 driver (...)
23 driver (...)
```

Listing 6.1: Configuration file example.

From code listing 6.1 the following observations are made:

- **Lines 1-7:** define a driver for the Stage simulator. The *name* of the driver is *stage* (line 3) and it is found in the *stageplugin* library (line 4). This driver *provides* data which conforms to the *simulation* interface (line 5). Lastly, the simulation environment is defined in a world file called *robot_6.world* (line 6).

  Consequently, when the configuration file gets invoked the first driver to be loaded is the Stage driver as shown in figure 6.1.



```
* Part of the Player/Stage/Gazebo Project [http://playerstage.sourceforge.net].
* Copyright (C) 2000 - 2013 Brian Gerkey, Richard Vaughan, Andrew Howard,
* Nate Koenig, and contributors. Released under the GNU General Public License.
* Player comes with ABSOLUTELY NO WARRANTY.  This is free software, and you
* are welcome to redistribute it under certain conditions; see COPYING
* for details.

invoking player_driver_init()...
 [Stage plugin] Stage driver init

 ** Stage plugin v4.3.0 **
 * Part of the Player Project [http://playerstage.sourceforge.net]
 * Copyright 2000-2013 Richard Vaughan, Brian Gerkey and contributors.
 * Released under the GNU General Public License v2.

success
 [Stage plugin] Loading world "robot_6.world"
```

Figure 6.1: An image showing Player loading the Stage driver to start the simulation.

- **Lines 8-13:** define a driver for a robot model, *robot1*. Line 10 specifies the name (*stage*) of the driver. The driver provides data that conforms to the *position2d*, *ranger* and *blobfinder* interfaces. Player exchanges this data through port 6665 (11). All these devices are linked to a model called *robot1* in the world file (12).

The rest of the lines of the configuration file can be explained in exactly like the lines 8-13. Once the world file has been loaded, the simulation is displayed on the screen through a simulation window entity that is defined in the world file. This is illustrated in figure 6.2.

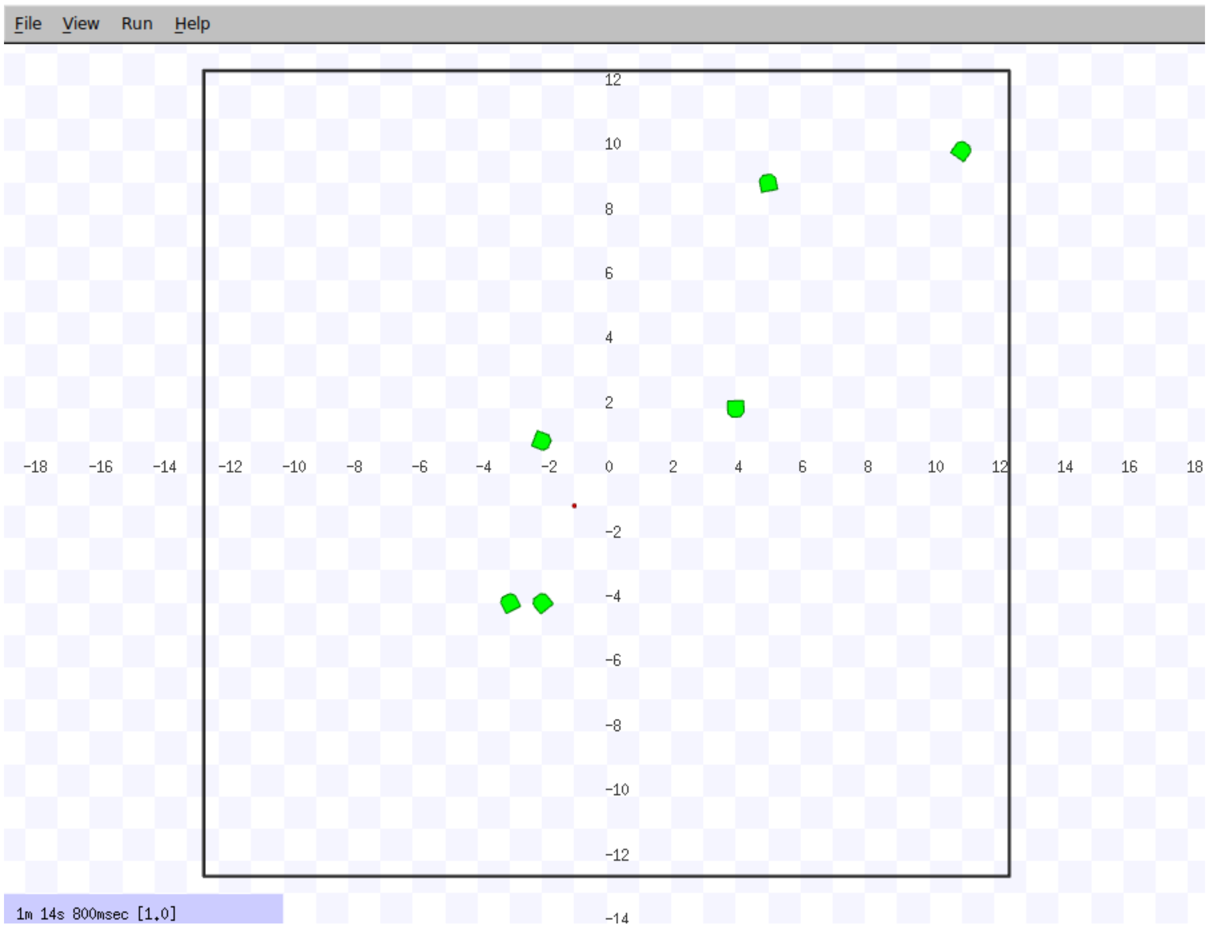
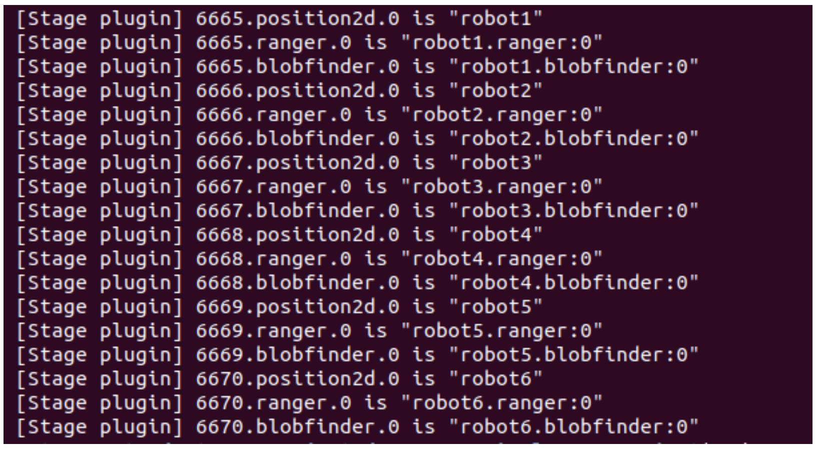


Figure 6.2: The simulation screen of the swarm of 6 robots.

Lastly, Player then listens for interactions with the devices from code through the ports specified in the driver definitions. This is illustrated in figure 6.3.



```
[Stage plugin] 6665.position2d.0 is "robot1"
[Stage plugin] 6665.ranger.0 is "robot1.ranger:0"
[Stage plugin] 6665.blobfinder.0 is "robot1.blobfinder:0"
[Stage plugin] 6666.position2d.0 is "robot2"
[Stage plugin] 6666.ranger.0 is "robot2.ranger:0"
[Stage plugin] 6666.blobfinder.0 is "robot2.blobfinder:0"
[Stage plugin] 6667.position2d.0 is "robot3"
[Stage plugin] 6667.ranger.0 is "robot3.ranger:0"
[Stage plugin] 6667.blobfinder.0 is "robot3.blobfinder:0"
[Stage plugin] 6668.position2d.0 is "robot4"
[Stage plugin] 6668.ranger.0 is "robot4.ranger:0"
[Stage plugin] 6668.blobfinder.0 is "robot4.blobfinder:0"
[Stage plugin] 6669.position2d.0 is "robot5"
[Stage plugin] 6669.ranger.0 is "robot5.ranger:0"
[Stage plugin] 6669.blobfinder.0 is "robot5.blobfinder:0"
[Stage plugin] 6670.position2d.0 is "robot6"
[Stage plugin] 6670.ranger.0 is "robot6.ranger:0"
[Stage plugin] 6670.blobfinder.0 is "robot6.blobfinder:0"
```

Figure 6.3: A screenshot of the devices that are simulated on Stage for each robot.

The robots are then created in code [1] as shown in code listing 6.2. The values passed to the robot constructor class match the values specified by the configuration file. Once the robots have been declared, they are then connected to the server, subscribed to devices, each robot's motor is enabled, and finally geometries are requested for each robot. The functions that perform all these actions (in the given order) are part of the robot wrapper class which was explained in section 5.4.1.

```
    int main(int argc, char** argv) {
    //----------------------------------------
    //declare robots
    //----------------------------------------
    robot robot1("localhost", 6665, "robot1");
    robot robot2("localhost", 6666, "robot2");
    robot robot3("localhost", 6667, "robot3");
    robot robot4("localhost", 6668, "robot4");
    robot robot5("localhost", 6669, "robot5");
    robot robot6("localhost", 6670, "robot6");
    .
    .
    .
    }
```

Listing 6.2: Declaring robots in controller code.

The first value passed to the robot class is the server address, the second is the port number through which Player expects to exchange data with the device and the last one corresponds to the name of the robot model as specified in the configuration file.

[1]all the code used to implement the system can be found on github: `https://github.com/einstein07/Simulating_Swarm_Behavior_for_Surrounding_a_Target`

Importantly, although the robot instances are given specific names, the algorithm still assumes that the robots are anonymous, i.e. robots have no specific roles whatsoever they all execute the same program.

6.2 Behavior synthesis

The distributed control methodology of this system is behavior-based, as alluded in the design section. This section seeks to explain how behavior coordination is implemented.

6.2.1 Behaviors

Listing 6.3 shows the interface of the *Movement coordination* class which implements the main behaviors of the swarm.

```
class movement{
      .
      .
      .
  public:
      void move( /*move to surround oil spill*/
                robot& robo,
                playerc_simulation_t *sim_proxy,
                double& forward_speed,
                double& turning_speed
            );
      void wander(/*target coordinates unknown*/
                robot& robo,
                playerc_simulation_t *sim_proxy,
                double& forward_speed,
                double& turning_speed
            );
      void avoid_collisions(/*avoid collisions */
                robot& robo,
                playerc_ranger_t *ranger_proxy,
                double& forward_speed,
                double& turning_speed
            );
}
```

Listing 6.3: Movement coordination class interface.

move-to-target

Individuals of the swarm use this behavior to move towards the oil spill location and to accordingly position themselves from the spill such that appropriate surrounding behavior emerges at the global level (see section 5.5.2 for this behavior's pseudocode). This behavior is the most complex of all the behaviors.

Lines 6 - 11 show the signature of this behavior. Its first parameter is a reference to a robot object (line 7), which provides information about the spill location thereby allowing the behavior to calculate the next appropriate step. The second parameter is a simulation proxy, which is provided by Player, to provide localization data(line 8). Lines 9 and 10 define two parameters that are used to set the translational and rotational commands for the motors.

wander

Individuals of the swarm use this behavior to move around the environment (in search of the spill) when the spill location is unknown. This behavior is relatively complex.

Lines 12 - 17 show the signature for this behavior. The parameters are the same as for the *move* function described above.

avoid-collisions

Individuals of the swarm use this behavior to avoid collisions within the swarm and with obstacles (i.e. walls) in the environment. This behavior can be classified as a survival behavior and it is based on reactive control rules, i.e. *condition-action*.

Lines 18 - 23 of code listing 6.3 show the function signature of this behavior. One of the parameters, line 20, is a ranger proxy which provides real time sensory data which is used by the function (behavior) to stimulate the necessary actions to ensure survival (i.e. not colliding).

6.2.2 Behavior coordination

To coordinate the behaviors, a behavior hierarchy is used such that commands from the highest ranking active behavior are sent to the robot motors and commands from lower ranking active behaviors are ignored.

The *collision-avoidance* (survival) behavior has the highest priority while the *wander* and *move-to-target* behaviors have the same ranking but are mutually exclusive thereby require no explicit action selection.

Figure 6.4 shows an activity diagram which illustrates how the various behaviors interact with each other to form the complete program executed by the individual robots of the system. The activity diagram does not explicitly show the *receive* component of communication because it is assumed that individuals instantaneously *receive* broadcasted information for as long as they are within communication range.

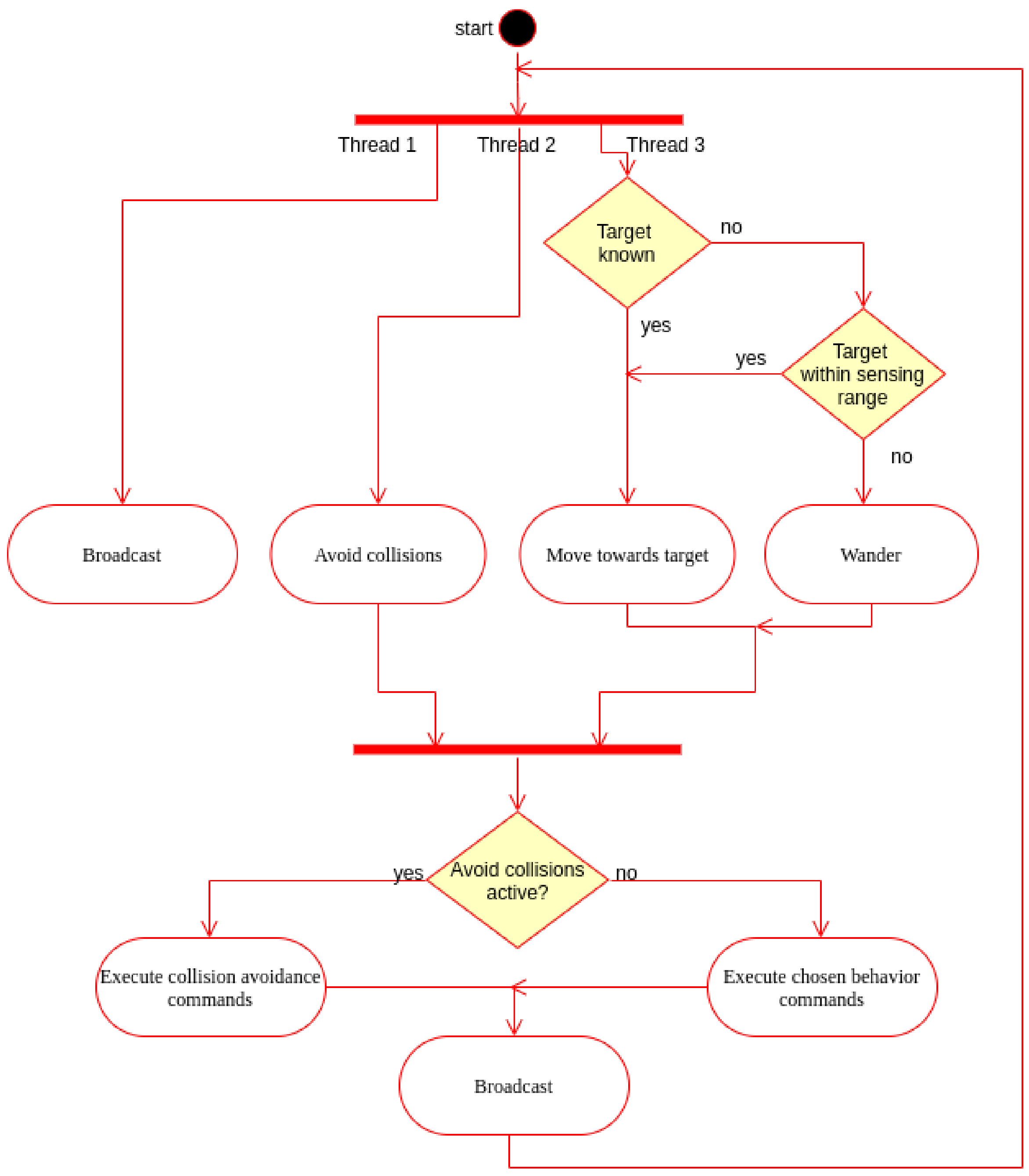
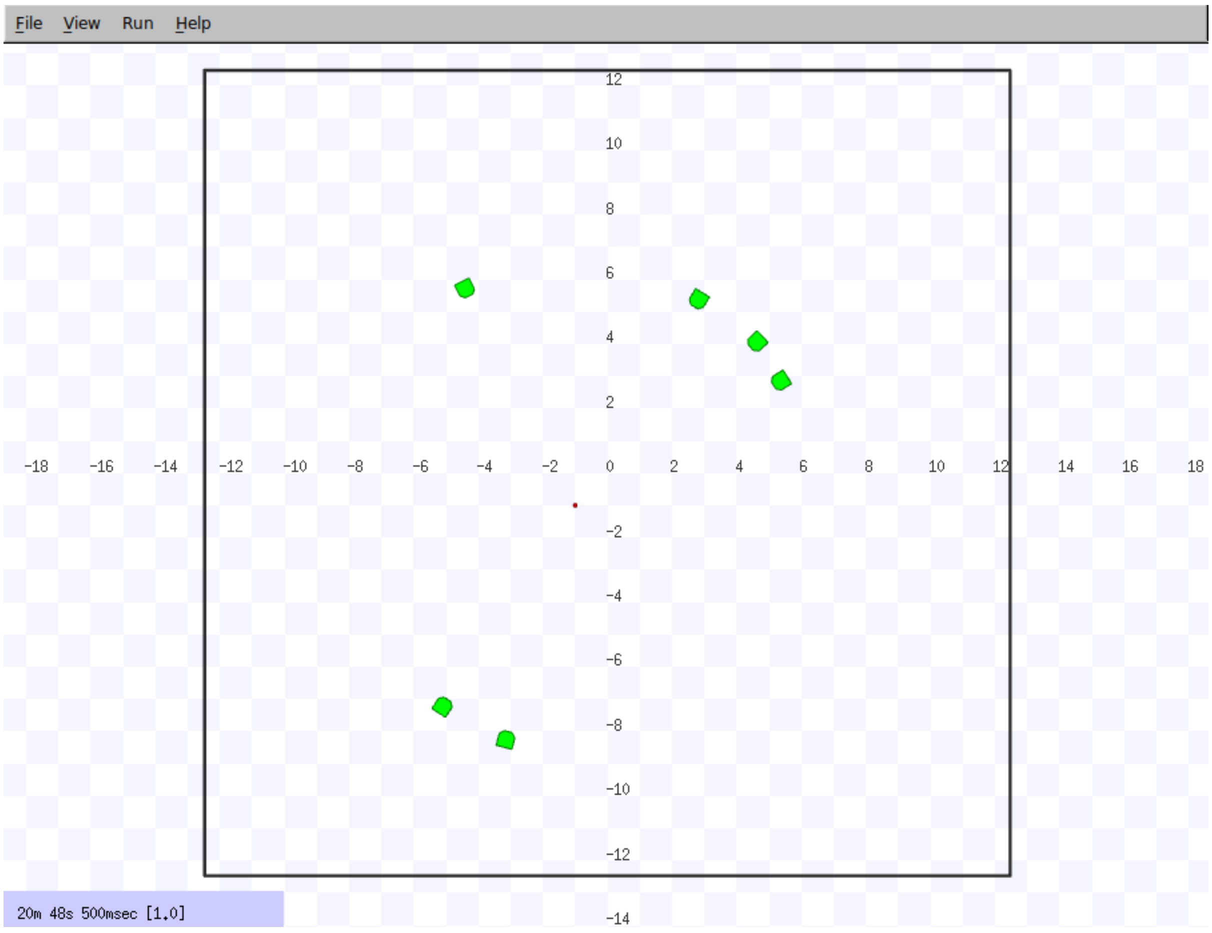


Figure 6.4: An activity diagram which shows how the code modules (behaviors) combine together to form a complete program which is executed by each robot of the swarm.

Figure 6.5 shows the final configuration of the swarm shown in figure 6.2 earlier after each robot has executed the program given by the activity diagram in figure 6.4 above. This behavior emerged (i.e. the program does not explicitly intruct the robots to form a circle) from the local interactions among the swarm as well as the interactions between the robots and the environment.



20m 48s 500msec [1.0]

Figure 6.5: A swarm of six robots demonstrating surrounding behavior emerging from local interactions.

Chapter 7

Testing

This chapter outlines the design of various tests that were carried out to verify that the subsystems work as expected as well as to evaluate the performance of the whole system. Firstly, testing of the various subsystems identified in section 5.3 is described, this is followed by the description of system level tests and lastly, tests that are associated with the acceptance tests formulated in section 2.1.4 are identified in the system verification section.

7.1 Subsystem Testing

7.1.1 Sensing

The aim of this test is to verify if individuals of the swarm are capable of sensing an oil spill when it is within a predetermined sensing range.

Experiment set-up

An individual with a sensing range set to 5 m, 10 m and 15 m is placed within a distance of 5 m, 10 m and 15 m respectively from the spill, in the direction of the spill. The number of times an individual is able to sense the spill is recorded. Each treatment is repeated 10 times.

Verification

In order to verify that the spill is indeed sensed, the spill coordinates are printed out upon detection, the number of times that a spill is sensed is recorded for each treatment. If the spill gets sensed more than five times out of the ten runs for each treatment then the subsystem is passed.

7.1.2 Actuation

The aim of this test is to verify if individuals of the swarm are able to operate in their environment.

Experiment set-up

An individual is set up to move in open space (i.e. no obstacles to be avoided or other agents in vicinity) towards a pre-assigned spill location. The number of times an individual is able to reach the spill location is recorded. This treatment is repeated 10 times.

Verification

If the spill location is reached more than five times out of the ten runs then the subsystem is passed.

7.1.3 Communication

The aim of this test is to verify if individuals are able to exchange information with other individuals within their communication range.

Experiment set-up

Two static individuals are placed in the simulation environment within communication range of each other. Each of the robots is set to continually broadcast its occupied coordinates while also receiving incoming coordinates. Communication ranges of 5 m, 10 m and 15 m are considered. Each treatment is repeated 10 times.

Verification

In order to verify that communication is happening between the robots, each robot is set to print out its received coordinates. The subsystem is passed if each robot is able to print out coordinates corresponding to the location of the robot within its communication range atleast five times out of the ten runs for each treatment.

7.2 Subsystem integration

7.2.1 Collision avoidance

The aim of this test is to verify if individuals are able to avoid collisions with other robots as well as obstacles in the environment. This test integrates two subsystems; sensing and communication.

Experiment set-up

Two static robots are placed in the simulation environment close enough to each other that the distance between them is less than the 'minimum allowable distance' yet also within communication range. Robots depend on received coordinates to determine the location of nearby robots, i.e. robots do not sense other robots as elaborated in the design section. Consequently, robots calculate the distance to the nearby robots from the received coordinates and compare this distance to the minimum allowable distance. We expect the robots to take a step from each other until the distance between them is greater than the minimum allowable distance.

The robot sensors only sense the target and obstacles. For this test, a robot is placed close enough to a wall in the environment that the distance inbetween is less than the 'allowable minimum distance'. We expect the robot to detect the wall and take a step away from it until the distance inbetween is greater than the minimum allowable distance.

The minimum allowable distance is varied between these values; 1.5 m, 1 m and 0.5 m.

Verification

According to the algorithm, if an agent is closer than the minimum allowable distance to a neighboring robot or an object then it ought to take a step away. This subsystem integration is passed if the agents are able to step away to avoid collisions.

7.2.2 Minimum allowable distance study

The aim of this study is to investigate the minimum allowable distance between agents and objects that does not result in moving agents colliding with each other or obstacles in the environment. This study integrates all three subsystems. Importantly, the value that will be obtained in this experiment will be used in all the subsequent experiments.

Experiment set-up

The position of an agent in the Player/Stage simulation is represented as a single coordinate, this refers to the position of the robots' origin (center of rotation). Since the robots are 0.5 meters in length and their center of rotation is their center, the minimum allowable distance is varied from 0.25 meters and subsequently incremented by 0.1 meter until a value which does not result in robots colliding with each other is obtained.

Each treatment is repeated 10 times. A fixed population size of 3 robots is used. There is no target to be surrounded but instead the robots are just allowed to wander in the simulation environment for 1000 iterations. Furthermore, translational speed is 0.8 meters per second and rotational speed is 60 degrees per second (these values also apply throughout the experiments, unless otherwise stated).

Verification

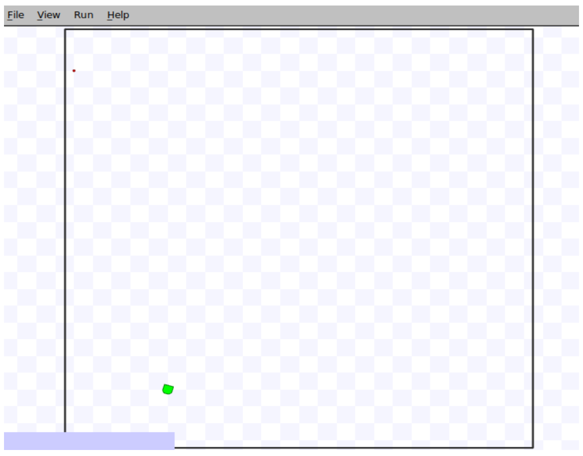
The number of collisions for each treatment is recorded. A value is accepted as fit for use if no collisions are recorded for all ten runs of the treatment.

7.2.3 Autonomy

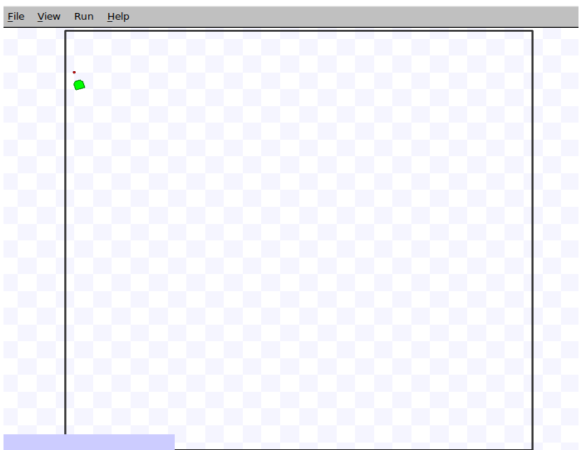
The aim of this test is to verify if individuals of the swarm are capable of navigating autonomously in the environment without colliding with obstacles. This test also verifies the effectiveness of the value obtained in the minimum allowable distance study.

Experiment set-up

A single individual of the swarm is set up to navigate towards a known spill location in the environment for a total of 1000 iterations. The treatment is repeated 10 times. For each run, the individual starts from a random location and the spill is also placed in a random location within the environment (see figure 7.1(a)). Essentially, the individual is tasked with forming a circle with a diameter of 0 m (i.e. circle diameter is set to 0 m) around the spill. However, we expect the individual to be able to stop at just less than a meter from the spill in order to avoid colliding with the spill (see figure 7.1(b)).



(a) An individual in its environment just before executing the task of navigating towards a known spill location.



(b) An individual in its environment just after executing the task of navigating towards a known spill location.

Figure 7.1: Autonomy subsystem test run.

Verification

To verify if the minimum allowable distance value obtained in the previous study is effective, the number of times the individual reaches the spill without colliding with obstacles in the environment (or even the spill) is recorded. If the individual is able to reach the spill without colliding with obstacles for all 10 runs then the individuals are truly autonomous and the value of the previous study can be used in the system level tests.

7.3 System Tests

7.3.1 Robustness

The aim of this experiment is to evaluate the performance of the system in aggressive environments (i.e. when individuals suddenly quit or malfunction in the middle of an operation).

Experiment set-up

A swarm of a total population size of six robots is set up in the simulation environment with the goal of surrounding a static oil spill. However, half the population of the robots are shutdown in the middle of the operation (after 500 iterations) to simulate malfunctioning robots. This treatment is repeated 10 times.

The communication range is set at 9 m and the circle being formed has a diameter of 15 m, table 7.1 shows the complete parameter values for this experiment set-up.

Verification

For this experiment the accuracy measure is given by the circularity error, which is defined by the following equation (which was explained in section 4.3.4):

$$C^{err} = 1/N \sum_{i=1}^{N} X_i ({D^t}_i - D/2)^2 \tag{7.1}$$

Parameters	**Values**
Initial population size	6 robots
Population size after 500 iterations	3 robots
Total number of iterations per run	1000
Rotational speed	60 degrees/s
Translational speed	0.8 m/s
Target location	(0, 0)
Target size	10 cm x 10 cm x 10 cm
Communication range	9 m
Target sensing range	9 m
Circle diameter	15 m
Minimum allowable distance between agents	0.8 m

Table 7.1: Parameter values for robustness test.

To pass this test the swarm should be able to form the required circle formation even after the population size is cut in half.

7.3.2 Scalability

The aim of this experiment is to evaluate the performance of the system across different population sizes.

Experiment set-up

A swarm of a total population size of three robots is set up in the simulation environment with the goal of surrounding a static oil spill. The population size is incremented by a single robot up until a population size of six robots is attained.

Each population size is treated as an independent treatment, and is repeated 10 times. The complete parameter values for this experiment are shown in table 7.2.

Verification

The circularity error (as given by equation 7.1) is used to measure the accuracy with which the required circle formation is formed around the oil spill. To verify if the system passed this test, performance should stay relatively constant across different population sizes.

Parameters	**Values**
Population size	Varied from 3 to 6 robots
Total number of iterations per run	1000
Rotational speed	60 degrees/s
Translational speed	0.8 m/s
Target location	(-1, -1)
Target size	10 cm x 10 cm x 10 cm
Communication range	9 m
Target sensing range	9 m
Circle diameter	15 m
Minimum allowable distance between agents	0.8 m

Table 7.2: Parameter values for scalability test.

7.3.3 Effects of target sensing and communication range on performance

The aim of this experiment is to determine how the target sensing and communication range affect the performance of the swarm.

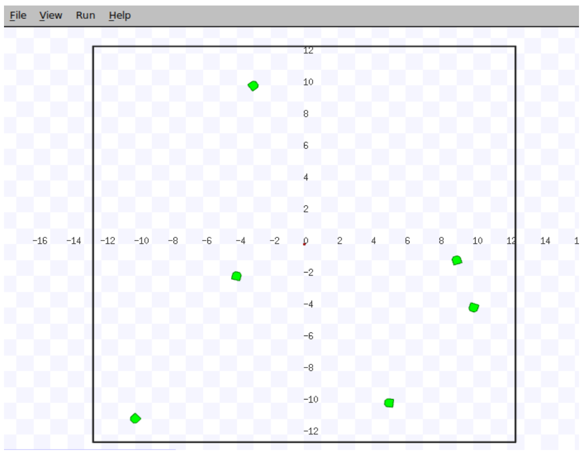
Experiment set-up

A swarm of a total population size of six robots is set up in the simulation environment with the goal of surrounding a static oil spill.

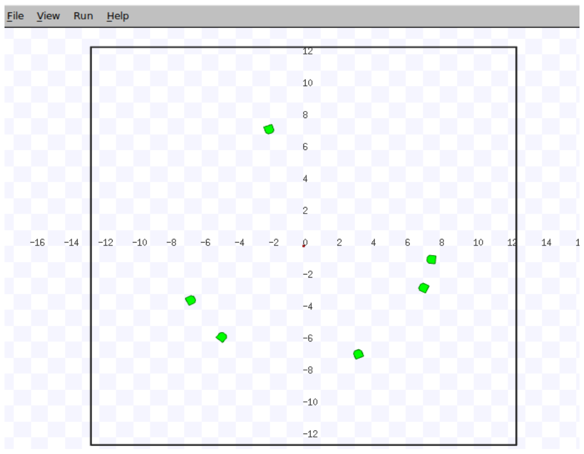
The communication range is set equal to the sensing range for all treatments.The following ranges are considered; 5 m, 10 m, 15 m and 30 m. The radius of the circle being formed is kept at a constant value of 15 m. Table 7.3 shows complete parameter values for this experiment design. The robots assume random initial positions for all runs. Figure 7.2(a) shows the initial position of the swarm for one particular run. Figure 7.2(b) shows the position of the swarm after 1000 iterations, the treatment considered corresponds to a communication and sensing range of 15 meters.

Parameters	**Values**
Population size	6 robots
Total number of iterations per run	1000
Rotational speed	60 degrees/s
Translational speed	0.8 m/s
Target location	(0, 0)
Target size	10 cm x 10 cm x 10 cm
Communication range	5 m, 10 m, 15 m
Target sensing range	5 m, 10 m, 15 m
Circle diameter	15 m
Minimum allowable distance between agents	0.8 m

Table 7.3: Parameter values for the communication range study.



(a) A swarm of six robots with a target sensing and communication range of 15 meters with the goal of surrounding a spill with the spill location initially not known.



(b) A swarm of six robots with a target sensing and communication range of 15 meters surrounding a spill after 1000 iterations.

Figure 7.2: Communication range study test run.

Verification

The circularity error, given by equation 7.1, is again used to measure the accuracy with which the required circle formation is formed around the spill. A simulation is used to evaluate the performance of the system and the variation of the circularity error for different range values is used to deduce the relationship between performance and the communication and sensing range.

7.4 System verification

In order to verify that all the acceptance tests are passed and that the respective user requirements and functional requirements formulated in section 2 are met, the ATPs are linked with the test and experiment designs described in the above sections of this chapter. This is outlined in table 7.4

ATP ID	**FRM ID**	**URM ID**	**Associated Test**
SRS-ATP-01	SRS-FRM-01	SRS-URM-01	Sensing (section 7.1.1)
SRS-ATP-02	SRS-FRM-02	SRS-URM-02	Autonomy (section 7.2.3)
SRS-ATP-03	SRS-FRM-03	SRS-URM-03	Communication (section 7.1.3)
SRS-ATP-04	SRS-FRM-04	SRS-URM-04	Robustness (section 7.3.1)
SRS-ATP-05	SRS-FRM-05	SRS-URM-05	Scalability (section 7.3.2)

Table 7.4: System acceptance tests procedures and their associated tests.

Chapter 8

Results

This section presents the results of the tests described in the previous section (section 7). The system is then analysed to verify that all the acceptance tests are passed and that the respective user requirements and functional requirements formulated in section 2 are met.

All simulations of the various tests and experiments were implemented in Player/Stage on a Linux Ubuntu 18.04.5 LTS PC with an Intel i5-6200U (@ 2.30GHz x 4) processor, Intel HD Graphics 520 (Skylake GT2) and 4 GB RAM. The simulation environment used accross all simulations was a 24 x 24 sq. meter area.

8.1 Subsystem tests

Table 8.2 below shows the results obtained from the subsystem tests described in the previous section (section 7.1).

Subsystem test	**Results**
Sensing (section 7.1.1)	Verification tests passed
Actuation (section 7.1.2)	Verification tests passed
Communication (section 7.1.3)	Verification tests passed

Table 8.1: Subsystem tests and their corresponding results

8.2 Integration tests

The two subsystem integration test designs (not including the Minimum allowable distance study) of the previous chapter sought to verify if the integrated subsystems worked as expected, more like the subsystem tests. Table 8.2 shows the results obtained from these tests.

Subsystem integration test	**Results**
Collision avoidance (section 7.2.1)	Verification tests passed
Minimum allowable distance study (section 7.2.2)	Results outlined below
Autonomy (section 7.2.3)	Verification tests passed

Table 8.2: Subsystem integration tests and their corresponding results

8.2.1 Minimum allowable distance study

The results of this experiment are illustrated by the graph in figure 8.1. The graph plots the mean across 10 runs and the error bars represent the standard deviation across runs. From the graph, we can see that the average number of recorded collisions is almost a maximum for distances less than 0.5 meters. The graph then decreases linearly for distances greater than 0.5 meters until eventually hitting the zero mark at 0.75 meters. From a distance of 0.75 meters and above no collisions occur between agents.

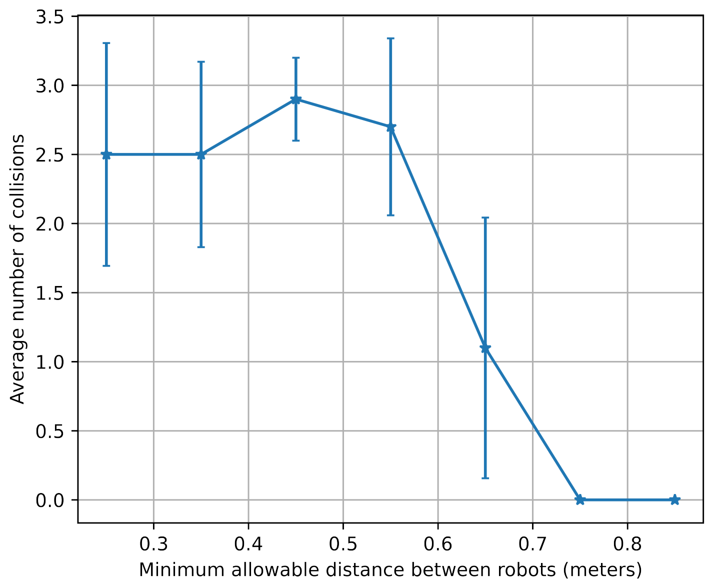


Figure 8.1: Results for the collision avoidance study.

To interpret these results one must consider the structure of the robots (0.5 meters in length), and their translational speed (0.8 m/s). The distance between agents is measured from their centers. Which means that when the allowable distance is 0.5 meters between agents then the edges of the robot bodies are just touching each other. Hence we expect maximum collisions for minimum allowable distances less than 0.5 meters.

Considering that the translational speed is 0.8 m/s, one possible conclusion to be made from this study is that an effective minimum allowable distance between agents should be equal or greater than the distance covered by agents per second for agents not to collide with each other or obstacles in the environment. The results of this study were tested through the Autonomy experiment and proved to be effective, consequently the minimum allowable distance was kept at 0.8 meters for all the system level tests.

8.3 System tests

The system level tests evaluate the performance of the whole system, and serve to verify if the system requirements are met or not. The results are presented in the same order as in the Testing section (section 7.3).

The results presented here were averaged over 10 runs and the error bars represent the standard deviation across runs.

8.3.1 Robustness

The graph in figure 8.2 shows the results obtained from this study. We can see from the graph that the circularity error starts very low as only few individuals have the oil spill location coordinates. The individuals who are aware of the spill information spread it through the swarm and consequently the circularity error increases as more individuals become aware of the spill location.

However, around 150 iterations the circularity error starts decreasing as individuals move towards the spill to form the circle. It decreases gradually until it becomes 0 just after 500 iterations. Which means that the swarm manages to stabilize on the desired circle trajectory.

Nonetheless, this stability does not last long because the circularity error starts increasing again just before 700 iterations. This means that new individuals get hold of the spill coordinates. This follows the reduction of the population size by half at 500 iterations. The system does well to restabilize itself soon after, which means that the robots which just got hold of the spill information are able to position themselves on the required circle formation and then the circularity error of 0 is maintained until termination after 1000 iterations.

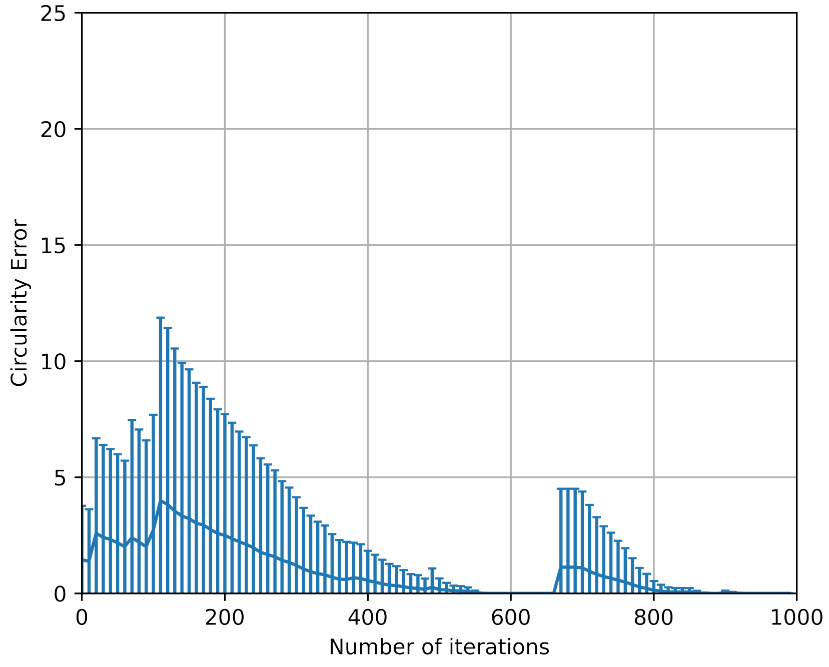


Figure 8.2: Results for the robustness study.

8.3.2 Scalability

As mentioned in section 7.3.2, population sizes of 3, 4, 5 and 6 robots were considered for this study. The results for all the different population sizes are shown in figure 8.3.

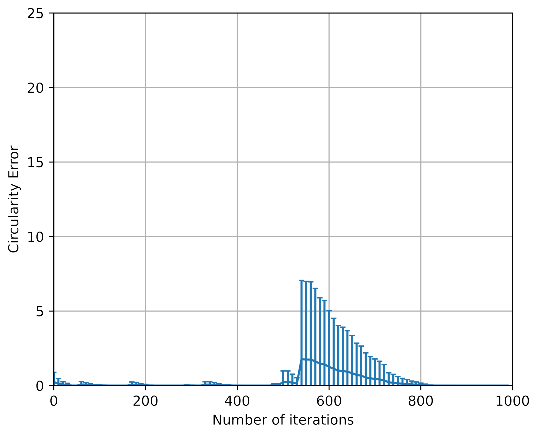
The results obtained do not seem to follow an obvious pattern. However, what seems to be a common theme across all of them is the initial value of the circularity error; the initial values of the error is zero or very close to zero because at the begining the spill location is not known to the swarm or very few individuals are aware of it. The general observation is that the circularity error then increases as more members become aware of the spill location. From then on it would help to consider the graphs one by one because the observations differ drastically.

Firstly, the circularity error for the study which considers 3 robots, shown in subfigure 8.3(a), stays at zero until at about 500 iterations. The spill information remains unknown for many iterations, i.e. it takes long for the spill to be located because the swarm is very small and as such does not cover much ground when wandering around searching for the spill. However, once the spill information is known it is quickly spread throughout the swarm and the robots are able to stabilize on the desired circle formation at around 800 iterations.

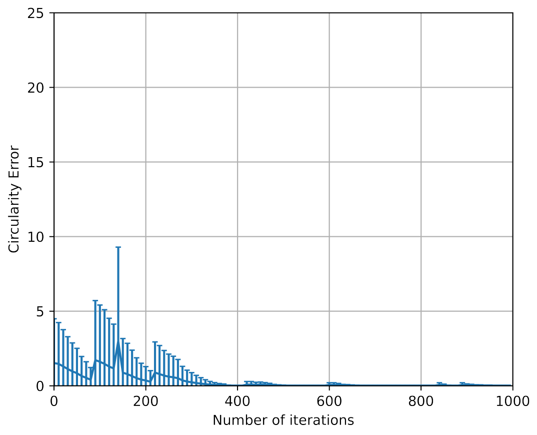
In subfigure 8.3(b), the circularity error graph shown corresponds to the study which considers a population of 4 robots. The error starts off somewhat around 2 and then it decreases to just above zero. However, it quickly increases as the spill information is spread throughout the swarm. The agents who just became aware of the spill location start forming the required circle and the circularity error decreases but it does not settle at zero, it increases once again as the spill information reaches to even more agents. Eventually at around 400 iterations the system stabilizes as the individuals distribute themselves as needed to form the required circle formation.

Subfigure 8.3(c) shows the results obtained from a population of 5 robots. Here yet again we see a completely different pattern from the two populations considered thus far. The system stabilizes soon after 200 iterations which means that all the individuals who are aware of the spill manage to spread out on the desired circle formation. However, the system destabilizes once again at around 400 iterations, which means that new individuals get hold of the spill information. These new individuals then start moving towards the spill to form the required circle formation and the error decreases until it reaches zero once again around 700 iterations and the system stabilizes until termination after 1000 iterations.

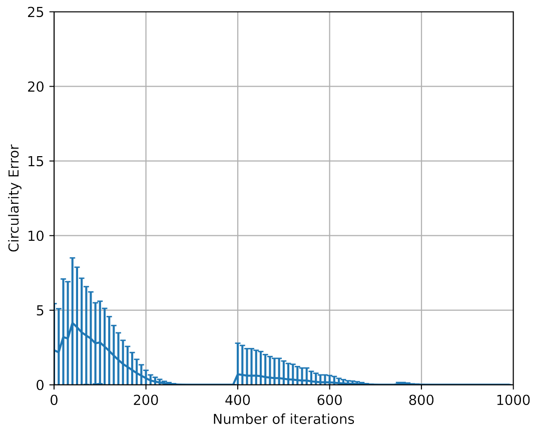
Subfigure 8.3(d) shows the results obtained from the study involving a population of 6 robots. The circularity error starts decreasing at around 100 iterations which means that the individuals are starting to form the required circle around the spill. The circularity error then remains constant just above zero, which means that the individuals have not accurately aligned themselves on the desired circle formation. This was shown to be caused by individuals not falling in line on the circle formation (i.e. one individual



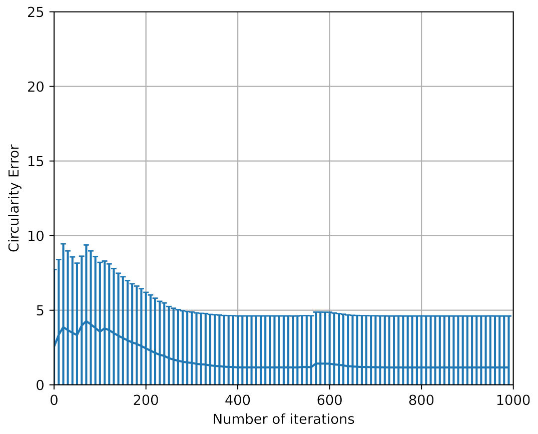
(a) Circularity error for 3 robots



(b) Circularity error for 4 robots



(c) Circularity error for 5 robots



(d) Circularity error for 6 robots

Figure 8.3: Results of the scalability study.

not moving around another individual infront of it), but rather standing behind other individuals so as to avoid collisions. This is illustrated in figure 8.4 (the two robots around coordinates (-5, 5)).

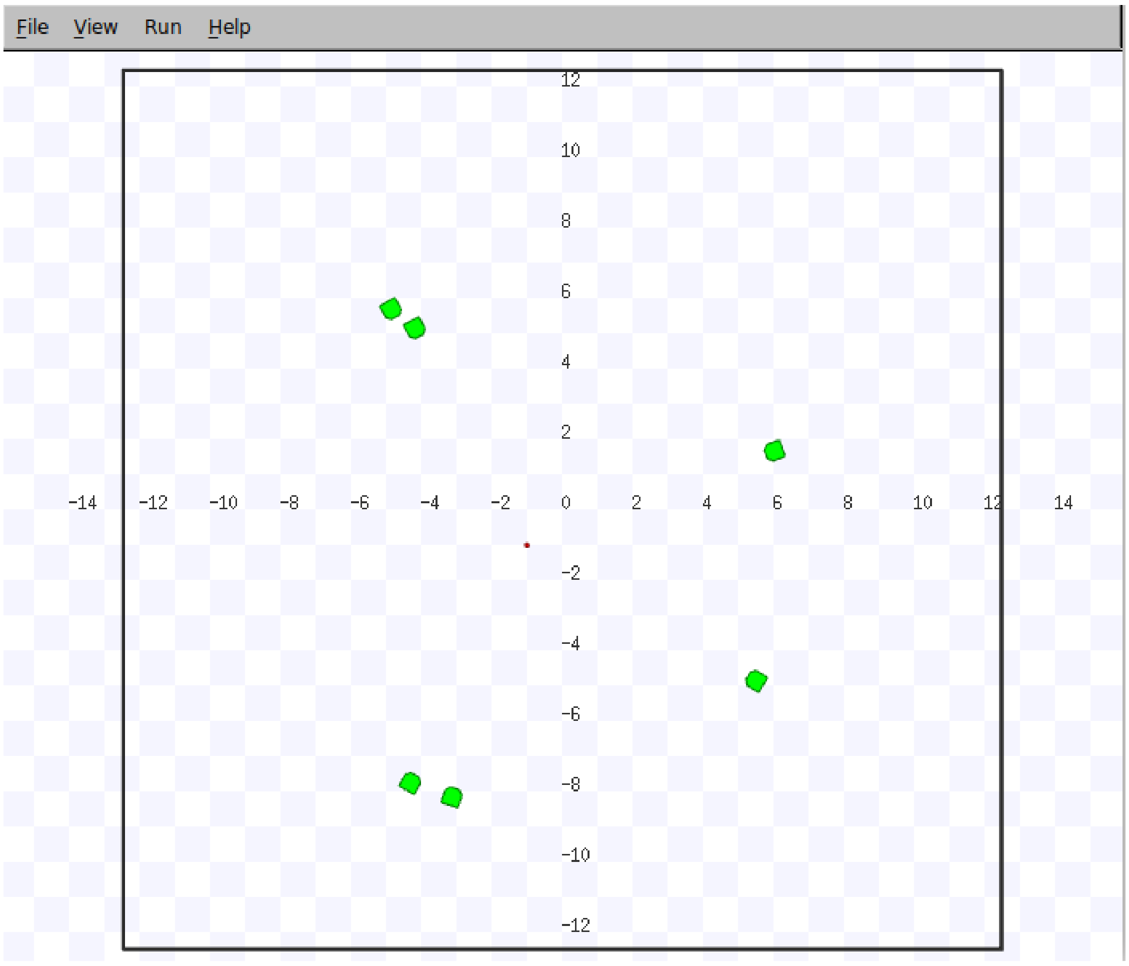


Figure 8.4: A robot stuck behind another to avoid collisions.

How does the spill information spread?

The graph in figure 8.5 shows how the spill information is spread through the swarm. This investigation was carried out concurrently with the scalability study which considered a population of six robots. However, the graph shows that the number of individuals aware of the spill location never reaches six robots even though the population size under consideration is six robots. Besides the fact that not all members get hold of the spill location in each run, this is also because in some runs the spill is never located (remember the graph plots average values across ten runs).

The general trend, however, is that the number of indiduals aware of the spill location increases as the run progresses. This makes sense because as the individuals which are aware of the spill move towards the spill they encounter (come into communication range of) clueless individuals and consequently, the spill location is shared with these unaware indiduals thereby increasing the number of robots aware of the spill.

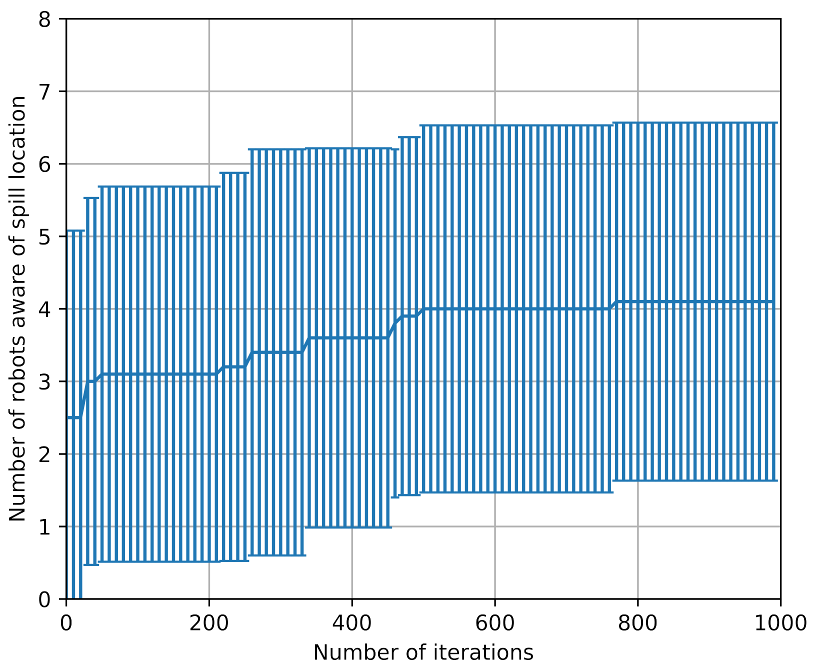
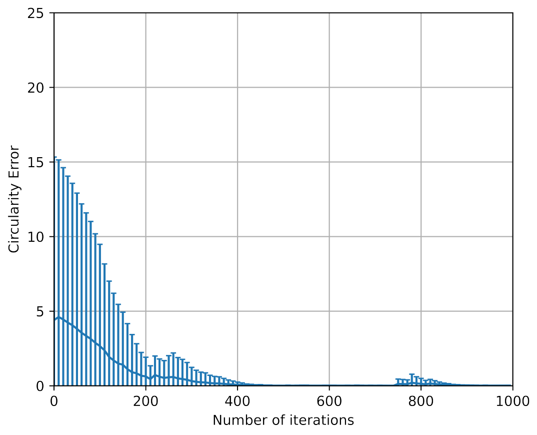


Figure 8.5: Graph showing how the number of individuals aware of the spill location varies for a population of six robots.

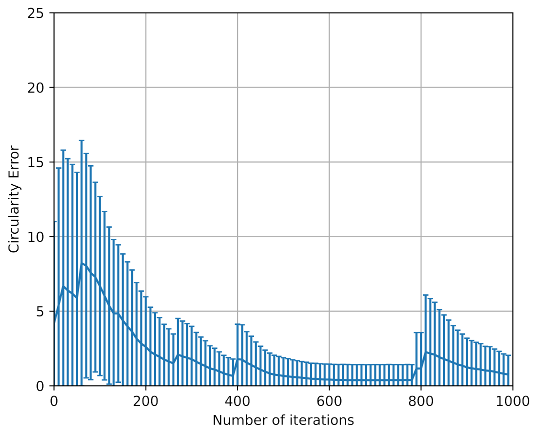
8.3.3 Effects of target sensing and communication range on performance

This study seeks to investigate the effects of the target sensing and communication range on the performance of the system. All system parameters are kept constant except for the target sensing and communication range. These two parameters are set to be equal at all times and take four different range values as deliberated in section 7.3.3. Figure 8.6 shows the results of this study. The graphs are each considered individually.

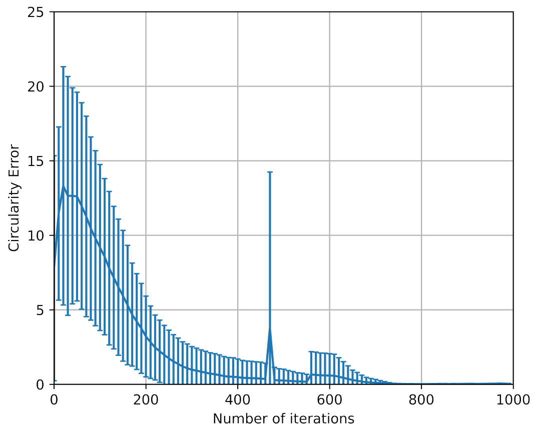
Subfigure 8.6(a) shows the results for a target sensing and communication range which is a third of the diameter of the circle being formed, i.e. 5 meters. The circularity error starts off at just under a value of 5, and then decreases fast enough for the system to stabilize at 400 iterations. This implies that the individuals who are aware of the spill



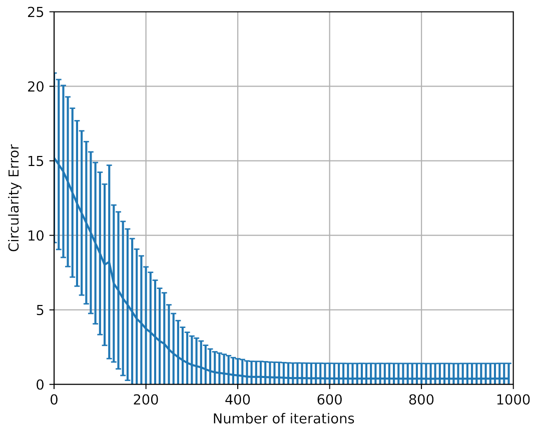
(a) Circularity error for 6 robots with target sensing and communication range of 5 meters



(b) Circularity error for 6 robots with target sensing and communication range of 10 meters



(c) Circularity error for 6 robots with target sensing and communication range of 15 meters



(d) Circularity error for 6 robots with target sensing and communication range of 30 meters

Figure 8.6: Results of the *range* study.

location are able to mobilise towards the spill and form the required circle formation to surround the spill. However, the nature of the circularity graph (it just decreases without ever going up) seems to suggest that individuals who are aware of the spill location do not share it with any more individuals. Tiny spikes are also observed at around 800 iterations, which are caused by individuals who had just discovered the spill but manage to quickly position themselves on the circle formation.

In certain cases the spill could not get located at all which resulted in the emergence of communities as illustrated in figure 8.7, this matches the observation made in [36].

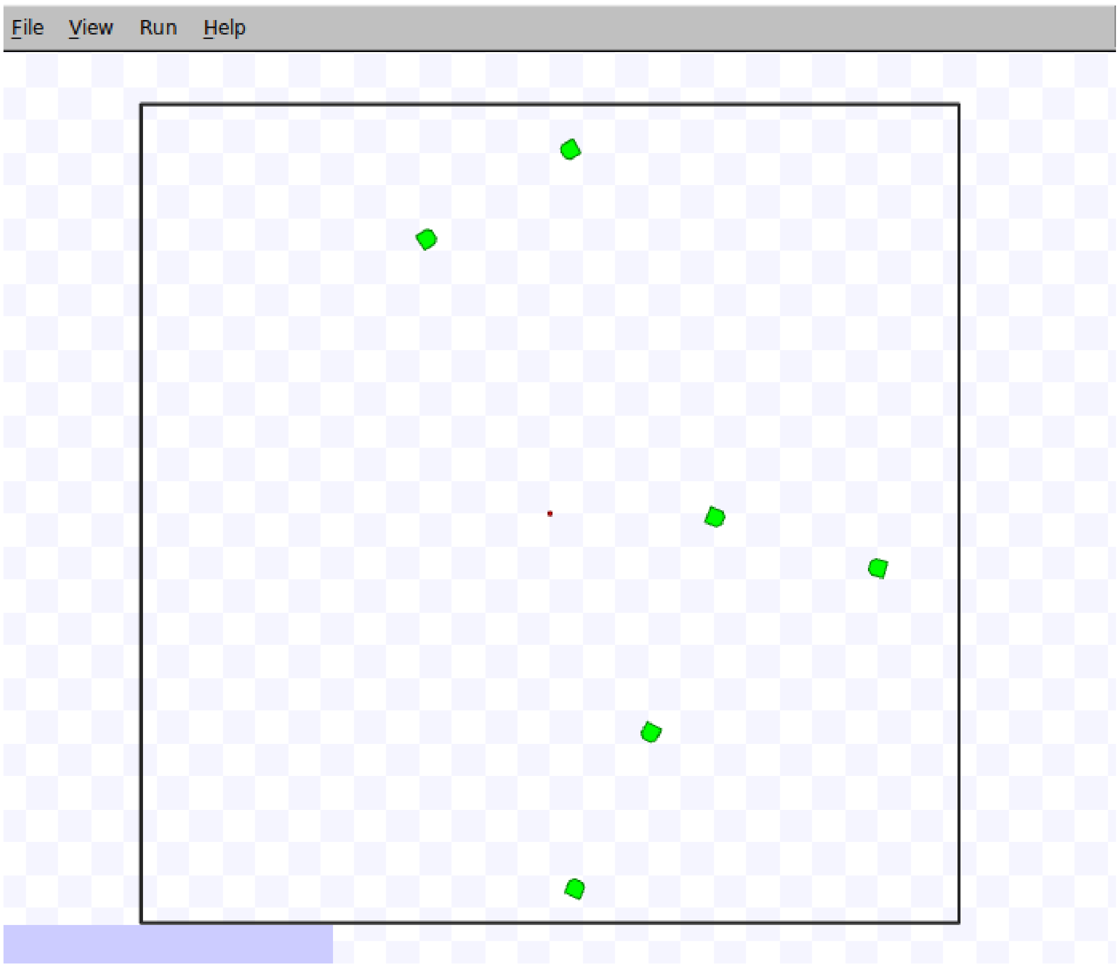


Figure 8.7: Emergence of communities in the swarm.

Subfigure 8.6(b) shows the results obtained when a target sensing and communication range of 10 meters was considered. The circularity error is initially low, and then it increases and peaks just before 100 iterations. At this point, the spill information has spread sufficiently throughout the swarm and individuals that are aware of the spill location move towards the spill to form the required surrounding behavior; which is illustrated by the decreasing circularity error. However, the circularity error keeps getting 'bumped' as it attempts to get to zero. This means that more individuals keep getting hold of the spill information as it spreads throughout the swarm. Consequently, the system does not manage to stabilize within the 1000 iterations as new individuals keep getting hold of the target information.

Subfigure 8.6(c) shows the results for a target sensing and communication range which is equal to the diameter of the circle being formed, i.e. 15 meters. The circularity error increases sharply as the spill location coordinates are quickly spread throughout the swarm. The individuals who are now aware of the spill location move towards the spill and start forming the required circle formation, as indicated by the exponential decrease in the circularity error. The circularity error does not settle at zero, it gets bumped just before 600 iterations as new individuals become aware of the spill location. These new individuals are also able to position themselves on the required circle formation soon enough and the circularity error settles at zero, as the system has stabilized and members are well distributed on the required circle formation; exhibiting the required global surrounding behavior.

Subfigure 8.6(d) shows the results for a target sensing and communication range which is twice the diameter of the circle being formed, i.e. 30 meters. The circularity error starts very high and never goes up after, which means that all the individuals are instantaneously aware of the target location from the get go (which is expected because this range covers the whole simulation environment). The error then decreases as robots start forming the required circle. However, the error does not settle to a value of zero. This can be attributed to the same phenomenon observed in figure 8.4. We do, however, observe that the error coverges relatively quickly for this study, i.e. surrounding behavior emerges relatively quicker.

8.3.4 System verification

A summary of the system verification results is shown in table 8.3. The results obtained for each acceptance test procedure are briefly described vis-a-vis how they link to the required acceptance condition in the following subsections.

ATP ID	**Results**	**Associated results section**
SRS-ATP-01	Passed	Section 8.1
SRS-ATP-02	Passed	Section 8.2
SRS-ATP-03	Passed	Section 8.1
SRS-ATP-04	Passed	Section 8.3.1
SRS-ATP-05	Passed	Section 8.3.2

Table 8.3: System acceptance tests procedures and their associated results.

SRS-ATP-01

This ATP is passed if the indiduals are able to sense the spill when the spill is within their sensing range. In the tests done in section 7.1.1, three different sensing ranges were considered and the indiduals demonstrated the capability to detect the spill and its location coordinates (which were subsequently printed out) in all three tests, respectively. Consequently, this ATP is passed.

SRS-ATP-02

This ATP is passed if the individuals demonstrate the capability to move around in their environment autonomously without colliding with obstacles.

This test was conducted after the *Minimum allowable distance* study, which investigated the minimum distance between individuals and objects which did not result in collisions.

The tests of section 7.2.3 demonstrated that the individuals are indeed capable of moving around their environment autonomously without collisions, and consequently this ATP is passed.

SRS-ATP-03

This ATP is passed if the individuals are able to communicate with each other when within communication range. In the tests conducted in section 7.1.3 the individuals were able to continually print the location coordinates of the other individual within their communication range. This indicates the ability of the individuals to both broadcast information to other indiduals (i.e. agents could broadcast their occupied coordinates) and to receive information (i.e. agents could receive coordinates occupied by agents within their communication range) from other indiduals. This ATP is passed.

SRS-ATP-04

This ATP is passed if the sudden failure of members of the swarm in the middle of an operation does not result in a catastrophic failure of the whole system.

In the test of section 7.3.1, the population size gets decreased by half nonetheless the system demonstrated the ability to still form the required surrounding behavior around the oil spill, and not only that but with the same accuracy as without any members quitting. This ATP is passed.

SRS-ATP-05

This ATP is passed if the performance of the system remains relatively stable across a range of population sizes.

In the tests conducted in section 7.3.2, the results show that as the population size increases the accuracy of the system decreases but overall performance (i.e. ability of the swarm to form the required circle formation) remains relatively stable. Therefore this ATP is passed.

Chapter 9

Discussion

The results obtained from the robustness study show that the sudden loss of individuals in the middle of an operation has no effect on the overall performance of the system, i.e. the swarm is still able to self-organise to form the required circle formation around the oil spill. This means that behavior-based control can be used to obtain reliable self-healing behavior. This is exactly what we had expected because behavior-based control calculates the steps to be taken by an individual robot based on real time sensor input. This input then specifies both activation conditions, which allow the behavior to generate actions, and stimuli, from which necessary actions are generated [38]. Consequently, the swarm can continually adapt to the loss of individuals. This is an important observation, especially considering the continually changing environmental conditions in the open seas.

The results obtained from the scalability study show that the system is able to achieve the required surrounding behavior for varying population sizes. However, it was observed that when the population size is very small (three robots) it takes longer (about 500 iterations) for the target to be located. This makes sense because larger population sizes can cover a lot of ground within a short period of time, and hence locate the target faster, compared to smaller populations.

The target sensing and communication range affects how fast the target is located as well as how fast the target information is spread through the swarm. A wider target sensing and communication range results in the target being located in shorter periods of time as well as a faster rate of information transmission through the swarm. Ultimately this affects the speed at which the surrounding behavior is achieved by the swarm. This observation is coherent with the observation made in [8]. However, a wider target sensing and communication range might not be desirable despite these benefits because it introduces the possibility of wireless traffic congestion as elaborated in [8].

Lastly, in one particular run of the target sensing and communication range (the range value was a third of the diameter of the circle that was being formed) study, the swarm failed to locate the oil spill and this resulted in the emergence of communities. The same observation was made in [36]. It would seem that when the swarm has no objective, i.e. when the oil spill information is not known, social behavior emerges as would be the case in mammals when they do not have any tasks. This observation could be important to biologists studying the evolution of collective behaviors because it seems to suggest some of the origins of collective social behavior.

Chapter 10

Conclusions

In this work we presented and reviewed four target surrounding algorithms for a swarm of robots. Based on the system requirements of this work, one was chosen and implemented. The performance and reliability of this algorithm was then evaluated using the Player/Stage simulator. The results obtained using simulated robotic swarms show that the algorithm can reliably obtain surrounding behavior for varying population sizes (i.e. scalable) as well as for operations in which other member robots suddenly quit or malfunction in the middle of the operation (i.e. robust).

Moreover, from the results obtained we conclude that behavior-based control can be used to obtain efficient surrounding behavior in a swarm with limited communication, in which distributed communication is achieved through a local communication strategy. Lastly, experiments show that a wider target sensing and communication range reduces the time taken for the surrounding behavior to emerge.

Chapter 11

Recommendations

The following recommendations are made for further work:

- A natural extension of this work would be the containment of a large oil spill that occupies more than a single coordinate. Such a study would require an investigation of an effective method of distributed path planning to map the spill contour. In this implementation, the swarm would not form a predetermined pattern, i.e. a circle, instead the swarm would need to evenly distribute itself on the spill contour as detected by the shape or pattern formed by the spill. One possible approach would be the execution of aggregation behavior once the spill is located, followed by chain formation behavior such that the chain lies on the boundary of the spill.
- Further investigations should be done of effective practical methods of implementing distributed communication in a swarm with limited hardware capabilities. Practical implementations of robot swarms typically rely on a central device for communication which limits the capabilities of practical systems. Such an investigation could include a quantitative analysis of the bandwidth requirements for different communication ranges and population sizes.

Bibliography

[1] Y. Tan and Z.-Y. Zheng, "Research advance in swarm robotics," *Defence Technology*, vol. 9, no. 1, pp. 18–39, 2013. doi: 10.1016/j.dt.2013.03.001

[2] D. E. Jackson and F. L. Ratnieks, "Communication in ants," *Current Biology*, vol. 16, no. 15, pp. R570–R574, 2006. doi: https://doi.org/10.1016/j.cub.2006.07.015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960982206018343

[3] T. Narumi, K. Uemichi, H. Honda, and K. Osaki, "Self-organization at the first stage of honeycomb construction: Analysis of an attachment-excavation model," *PLOS ONE*, vol. 13, no. 10, pp. 1–15, Oct 2018. doi: 10.1371/journal.pone.0205353. [Online]. Available: https://doi.org/10.1371/journal.pone.0205353

[4] H. Wallraff, *Avian Navigation: Pigeon Homing as a Paradigm.* Springer, Verlag Berlin Heidelberg, Nov 2010, vol. 208. ISBN 978-3-540-26432-3

[5] K. James, E. Russell C., and S. Yuhui, *Swarm Intelligence.* Morgan Kaufmann Publishers, San Francisco, 2001, ch. On Our Nonexistence as Entities: The Social Organism, pp. 81–132. ISBN 1-55860-595-9

[6] R. Arnold, K. Carey, B. Abruzzo, and C. Korpela, "What is a robot swarm: A definition for swarming robotics," *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019. doi: 10.1109/uemcon47517.2019.8993024

[7] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30552-1 pp. 10–20.

[8] K. Swaminathan, "Self-organized formation of geometric patterns in multi-robot swarms using wireless communication," Master's thesis, University of Cincinnati, Cincinnati, Ohio, 2005.

[9] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, Mar 2013. doi: 10.1007/s11721-012-0075-2. [Online]. Available: https://doi.org/10.1007/s11721-012-0075-2

[10] H. Singh, N. Bhardwaj, S. K. Arya, and M. Khatri, "Environmental impacts of oil spills and their remediation by magnetic nanomaterials," *Environmental Nanotechnology, Monitoring & Management*, vol. 14, p. 100305, 2020. doi: https://doi.org/10.1016/j.enmm.2020.100305. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2215153219302338

[11] G. Beni, "From swarm intelligence to swarm robotics," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30552-1 pp. 1–9.

[12] M. A. Hsieh, Á. Halász, S. Berman, and V. Kumar, "Biologically inspired redistribution of a swarm of robots among multiple sites," *Swarm Intelligence*, vol. 2, no. 2, pp. 121–141, Dec 2008. doi: 10.1007/s11721-008-0019-z. [Online]. Available: https://doi.org/10.1007/s11721-008-0019-z

[13] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020. doi: 10.3389/frobt.2020.00036. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00036

[14] O. Soysal and E. Şahin, "A macroscopic model for self-organized aggregation in swarm robotic systems," in *Swarm Robotics*, E. Şahin, W. M. Spears, and A. F. T. Winfield, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-71541-2_3. ISBN 978-3-540-71541-2 pp. 27–42. [Online]. Available: https://doi.org/10.1007/978-3-540-71541-2_3

[15] B. Shucker and J. Bennett, *Distributed Autonomous Robotic Systems.* Springer, Tokyo, Jan 2007, vol. 6, ch. Scalable Control of Distributed Robotic Macrosensors, pp. 379–388. ISBN 978-4-431-35873-2

[16] V. Sperati, V. Trianni, and S. Nolfi, "Self-organised path formation in a swarm of robots," *Swarm Intelligence*, vol. 5, no. 2, pp. 97–119, Jun 2011. doi: 10.1007/s11721-011-0055-y. [Online]. Available: https://doi.org/10.1007/s11721-011-0055-y

[17] A. L. Christensen, R. O'Grady, and M. Dorigo, "Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots," *Swarm Intelligence*, vol. 2, no. 2, pp. 143–165, Dec 2008. doi: 10.1007/s11721-008-0012-6. [Online]. Available: https://doi.org/10.1007/s11721-008-0012-6

[18] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella, "Communication assisted navigation in robotic swarms: Self-organization and cooperation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Jan 2011. doi: 10.1109/IROS.2011.6094454 pp. 4981–4988.

[19] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2, pp. 97–120, Dec 2008. doi: 10.1007/s11721-008-0016-2. [Online]. Available: https://doi.org/10.1007/s11721-008-0016-2

[20] M. A. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo, "Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making," *Swarm Intelligence*, vol. 5, no. 3, pp. 305–327, Dec 2011. doi: 10.1007/s11721-011-0062-z. [Online]. Available: https://doi.org/10.1007/s11721-011-0062-z

[21] I. Navarro and F. Matía, "An introduction to swarm robotics," *ISRN Robotics*, vol. 2013, p. 608164, Sep 2012. doi: 10.5402/2013/608164. [Online]. Available: https://doi.org/10.5402/2013/608164

[22] A. Moseman, "Mit invents a swarm of sea-skimming, oil-collecting robots," Nov 2019. [Online]. Available: https://www.discovermagazine.com/technology/mit-invents-a-swarm-of-sea-skimming-oil-collecting-robots

[23] P. Scharre, "Robotics on the battlefield part ii: The coming swarm," 2014. [Online]. Available: https://s3.amazonaws.com/files.cnas.org/documents/CNAS_TheComingSwarm_Scharre.pdf

[24] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 3, no. 1, p. 18, Dec 2018. doi: 10.1186/s41018-018-0045-4. [Online]. Available: https://doi.org/10.1186/s41018-018-0045-4

[25] K. Mizokami, "The pentagon's autonomous swarming drones are the most unsettling thing you'll see today," Jun 2018. [Online]. Available: https://www.popularmechanics.com/military/aviation/a24675/pentagon-autonomous-swarming-drones/

[26] L. Blázovics, T. Lukovszki, and B. Forstner, "Target surrounding solution for swarm robots," in *Information and Communication Technologies*, R. Szabó and A. Vidács, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32808-4 pp. 251–262. [Online]. Available: https://doi.org/10.1007/978-3-642-32808-4_23

[27] B. Khaldi and F. Cherif, "Swarm robots circle formation via a virtual viscoelastic control model," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, 2016. doi: 10.1109/ICMIC.2016.7804207 pp. 725–730.

[28] Q. Chen, S. M. Veres, Y. Wang, and Y. Meng, "Virtual spring-damper mesh-based formation control for spacecraft swarms in potential fields," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 3, pp. 539–546, 2015. doi: 10.2514/1.G000569. [Online]. Available: https://doi.org/10.2514/1.G000569

[29] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," Dec 1998. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1097-4563(199603)13:3⟨127::AID-ROB1⟩3.0.CO;2-U

[30] A. M. Deshpande, R. Kumar, M. Radmanesh, N. Veerabhadrappa, M. Kumar, and A. A. Minai, "Self-organized circle formation around an unknown target by a multi-robot swarm using a local communication strategy," in *2018 Annual American Control Conference (ACC)*, 2018. doi: 10.23919/ACC.2018.8431109 pp. 4409–4413.

[31] M. S. Gzel, E. C. Gezer, V. B. Ajabshir, and E. Bostanc, "An adaptive pattern formation approach for swarm robots," in *2017 4th International Conference on Electrical and Electronic Engineering (ICEEE)*, 2017. doi: 10.1109/ICEEE2.2017.7935818 pp. 194–198.

[32] T. Collett, B. Macdonald, and B. Gerkey, "Player 2.0: Toward a practical robot programming framework," *Proceedings of the 2005 Australasian Conference on Robotics and Automation, ACRA 2005*, pp. 10–17, 08 2008.

[33] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2, pp. 189–208, Dec 2008. doi: 10.1007/s11721-008-0014-4. [Online]. Available: https://doi.org/10.1007/s11721-008-0014-4

[34] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004. doi: 10.1109/IROS.2004.1389727 pp. 2149–2154.

[35] O. Michel, "Cyberbotics ltd. webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004. doi: 10.5772/5618. [Online]. Available: https://doi.org/10.5772/5618

[36] C. Pinciroli, V. Trianni, R. Ogrady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. D. Caro, F. Ducatelle, and et al., "Argos: A modular, multi-

engine simulator for heterogeneous swarm robotics," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011. doi: 10.1109/iros.2011.6094829

[37] L. E. Parker, *Multiple Mobile Robot Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 921–941. ISBN 978-3-540-30301-5. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_41

[38] M. J. Matarić and F. Michaud, *Behavior-Based Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 891–909. ISBN 978-3-540-30301-5. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_39

Appendix A

Addenda

A.1 Ethics Form

ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Sindiso Mkhatshwa
Department		Electrical Engineering
Preferred email address of applicant:		mkhsin035@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc in Electrical and Computer Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Jarryd Son
If this is a researchcontract, indicate the source of funding/sponsorship		N/A
Project Title		Simulating Swarm Behavior for Surrounding a Target

I hereby undertake to carry out my research in such a way that:
- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Sindiso Mkhatshwa	[illegible]	14-08-2020
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Jarryd Son	[illegible]	17-08-2020

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).	A/Prof F Nicolls pp J Buxey	Buxey Dept. Manager: Elec Eng Authorised to sign obo HOD	28.8.2020
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			