# Search and Restore: A Study of Cooperative Multi-robot Systems

Matthew Samuel Haire

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

March 2021

I hereby declare that:

[1] I have not been enrolled for another award of the University, or other academic or professional organisation, whilst undertaking my research degree.

[2] None of the material contained in the thesis has been used in any other submission for an academic award.

[3] I am aware of and understand the University's policy on plagiarism and certify that this thesis is my own work. The use of all published or other sources of material consulted have been properly and fully acknowledged.

[4] The work undertaken towards the thesis has been conducted in accordance with the SHU Principles of Integrity in Research and the SHU Research Ethics Policy.

[5] The word count of the thesis is 47000.

| | |
|---|---|
| Name | *Matthew Samuel Haire* |
| Date | *March 2021* |
| Award | *PhD* |
| Faculty | *The Department of Engineering and Mathematics* |
| Director(s) of Studies | *Dr. Xu Xu* |

# Dedication

To Xu, my trusted mentor and friend, it is not an exaggeration to admit that I could not have done this without you. There were many times during this journey where I believed I would not make it but your dedication, wisdom and kindness continuously inspired me to never give up. I will never forget what you have done for me and I am eternally grateful.

To Kat, my beloved Fiancée, everyday I'm thankful you came into my life. I cannot express in words how much your support and love has helped me over the years. You were there for me even in the darkest of times and without you I don't know if I would have ever achieved this long held dream. Thank you for staying by my side, I love you now and always.

To my mother and father, thank you for believing in me and supporting me through all of this. Whenever I was unsure or troubled I knew you would always be there to help me. You always believed I could do it, even when I didn't believe in myself. It's been a long journey but I finally made it and I hope I did you proud!

Finally, to everyone who helped me along this path including my dedicated second and third supervisors Lyuba and Hongwei, my study partner Alexander, and my best friend David-Jeremy; your support means so much to me and I am without a doubt a better scholar and individual for your advice and friendship. You are all prime examples of how if we work together, anything is possible. Thank you!

## Abstract

Swarm intelligence is the study of natural biological systems with the ability to transform simple local interactions into complex global behaviours. Swarm robotics takes these principles and applies them to multi-robot systems with the aim of achieving the same level of complex behaviour which can result in more robust, scalable and flexible robotic solutions than singular robot systems. This research concerns how cooperative multi-robot systems can be utilised to solve real world challenges and outperform existing techniques.

The majority of this research is focused around an emergency ship hull repair scenario where a ship has taken damage and sea water is flowing into the hull, decreasing the stability of the ship. A bespoke team of simulated robots using novel algorithms enable the robots to perform a coordinated ship hull inspection, allowing the robots to locate the damage faster than a similarly sized uncoordinated team of robots. Following this investigation, a method is presented by which the same team of robots can use self-assembly to form a structure, using their own bodies as material, to cover and repair the hole in the ship hull, halting the ingress of sea water.

The results from a collaborative nature-inspired scenario are also presented in which a swarm of simple robots are tasked with foraging within an initially unexplored bounded arena. Many of the behaviours implemented in swarm robotics are inspired by biological swarms including their goals such as optimal distribution within environments. In this scenario, there are multiple items of varying quality which can be collected from different sources in the area to be returned to a central depot. The aim of this study is to imbue the robot swarm with a behaviour that will allow them to achieve the most optimal foraging strategy similar to those observed in more complex biological systems such as ants. The author's main contribution to this study is the implementation of an obstacle avoidance behaviour which allows the swarm of robots to behave more similarly to systems of higher complexity.

# Table of Contents

# Chapter 1. Introduction

## Section 1.1 Background

Swarm intelligence is the study of natural biological systems with the ability to transform simple local interactions into complex global behaviours. Swarm robotics takes these principles and applies them to multi-robot systems with the aim of achieving the same level of complex behaviour which can result in more robust, scalable and flexible robotic solutions than singular robot systems. The key to overcoming individual shortcomings in multi-agent systems is communication. Isolated individuals only have access to their immediate surrounds which reduces the information available to them and limits their ability to make informed decisions. By communicating with others, these limits are removed and the extent of their knowledge is no longer restricted to their individual reach, but to the reach of the collective. Inter robot communication creates opportunities for individuals to collaborate, enabling them to achieve tasks they would be incapable of performing alone, and increasing the speed at which achievable tasks can be completed. This research demonstrates these principles by showing how cooperative multi-robot systems can be utilised to solve real world challenges and outperform existing techniques.

## Section 1.2 Importance of the research

Researchers have begun to recognise the power of swarm intelligence and the solutions it could provide if appropriately applied to multi-robot systems from the early 2000s to the present day. Solutions which rely on a single highly complex robot may be capable of performing a variety of tasks simple robots would be unable to achieve themselves, however this comes with a number of drawbacks, prime among these being poor scalability and the infamous single point of failure (SPOF). When aspects of a problem grow (such as the size of an environment to be monitored or explored) it becomes more difficult for a single robot system to scale its solution, reducing efficiency and increasing the time taken to complete tasks. Should a single robot break or malfunction the entire system must come to a halt until repairs have been completed.

Cooperative multi-robot systems offer a viable alternative to the conventional single-robot solution which can overcome both of these issues. Multi-robot systems experience less reduction in performance when scaling their approach to a growing problem thanks to the ability to easily add more robots and expand the system's reach. If a single robot breaks or malfunctions in a multi-robot system the team can still continue to function, albeit at a reduced efficiency, removing the SPOF associated with single robot solutions. When designed correctly, multi-robot systems are even capable of changing their collective approach to match changing problems such as navigating dynamic environments.

It stands to show that cooperative multi-robot systems may offer many advantages compared to more complex individual robots, but this is highly dependent on the behaviours built into each robot. Robots which are incapable of communicating or coordinating with other robots do not offer the benefits of multi-robot systems as listed above. As such, the subject of how to design individual robot behaviours which result in desired complex global behaviours is of paramount importance to the field of swarm robotics research. This thesis presents three novel demonstrations of how individual robot behaviour and communication are leveraged to create complex global behaviours applied to an entirely new approach to emergency ship hull repair, and nature inspired foraging scenarios. This research on multi-robot systems performing emergency ship hull repair is a first of its kind study and is the most significant contribution to knowledge. All of the studies serve as new examples of how cooperative multi-robot systems may be applied to address real-world problems and showcase the possibilities of swarm robotics.

## Section 1.3 Motivation

The motivation behind pursuing this research came from the author's interest applying robots to efficiently solve real world problems that are deemed hazardous to human life. During the author's time in the British Royal Fleet Auxiliary, he was instructed on conventional methods of emergency ship hull repair and the importance of regaining ship stability quickly. The task of emergency ship hull repair is entrusted to human crew members, but being a dangerous and time constrained procedure it increases the risk of injury to the crew. It was during the author's research into swarm robotics when a method of delegating the emergency ship hull repair procedure to a multi-robot

system was first formed. Using robots to autonomously repair ship hull damage would reduce the risk to human life by removing them from the situation and allowing them to focus on other tasks. Studying at Sheffield Hallam University allowed the author to learn from some of the most respected swarm robotics researchers while undertaking his research. This exchange of knowledge and tutelage increased the author's knowledge of cooperative multi-robot systems such that he began consider how this technology could be applied to solve other issues beyond ship hull repair. This interest encouraged the author to participate in a joint research project with another team of swarm robot researchers at the University of Sheffield and the collaboration resulted in a comprehensive study on designing optimal foraging behaviours in multi-robot systems, which served to expand the author's knowledge of open issues in swarm robotics, but also how current swarm robots could be applied to solve other real-world problems.

## Section 1.4 Research challenges

There are several open issues in the field of swarm robotics yet to be full addressed which had to be taken into consideration when proceeding with the studies. One of the more significant issues to address is the lack of a general design pattern for swarm robot systems – how to achieve any desired global behaviour from the design of individual robot behaviours and vice versa. While some progress has been made towards realising formal design patterns for some specific global behaviour, a general design pattern has not yet been formulated. Furthermore, many existing design patterns are highly dependent on the robots physical morphology which makes them difficult to implement on multi-robot systems using different robots from the example. Without a general design pattern, the author had to utilise the latest body of research when designing the individual robot behaviours to create the desired global behaviours for the studies.

This research presented a number of additional challenges for the author to overcome. The ship hull repair scenario made use of bespoke simulated autonomous underwater robots, whose design was based on existing technologies which had been demonstrated in other underwater robots. There is a variety of open-source and proprietary robot simulators available for carrying out experiments but few of these have been optimised for swarm robotics research. The simulators which are better

designed to run multi-robot simulations also vary in their ability to model different environments such as air, ground, and water. Of the fraction of simulators suitable for multi-robot systems, only a very small portion of these can simulate fluids appropriately. This presented a challenge to the researchers in choosing a suitable simulator to carry out the ship hull repair experiments while minimising the reality gap.

## Section 1.5 Outline of Thesis

The research is separated into six chapters. Chapter 1 opens with a brief introduction to the subject of artificial swarm intelligence, the main subjects explored within this thesis, and the significance of the research. This is followed by an explanation of the author's interest in pursuing cooperative multi-robot systems research and a discussion of how the author addresses the more prominent open issues and challenges in swarm robotics through these studies. The Chapter concludes with this outline of how the paper will proceed, providing a summary of the contents contained in the main body of the thesis.

Chapter 2 provides a comprehensive review of the historical developments of swarm robotics and multi-robot systems from their inception to the present day. Included are key publications which established the theories and methodologies found to be most relevant to the studies performed in this thesis such as: the main principles of swarm robotics, historical developments, behaviour-based robotics, multi-robot modelling, pattern-formation, coordinated motion, localisation and mapping, multi-robot exploration, aggregation, modularity, self-assembly, and foraging behaviours. The methodologies present in each subsequent chapter are linked to this literature review, to show how the approach was informed by established swarm robot methods and theories. The final section identifies gaps in existing research and indicates how the studies in this thesis contribute to bridging these gaps.

Chapter 3 is an extension of the published work by Haire, et al. (2019a) and presents the emergency ship hull repair scenario and proposes solutions which use a group of cooperative autonomous underwater robots to perform inspection. The methodology for the ship hull inspection is explained in-depth and is followed by a presentation, analysis and discussion of the results from the experiments. The morphology of the

individual robots and their design is discussed along with the simulated environment. The chapter concludes with a detailed discussion of the results and their implications on future experiments concerning emergency ship hull repair and complete area coverage (CAC) algorithms applied to swarm robot systems.

Chapter 4 is an extension of the published work by Haire, et al. (2019b) and presents the next stage of the emergency ship hull repair process, providing an in-depth explanation of the methodology used for the swarm robots performing self-assembly. Differences in robot morphology from those used in Chapter 5 are identified here along with the simulated environment used to carry out experiments. The chapter proceeds to discuss the experimental setup, presenting the results of the experiments, and concludes with a discussion of the implications of the findings on future experiments concerning emergency ship hull repair and self-assembly algorithms applied to swarms of homogeneous modular robots.

Chapter 5 is an extension of the published work by Talamali, et al. (2020) to which the author of this thesis contributed. The chapter discusses how nature-inspired swarm robot systems can be applied to solve foraging scenarios and obstacle-avoidance tasks, and then delves into the methodologies used in the experiments. The study examines swarm-size dependant foraging strategies, how these influence the performance of a swarm of robots, and how the author's implementation of obstacle avoidance benefited this collaborative study. The chapter concludes with a discussion of the implications of these studies, and how they impact the field of swarm robots and will influence future studies of the subject.

Chapter 6 is the final chapter which provides a succinct conclusion for each of the studies presented in the thesis. Each of the studies provides a contribution to the existing knowledge of swarm robotics research and these are identified here. The chapter ends by proposing a collection of recommended future studies that could further advance the field of swarm robotics with respect to the studies presented within this thesis. All references are provided in the section following this along with appendices containing relevant code, supplemental figures, tables and graphs.

## Section 1.6 Main Contributions

The research presented within this thesis contains three novel contributions to the field of cooperative multi-robot systems. Chapter 3 presents an application that utilises theories of cooperative multi-robot exploration and communication to create a complete area coverage search method for a swarm of robots tasked with inspecting a damaged ship hull. The cooperative search algorithm was proven in simulation to be more effective at achieving complete area coverage in less time than the same multi-robot system using an uncoordinated search algorithm. Additionally, the chapter presents a simulated robot sensor arrangement that would allow robots to maintain a set distance from a 3D object, allowing them to treat their environment more akin to a 2D plane, which allows for simpler implementations of the search algorithm.

Chapter 4 expands on the scenario presented in Chapter 3 with respect to autonomous ship hull repair using a swarm of robots. The main contribution of this research is a method of self-assembly that would allow modular robots to form a repair patch capable of coving a hole in a ship hull. In addition, the results from the experiments informed an improved self-assembly approach which suggests a method of enhancing the initial approach by controlling the angle of approach the robots use when navigating their way to the damage, or by allowing more than one assembly location for the repair patch.

The main contribution of chapter 5 is the implementation of obstacle avoidance behaviour with low computational overhead on a large swarm of robots tasked with collective foraging in environments. The swarm of robots are able to tune their responses to their environment to create the best distribution of agents balancing quality of items to collect against the distance required to retrieve them. The obstacle avoidance behaviour solved a major issue of physical robots becoming stuck against the walls of their bounded arena and other robots, which improved the performance of the swarm, and created a system more capable of emulating the collective foraging behaviours observed in biological swarm such as ants.

# Chapter 2. Literature Review

## Section 2.1 Main Principles

Swarm intelligence is considered the study of natural biological systems with the ability to transform simple local interactions into complex global behaviours, such as bees working together to build nests, ants exploring environments and foraging for food, or the pattern formation in schools of fish evading predators (Bonabeau, Dorigo, and Theraulaz, 1999; Camazine, Deneubourg, Franks, Sneyd, Theraulaz, and Bonabeau, 2003). Swarm robotics takes these same principles and applies them to multi-robot systems with the aim of achieving the same level of complex global behaviour from simple local robotic interactions, which can result in more robust, scalable and flexible robotic solutions (Beni, 2005; Şahin, 2005). The first definition of the term swarm robotics, which is still regarded as the most complete description of the discipline (Barca and Sekercioglu, 2013; Brambilla, Ferrante, Birattari, and Dorigo, 2013; Navarro and Matía, 2013; Bayindir, 2016; Nedjah and Junior, 2019), was proposed by Şahin (2005) in his seminal paper 'Swarm Robotics: From Sources of Inspiration to Domains of Application':

> "Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment."

Following this definition, Şahin (2005) identified the main principles of swarm robot systems with a focus on three desired properties: robustness, flexibility, and scalability. Robustness is the ability of a system to continue to function, albeit at lower performance, when a portion of the system fails or in the presence of disturbances in the environment. Scalability is the ability of a system to increase and decrease the number of individuals of the group and continue to function using only the same local interaction rules. Flexibility is the ability of a system to adapt and address changing demands such that the system can reconfigure its group members or approach to address various tasks.

These are the main aspects of a swarm robot system, but in order to further distinguish them from other closely related subjects and more general multi-robot

systems, Şahin (2005) identified additional sets of criteria: The individuals that make up the swarm should be autonomous – they should possess physical embodiment with the ability to interact with the environment. Studies of social insects (Camazine et al., 2003) showed that natural systems are able to achieve robust, flexible and scalable behaviours without the need for a centralised control; these same attributes are desired for swarm robot systems and so designers should make effort to ensure their systems are also decentralised. The abilities of the system should involve coordination of large numbers of robots, or at least smaller numbers of individuals with the ability to scale to higher population sizes without the need to change the simple local rules of interaction. The individuals that make up the group should be homogeneous with no variance between robots – heterogeneous groups of robots with predefined roles and different rules of interaction are less scalable and robust than homogeneous groups and as such rarely meet the criteria to qualify as a swarm system. The individual robots should be relatively simple compared to the task at hand, such that an individual would be incapable of carrying out the task by itself, or completes the task much less efficiently than a group of individuals would. Finally, the robots used to make up these systems should only need to utilise limited sensing and local communication for the swarm to achieve its desired behaviour. This form or distributed coordination removes the need for global communication methods that would likely hinder the scalability of swarm.

These definitions were initially intended as a means to determine to what degree the term swarm robotics might apply to a given multi-robot system, but had since evolved to serve as the corner stones for defining most swarm robot systems in use today. Most importantly, true swarm robot systems today are described as multi-robot systems which are capable of generating complex global behaviours from simple local interaction; these are systems which are capable of performing more than the mere sum of its individual parts. In the following section, the origin of the research of swarm robotics is discussed with insights into how the subject was originally formed from fusions of the studies of multi-robot systems and collective intelligence observed in nature.

## Section 2.2 Historical Developments of Swarm Robotics

Swarm robotics is a relatively new area of research, but its founding extends back to the early 1980s when researchers were using cellular automata to model and replicate the patterns and behaviours observed in nature (Wolfram, 1983). Wolfram's studies of self-organisation and the ability of cellular automata to produce complex patterns from simple rules sparked the imagination of scientist giving new momentum to the study of how such natural complex behaviours may be replicated in artificial systems.

In the late 1980s concepts and studies of multi-robot systems with the ability to self-organise began to emerge and a new term to describe them; cellular robots (Fukuda and Nakagawa, 1988; Beni, 1988). This term was intended to indicate how these groups of simple robots could behave like the cells of an organism, assembling to form more complex structures. The term swarm intelligence began being used by Beni (1988), Beni and Wang (1989; 1991), and Hackwood and Beni (1991; 1992) to describe the ability of these cellular robot systems to generate patterns and complexity through simple local interactions. However, research into biological systems displaying collective intelligence such as insect colonies by Pratte, Gervet, and Theraulaz (1990) was also being conducted at the same time, and the crossover between the disciplines quickly became apparent. These biologists found that the concepts of swarm intelligence could be used to describe the behaviours they had been observing in nature. After all, the systems they were describing were also decentralised, homogeneous and made up of large groups of relatively simple individuals, but capable of displaying complex behaviours.

Biologists and roboticists alike began utilising the concepts of swarm intelligence in their research to find new ways of understanding how natural systems functioned and how these discoveries could be applied to artificial systems to generate complex behaviours (Kube and Zhang, 1993; Balch and Arkin, 1994; Dorigo, Maniezzo, and Colorni, 1996; Bonabeau, Theraulaz, Deneubourg, Aron, and Camazine, 1997; Balch and Arkin, 1998; Arkin, 1998). It soon became apparent that the term swarm intelligence could be used to describe both the behaviours of natural and artificial systems, and by the late 1990s the definition was extended to include attempts to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies (Bonabeau et al., 1999).

In the early 2000s it had been revealed that social insects indeed functioned without centralized coordination and yet their interactions and behaviours formed a natural system that was robust, flexible and scalable (Camazine et al., 2003) – properties considered desirable for distributed multi-robot systems. This helped solidify the idea that artificial systems with these properties could be developed if behaviours from natural systems could be replicated, which boosted research into reproducing the behaviours observed in ants, bees, fish, and birds in multi-robot systems. By this point in time, there were a variety of terms being used to describe these kinds of multi-robot systems such as the earlier mentioned cellular robotics, robot colonies, distributed robotics, and collective robotics (Kube et al., 1993; Arkin and Bekey, 1997; Martinoli, 1999). With no universal terminology yet in place Sahin (2005) sought to establish the term swarm robotics as the title of this disciple, distinguishing the subject from general multi-robot systems. He provided the first definition and listed the three main principles of robustness, flexibility, and scalability – which are still recognised as the defining points of swarm robot systems today (Nedjah et al., 2019).

There were a variety of suggested applications of swarm intelligence prior to Sahin's definition of swarm robotics, but by the mid-2000s some of the more sought-after domains of application had become clearer. These domains included: tasks that covered a region, such as space exploration (Burgard, Moors, Stachniss, and Schneider, 2005), environmental monitoring (Dhariwal, Sukhatme, and Requicha, 2004), surveillance (Solomon, 2004), or hazard detection (Zarzhitsky, Spears, and Spears, 2005); tasks considered too dangerous for humans, such as robot mine detection (Kumar and Sahin, 2003); tasks that may scale up or down in time, such as containment of oil spills (Kakalis and Ventikos, 2008); and tasks where redundancy is a benefit, such as forming dynamic communication networks in disaster scenarios (Witkowski, El-Habbal, Herbrechtsmeier, Tanoto, Penders, Alboul, and Gazi, 2008).

The subject area of swarm robotics only continued to grow with researchers tackling a plethora of problems with the aim of one day realising many of the suggested applications of this new technology. Along with the advances came a number of taxonomies on the subject of swarm robotics each identifying the most prominent problems being tackled by researchers and categorising them into various subject areas. Of the variety of suggested classifications of the subject, Brambilla et al. and

Bayindir's taxonomies are currently the most accepted in the literature (Nedjah et al., 2019). Brambilla divided the works of swarm robotics into the two classes of methods and collective behaviours as shown in Table 2.1. While Bayindir divided the subject into the five main axis of modelling, behaviour, design, communication, analytical studies, and problems shown in Table 2.2.

*Table 2.1 Brambilla et al.'s (2013) taxonomy of swarm robotics research*

| | | |
|---|---|---|
| Methods | Design methods | Behaviour-based design methods |
| | | Automatic design methods |
| | Analysis methods | Microscopic models |
| | | Macroscopic models |
| | | Real-robot analysis |
| Collective behaviours | Spatially-organising behaviours | Aggregation |
| | | Pattern formation |
| | | Chain formation |
| | | Self-assembly and morphogenesis |
| | | Object clustering and assembling |
| | Navigation behaviours | Collective exploration |
| | | Coordinated motion |
| | | Collective transport |
| | Collective decision-making | Consensus achievement |
| | | Task allocation |
| | Other collective behaviours | |

*Table 2.2 Bayindir's (2016) taxonomy of swarm robotics research.*

| | | |
|---|---|---|
| Modelling | Sensor-based | |
| | Microscopic | |
| | Macroscopic | |
| | Cellular Automata | |
| Behaviour design | Nonadaptive | |
| | Learning | Reinforcment Learning |
| | Evolution | |
| Communication | Interaction via Sensing | |
| | Interaction via Communication | |
| Analytical Studies | | |
| Problems | Pattern Formation | |
| | Aggregation | |
| | Chain Formation | |
| | Self-assembly | |
| | Coordinated Movement | |
| | Hole Avoidance | |
| | Foraging | |
| | Self-Deployment | |

Both Brambilla et al. and Bayindir's taxonomies can be used to identify which subjects a study belongs to and help identify how it may relate to other research categories. For instance, in this thesis the following subjects could be categorised according to Brambilla as follows: Complete area coverage algorithms using a swarm of robots in Chapter 4 can be categorised as collective behaviours, navigation behaviours or spatially-organizing behaviours, collective exploration, coordinated motion, and chain formation. Self-assembly using a swarm of robots in Chapter 5 can be categorised as collective behaviours, spatially-organizing behaviours, aggregation, self-assembly and morphogenesis, collective decision-making, and consensus achievement. Foraging with obstacle avoidance in Chapter 6 can be categorised under methods, analysis methods, and real robot analysis. It could also be categorised under collective behaviours,

collective decision making, task allocation, or navigation behaviours and collective exploration.

The following sections and studies within were mainly selected according to the catagories outlined in Brambilla and Bayindri's taxonomies, but only lists those that are most applicable to the subsequent research presented in Chapters 3, 4, and 5 of this thesis. They include discussions of the most prominent studies that have contributed to the advancment of swarm robotics, the methods and approaches that have emerged, and identifies the papers that have had a significat influence on this thesis and helped inform the design process for each robot system.

## Section 2.3 Behaviour Based Robotics

The most widely used approach to designing robots with artificial intelligence (AI), prior to the mid-1980s, used what became known as the symbolic system, where robots used symbols to represent the world around them and perform mathematical functions to solve various scenarios (Feldman, and Sproull, 1977). This approach to AI saw many successes in solving problems encountered by robots, but as the scenarios to solve became more complex, the computation needed to obtain solutions became increasingly expensive. To solve this dilemma, a new approach to achieving robotic solutions was proposed: behaviour-based robotics (Brooks, 1986; 1990). The symbolic system approach to AI relied heavily on high-level cognitive processes such as representation and reasoning to achieve desired robot behaviours, but in the behaviour-based approach the perceptions of the robots were directly coupled with actions resulting in solutions that were much less computationally expensive - the key to this is in how the task to be performed is decomposed into subtasks. In the symbolic system approach the control system of the robot is divided into separate modules to find solutions via a process of functional decomposition, where the problem is split into series of sequential processes such as perception, modelling, planning, and execution as shown in Fig.2.3.1. Conversely, behaviour-based robot control systems develop solutions using behavioural decomposition, where the solution is represented as separate independent processes running simultaneously following the subsumption architecture as shown in Fig.2.3.2.

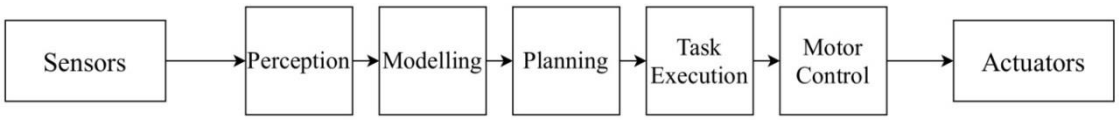



*Fig.2.3.1 Functional decomposition of a desired robot task, adapted from Brooks (1986).*

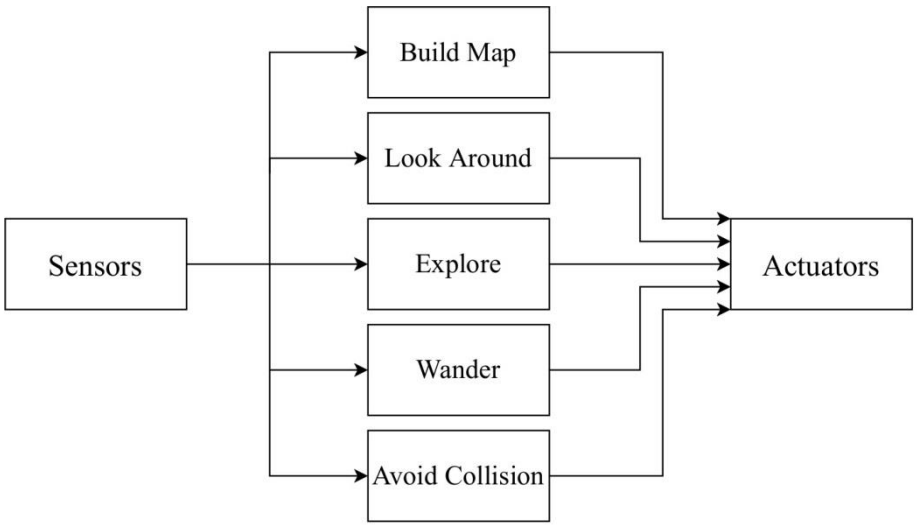



*Fig.2.3.2 Behavioural decomposition of a desired robot task, adapted from Brooks (1986).*

Contrary to the sequential function blocks of the symbolic system, behaviour-based architectures are typically represented as stacks of parallel concurrent behaviours. One of the first methods of dictating how the layers interact is known as subsumption architecture (Brooks, 1986), called such due to the way it subsumes lower levels of behaviour. In subsumption architecture the bottom layers deal with the most crucial behaviours to the robot's survival such as obstacle avoidance and the top levels control more complex processes such as object recognition, localization or mapping. These systems are designed with a bottom-up approach, starting with the simplest most essential behaviours and only adding higher behaviour once the lower-level behaviours have been tested, refined, and proven functional and robust. Although higher level behaviours can rely on the functioning of lower-level behaviours, they do not explicitly use the lower levels as subroutines, only as a set of existing competences. Subsumption is one of the better documented methods of coordinating the different levels but there are alternative methods of showing how the different levels of

behaviours correspond to one another which are compared in Arkin's (1998) review of behaviour-based robotics.

When subsumption architecture was first proposed, Arkin (1989) authored an alternative approach to behaviour-based robotics using the concept of motor schema. Motor schema theory is a method that is able to describe the behaviour of agents using a higher level of abstraction and representing them as modules. There are a number of definitions of schema which depend on the area of application but for the purpose of encoding robotic behaviours, Arkin (1998) defined schema as follows:

> "A schema is the basic unit of behaviour from which complex actions can be constructed; it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted."

Much like other behaviour-based methods, motor schema demonstrated advantages over the symbolic system approach to the design of control systems for autonomous robots. In motor schema the modules that represent different behaviours execute concurrently and all of the responses formed by the modules are represented as vectors using potential fields. Unlike subsumption architecture, coordination between the modules is achieved using vector addition and there is no pre-defined hierarchy for this cooperation. However, the biggest distinction between the approaches is the inclusion of a second layer between the schema and the output of the motors, where the information generated in each schema is fused to form a single resultant action. This is best illustrated by Arkin (1989) in Fig.2.3.3 where he applied motor schema theory to solve robot navigation with his perception-action schema. This method results in extremely fast computation since only a single vector is required to be computed at the robot's current location.

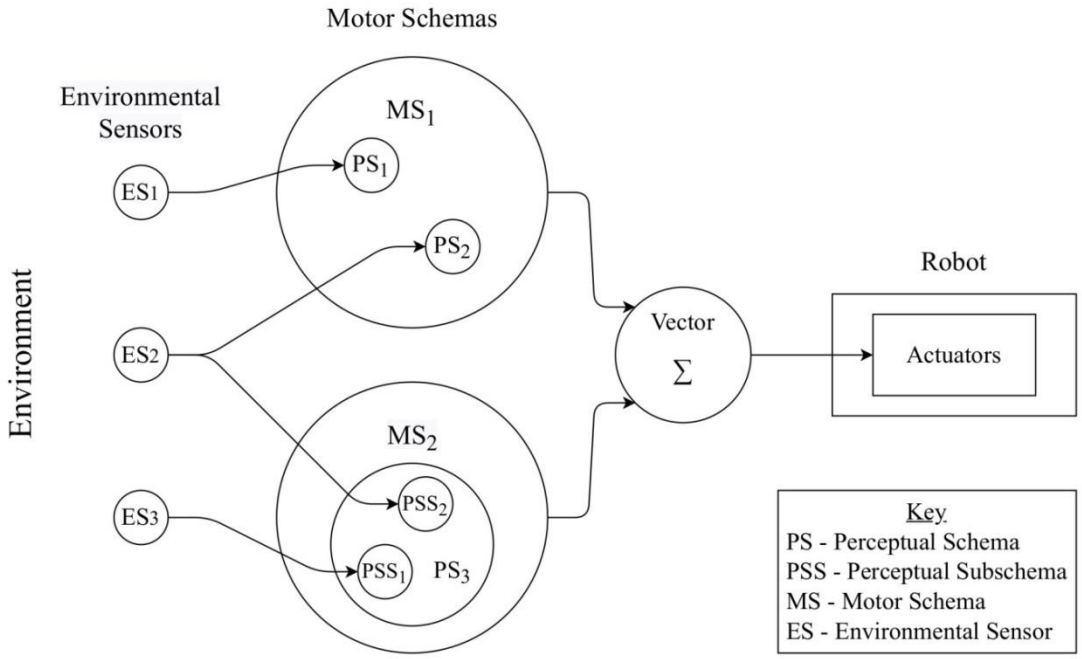



*Fig.2.3.3 Motor schema theory applied to robot navigation; perception-action schema relationship, adapted from Arkin (1998).*

Floreano and Mattiussi (2008) provided a good summary of the main benefits and drawbacks of the symbolic system design and behaviour-based design. The symbolic approach to design excels at producing robotic systems that are precise, controllable, and predictable – qualities well suited to domain applications such as surgical robotics or assembly line machines. The main drawback of this approach is its failure to cope well with noise and uncertainty, which are commonly encountered in autonomous robots. Furthermore, each function is dependent on the preceding stage of the process, which is less robust as failures at earlier stages can greatly impact the functioning of the system as a whole. It is also a computationally expensive process due to the systems needs to build models and produce plans at the same time in order to function.

The main advantages of behaviour-based robotics over the symbolic approach, such as faster reactions, stem from the systems method of directly connecting sensory information onto motor actions. It is a more robust design since processes run in parallel and can operate independent of one another. This means that if one of the behaviours fails the remaining behaviours can continue to function, although they may see a minor impact on performance dependant on the task. It can also handle multiple

goals which can be dealt with by individual behaviours at different levels, with the need for significantly higher computational power.

Both subsumption and motor schema approaches have been proven to be appropriate methods of creating robot control systems using behaviour-based design, breaking away from the mainstream method of using representational knowledge and instead emphasise the use of behavioural decomposition, and tight coupling between sensors and actuators. The robots presented in Chapters 4 and 5 were designed using the bottom-up approach of behaviour-based design, with motor schema approach selected as the base architecture due to its focus on non-layered cooperative interaction between the separate behaviours.

For instance, the simulated ship hull inspection robot described Chapter 4 performs a number of separate behaviours that execute in parallel to allow the robot to navigate the ship hull. The data obtained from its forward facing distance sensors is passed through a controller which links directly to a corresponding propeller, which allows the robot to maintain a set distance from the ship hull at all times. While this process is being executed, the robot uses additional proximity sensors to detect the presence of other robots or obstacles and adjust its position accordingly. These two behaviours execute simultaneously, demonstrating a method of generating formation control.

## Section 2.4 Multi-Robot Models

When designing cooperative multi-robot systems, mathematical models of the swarm are essential to evaluate several aspects, such as the feasibility of the task to be carried out, the minimum number of robots necessary to achieve the desired behaviour, and the effect of any disturbance to the system. There are two main methods of describing system behaviours and in swarm robotics that fall under the categories of microscopic and macroscopic studies. Microscopic models use a bottom-up design and focus on the individual behaviour and interaction between members of the swarm, while the macroscopic approach is more of a top-down design concerned with the function of the swarm as a whole (Brambilla et al., 2013). Microscopic models of swarm robots are typically described at different levels of abstraction from simple points representing robots on a 2D plane, to full 3D simulations where environmental forces, sensors, and actuators are modelled. These different levels of abstraction come

with inherent reality gaps, such that when the behaviour is implemented on a real robot system the results may not align with the simulations. This is an important factor to consider when selecting an appropriate simulator to represent swarm robot models and is discussed further in Section 3.4.

Macroscopic models typically use mathematical formula to describe collective behaviours and one of the most popular categories of these are the rate and differential equations. In swarm robotics, rate equations can be used to describe the different proportions of robots exhibiting a set number of states which are derived from probabilistic finite state machines (PFSM). PFSM consist of different states with descriptions of how an agent transitions between them. The transitions that govern the shift between states can be determined by more than just the previous event, such as specific interactions with external processes. PFSM are a form of non-deterministic finite state automata where the probability of a given transitions between states is also provided. Eq. (2.1) shows an implementation of a PFSA applied to a swarm of robots tasked with collaboratively collecting sticks while avoiding the wall of their arena (Ijspeert, Martinoli, Billard, and Gambardella, 2001).

$$\mathrm{P_W} = \frac{\mathrm{A_W}}{\mathrm{A_A}} \quad \mathrm{P_R} = \frac{\mathrm{N_R} \cdot \mathrm{A_R}}{\mathrm{A_A}} \quad \mathrm{P_{G1}}(\mathrm{t}) = \frac{\mathrm{N_{G1}}(\mathrm{t}) \cdot \mathrm{A_S}}{\mathrm{A_A}} \quad \mathrm{P_{G2}}(\mathrm{t}) = \frac{\mathrm{N_{G2}}(\mathrm{t}) \cdot \mathrm{R_{G2}}(\mathrm{t}) \cdot \mathrm{A_S}}{\mathrm{A_A}}$$

$$\mathrm{P_N}(\mathrm{t}) = 1 - \left(\mathrm{P_W} + \mathrm{P_R} + \mathrm{P_{G1}}(\mathrm{t}) + \mathrm{P_{G2}}(\mathrm{t})\right) \tag{2.1}$$

where $\mathrm{P_N}$ represents the number of probabilities at each iteration, $\mathrm{P_W}$ is the probability of encountering a wall, $\mathrm{P_R}$ for encountering a robot, $\mathrm{P_S}$ for finding a stick, $\mathrm{P_{G1}}$ and $\mathrm{P_{G2}}$ for holding a stick and another robot respectively. $\mathrm{A_W}$ is the surrounding wall of the arena, $\mathrm{A_A}$ is the entire arena, and $\mathrm{A_R}$ is a single robot.

In rate equations, the states from the PFSM are represented as variables with an equation assigned to each of them much like those of Eq. (2.1). These variables can be used to track the number of robots in a given state as time evolves and show how many transitions between states occur within a given time frame and under which conditions. Indeed, rate equations has been proven effective at modelling swarm robot systems in foraging scenarios in the presence of interference (Lerman and Galstyan, 2002), when foraging from multiple sources (Campo and Dorigo, 2007), and when collecting energy units (Liu and Winfield, 2010). The experimental challenges of

microscopic and macroscopic design were investigated by Mermoud, Upadhyay, Evans, and Martinoli (2014), where they compared the two design methods when used to solve a given scenario. Their results indicated that for both models, top-down approaches were less effective than the bottom-up approaches for designing distributed controllers, but concluded that a model-based control design methodology that incorporated the aspects of both top-down and bottom-up approaches would be the most effective.

Both the bottom-up microscopic and top-down macroscopic approaches have seen success in designing swarm systems capable of carrying out simple tasks, and a combination of the design methods may be more beneficial than focusing on a single approach. However, determining exactly which local interactions between agents at the microscopic level leads to a desired global behaviour at the macroscopic level and vice-versa is a difficult task. Some promising work towards achieving a quantitative link between these macroscopic and microscopic behaviours was conducted by Reina, Miletitch, Dorigo, and Trianni (2015) where they identified quantitative links between the dynamics of the microscopic implementation of a robot swarm tasked with shortest-path discovery, and the dynamics of a macroscopic model of a foraging task based on best-of-n site selection in honeybees.

Their study used central-place foraging strategy in an environment consisting of a bounded space, a single central nest, and two resources sites at different distance from the nest. The microscopic behaviour of each robot was implemented as a probabilistic finite state machine (PFSM). The states indicated which resource site the agent was committed to, if it were uncommitted, and whether it was in an interactive or latent state – indicating if the state of a neighbour would affect its own commitment state – with probabilities dictating each transition. Their microscopic implementation was evaluated in simulation, the results of which were compared to their macroscopic model by investigating the decision-making dynamics for varying probabilities and for sets of different decision problems by varying the distance of the resource sites. Their results revealed that the final distribution of agents according to their macroscopic model and the multi-agent simulation were in agreement, confirming the existence of a quantitative micro-macro link. This work represented a significant step toward achieving a formal design pattern which was later refined to

address the spatial and topological factors that impact the micro-macro link (Reina, Valentini, Fernández-Oto, Dorigo, and Trianni, 2015), though more research is required before a general design pattern for swarm robots can be established.

As observed, many existing swarm robot systems are modelled through either microscopic or macroscopic lenses and see implementation and validation in simulation or real robots. However, Kazadi et. al (2007; 2009) found another way to describe the properties and performance of a swarm using mathematical language to prove their validity. Their model-independent approach used a combination of bottom-up and top-down design to describe a desired global behaviour in terms of tangible quantities and measurements. In their 2007 study, they used their method to form a hexagonal pattern using a swarm whose movement was dictated by the summed forces of individuals using artificial physics. This was extended in their 2009 study to propose that desired behaviours can be more readily achieved when a swarm system can be described in terms of measurements within the environment. The methods by which individual robot obtain these measurements are left open to the interpretation of designers which they argue allows for implementation of the behaviours across different swarms with varying morphologies.

There exist alternative methods to modelling swarm behaviour, but methods and equations provided in this Section were highlighted as they are most applicable to swarm robot studies discussed in Chapters 4, 5 and 6 of this thesis. For instance, the robots presented in the ship hull inspection and foraging scenarios were each designed using either non-deterministic finite state machines or PFSM to describe their microscopic behaviour with the aim of achieving a desired macroscopic behaviour. The yield variable R of chapter 5 is an example of how the optimal performance of a swarm can be quantified using measurements directly obtained from the environment.

## Section 2.5 Exploration

Swarm robot systems are inherently mobile and many applications require agents to move within a given environment in order to accomplish their tasks. This raises the question of how these agents should move in the environment and if there is an optimal method of exploration that can be used for tasks such as searching an area, building maps, or monitoring changes in an environment. Exploration of unknown

environments has been a subject of immense interest to researchers over the years for both singular and multi-robot systems since solutions can be used to solve many real-world problems of navigation in autonomous systems. These studies mainly focused on robots gathering information about their surroundings to better inform their decisions of how to best reach a specified goal location. *C*onsequently, many algorithms and methods have been developed to solve issues concerning optimal exploration techniques such as dispersion, coverage, pattern formation, path planning, flocking, localization, and mapping. This section discusses the various methods employed in exploration which we found most applicable to the studies of ship hull inspection and ship hull repair using multi-robot systems presented in Chapters 4 and 5 respectively.

### Section 2.5.1 Dispersion and Pattern Formation

Dispersion is a method used by a group of robots to distribute themselves in a given environment, without falling out of communication range in order to maximise their coverage of an area. In order to increase the robustness of the technique, dispersion is typically designed as a process that is not centrality planned. Effective dispersion techniques should result in a network of distributed robots which have maximised the area they can monitor while remaining able to communicate with their nearest neighbours. Dispersion is a useful tool for scenarios where a swarm of robots is tasked with monitoring environments for hazards (Zhang, Fricke, and Garg, 2011), mapping of unknown environments (Wang, Liang, and Guan, 2011), or searching for objects or landmarks in unknown environments (Liu and Nejat, 2013). Pattern formation can be interpreted as a variant of the dispersion task, where robots tasked with occupying a space display a repeatable pattern. Swarms that incorporate pattern formations in their dispersion technique often result in systems more robust to the failure of units or sensor errors (Turgut, Çelikkanat, Gökçe, and Şahin, 2008), which increases the ability to recovery from gaps formed in the swarm and minimises the risk of leaving spaces uninspected. There are many proposed approaches to solving area coverage using dispersion and pattern formation and the following section discusses a selection of notable studies.

When robots are tasked with dispersion and pattern formation, they require a way of ensuring they maintain a specific distance from their closest neighbours and do not fall

out of communication range, and there exist a number of ways this has been achieved in literature such as inter-robot communication (Batalin and Sukhatme, 2002; McLurkin and Smith, 2004; Falconi, Sabattini, Secchi, Fantuzzi, and Melchiorri, 2015). Approaches that utilise inter-robot communication rely on either the direct exchange of information between agents, or the ability of the robot to react to the presence of other robots that fall within their sensor range. Another approach to modulating the distance between robots in a group is to instead use the intensity of received wireless signals from neighbouring robots to determine how far these agents are from each other – a method referred to as distance estimation using wireless signal strength (Ludwig and Gini, 2006; Ugur, Turgut, and Sahin, 2007). An alternative method to achieving dispersion uses virtual forces (Spears, Spears, Hamann, and Heil, 2004; Sallam and Baroudi, 2015). The virtual forces approach takes inspiration from models in physics, assigning forces to each robot and using the resultant vectors to determine the directions agents should travel relative to their neighbour's trajectories.

One of the most debated methods of dispersion is the use of artificial potential fields (Reif and Wang, 1999; Balch and Hybinette, 2000; Howard, Matarić, and Sukhatme, 2002; Poduri and Sukhatme, 2004; Mikkelsen, Jespersen, and Ngo, 2013). Artificial potential fields assign attractive and repulsive forces to all of the robots, obstacles, and goals within an environment and use the resultant forces to achieve optimal dispersion and path planning. While effective at allowing robots to navigate known environments, artificial potential fields has been criticized as being ill suited to real-world environments as it often times relies on environment features such as obstacles and goal location to be know prior to execution. Each of these approaches discussed has proved successful in achieving dispersion and pattern formation for the purpose of area coverage which is used to inform the approach to complete area coverage in chapter 3. They also provide valuable insights into optimal dispersion theory which is discussed below.

Batalin et al. (2002) proposed two methods of dispersion for a swarm of autonomous robots in order to maximise their sensor coverage; their informative approach where robots explicitly communicating with other agents to determine where they should move, and their molecular approach which communicated implicitly, following boundary conditions with the ability to distinguish between robots and obstacles.

These were compared against a basic approach which used obstacle avoidance only. Their results showed that control strategies that allow agents to communicate with each other outperform simple obstacle avoidance techniques when performing dispersion, and the approach which did not explicitly communicate with neighbours but could distinguish between robots and other obstacles converged to optimal distributions the fastest. The theory that control algorithms which allow robots to distinguish other robots from other obstacles can outperform algorithms that neglect this distinction was reinforced in a study by Morlok and Gini (2007) which proved that not only does knowledge of the locations of the other robots help to speed up the exploration process, but that cooperative exploration can outperform random walks and simple wall following behaviours in maximising area coverage of enclosed spaces. These studies are prime example supporting the supposition that a coordinated swarm of robots could perform a complete area coverage search more efficiently than an uncoordinated swarm of robots.

McLurkin et al. (2004) also conducted experiments with swarms of robots to test their algorithms of directed dispersion within bounded spaces where their robots spread out according to information received from local neighbours about their positions. Further, they proposed an algorithm that allowed a swarm of robots to explore an arena larger than the maximum distributed formation of the swarm using a pulling strategy which guided the whole swarm into unknown regions without losing connectivity or breaking the achieved pattern. Their results showed that path planning and directed motion algorithms become easier to develop when the primary input is the positions of other nearby robots. This guided the decision to design the robots of chapter 3 so they would seek to maintaining contact with at least one other robot at all times to prevent formations splintering into different groups which could increase the risk of missing sections of ship hull while performing a search.

Falconi et al. (2015) also provide a good example of how robots using the positions of neighbours can be leveraged by introducing a method of consensus-based formation control which allows groups of robots to maintain a given formation even in the presence of communication delays. This method relies on direct communication between robots and could be used as another method of exploring unknown environments in a given formation if the optimal dispersion of a swarm of robots does

not cover the entire area to be explored. Their results compare favourably against other formation control techniques using potential fields which are more susceptible to communication errors and propagation delays.

Ludwig et al. (2006) identified a vulnerability of inter-robot communication approaches to dispersion in that these approaches relied heavily on receiving accurate information of the relative distance and bearing of other robots through sensors. Their solution was to propose an approach that instead used the strength of received wireless signals from other robots to approximate their distance and use this information to effectively disperse. This promised to be an effective alternative which they proved through simulation. Ugur et al. (2007) took this approach a step further with experiments in both simulation which more accurately modelled the sensors and on real robots. In addition, they applied attractive and repulsive forces to robots based on the received wireless signal intensities, similar to approaches used in potential fields, to modulate the distance robots would travel from neighbours to ensure they did not travel out of range or remain too tightly clustered. Their results reinforced that this was an effective method of dispersion, but demonstrated that the detected signal strength was largely susceptible to the orientation of the communicating robots, which highlighted the necessity of selecting appropriate hardware and contingencies for signal errors in such systems.

Spears et al. (2004) were one of the first to propose a method of creating pattern formations in large groups of robots using physics inspired virtual forces referred to as physicomimetics. In their approach the robots display repulsive or attractive forces acting on neighbouring robots that fall within range of their sensors. Each robot is given a threshold value within their sensor range allowing the force on their neighbours to transition from attraction to repulsion and vice versa. This method was demonstrated in 2D and 3D simulation to be capable of forming square and hexagonal lattices which were capable of adapting to the loss of agents – such robustness is greatly desired in real world swarm robot systems where loss of agents is a possibility. Sallam et al. (2015) adapted the virtual forces framework to develop their own method (COVER) of cooperative area coverage with robots using virtual forces to achieve desired formations and population densities around landmarks in an unknown

environment. Their results demonstrated a way of deploying this technique to solve scenarios concerning discovery and monitoring of areas of interest, such as inspection.

Reif et al. (1999) were the first to propose social potential fields for distributed behavioural control of swarms of robots. In their approach, they apply artificial force laws to all robots giving agents both attractive and repulsive forces. As such, each robot's motion is determined by the resultant artificial force imposed by other robots and components of the system. Balch et al. (2000) employed a similar technique on a simulated swarm of goal-oriented robots in a bounded arena with goals and obstacles. These distributed control techniques, where calculations of motion are performed asynchronously, proved successful at demonstrating pattern formation and obstacle avoidance when navigating towards goals. However, these experiments did identify issues with the approach such as scenarios where agents converged to sub-optimal solutions and local minima.

Finding an optimal solution to local minima avoidance (LMA) and local minima escape (LME) is a subject which has received much attention since the first applications of social potential fields to swarm robots. Notable examples of such solutions include works by Mabrouk and McInnes (2008) who allow agents to use their internal states to influence the potential field in way that allows them to achieve LME. Alternatively, Couceiro, Rocha, and Ferreira (2011) implemented a social inclusion and exclusion concept which formed a punish-reward system allowing agents close to becoming stuck in sub-optimal solutions to achieve LMA and LME.

Despite these limitations, researchers such as Howard et al. (2002) and Poduri et al. (2004) were still able use social potential fields to develop effective systems of deployable sensor networks, which successfully tackled area coverage scenarios with results comparable to other techniques being employed at the time. One of main criticisms of potential field approach is the difficulty of implementation of real robot swarm without use of centralised control but researchers are continuing to develop new methods to address this shortcoming such as the Probabilistic Communication based Potential Forces (PCPF) model proposed by Mikkelsen et al. (2013). PCPF assigns both attractive and repulsive forces based only on the probability of communication between robots and the received signal strength, resulting in a method which is more

robust to unreliable sensor readings and external noise. This makes PCPF arguably easier to implement on real robots than the basic potential fields approach since PCPF better compensates for such factors which are likely to be encountered in real robot systems.

### Section 2.5.2 Coordinated Motion/Flocking

Another key feature of efficient exploration in swarms of robots is the ability to achieve coordinated motion. Coordinated motion, or flocking behaviour, is the term ascribed to collections of robots capable of navigating towards a common goal in a given formation or pattern while retaining the ability to avoid collisions with both obstacles and other robots. Such techniques are especially useful in scenarios where the maximum area of dispersion for a group of robots performing area coverage is smaller than that of the environment to be explored (Falconi et al., 2015). Robots performing coordinated motion must remain within communication range of neighbouring robots in order to avoid splintering into separate groups, much like robots tasked with pattern formation. Indeed, pattern formation is considered a necessary precursor to achieving effective coordinated motion and studies on both subjects are often complimentary.

Flocking behaviour was originally inspired by the abilities of groups of social animals to move with a coordinated motion such as flocks of birds flying in formation, or schools of fish evading predators (Okubo, 1986). Reynolds (1987) was the first to reproduce flocking behaviour in simulated agents, which he achieved by instilling members of the swarm with three rules: collision avoidance, velocity matching, and flock centring. This seminal paper demonstrated that any multi-agent system made up of individuals that can sense the distance and relative heading of other members of the swarm are capable of achieving coordinated motion with the appropriate behaviour. These three rules served as the basis for subsequent studies into achieving coordinated motion in swarm robots, even though more recent studies have since demonstrated that flocking behaviour can still be achieved without exchanging heading information (Antonelli, Arrichiello, and Chiaverini, 2010; Moeslinger, Schmickl, and Crailsheim, 2010; Stranieri et al., 2011; Ferrante et al., 2012).

Balch and Arkin (1998) advanced the field of coordinated motion in robot teams by identifying three methods agents could use to maintain a given formation: unit-centre-reference, leader-reference, and neighbour reference (Fig.2.5.1). In the unit-centre-reference approach, each robot computes the centre of the formation by averaging the x and y coordinates of all of the robots involved in the formation and determines its position relative to that centre. In the leader-referenced approach, each robot determines its position based on the position of a leading robot, except the leader who does not attempt to maintain the formation, but whose decisions affect the actions of its followers. The neighbour-reference method tasks each robot with maintaining a position relative to a pre-determined neighbour only.

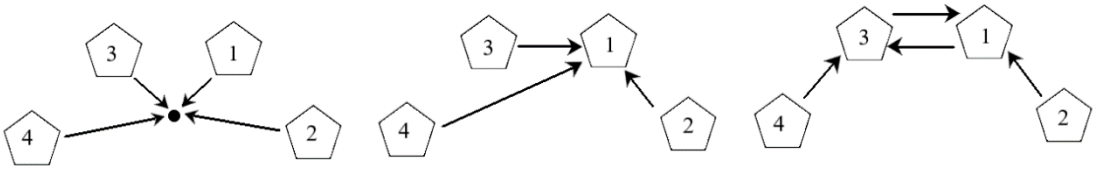



*Fig.2.5.1 Formation position determined by the three referencing techniques (From left to right: unit-centre, leader, neighbour), reproduced from Balch et al. (1998).*

The effectiveness of these referencing techniques to achieve coordinated manoeuvres, such as 90° turns and maintaining formation across an obstacle field, were tested on four formations common to mechanised infantry units used in the military: line, column, diamond, and wedge. The results from these experiments demonstrated that the unit-centred approach performed the best at both turns and formation control across obstacles for all formations, but identified there are scenarios where this approach would be less suitable. Unit-centre is very dependent on the ability of member to sense the position of every other member of the swarm which becomes impracticable in systems made up of many more units with limited sensing capability. It is also a technique ill-suited to scenarios where communication is restricted. In such scenarios, the leader-referenced or neighbour-referenced approaches would prove more practicable.

Neighbour-referenced approach presents its own issues, such as scenarios where an agent fails resulting in a formation that splits into two or more separate groups. Balch and Hybinette (2000) remedied this shortcoming in an alternate study which used

virtual forces (social potentials) to create flocking behaviour and enact formation control. In this approach, the position of each robot was calculated relative to the positions of multiple neighbours that fell within its short sensor range. This modification allowed their swarm of robots to form and maintain more complex formations such as lattice structures while navigating to goals and avoiding obstacles (Fig.2.5.2), resulting in a system that was more robust to unit failure. The success of this study and subsequent works concerning flocking was so distinct from the original neighbour-referenced technique it lead to a new classification, known as multi-neighbour-reference (Navarro and Matía, 2013), which remains a popular method used to achieving flocking behaviour.

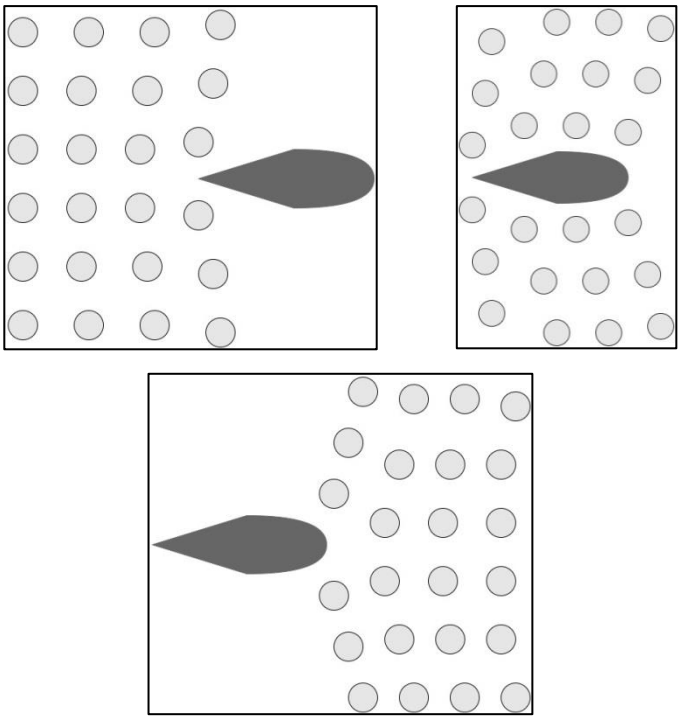


*Fig.2.5.2 Multi-neighbour-referenced approach to coordinated motion; a formation of 24 robots following square attachment geometry successfully navigates around an obstacle reforming on the other side, abstracted from Balch et al. (2000). The small grey circles represent the robots and the large dark grey object is the obstacle to avoid.*

Studies on achieving coordinated motion in swarm robotics fall under two broader categories: direction by global target and emergent direction (Bayinder, 2016). In the direction by global target category, some or all members of the swarm have access to a global target location which can be used to guide them to their goal and help maintain formations while in transit and avoiding obstacles. This was the approach used by Balch et al. (1998) in the studies discussed previously and similarly by Hayes and

Dormiani-Tabatabaei (2002) in their work concerning leaderless distributed flocking algorithms for swarm robots.

Allowing agents access to global information can serve as an advantage in that all agents know where they must navigate to without needing to communicate this with neighbours, so communication between agents is only needed for maintain formations. In groups where only a fraction of the swarm has access to global information, communication between agents is also used to spread this knowledge throughout the swarm to inform them of the heading (Çelikkanat and Şahin, 2010), and assist in reaching consensus on priority targets when there are multiple goals (Ferrante et al., 2014). However, swarm systems that rely on prior knowledge are only applicable in known or partially known environments – and so are ill suited to exploration of unknown areas. Coordinated motion algorithms that function on emergent direction are preferred for scenarios where prior information is not available and the area to be explored is unknown.

In the emergent direction category, swarms achieve coordinated motion without using shared knowledge of global information, but from using only local interactions between agents. Turgut et al. (2008) implemented such a flocking algorithm on a swarm of real and simulated robots, using only proximal control and heading alignment to achieve coordinated motion. Their approach was successful in navigating arenas with obstacles in the absence of global information. Their system was also shown to be more robust to errors in relative heading measurements shared between swarm members – a resilience which only increased when more agents were added to the swarm. Moeslinger et al. (2010) demonstrated how flocking could be achieved using emergent direction with their implementation of a low-end flocking algorithm which was based on simple rules of collision avoidance, separation, and cohesion. Their results showed that with appropriate distance threshold applied to the infrared sensors of the robots; flocking behaviour could emerge even without communication or preassigned tasks of alignment.

Vásárhelyi et al. (2014) implemented a decentralised flocking algorithm on flying robots which controlled the distance between agents using GPS data and wireless communication between agents. Their approach used a repulsive distance-based force

between neighbouring units to avoid collisions and defined an upper threshold for repulsion to avoid over-excitation. To compensate for time lag in communication, robots close to each other damp their velocity difference to reduce oscillations and synchronise their collective motion with a viscous friction-like term. Their control algorithm resulted in a swarm robot system with a high stability with resistance to noise and delays in communication and sensing. This study was of particular importance because it was one of the first that identified how to address real world limitations, such as time lag, in swarm robot systems performing coordinated motion.

### Section 2.5.3 Localization and Mapping

Navigation of any unknown environment presents challenges for both singular and multi-robot systems, but there are two particular problems which have a distinct effect on the effectiveness of the exploration of these environments: localization and mapping. Localization is the ability of a robot to determine its position relative to objects, landmarks, and other robots in either its immediate surrounding or globally, and mapping is the process by which the robots construct a record of these features for future reference. There are a number of prospective multi-robot systems capable of performing path finding without localization and mapping by instead utilising communication and the dispersion of team members within an environment as their method of navigating to a desired goal (Cohen, 1996; Payton et al., 2001; Ducatelle, Förster, Di Caro, and Gambardella, 2009; Mullins, Meyer, and Hu, 2012). However, a greater variety of complex behaviours become possible to achieve by implementing localization or mapping in multi-robot systems, such as path planning within unknown dynamic environments. In this section, key approaches to achieving decentralised localization and mapping in multi-robot systems and their benefits are discussed.

In decentralised multi-robot systems, the task of determining the position of robots without the aid of external references such as global positioning system (GPS) is non-trivial. This challenge is known as the localization problem and the best solutions devised to solve these issues can be categorised into two classes: range based methods and range-free methods. Range-based methods rely on the ability of individuals to measure the distance between themselves and global references or neighbouring robots using the Received Signal Strength (RSS), Time of Arrival (TOA) or Time Difference of Arrival (TDOA) of two signals known to have different speeds of

propagation (Mao, Fidan, and Anderson, 2007). Range-free methods are able to estimate the position of robots without measuring distance, instead relying heavily on external references or the presence of recognisable markers within the environment. Range-free solutions typically require fewer resources than range-based methods making them more economical, but their results are not as accurate (Yun, Lee, Chung, Kim, and Kim, 2009).

Fox, Burgard, Kruppa, and Thrun (2000) developed one of the earlier range-based localization techniques for multi-robot systems working in indoor environments. Based on Markov localization, their approach allowed a team of heterogeneous robots equipped with sensors of different granularity to achieve localization faster than robots performing the task individually by working collaboratively. Their results support the theory that robots performing localization cooperatively could outperform uncoordinated individual efforts, but also identified several limitations with their approach such as only operating if the robot is able to detect and identify the robot it has seen, and a lack of error handling for false-positive detection of robots greatly reducing robustness.

Roumeliotis and Bekey (2002) set out to address some of the limitations of previous approaches by devising a multi-robot localization technique based on the popular extended Kalman filter (EKF). In their approach, they devised a centralised EKF designed to account for the position and orientation of all members of the swarm and split it into component equations which they distributed across the team of robots. Each robot collected information from their proprioceptive and exteroceptive sensors and used their respective equation to make estimations of position and orientation, which was made more accurate by comparing estimations from neighbouring robots within communication range. This approach required less computation and communication than previous approaches and was scalable to larger teams. Furthermore, they showed that information sharing between robots with different levels of capability allowed the fully functioning robots to improve the estimates of malfunctioning or less capable robots, increasing robustness of the swarm.

Roumeliotis et al.'s (2002) application of EKF to a distributed sensor network was seminal to multi-robot localization studies and many papers which followed adopted

EKF as the leading method. Martinelli, Pont, and Siegwart (2005) built on the original paper by using a similar implementation of the EKF to achieve decentralised localization using the relative observations between robots, such as relative bearing, relative distance, and relative orientation to successfully increase the accuracy of the estimations than had been achieved previously. Madhavan, Fregene, and Parker (2004) used an EKF to propose a scheme for distributed outdoor localization and terrain mapping, which was a significant step for addressing how to achieve multi-robot localization in uneven environments. Their approach was shown to operate well in unmapped and unknown environments, and was further distinguished from previous studies for being the first that required no restriction on the number of robots that could move at any one instant while performing localization.

More recent studies have begun to move away from the use of external references such as GPS in efforts to increase the type of environments their methods could be applied to, such as underwater environments where access to such systems is not possible but localization is still required. To this end, De Sá, Nedjah, and De Macedo Mourelle (2016) proposed two algorithms to aid in localization without the use of external references such as GPS; one based on the Particle Swarm Optimization (PSO), and another based on the Backtracking Search Algorithm (BSA). In both approaches, the robot locations are determined relative to neighbouring robots using range-based methods and applying confidence values to the measurements obtained to better determine how accurate the reading is. Their inclusion of the confidence factor improved the reliability of their techniques, which was shown to be more significant when fewer neighbouring robots were available for the calculations.

Understanding the objects surrounding a robot at any instant via localization is highly beneficial in robotic systems performing path planning, and navigation can be improved further by using this information to create records of previous instances via the process of mapping. Maps are representations of the physical environment surrounding a robot created by transforming data from sensors into spatial models, which are typically either topological or geometric (Thrun, 2002). The task of constructing a high resolution map when the location of a robot is already known has already been achieved in previous studies using sonar sensors (Moravec and Elfes, 1985), and vice versa using various algorithms (Borenstein, Everett, and Feng, 1996).

However, the task of mapping becomes much less trivial when the locations are yet to be determined, and this complexity increases when the task is to be performed by distributed swarm robot systems working cooperatively due to the lack of centralised control, and limited resources such as memory, computation and communication. There are numerous studies which have attempted to overcome these limitations, but of the many approaches dedicated to finding an optimal solutions to localization and mapping, the most effective methods developed to date involve a process that undertakes both of these tasks at the same time; Simultaneous Localization and Mapping (SLAM) (Durrant-Whyte and Bailey, 2006).

Robots performing SLAM estimate their trajectory and the locations of landmarks using on-board capabilities and without the need for a priori knowledge. These estimates of landmark locations carry a degree of error, however the differences between true and estimate landmark locations is common between the landmarks due to the observing robots initial error in estimating its own location. This means the relative locations between any two observed landmarks are known with high accuracy even when the true location of a given landmark is uncertain. These discoveries led to one of the more important insights into the SLAM technique; increasing the number of observations always improves the estimates of relative landmark locations (Bailey, Nieto, and Nebot, 2006), and as the accuracy of the map increases the estimate of the location of the robot relative to these landmarks also improves resulting in highly accurate localization.

However, building maps with this technique requires that the individual robots performing the mapping process have access to a significant amount of memory and computational power. The multi-robot systems examined within this study are fully decentralised and only have access to very little memory and computational ability, which significantly reduces the feasibility of implementing such mapping techniques. As such, it was decided that for these studies localization techniques alone would suffice, while implementation of advanced mapping techniques would be delegated to future studies.

## Section 2.6 Self-Assembly

Self-assembly is a complex spatially organised behaviour employed in swarm robot systems which can allow the swarm to perform functions individual robots are not capable of, such as navigating difficult terrain (Mondada et al., 2005; O'Grady, Groß, Christensen, and Dorigo, 2010) and collaborative transportation of objects (Groß and Dorigo, 2009). However, self-assembly in swarm robot systems are typically facilitated only through local interactions between agents and so require many members to be within communication range of one another. Many swarm robot scenarios involve members initially being dispersed within an environment and so require a method they can follow to regroup at a common location - this is the task of aggregation. Both aggregation and flocking behaviours (only the latter was discussed in Sec 2.5.2) can be considered precursors to achieving self-assembly in swarm robots. This section discusses the various approaches to aggregation and self-assembly which were the most influential to the design of the multi-robot system and methodology used in emergency ship hull repair study in Chapter 4.

### Section 2.6.1 Aggregation

Like many swarm robot studies, the task of aggregation was originally inspired by behaviours observed in social insects which saw them gathering at common locations under specific conditions. Some notable cue-based artificial behaviours, where the gathering of agents is influenced by environmental conditions, were developed to mimic those observed in nature, such as bees choosing to rest in areas of high temperature (Schmickl and Hamann, 2011) or cockroaches being drawn to areas with less light to safely rest (Garnier, Gautrais, Asadpour, Jost, and Theraulaz, 2009). In these examples, aggregation is guided by both external stimuli and inter-robot communication which was shown to be more effective at achieving aggregation than relying on environmental information alone. Other studies indicate that it is also possible to achieve aggregation in systems that do not use environmental cues, known as self-organised aggregation, utilising only inter-robot communication and artificial forces instead (Mogilner and Edelstein-Keshet, 1999). The methods of control used to achieve aggregation in artificial systems which are most pertinent to the studies in this thesis can be categorized into two types: virtual forces, and probabilistic finite state machines.

Virtual forces are a popular method used in swarm robotics to maintain set distances between agents as discussed in Section 2.5.1 for applications of pattern formation. When non-local virtual attractive and repulsive forces are applied to the components of a swarm robot system, it can influence the movement of agents across great distances, allowing them to stay grouped together while avoiding collisions between themselves and objects. The magnitude of the attractive and repulsive forces acting between a robot and its neighbours is typically dictated by distance, such that robots will move towards each other when the distance between them is large, but will transition to repulsing one another once they cross a given distance threshold. This allows swarms to gather at common locations and form clusters while maintaining safe distances between agents so as to avoid collisions (Mogilner et al., 1999; Vanualailai and Sharma, 2010; Fetecau and Meskas, 2013). These non-local virtual force techniques have successfully achieved aggregation behaviour in simulated environments, but it is significantly harder to implement such behaviours in real robot systems where the robot sensing capabilities necessary to perform such techniques are not considered cost effective, or as scalable as more distributed techniques. These are some of the main reasons why there are relatively few studies on implementing non-local virtual forces for the purpose of aggregation in real multi-robot systems.

Another method of achieving aggregation in swarm robot systems is to employ probabilistic strategies. In probabilistic finite state machines, the behaviour of the robot is represented as various states with a given probability of transitioning between them. When applied to swarm systems performing aggregation, robots decide stochastically whether to transition between: 1) approaching other robots, 2) remaining still, or 3) moving away from other robots (Soysal and Sahin, 2005). The probability of transitions can be fixed or vary according to influences from environmental cues, such as the number of robots present in their current location or more complex inter-robot communication (Sahin et al., 2002). One of the main reasons studies modelling aggregation behaviours using finite state machines employ probabilistic strategies over deterministic methods is the ability of PFSM to form unstable aggregates where robot join and break from existing clusters at random intervals. Introducing such instability has proven effective at ensuring single large aggregates form while reducing the risk of stagnation in sub-optimal solutions which

form several separate clusters (Garnier et al., 2005; Hamann, Schmickl, Wörn, and Crailsheim, 2012).

Bayinder (2016) categorised the various aggregation algorithms employed in swarm robotics into two main types: free aggregation and environment-mediated aggregation. Free aggregation algorithms are designed to allow multiple robots to form aggregates anywhere in an environment, without preference for any particular location. In environment-mediated aggregation algorithms, the conditions of the robot's surroundings influence the robot behaviour such that certain locations and conditions increase the likelihood of robots forming groups. The studies within this thesis are more concerned with achieving aggregation at specific locations, thus environment-mediated aggregation algorithms are more relevant. In particular, Arvin et al. (2014) demonstrated an aggregation algorithm which allowed a group of dispersed robots to aggregate at a specific location using an acoustic signalling system. In their approach, the area of aggregation was specified by emitting a sound from that location using a speaker. The robots used microphones to detect the direction and intensity of the signal and move accordingly. This system resulted in a successfully formed group at the specified location. Schmickl, Möslinger, and Crailsheim (2006) also provided a notable method of enabling swarms of robots to aggregate at two assembly points of different size with the requisite that the number of robots at each site should be proportional to size of the assembly location. Their agents were equipped with minimal sensors capable of detecting when they were at one of the specified regions but unable to determine its size, and communicating with neighbours. Their system of communication between agents resulted in a collective perception capable of collectively measuring the size of the target areas and to communicate these sizes with the whole swarm.

### Section 2.6.2 Self-Assembly

One of the most prominent advantages of multi-robot systems is their ability to perform tasks which individuals alone are not capable of, and there is no task in swarm robotics which exemplifies this better than self-assembly. In swarm robot systems, self-assembly refers primarily to multi-robot systems where agents have the ability to communicate and connect with one another to form structures and configurations capable of more than the sum of the individuals acting independently. Studies in self-

assembling swarm robot systems focus on two major aspects: robots autonomously connecting with each other in order to create a desired target structure, known as morphogenesis, and controlling the resultant structure to perform novel tasks. In their comprehensive review, Groß and Dorigo (2008) categorised studies on morphogenesis in macroscopic systems by their primary function such as formation, growth, self-reconfiguration, self-repair, and template replication.

Formation studies focused on using swarms to produce one or more objects of a predefined size and structure (Hosokawa, Shimoyama, and Miura, 1994). Growth studies were concerned with increasing the number of robots that make up a given structure, which is considered an essential feature of all self-assembling robot systems (Fukuda, Husband, and Ueyama, 1994). Self-reconfiguration studies worked towards designing systems capable of changing an existing entity structure to form a new entity better adapted to changes in the environment or capable of performing different functions than the original (White, Kopanski, and Lipson, 2004; White, Zykov, Bongard, and Lipson, 2005). Self-repair studies investigate ways entities could replace faulty or damaged modules with other fully functioning modules (Bererton and Khosla, 2001). Template replication studies use modules to recreate templates of objects with a known size and structure (Griffith, Goldwater, and Jacobson, 2005). The rest of this section identifies studies which have advanced the field of self-assembly with respect to swarm robot systems.

There are three notable aspects of morphology that are routinely considered when designing self-assembling swarm robot systems: binding mechanisms, sensors, and communication methods. Swarm robot systems that utilise passive binding techniques such as the use of permanent magnets and electromagnets (Hosokawa et al., 1994; White et al., 2004; Doyle et al., 2016) are advantageous due to their relative simplicity and low power consumption, but they come at the cost of limited connection strength. Alternatively, passive mechanical connection methods can be used in swarm robot systems to address connection strength, such as the pin and hole connection method (Yim, Duff, and Roufas, 2000; Castano, Behar, and Will, 2002) in which robots are designed with faces and pins that correspond to holes on the face of another robot. This form of attachment results in links more resistant to shear stress, but the robots require a higher degree of accuracy for the task of aligning faces. Active mechanical

links such as actuated mechanical hooks can ensure much stronger links between robots with lower accuracy requirements than passive mechanical techniques (Fukuda and Kawauchi, 1990; Mondada et al., 2004; Wei, Chen, Tan, and Wang, 2010), but typically consume more power and have a higher risk of failure than their passive counterparts.

Sensors can be assumed to be essential to all swarm robot systems, but in studies concerning self-assembly, sensors have played a smaller role in externally propelled systems than self-propelled systems. Externally propelled robots which rely on external manipulators to move such as magnets guiding agents and thus designing robots to store information about their surrounds has been considered less essential to the functioning of the system. However, self-propelled robots which use internal power sources to move themselves with propellers or wheels require more data about their surroundings to make informed decisions, and a variety of sensors have been used to ensure this. There are many types of sensors used to gather information about robots surroundings in self-assembling swarm robot systems, including the use of bump switches to detect collisions and confirm physical interactions between agents (Bererton at al., 2001), infrared detectors and ultrasonic distance sensors for detecting the presence of obstacles or other robots (Fukada et al., 1194; Castano et al., 2002; Wei et al., 2010), inclinometers to detect changes in angles of slope or elevation of a robot (Yim et al., 2003; Murata, Kakomura, and Kurokawa, 2006), and cameras to gather addition information about obstacles, robots and environmental features (Yamakita, Taniguchi, and Shukuya, 2003; Mondada et al., 2004; Bonani et al., 2010).

Communication is a vital component to achieving many of the behaviours in swarm robot systems, and self-assembly is no exception. Some of the most popular communications methods for self-assembly swarm robots include infrared emitters and receivers for line-of-sight communication (Fukuda et al., 1990; Yim et al., 2000; Castano et al., 2002; Murata et al., 2006), Wi-Fi, Bluetooth, and Zigbee for more reliable wireless communication in crowded environments (Groß, Bonani, Mondada, and Dorigo, 2006; Wei, et al., 2010; Bonani et al., 2010), and LEDs for close range communication between individual modules and signalling of states (Groß et al., 2006; O'Grady et al., 2009; Doyle et al., 2016). The method of communication chosen for the system can greatly impact the complexity of formations and reconfigurations possible

to create. The chosen method is also subject to the environment the robot is expected to perform the assembly in, for instance, direct line-of-sight communication methods are ill equipped to function in environments with many obstacles and wireless communication may be a more appropriate choice.

The effective synthesis of these technologies has led to a number of notable achievements for swarm robot systems performing self-assembly. Some of the notable platforms developed for multi-robot self-assembly experiments include the Swarmbot, MarXbot, Kilobots, Symbrion, and Mori. Groß et al. (2009) used Swarmbots to demonstrate self-assembly for the purpose of collaborative object transportation, where a group would surround objects of different shapes and sizes, connect to each other, and pull the object to a desired location with their increased pulling power. O'Grady at al. (2009) used tis same platform to demonstrate their SWARMMORPH protocol which could guide Swarmbots into achieving different morphologies using LEDs to inform where the robots should approach and connect to each other. Bonani et al. (2010) developed the MarXbot to improve on various aspects of the Swarmbot design including an improved binding mechanism and methods of communication. Mathews et al. (2011) were able to utilise the MarXbot to perform directional self-assembly, which robots forming part of a desired structure guided other robots using radio signals, informing them where they could best attach in order to complete the entity. The Symbrion and Replicator projects (Levi and Kernbach, 2010) investigated many aspects of self-assembling swarms but focused primarily on the realisation of symbiotic multi-robot organisms. The resultant Symbrion modules (Liu and Winfield, 2010) were capable of operating as fully autonomous agents in swarm mode, but could also transition to form part of a greater structure in organism mode where energy and computational resources could be shared between neighbours. Rubenstein, Cornejo, and Nagpal (2014) were some of the first to demonstrate self-assembly and pattern formation in very large swarms using one thousand Kilobots. Their approach allowed the swarm to form various shapes using four stationary robots to serve as an anchor point and having agents connect to them appropriately. Doyle et al. (2016) developed a prototype floating robot capable of controlling the motion of a structure built from their modules using modular hydraulic propulsion, demonstrating how such technology could be used to guide such structures. Belke and Paik (2017) developed

the Mori platform; a triangular two-dimensional lattice type reconfigurable modular origami robot, which is notable for its genderless connection mechanism and flexibility with regards to the variety of complex shapes it can assume from simple component modules.

## Section 2.7 Foraging

Foraging is considered to be one of the more complex forms of collective behaviour to replicate in multi-robot systems as it relies on the correct execution of a number of behaviours considered difficult in their own right, such as exploration, global and local communication, collective transport, and collective decision making. From an individual agent's perspective, the foraging task can be described as a sequence of the following behaviours: exploration of an environment surrounding a nest, identifying objects and areas of interest, returning the objects to the nest, communicating its discovery with other robots, and returning to the area of interest to collect more objects (Dorigo and Di Caro, 1999). The task of foraging in swarm robot systems was inspired by observations from biological collectives such as bee swarms (Montague, Dayan, Person, and Sejnowski, 1995) and ant colonies (Traniello, 1989) and their ability to use local interactions between individuals to exploit resources surrounding their nests. One notable extension of this behaviour is multi-foraging, where there are multiple types of retrievable objects in an environment (Campo and Dorigo, 2007), which presents a promising basis for accomplishing complex practical tasks using multi-robot systems such as mining or search and rescue operations. In this Section, some prominent foraging strategies applied to swarm robot systems are presented according to their applicability to the studies presented in Chapters 4 and 5.

It has been proven that the problem of resource collection in dynamic environments can be solved by social insect colonies using collective central-place foraging (Olsson et al. 2008; Detrain and Deneubourg 2008), and it is this success that has spurred research into recreating such efficient and scalable approaches in swarm robot systems. A popular method of achieving foraging behaviours in swarm robot systems involves first deconstructing the behaviour into simpler tasks that flow in sequence. However, the defining features of these systems often lie in the methods they use to communicate information between individuals. There are a number of methods researchers have implemented to achieve foraging behaviour analogous to those

observed in biological super-organisms and these can be categorised under the following two categories: direct communication as outlined in Section 2.6 and stigmergic communication where information is shared via modification of the environment (Bayinder, 2016).

Swarm robots systems using global forms of direct communication such as signal broadcasting can share information between robots over moderate distances. This approach can aid in aggregation behaviour (Arvin et al., 2014) for the task of foraging so that robots can inform others of an area of interest (Vaughan, Støy, Sukhatme, and Matarić, 2000). However, the performance and reliability of these methods tend not to scale well to very large numbers of robots or over increased distances which is an undesirable feature of true swarm robot systems (Şahin, 2005). It can also be difficult to implement such features on simple robots with limited capabilities, of which most swarm robot systems consist, making it a more impractical option for certain platforms. Thus, this method of communication may be considered appropriate for multi-robot systems that use fewer agents and operate over short distances, but sub-optimal for swarms consisting of many more agents that operate in larger arenas or unbounded search spaces.

Conversely, local direct communication methods which rely on exchange information between neighbouring robots that fall within a given range can be considered highly effective at facilitating effective foraging behaviour in swarm robot systems. Direct explicit exchange of data can be used to report a robot's respective state or to indicate the direction of objects, areas of interest, or the location of a central nest to neighbouring robots (Arkin, Balch, and Nitz, 1993; Rybski et al., 2004). This information can be used to improve the robots present behaviour and help it achieve its current goal more effectively, be it searching for objects, or returning them to the nest. In addition to direct data exchange, local sensing strategies which simply detect the presence of nearby robots or obstacles can also be used as an effective tool to aid foraging behaviour (Hoff, Sagoff, Wood, and Nagpal, 2010). These direct communication methods can be used to improve the swarm's ability to reduce overcrowding (Goldberg and Mataric, 2000) and form more organised paths between nests and areas of interest (Sadat and Vaughan, 2010; Penders and Alboul, 2012), enabling more effective foraging strategies.

Bee algorithms are a notable class of nature-inspired collective behaviour that use direct communication techniques to mimic the foraging strategies of honey bees (Karaboga and Akay, 2009). In bee colonies, foraging consists of a sequence of behaviours starting with exploration of the sear surrounding the central nest. On discovery of a food source, the bee collects the precious nectar resource and returns to the hive to deposit what it has gathered. After completing its deposit, the bee then performs a special dance of varying direction and intensity to indicate the direction and distance of its last collection source in the hope of recruiting more bees to assist in retrieval. This dance is the aspect of bee communication which when applied to swarm robot systems has been shown to yield effective methods of task-allocation and foraging (Jevtic, Gutiérrez, Andina, and Jamshidi, 2011; Schmickl et al., 2011). These individual interactions between agents can be combined to produce an effective collective decision-making process when the correct tuning parameters are selected, as demonstrated by Reina et al. (2015) in their shortest-path selection study.

Making changes to the environment in order to communicate between agents, known as stigmergic communication, is perhaps the most well studied form of indirect communication found in biological super-organisms and applied to swarm robots performing coordinated resource collection (Goss et al., 1992; Werger and Matari'c, 1996; Payton et al., 2001; Nouyan et al., 2009; Campo et al., 2010). In natural systems, this form of communication is best exemplified by certain species of ants which can secrete and detect pheromones – a chemical substance they can use to mark the environment (Hölldobler and Wilson, 1990). Ants deposit this pheromone on return to the nest from a resource site to serve as a mass recruitment mechanism helping to guide other ants to the same source of forage (Sumpter and Pratt, 2003). Foraging ants follow these trails, gravitating to paths with a high concentration of pheromone to exploit the best resource. This system allows ant colonies to form consensus on selecting the best resource site in the environment according to factors such as food quality (Beckers et al., 1990), path length (Goss et al., 1989), and predation risk (Nonacs and Dill, 1990). This positive feedback mechanism is typically disadvantageous to systems seeking to maintain adaptability and flexibility to a changing environment. However, there are alternative mechanisms observed in other ant species capable of overcoming this limitation such as: repellent pheromone to mark off undesirable paths

(Stickland et al. 1999; Robinson et al. 2005), using tandem running to recruit ants to newly available higher-quality food sources (Beckers et al., 1990), or using quality-dependent linear recruitment and quality-dependent abandonment (Shaffer et al., 2013).

To recreating stigmergic communication in swarm robots using techniques that mimic pheromones is a challenging task that must take into account how the pheromones are deposited, detected by others, and how the resultant trails change over time. The three most advanced approaches found in literature rely on either using robots as beacons, robots with on-board sensors and actuators, or smart environments. Beacon robot techniques use the robots themselves to act as a physical embodiment of pheromone, commutating the presence and strength of pheromone to neighbouring robots (Goss et al., 1992; Werger and Matari'c, 1996; Payton et al., 2001; Nouyan et al., 2009; Campo et al., 2010; Ducatelle et al., 2011; Hoff et al., 2012). This approach is beneficial since it can be implemented on many simple robots, but is limited by beacon robots being unable to contribute to the item collection task, ever increasing population size requirements to address larger environments, and beacon robot robots serving as obstacles in the environment also. These issues can be addressed by allowing the beacon robots to remain mobile and contribute to item retrieval (Sperati et al., 2011; Ducatelle et al., 2011), but performance of this approach relies on balancing the swarm size and communication range with the size of the search space.

There are a variety of ways researchers have tried to implement stigmergic communication in swarm robots using on-board actuator and sensors such as using marker pens to draw lines on a path to represent pheromone (Svennebring and Koenig 2004), emitting gas which other robots can detect (Purnamadjaja and Russell, 2007), energising phosphorescent paint using UV-LEDS (Mayet et al. 2010), and using ethanol (Fujisawa et al. 2008, 2014). Of these varied attempts, only the ethanol experiments of Fujisawa et al. (2008, 2014) were able to model the four critical characteristics of pheromones observed in natural systems: evaporation, diffusion, locality, and reactivity. The evaporation aspect is considered especially important to avoid runaway positive feedback (Garnier et al. 2007, 2013) which can cause swarms to become mired in sub-optimal solutions or become unable to break from expended resource sites.

The final category is smart environments which have the ability to store and supply virtual pheromone information to swarm robots in real-time (Sugawara et al. 2004; Garnier et al. 2007; Hecker et al. 2012; Garnier et al. 2013; Arvin et al. 2015; Valentini et al. 2018). Smart environments are considered one of the most popular approaches to implementing indirect communication in swarm robots, due to their low cost and adaptability to different sizes of swarms and search spaces. However, it is far less practical to use smart environments in real applications than the previously discussed alternative methods, so its use is instead delegated to targeted research. Mimicking pheromone trails using smart environments can be accomplished using radio-frequency identification (RFID) tags (Mamei and Zambonelli 2005, 2007; Herianto et al. 2007; Herianto and Kurabayashi 2009; Bosien et al. 2012; Khaliq et al. 2014), simulated pheromones using projected lights or other custom hardware (Sugawara et al. 2004; Garnier et al. 2007, 2013; Arvin et al. 2015; Valentini et al. 2018), or augmented reality tools in which a virtual environment is interacted with by robots using virtual sensors and actuators (Reina et al. 2015, 2017).

Determining what constitutes an optimal foraging model requires the selection of appropriate metrics with consideration given to currencies of costs (quantities to be maximised in order to achieve optimality) and benefits. The two metrics most often selected to measure success in foraging theory are the net rate gain of energy and efficiency (Kacelnik 1984; Houston and McNamara 2014). The net rate of energy gain is the difference between the forager's gross rate of gain and its rate of energy expenditure, while efficiency is the gross rate of energy gain divided by the rate of energy consumption (Houston and McNamara 2014). However, optimal foraging theory does not always apply to real systems and developing a theory that works for several foraging species seems inherently difficult, as the mechanisms underlying foraging can be quite different (Traniello 1989). Though there are many ant species where the production of pheromone trails is crucial in the foraging process, other aspects which are more generally related to the state of the forager and the environmental conditions should also be considered when developing an optimal foraging model.

In foraging scenarios, the problem of inter-robot interference also tends to arise frequently with multiple robots sharing a confined space. This increase in robot

congestion is noteworthy for the effect it can have on the efficiency of the overall swarm with respect to foraging. Increases in robot avoidance events or the length of time taken to overcome a near collision can increase the gross energy expenditure and time taken to complete the task. There are two methods of measuring the quantity and frequency of these occurrences: the number of collisions between robots (Maes et al., 1996; Goldberg and Matarić, 2000) or the time spent avoiding a robot while trying to perform another task such as transport an object to the nest (Krieger and Billeter, 2000). Both of these methods can be used in combination with other establish metrics to assess the impact increased collisions or manoeuvring time has on system efficiency and net energy gain.

## Section 2.8 Summary

The literature explored in the above sections tells a story of how far the field of swarm robotics has progressed over the past few decades, identifies the most predominate methods that have evolved out of the research, and can provide clarity on what could be done to ensure swarm robotics research continues to mature. This section identifies some of the gaps in existing knowledge that motivated the studies within this thesis.

The subjects of obtaining effective dispersion, pattern formation, coordinated motion, localization, and self-assembly in multi-robot systems has been explored at length in ground and air-based scenarios but significantly less so in underwater environments. This is in part due to the difficulty in translating these techniques, many of which rely on high frequency sensors and telemetry such as GPS (global positioning system) coordinates, into the underwater realm where such communication techniques do not work effectively due to the high absorption of the surround medium (water). Nevertheless, there are many underwater problems that could benefit from multi-robot solutions such as underwater inspection of ship hulls or off-shore rigs, monitoring and surveillance of marine life, and underwater construction. This gap in knowledge is partially what motivated the research into using swarm robots to perform underwater inspection and repair of ship hulls.

Foraging strategies in swarm robot systems can be considered a more mature field of research given the many studies concerned with how to achieve optimal foraging strategies. However, perfect emulation of an ant colony has not yet been achieved due

in part to the complexity of such systems. Indeed biological swarm intelligence is still a thriving field of study to this day, helping to inform how swarm roboticists may improve on their own designs. The work being undertaken at the University of Sheffield, with respect to swarm robot foraging strategies, represented another step toward creating a swarm robot system more capable of emulating the emergent behaviour observed in ant colonies, and was the main motivating factor behind developing obstacle avoidance behaviour for the robots. Ultimately, allowing the swarm of robots to more accurately represent the biological ant colony their behaviour was modelled after.

# Chapter 3. Ship Hull Inspection: Complete Area Coverage Algorithm

In this chapter, a novel approach to emergency ship hull repair using a swarm of autonomous underwater robots is introduced. This research uses theories of cooperative multi-robot exploration and communication to inform the design of a complete area coverage search method for a swarm of robots tasked with inspecting a damaged ship hull. The results from this Chapter show how the cooperative search algorithm is more effective at achieving complete area coverage in less time than the same multi-robot system using an uncoordinated search algorithm. Additionally, the chapter presents a simulated robot sensor arrangement that would allow robots to maintain a set distance from a 3D object. This novel utilisation of an additional constraint enables the robots to treat their environment more akin to a 2D plane, which allows for simpler implementations of search algorithms.

The general approach to emergency ship hull repair is presented in Section 3.1 but the majority of the chapter focuses on the first major stage of the ESHR scenario: ship hull inspection using a collaborative multi-robot system. This task poses the distinct challenge of how to fully inspect the submerged hull of a ship using multiple robots, how to do so effectively, and in a timely manner. To address this challenge, two complete area coverage (CAC) algorithms were devised: a sweeping search pattern and a lawnmower search pattern which are described in more detail in Section 3.2. The search patterns are intended to be used by homogeneous multi-robot systems to inspect the ship hull while it is still in the water as this is the repair process intended to take pace immediately following damage. To test the effectiveness of the algorithms and compare their results, the code was implemented on a simulated group of custom designed robot modules.

The simulated robot modules used to test the algorithms do not yet have a physical counterpart and as such, the robot module specifications are restricted to their geometric shape, key sensors and actuators, and descriptions of their capabilities which are based on existing technologies currently employed in mobile robotics and machine vision. A more detailed description of the technical and physical aspects of the robots is provided in Section 3.3, however it should be noted that these are features the simulated robot modules are assumed to possess for the purpose of the

algorithms. The experiments were wholly conducted in a simulated 3D environment built using Webots; a simulation suite which is renowned for its ability to correctly model mobile robots. The key features and reasons for its use in this study, along with the experimental setup used to compare the effectiveness of the CAC algorithms, are presented in Section 3.4. The results of the experiments are presented in Section 3.5 and are followed by a discussion of the findings and their implications in Section 3.6.

## Section 3.1 Emergency Ship Hull Repair

### Section 3.1.1 Background

Emergency ship hull repair (ESHR) is one of many stages of damage control that takes place in the event of a hull breach while at sea. Innovations in materials, mechanical engineering, and naval architecture have ensured that the strength and resilience of ship hulls has remained steadfast this past century, but no sea-faring vessel is immune to accidental or deliberate damage. When a ship finally suffers a fracture or hull breach, the race to prevent the loss of the ship begins.

Repairing hull damage immediately after an incident is necessary to prevent the loss of a ship. Reducing the ingress of water minimises the effect of flooding and supports efforts to restore buoyancy and stability to the damaged vessel, enabling it to either continue its course or return to a ship yard for extensive repair. There are numerous types of breaches that vary in size, shape, depth, and location; each of which affects whether the breach can be addressed by conventional means.

The standard approach to repairing ship hull breaches, known as shoring, has remained mostly unchanged from the end of the second world war and amounts to three general methods: (i) plugging the hole from the interior of the ship using soft wooden plugs, (ii) covering it with prefabricated patches from the exterior of the ship, and (iii) establish and maintain flooding boundaries within the ship to prevent further progress of the flooding (Center, 2013; Press, 1945). These are intended as temporary repairs and in most cases are not perfectly watertight, but even reducing water ingress by half can allow crew to quickly bring flooding under control using pumps.

These techniques serve to mitigate damage but are far from optimal given the delay between detecting a breach, assessing the damage, transporting materials, and

carrying out the repair. They are dangerous, time constrained procedures and with modern naval services moving towards greater autonomy with fewer crew members (Levander, 2017) it is beginning to stand out as a point of vulnerability. To remedy this situation, a modern approach to emergency ship hull repair is proposed, using a swarm of autonomous underwater robots to investigate the ship hull and carry out repairs. If realised, this solution could remove the requirement for engineers to carry out inspections to locate the damage and deal with most of the repairs, promoting greater autonomy of large sea-faring vessels and helping to safeguard the lives of the ship's crew.

### Section 3.1.2 General ESHR method

The EHSR method discussed is intended to address hull breach scenarios where ingress of water must be halted to prevent excess listing and quickly restore the stability of the vessel. The proposed approach suggests using a decentralised group of homogeneous autonomous underwater robots to collectively carry out ship hull inspection, aggregate at the hull breach location, self-assemble to form a sheet of connected robots, and use the resultant structure to cover and seal the hull breach. If carried out correctly, this would significantly decrease the ingress of water and allow human crew members to safely deploy pumps to drain the flooded compartments, restoring stability.

Using robots to operate in hazardous environments in place of human operators has been shown to be an effective solution to reducing the risk to human life and equipment while making processes faster and more reliable. Using multiple robots which work cooperatively to complete tasks, rather than individual robots, compounds these advantages by making the system more robust, flexible and scalable as discussed in Section 2.1. For this reason, the proposed ESHR solution suggests the use of multiple robots working together in order to maximise performance in terms of speed of completion, robustness to failure, and even distribution of workload.

When using multiple robots to carry out a coordinated task, it can be beneficial to employ decentralised control schemes. This is so that the system can scale its response to address more demanding scenarios without sacrificing performance due to increasing computational requirements observed in multi-robot systems which rely on centralised control schemes, as identified in Section 2.1. The ESHR method is intended

to address scenarios which require the detection and repair of ship hull breaches of various sizes and shapes, making the ability of the system to scale according to the requirements essential – thus the proposed solution uses a decentralised control scheme without a master control.

Multi-robot systems can help perform inspection of the ship hull more quickly, but utilising their greater numbers for the repair task would also be advantageous. One of the more commonly employed methods of repairing ship hull breaches from the exterior of the ship involves the use of patches to cover and seal the breach. This approach could be adapted for use by robot systems in two ways: collective transport or self-assembly. Using a swarm of robots to collectively transport prefabricated patches to the hull breach presents a number of issues such as patches being ill-fitted to the hole, difficulty of transporting objects underwater due environmental disturbances, or accidental damage to the patch serving as a single point of failure in the system.

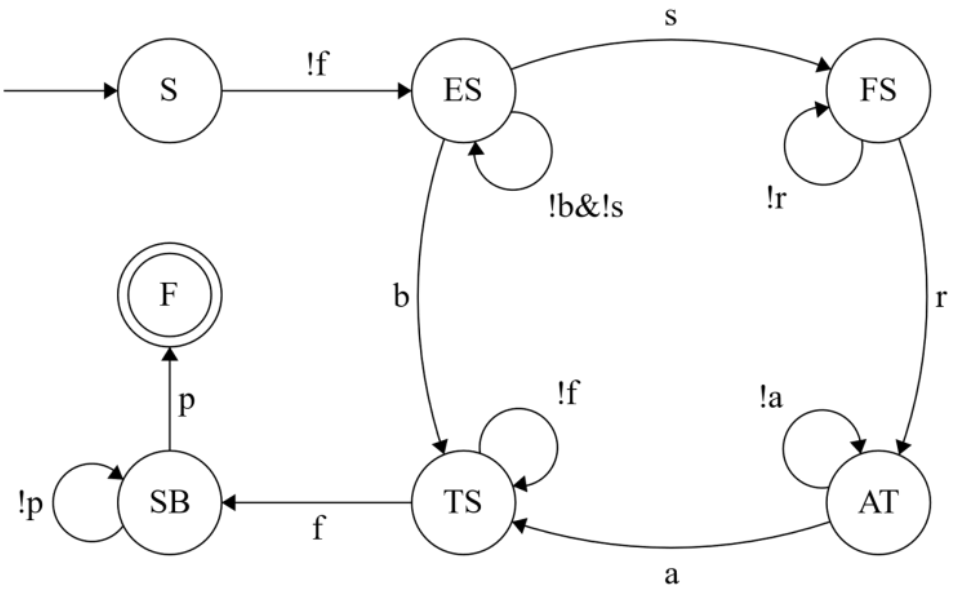
Self-assembling techniques such as those discussed in Section 2.6.2 could be employed to address the shortcomings of the collective transport approach. If the robots were designed as modular homogenous units, they could be programmed to form larger structures using their bodies which could then be used to cover holes of various shapes and sizes. The self-assembly approach was selected as the repair method as the robots can adapt their resultant structure to more accurately address damage while reducing the number of points of failure. The modular robots are homogenous because using heterogeneous robots to conduct self-assembly has been shown to decrease the scalability of the system.

Forming a structure of appropriate shape and size is a non-trivial task, however ensuring the structure can remain attached to the vessel once it has covered the hole is equally challenging. The precise method of underwater adhesion falls outside of the scope of this thesis, however the leading suggestion could involve the use of an underwater epoxy or fibre reinforced polymers (FRP) to be administered by the robot modules. Rubino, Nisticò, Tucci, and Carlone (2020) performed an extensive review of the use of FRP in underwater construction and repair of ship, and off-shore platforms. Their findings show that while the industry still prefers using metal as the primary

material for construction and long-term repair, FRP remains a promising alternative with marked success in the restoration of structures damaged by exposure to the marine environment, chemical agents, or marine life.

As discussed in Section 2.4, when designing cooperative multi-robot systems it is important to select a model which can be used to predict how the system will function. This allows of the evaluation of aspects such as feasibility of the task, number of robots required, and the effect of disturbances. Finite state machines (FSM) are a prominent method of modelling multi-robot system behaviour which has be used to solve various tasks such as exploration, pattern formation and collaborative mapping – at both the macroscopic ad microscopic scale. These tasks are closely aligned with the ESHR scenario and thus FSM was selected as the most appropriate model for the robot behaviour. The FSM of Fig.3.1 describes the robot behaviour and shows how inspection, assembly, and repair process is expected to unfold.

The emergency ship hull repair process begins with the robot modules being deployed into the water, entering the start state (S). If the robots receive no signal to indicate that a complete repair structure has been formed (!f) they immediately transition to begin searching the ship hull for damage (ES). The robots will continue to explore and inspect the ship hull until they either locate a hull breach (b) or detect a signal from another robot that has found a breach (s). If a robot is the first to locate a breach (b), it changes to the transmit state (TS) and begins broadcasting a short-range acoustic signal to other robots in its vicinity, notifying them of the location of damage they have discovered. However, if robots in the search state (ES) have not located the breach but have instead detected a signal from a robot that has (s), they transition to the follow state (FS) where they will move toward the origin of the signal until they find a robot in the area matching the location of the signal (r). This method of guiding robots to a specific location was inspired by studies of signal-assisted aggregation and self-assembly which were discussed in Section 2.6.1.




| States | | Transitions | |
|---|---|---|---|
| **S** | Deploy and start | **f** | Complete repair structure formed |
| **ES** | Explore ship hull | **b** | Hull breach located |
| **FS** | Follow signal | **s** | Signal detected |
| **AT** | Attach to robot | **r** | Robot located |
| **TS** | Transmit location signal | **p** | Repair structure attached to ship hull |
| **SB** | Seal breach | **a** | attached to robot |
| **F** | Finished | | |

*Fig.3.1. Finite state machine (FSM) of the emergency ship hull repair robot behaviour, showing the stages the robots move through: Searching for the damage, aggregating at the location, forming a repair structure, and sealing the hull breach.*

Once two or more robots rendezvous at the hull breach location, they can begin communicating with each other using their local sensors to determine where to attach to each other in order to best form a repair structure (AT). Successfully attaching to an appropriate part of the structure (a) will allow the robot module to transition to the transmit state (TS). In order to avoid transmitting multiple signals at once, all robots in the transmit state (TS) will communicate with each other via local sensors and reach a consensus which single robot should transmit the signal based on factors such as location and remaining power. This approach to choosing a unit for signal transmission was inspired by studies of collaborative decision making as discussed in Section 2.7.2.

The structure will continue to form by using a robot in the transmit state to guide robots in the follow state (FS) to optimal attachment positions until a structure of appropriate shape and size has been full constructed (f). The fulfilment of this transition condition will be determined as a result of robot modules communicating the number of robots connected to them and their location in relation to intact and damaged sections of the ship hull using local sensors. With a repair structure fully formed, the robot modules transition to the seal state (SB) during which they will collectively move to cover the breach and being adhering to the intact sections of hull surrounding the damage. When the modules have completed sealing themselves to the hull (p) they will transition to their final state (F) indicating that the operation is complete and that it is safe to deploy pumps into the flooded compartments.

## Section 3.2 Simulated Robot Morphology

In order for the proposed CAC algorithms to be assessed, a suitable robot model on which the code can be implemented is required. Section 3.2.1 specifies the robot functions and physical capabilities required to carry out the algorithms, demonstrate how existing underwater robots only fulfil some of these requirements, and identify the need for a bespoke simulated robot design. Section 3.2.2 delves into the specifics of the simulated robot morphology giving details of the various sensors, actuators, and communication techniques used with explanations of their function with respect to coordinated exploration of the ship hull.

### Section 3.2.1 Robot Specification

There are five main abilities the autonomous underwater vehicles (AUV) must possess in order to carry out the CAC algorithms: they must be able to move freely underwater, inspect the ship hull, detect objects and other robots, communicate with other robots over short distances, and self-assemble to form larger water-tight structures. Since their inception in the mid-20$^{th}$ century, there have been many AUVs developed for the purposes of underwater inspection, environmental monitoring, and various military applications. As a result, a plethora of methods and mechanisms have emerged with the aim of achieving more efficient navigation, communication, localisation, and mapping in the underwater domain (Paull, Saeedi, Seto, and Li, 2013; Aguirre, Vargas, Valdes, and Tornero, 2017). The ability to move freely underwater can be attributed to factors such as hull shape, method of propulsion, and buoyancy control. The geometry

of an AUV's hull plays an important role in how fast the robot will be able to move through the water – hydrodynamic designs such as the torpedo hull type (McPhail, 2009) are commonly employed in the design of many AUVs primarily for their ability to generate low drag force (Aguirre et al., 2017). However, alternative hull geometries such as open structure types (Boeing and Bräunl, 2012) are also worth considering for applications less concerned with maximum speed and more focused on incorporating irregular sensors and actuators for achieving applications such as underwater construction.

With the rise of biomimetic underwater robots designed to move using actuation similar to biological counterparts such as fish (Wang, Hang, Wang, and Xiao, 2008) a variety of methods of locomotion now exist for AUVs. However, these new technologies have yet to be implemented in commercial products and without further testing and verification, propellers and water jets remain as the most reliable methods of movement for AUVs. There are many ways propellers have been incorporated into AUVs to achieve systems with powerful forward thrust and the technology has continued to mature. In recent years, modifications to improve factors such as protection from marine debris using enclosed propellers (Kopman, Cavaliere, and Porfiri, 2011), and using quad-coper configurations to improve manoeuvrability underwater (Ranganathan, Thondiyath, and Kumar, 2015) have seen increased utilisation in robot designs. AUVs have also been shown capable of move freely using internal pumps that create water jets (Mazumdar, Triantafyllou, and Asada, 2015) and while this method is typically less effective at generating thrust than propellers, it can at least be used as an additional tool to assist in positional control. Pumps have also been shown to be an effective tool for injecting and ejecting water from internal ballasts enabling active buoyancy control in AUVs (Woods, Bauer, and Seto, 2012).

Every AUV requires methods of sensing its surrounds in order to achieve behaviours necessary for navigation such as obstacle avoidance, localization, and mapping. There are a variety of sensors available but additional considerations must be made, largely due to the difficulties associated with operating underwater. A common method of distance sensing for robots operating on the ground or in the air is infrared (IR) sensors, but when placed underwater the effective range of these devices is heavily restricted due to the absorption of rate of the water (Farr et al., 2010). Though even

with such restrictions, there are still of AUVs that prove it is possible to perform basic obstacle avoidance using such sensors, albeit at a restricted range (Deng et al., 2015). Acoustic signals such as sonar are much lower frequency than IR signals which allows them to propagate much further through water. Sonar sensors can reliably detect objects at range and this leads to the widespread implementation of these sensors in AUVs for applications such as obstacle avoidance, localization, and mapping (Teo, Ong, and Lai, 2009; Mallios et al., 2010). Acoustic sensors are especially useful for underwater localization tasks where technologies that rely on radio frequencies such as GPS cannot be used. Instead AUVs can perform localization using underwater positioning systems (UPS) which use beacons on the surface of the water to determine the relative position of a robot using trilateration (Tan, Diamant, Seah, and Waldmeyer, 2011). Recent studies have also demonstrated the benefit of combining acoustic sensors with cameras (Evans et al., 2003) showing that machine vision techniques and sensor-fusion can accomplish more accurate mapping and feature detection – ideal for visual applications such as autonomous docking or underwater inspection (Hover et al., 2012).

Using acoustic sensors to detect the presence of objects can allow AUVs to perform obstacle avoidance, but they also allow for indirect communication between robots. Stigmergic communication is a powerful tool that can be exploited to produce formation control or obstacle avoidance behaviours simply from detecting environmental changes made by other robots (Dorigo et al., 2006), or inferred positions of other members of the swarm (Balch et al., 2000). Some complex behaviour, such as self-assembly, are at present too complex to be achieved with indirect communication alone and require a more direct method of communication. However, direct communication is not necessary for the CAC algorithms to function and so discussion of this can instead be found in Chapter 4 where exploitation of direct communication for self-assembly and signalling is addressed more fully.

The existing AUVs identified in the preceding section reveal that there are several robots capable of performing some of the four major tasks necessary to carry out CAC algorithms. However, none of the robots identified in the literature possess a geometry which would allow for self-assembly – a fifth ability which is required for the formation of larger water-tight structures. The ability to self-assemble is not necessary

for the CAC algorithm, but is of paramount importance to the success of the complete emergency ship hull repair method and failure to account for this necessary quality in the early stages of design could cause issues later in the design process. While there is no existing robot that possesses all of the necessary qualities required for the CAC and self-assembly experiments, the varied abilities of existing models arguably shows that a robot capable of performing all of the tasks should be possible to design and construct. For this reason, the CAC algorithms were chosen to be examined in simulation using robots with a bespoke design, with a geometry that would allow for self-assembly, and based on existing technologies demonstrated in current AUVs.

### Section 3.2.2 Simulated Robot Design

The morphology of the simulated robot was determined according to the requirements identified in section 3.2.1 and focuses on the following aspects: robot hull shapes that would allow for self-assembly, methods of propulsion that allow for high manoeuvrability and buoyancy control, and sensors that accurately and reliably retrieve information about the environment and allow for indirect communication between robots. One of the most limiting factors of the robot design was hull shape. Typical AUV hull designs tend towards torpedo shapes due to their many advantages with respect to high manoeuvrability and low drag generation, but formation of larger water-tight structures can hardly be achieved with such irregular shapes. Therefore, simpler geometries such as triangles, squares, or hexagons may be a more appropriate option as these are more commonly selected to carry out self-assembly in existing ground and air robot systems as discussed in Section 2.6.2. A square geometry was ultimately selected due to the ability to more easily incorporate multiple propellers and water jets for positional control, and to simplify the design of the self-assembly algorithms addressed in Chapter 4. Figure 3.2 illustrates the square structure of the simulated robot with simple representations of each of its sensors and actuators – details of which will follow.

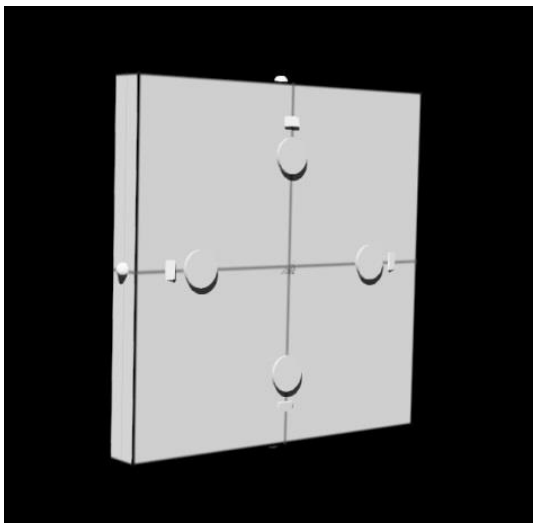


*Fig.3.2. shows a visual representation of the simulated repair robot used in CAC experiments. The larger circles represent the four enclosed bi-directional thrusters, the smaller rectangles bordering the thrusters represent the distance sensors, the small square in the centre of the robot represents the forward-facing camera, and the semi-spheres on each of the smaller sides of the robot represent both the sonar sensors and water-jets to control distance from other robots and obstacles.*

The simulated robots measure 50cm×50cm×5cm; a similar scale to that of larger ariel quadcopters which served as inspiration of the propeller arrangement and control scheme discussed later. This scale makes the implementation of embedded electronics and mechanical parts more feasible than using smaller casings but decreases it resistance to shear stress from ocean currents due to its larger cross-sectional area. However, propellers powerful enough to compensate for these increased stresses can be implemented, allowing it to maintain its position even in the presence of greater forces exerted on the robot. The selected scale also allows for the use of mechanical links with greater strength and size, to be used between robots for linking together in the self-assembly process. Using relatively large robots reduces the total number of robots necessary to form a repair patch of adequate size, which means the total number of mechanical links between agents is less than a larger swarm of smaller robots, which in turn reduces the likelihood of linkage failures occurring.

The robots used to cover the hull breach should not be too large or singular for several reasons. For instance, if a single very large robot were used to address the repair and the unit was to malfunction, the repair process would fail making the system less robust than using multiple agents. A singular robot would only be able to address breaches of a given size and shape, as opposed to swarms of robots which are able to

scale their approach to repair breaches of any reasonable size and shape. Using multiple robots with semi-flexible casing would also allow the resultant repair patch to better conform to the shape of ship hulls which are typically curved. A robot diameter of 50cm was selected because a collection of 20 robots with these dimensions would be adequate to repair breaches measuring up to 2m×2m. A single hull breaches that measure more than 2m×2m on a standard bulk carrier ship which measures 100m×80m×10m would be considered unsalvageable by conventional methods, so designing the system to try and repair damage of a much larger scale could be considered unrealistic. There is a limit to the amount of damage a structure can suffer and still be considered salvageable, and the approaches discussed in this thesis are only intended to repair damage that falls within these limits.

The simulated robots use a combination of propellers and water jets to control the position of the robot underwater, which were selected due to their proven reliability. Similarly, to unmanned air vehicles (UAV), vehicles using rotary blades typically require three or more propellers to control their position and orientation. The simulated robot was designed to operate using four bi-directional propellers which are primarily sued to maintain a set distance from the ship hull during inspection. The decision to use four propellers was inspired by the methods of movement employed in quadcopter UAVs which commonly use a combination of four or more rotary blades, and sophisticated controllers to control their position and orientation (Quan, 2017). Unlike the quadcopter however, the simulated robots will be operating in a denser fluid environment which must be accounted for in the control strategy and selecting an appropriate method of buoyancy control can help compensate for these additional constraints. To this end, the simulated robots are assumed to possess passive neutral buoyancy which allows them to stay submerged at 2 meters below the surface of the water without needing to engage its actuators to maintain this depth. Using a passive buoyancy system, rather than an active system allows for a simpler design of the robot and its controllers.

The four propellers and the neutral buoyancy of the robot serve as sufficient actuation for most tasks requiring underwater manoeuvring, but to increase its ability further the simulated robot also possesses internal pumps to create water jets. These pumps push and pull fluid through the main body of the robot via connected channels that

run between the four faces of the robot with the smallest cross-sectional area, allowing it to further control its movement when the four main propellers are working to maintain a set distance from the ship hull. This method of routing water through the body to generate thrust is known as hydraulic propulsion (Doyle et al., 2016) and is illustrated in Fig. 3.3.

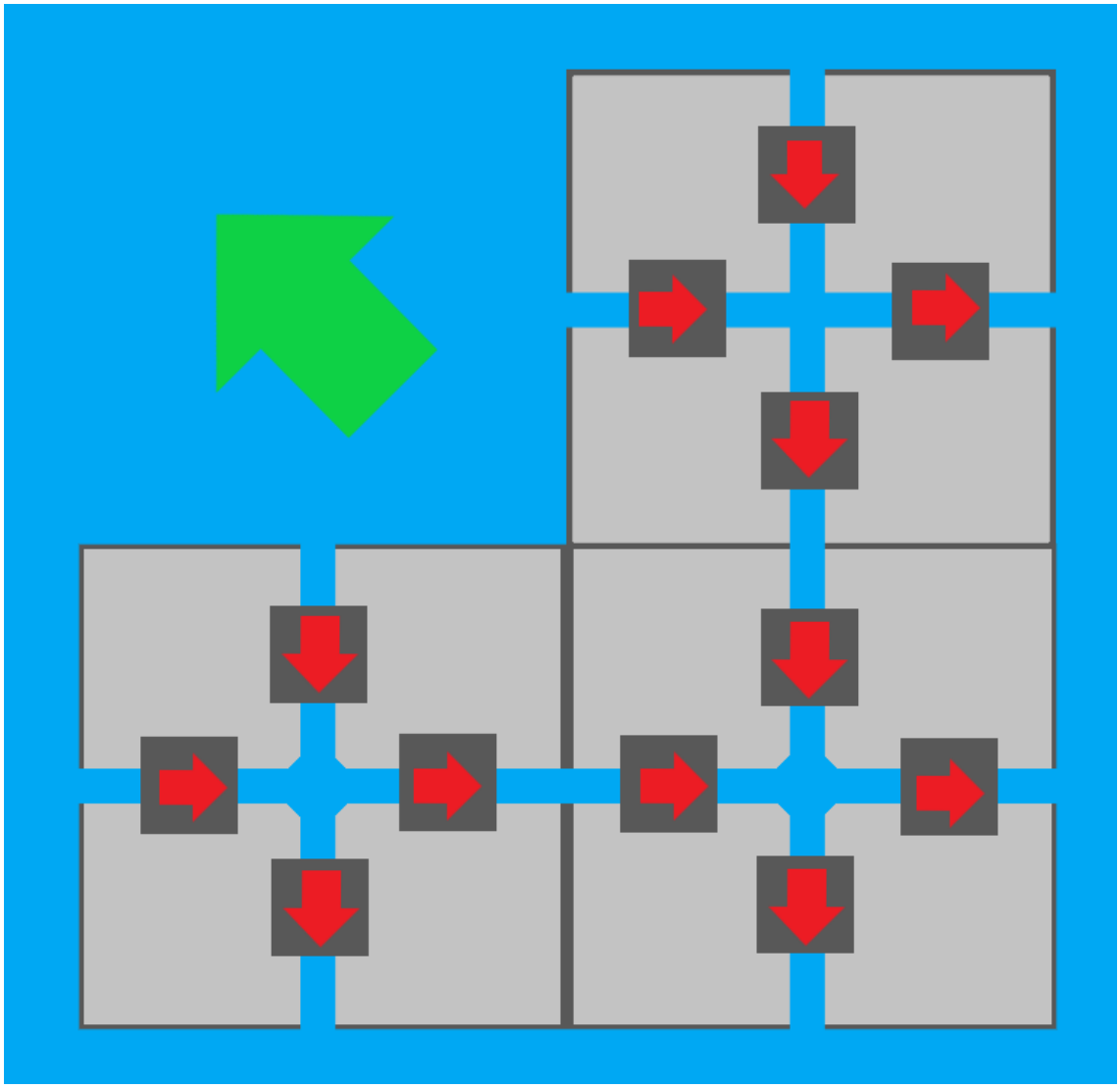


*Fig.3.3. Three connected robot modules creating a network of internal pumps showing the resultant motion of the structure from different pumps being activated. The red arrows indicate the motion of the fluid pushed through the robot using the pumps, and the green arrows indicate the resultant direction of motion of the structure.*

One of the advantages of this method, as demonstrated by Doyle et al.'s modular hydraulic propulsion robots, is that it can continue to function when robot modules become connected to each other. When two or more robot modules connect to each other as shown, their channels become linked and the hydraulic propulsion of each

module continues to function, strengthens the connection between modules, and influences the resultant direction of movement of the new robot configuration. Due to the dimensions of the robot selected earlier in the design process, the size of the motors that could feasibly be implemented to enable these methods of movement is also limited. Thus, the potential maximum speed of each robot is restricted to less than or equal to 5 m/s in any direction the water inlets are facing.

The selected hull shape and methods of propulsion provide an appropriate base for the repair robot, but in order to carry out the inspection in earnest the robot requires methods of sensing its surrounds so it can infer its position in the environment. In section 2.5.2 one of the more common methods of localisation for swarm robots was identified – the use of beacons or landmarks. The purpose of the CAC algorithm is to carry out inspection of a ship hull, and this is something we can exploit to our advantage by selecting the ship hull as a known landmark in an otherwise boundless environment. To enable a positive identification of the ship hull, the robot is fitted with a forward-facing camera close to the centre of the robot, and four infrared distance sensors that border the four propellers as described in Fig.3.2. Section 3.2.1 discussed how IR sensors are ill equipped for long-range underwater sensing but can function adequately over short ranges such as the 2 meters distance the robots will be working to maintain from the ship hull.

The simulated robot also possesses four sonar sensors; one on each of the 4 faces of the robot with the smallest cross-sectional area, which it can use to confirm the location of neighbouring robots and potential obstacles in the water, such as debris, seaweed, or moving sea creatures. The robots use these proximity sensors to detect when another robot is in range of the sensor closest to the waterline, and in range of the sensor furthest away from the waterline ensuring that it always remains in contact with these other modules throughout the inspection process. Each robot works to equalize the measured distance between itself and its two closest neighbours using virtual forces techniques much like those discussed in sections 2.5.1 and 2.5.2. The robots use the error between their two opposing proximity sensor readings to affect the magnitude and direction of their internal pumps which control the position of the robot using hydraulic propulsion, enabling effective pattern formation and control. By ensuring each robot follows this protocol, the formation of robots shown in Fig. 3.4 can

be created and maintained, enabling a more complete examination of the ship hull to be performed.

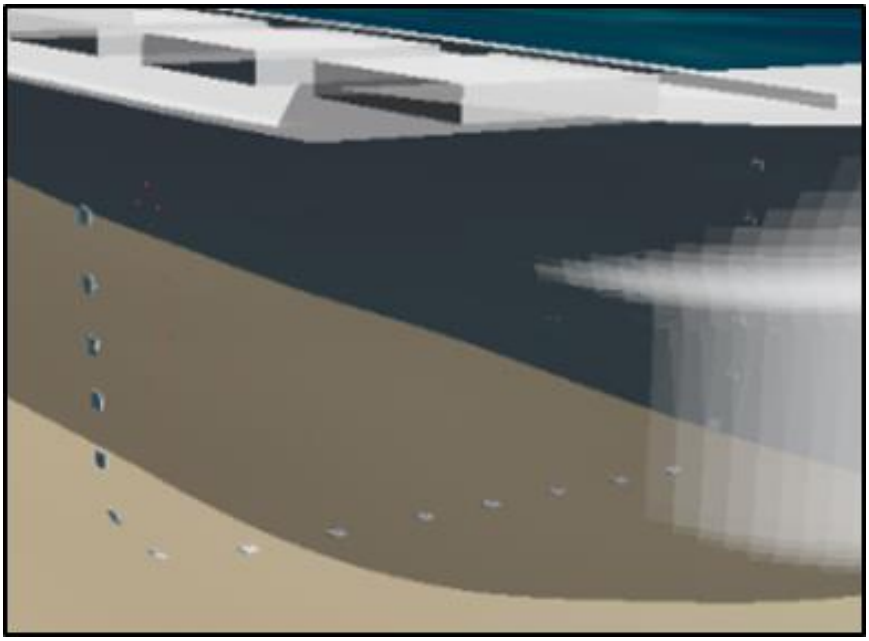


*Figure 3.4. Optimal configuration of the swarm of repair robots for conduction ship hull inspection. Each robot works to maintain this formation as it conducts the inspection by first positively identifying two robots either side of itself and moving to equalize the distance between the two. The exception to this rule is the robot closest to the waterline, which always works to stay with 1 meter of the waterline.*

The main method of inspection of the ship hull is to be performed by a forward-facing camera and accompanying light which sits close to the centre of the largest face of the robot. There are a number of aspects to consider when choosing an appropriate camera including lighting, camera field of view, pixel resolution and the subject of the images captures, and whether it is stationary or in motion. Each of these features can be used to dictate the type, resolution and size of camera that would be most suitable for the job. However, the visual computation method used to detect defects and deformations in the ship hull lies outside the scope of this study on complete area coverage. Instead, the field of view of each robot (2mx2m) is the main metric used to determine if sections of the ship hull are being inspected by more than a single robot, which we can use to discern if the area coverage of the ship hull is complete or not. However, there are examples of machine vision techniques that have been successfully applied to autonomous underwater vehicles with limited computational power to enable visual detection and feature recognition. For instance, edge detection and line

extraction which are commonly employed in machine vision and work well when the robot has access to information about the structure being imaged, which could be suitable for this scenario where the object of inspection is known to be a ship hull. Gamroth (2010) demonstrated how automatic detection and tracking of man-made objects in subsea environments can be achieved with such techniques in the presence of marine snow and poor visibility.

In simulation, the distance sensors the robot is equipped with are visualised as red dots wherever they intersect with the ship hull, and the forward-facing cameras can detect this. This information is used to confirm the overlap of camera fields of view between two or more robots inspecting the ship hull near one another as shown in Figure 3.5. If the camera detects more than four red dots, this indicates that the extra red dots are from the distance sensor of another neighbouring robot. Each camera has a field of view that allows it to examine a $4\mathrm{m}^2$ section of the ship hull at any one time while maintaining a distance of 2 meters from the ship hull in accordance with the constraints enacted on the mobile robots carrying out the inspection.

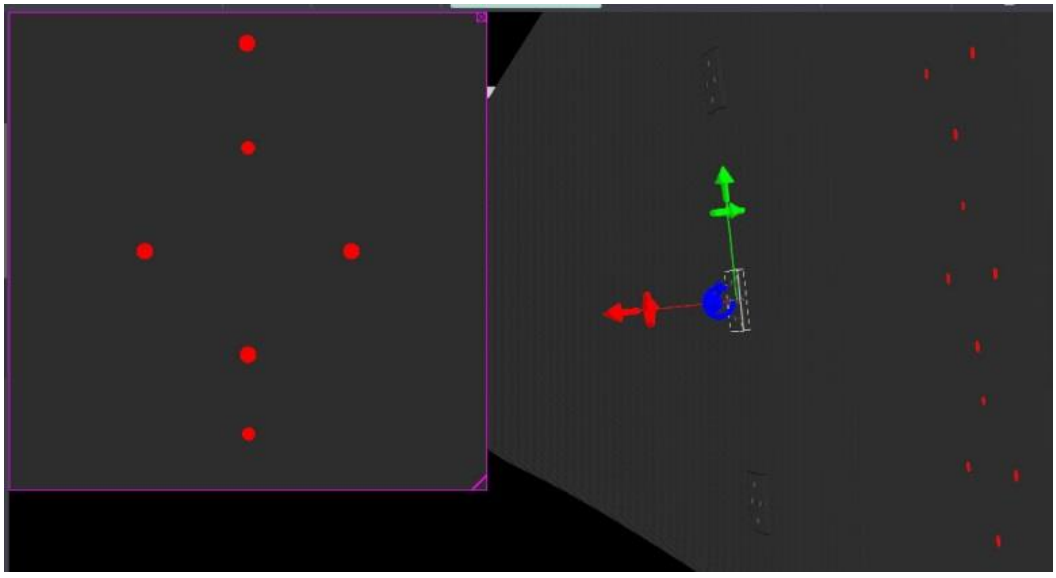


*Fig 3.5. Overlap of camera field of view, signified by the presence of more than four simulated red dots, from the robots' distance sensors, on the camera image.*

With appropriate sensors and actuator selected, the simulated robot now possesses the ability to move freely underwater and sense its surrounds – all of which are necessary for carrying out the CAC algorithms in this study. In addition, it possesses capabilities that will allow for effective self-assembly behaviours to be implemented

such as direct communication over short ranges, interlocking mechanisms, and an appropriate hull geometry which are discussed further in Chapter 4. The next section delves into the methodology and discusses how these functionalities are utilised to perform a ship hull inspection using CAC algorithms.

## Section 3.3 Ship Hull Inspection Methodology

The proposed CAC algorithms are designed to work on multi-robot systems in which each individual robot possess the same morphology and programming. As described previously in section 3.2, the robots' modules are capable of moving freely underwater in any direction, but to simplify the search algorithms and the subsequent controllers for the robots a constraint was added to the robots' working-space with respect to the ship hull. This constraint compels the robot to use its four forward-facing distance sensors and corresponding bi-directional propellers to maintain a set distance of 2 meters from the ship hull, as illustrated in Fig. 3.6. Ensuring the robot stays aligned with the ship hull reduces the chance that the robot will lose contact with the target and its neighbours while allowing for simpler control schemes to be considered. More discussion on the benefits of this additional constraint and how it affects the outcome of the simulated experiments is presented in Section 3.6.

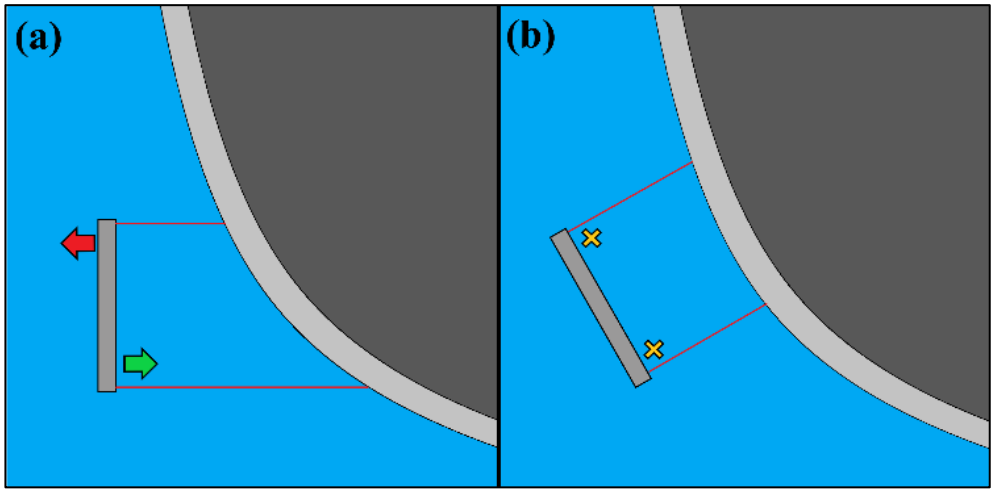



*Fig 3.6. Robot works to achieve a set distance and orientation to the ship hull, by maintaining equal readings on their four forward facing distance sensors. (a) The forward-facing distance sensors detect a difference in measured distance from the ship hull, indicating the robot is not parallel to the hull as required, and (b) The bi-directional propellers have adjusted the magnitude and direction of their thrust to equalise the distance sensor readings, giving a better indication that the robot is more parallel to the ship hull than was previously recorded.*

The PID controller as shown in Fig. 3.7 and described in Eq. (3.1) was implemented on each of the four forward-facing propellers individually, the collective result of which enables the robot to align with the ship hull as parallel as possible. Each propeller generates the most appropriate direction and magnitude of thrust using an error signal which is determined by the difference between the desired distance between the propeller and the ship hull (2 meters) and the measured distance from each propeller's corresponding distance sensor. There are many closed-loop controllers that could have been used to achieve the desired set point, but the classical PID controller was chosen because it is well understood, has been proven highly reliable in the control of motors and positioning (Åström, Hägglund, and Astrom, 2006; Visioli, 2006), and continues to prove successful in recent applications to AUV control (Khodayari and Balochian, 2015; Sarhadi, Noei, and Khosravi, 2016).

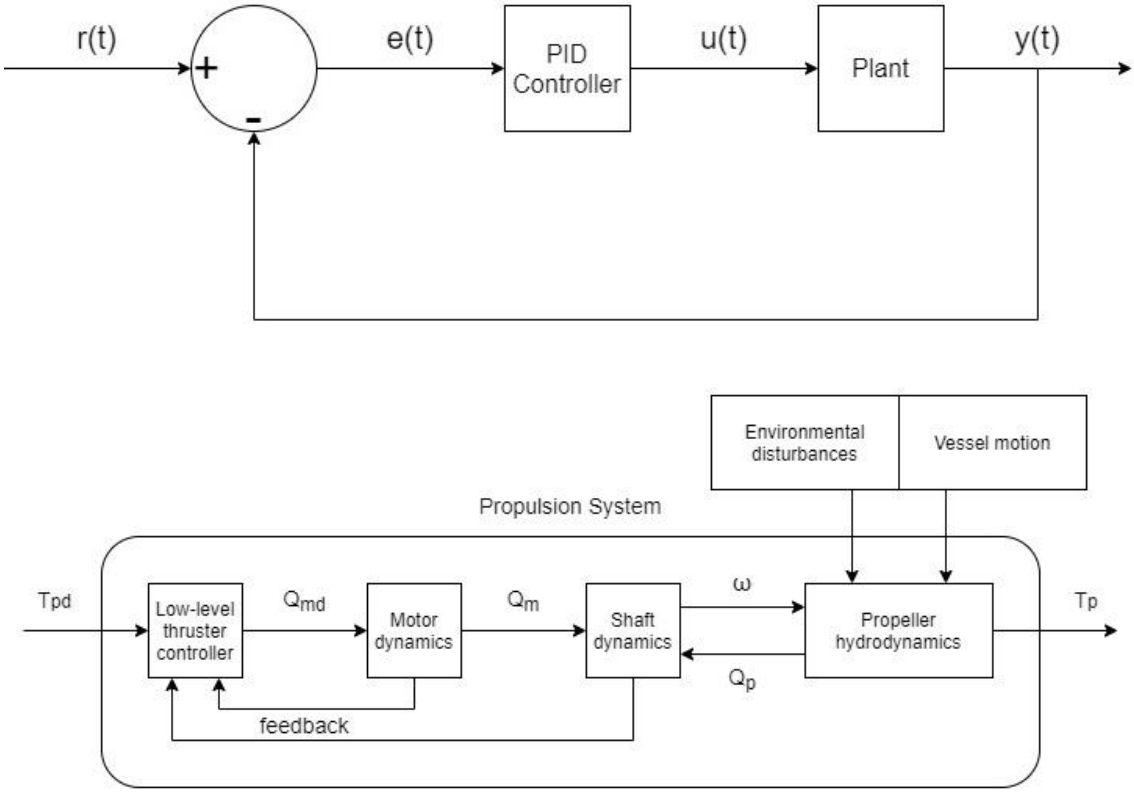



*Fig.3.7 Block diagram showing (a) the PID controller implemented on forward facing propellers of the robot to control its distance from the ship hull, and (b) the internal working of the plant Smogeli, (2006).*

The desired set point $r(t)$ of the controller represents a distance sensor reading of 2 meters between the sensor and the ship hull. This set point $r(t)$ is compared against the measured output $y(t)$ creating an error signal $e(t)$ which represents the difference

between the current state and the desired state. The PID controller applies proportional, integral, and derivative gains to this signal as described in Eq. 1 to create a control signal$u(t)$. This signal is then passed through the plant - composed of the motor, gearbox, and propeller - to affect the speed of rotation and resultant position of the robot, changing the measured output signal $y(t)$ from the corresponding distance sensor, which is fed back to the comparator to generate a new error signal$e(t)$.

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt} \tag{3.1}$$

*Where $u(t)$ is the control variable, $e(t)$ is the error between the desired set point and the measured output, and $K_p$, $K_i$ and $K_d$ are the respective proportional, integral, and derivative gains (Visioli, 2006).*

The complexity of propulsion systems such as that of Fig.3.7 (b) and their design is not to be understated, with the many factors that affect the efficiency and effectiveness of the propeller performance that need to be accounted for. In this example from Sogeli (2006), $Q_m$ is the motor torque, $Q_{md}$is the desired motor torque, $\omega$ is the angular velocity of the propeller, $Q_p$ is the propeller torque, $T_{pd}$is the desired thrust, and $T_p$ is the actual propeller thrust. Comprehensive low-level system design involving such parameters falls outside the scope of this study, but has been examined extensively by other researchers investigating marine propulsion (Smogeli, 2006; Pivano, 2008). Instead, these aspects of the system will be explored in future studies concerning the use of propellers for positional control of AUVs in dynamic environments.

With the robots' movements constrained to maintain a set distance from the ship hull, the CAC algorithms become more comparable to those used to explore 2D spaces where only the XY-plane considered. Both CAC approaches described here after, operate using the same conditions described above, maintaining an equal distance from the ship hull as often as the controllers will allow. The two methods, referred to as the lawnmower search (uncoordinated) and the sweeping search (coordinated), are both designed to fully inspect the middle section of a ship hull. Studies on area coverage using multiple robots have demonstrated the benefit of minimising turns in such approaches (Vandermeulen, Groß, and Kolling, 2019), which indicates that the sweeping search should marginally outperform the lawnmower approach. The results

of the following experiments should confirm this while also indicating the effect of allowing coordination among the robots.

The lawnmower search (Fig. 3.8) is an un-coordinated complete area coverage method which can be used to measure the performance of a swarm of homogeneous robots where each robot operates independently of the actions of its neighbours. In this method, the robots are evenly distributed along one side of the vessel at the waterline, allowing for an initial overlap of their forward-facing camera field of view. Note that this initial even distribution of robots is not controlled by the robots themselves, but by the mechanism used to deploy the robots into the water from the side of the ship, and once deployed the robots do not communicate with each other.

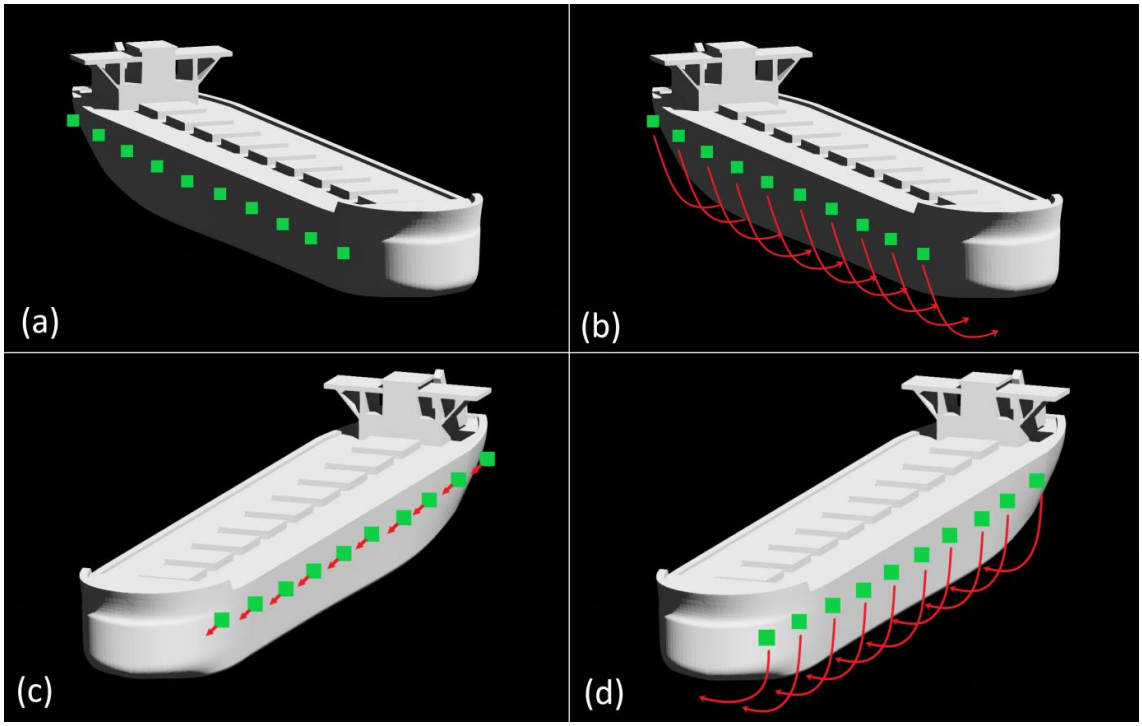



*Fig.3.8. the four distinct phases of the un-coordinated lawnmower search pattern with robots represented by green squares and movement pattern represented by red arrows. (a) Shows the initial distribution of the robots, (b) shows their direction of movement for the first pass under the ship hull, (c) shows the lateral movement of 2m, and (d) shows the next pass back under the ship hull.*

Each robot performs and individual search in a straight line that stretches under the vessel until the waterline on the other side of the ship hull has been reached. Once this point has been reached, the robot will turn and move parallel to the ship hull for 2 meters (half the width of its cameras field of view). The robot then completes the initial pattern by performing the same straight line search under the vessel once more

until the original side is reached. This pattern then repeats until the entire hull has been examined. In this approach sensors are only used to maintain a set distance from the ship hull, inspect its condition with the camera, and perform basic obstacle avoidance to prevent collisions and ensure completion of the search – formation control between robots is not used.

The sweeping search (Fig. 3.9) is the coordinated approach which is intended to outperform the uncoordinated lawnmower search in terms of time to complete the search, and robustness to sensor noise or population failure in agreement with the findings of section 2.5 of the literature review. The robots are initially evenly distributed underneath the vessel, forming a line that follows the curvature of the hull. The search stretches the length of the ship hull from front to back and terminates once the main body of the ship hull has been examined. Although this approach starts with a different initial configuration, the main distinction of this approach is that the robots are instructed to stay within sensor range (4 meters) of one another while performing their search of the ship hull. The robots take measurements of the distance themselves and their two closest neighbours using their proximity sensors on opposing sides. This data is passed through a PID controller Eq. (3.1) to minimise the difference between these two values, which would indicate an equal distance between the robot and each of its neighbours has been achieved. The maximum allowed space between each robot is defined by the point at which the overlap of their forward-facing camera field of view falls to zero.

In the lawnmower approach, coordinated motion is achieved using a method of formation control which enables each robot to set the direction and velocity of its internal propellers, responsible for moving the robot about the x-y plane, using hydraulic propulsion. The direction and velocity of these propellers are determined using the readings from the proximity sensors which measure the difference in distance between its two closest neighbours, as described in Eq. (3.2).

$$e(t) = \min\{PS_L, 4.0\} - \min\{PS_R, 4.0\} \tag{3.2}$$

The error value (e(t)) is generated by subtracting the minimum distance measurements of its left proximity sensor ($PS_L$), and its right proximity sensor ($PS_R$) which indicates whether it needs to move closer or farther away from its respective neighbours. A

minimum is used to limit the speed at which the robot moves to equalise the distance between itself and a missing neighbour, as using the maximum range of the sensor instead of a cap of 4.0 (representing 4 meters) could cause the robot to accelerate faster than is desired, which could cause collisions with newly discovered neighbours. Negative error values indicate the robot would need to move closer to its neighbour on the left, while positive error values indicate that it would need to move closer to its neighbour on the right. Passing this error value though a PID controller, as described in Eq. (3.1) would allow the robot to safely equalize the distance between its neighbours at a controlled speed, forming a more stable formation.

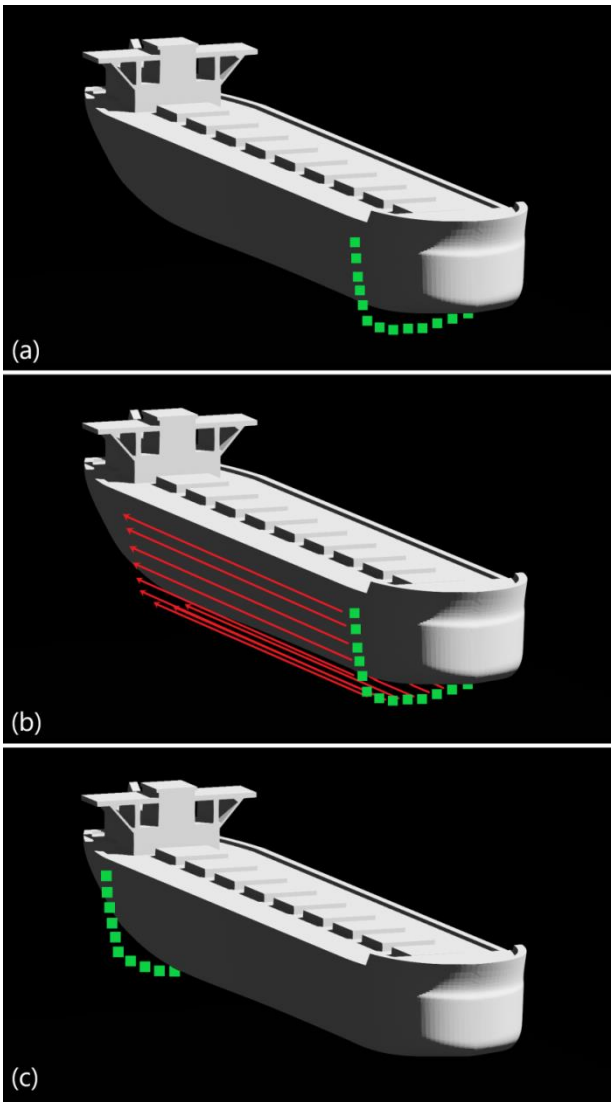



*Fig.3.9. the three distinct phases of the coordinated sweeping search pattern with robots represented by green squares and movement pattern represented by red arrows. (a) Shows the initial distribution of the robots, (b) shows their direction of movement for their pass under the ship hull, and (c) shows their final distribution following a successful inspection.*

In scenarios where the robot has lost sight of one of its neighbours – for instance when a robot has broken down - Introducing the cap of 4.0 in the error calculation also helps prevent the robot from losing contact with the neighbour that is visible while searching for a neighbour on its unoccupied side. The robots are assumed to be capable of discerning when they are within 1 meter of breaching the waterline. When a robot detects this, it will ignore a lack of neighbours closer to the water line that themselves and work to maintain a set distance of 1m below to waterline, serving as one end of the line formation of robots under the hull.

## Section 3.4 Experiment Setup

Webots is the simulation software that was selected to carry out the ship hull inspection experiments. This allowed for the creation of more realistic models of the swarm of swimming robots, the underwater environments, and the ship hull to be inspected. The experimental setup for the lawnmower and sweeping search experiments is kept relatively simple by modelling only the ship hull, the robots, and the fluid environment, but omitting the inclusion of additional obstacles. The body of water was modelled with a high clarity to ensure that the image quality of the robots forward-facing camera was not impacted by anomalies such as mud, oil, or other impurities. In order to assess the performance of the system under ideal conditions, fluid qualities such as turbulence, complex currents, and tides were not initially modelled and instead a static body of water is used so that only the viscosity of the fluid and the buoyancy of the robots are considered. Exactly how the results and system performance are expected to change when implemented in a turbulent environment is a subject which has been delegated to future experiments. The simulated ship used in the experiments is that of a bulk carrier ship, the second most common sea faring vessels used in international shipping of dry cargoes with a high weight to cost ratio such as coal, grain, and ore (Global merchant fleet - number of ships by type 2019 | Statista, 2020). The ship hull inspection technique discussed in this study could also be applied to the more common general cargo tankers, but the bulk carrier ship hull was selected because these types of ship typically carry more valuable cargo and as such are at the greatest risk of loss. The scale of the modelled bulk carrier ship is relatively small, measuring 100m×8m×10m in length, height and width respectively (Fig. 3.10). If a different size of vessel or a ship with a wholly

different shape of hull were to be used the CAC algorithms for inspection should in theory not need adjustment. However, the number of robots deployed to conduct the search may need to be increased or decreased to avoid sparsity or overcrowding of robots carrying out the work. Given the size of the simulated ship, the total area of the hull section to be inspected, and the maximum size of breach that could be consider salvageable (as discussed in Section 3.2.2), a population of 20 robots with dimensions of 50cm×50cm×5cm per module and an average field of view of $4\mathrm{m}^2$, would be sufficient to achieve complete area coverage and conduct subsequent repairs by following the ESHR approach outlined in Section 3.1.2.

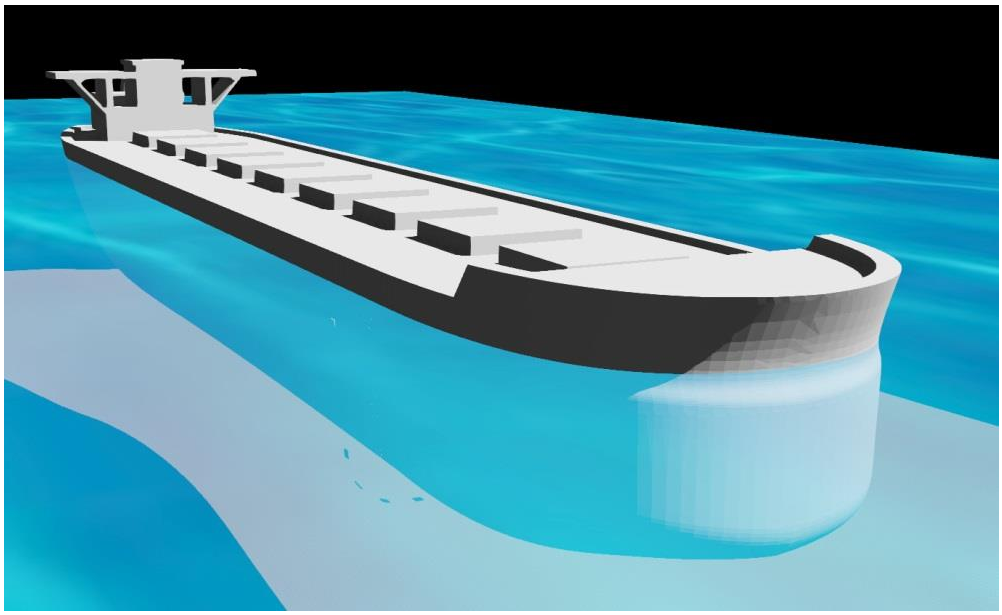


*Fig.3.10. Three-dimensional model of the bulk carrier ship used in the ship hull inspection simulations with fluid environment colorized.*

In the experiments, the ship's propellers and thrusters are not activated so that the ship remains in place, simply floating in the body of water to allow the robots to conduct search. The ship sits very low in the water so that the majority of the hull is submerged, which would be the case if the ballast tank of the vessel was full. In addition, since both methods of ship hull inspection are intended to examine a ship which has taken on additional water, the hull is submerged even further to the point where only a meter of the hull section sits above the surface, allowing realistic simulations of scenarios where the ship is holding even more water than the ballast would allow. This configuration represents the largest area to be examined and the worst-case scenario before the ship begins to sink in earnest.

The two methods of ship hull inspection described in section 3.3 as the un-coordinated lawnmower approach and the coordinated sweeping approach are tested, and three scenarios are examined for each of the two methods in environments with no additional obstacles. The first scenario tests the system performance in an ideal setting, where every robot in the swarm is fully functioning throughout the experiment and there is no excessive noise or errors present in any of the sensor measurements or camera images. The second scenario examines the performance of each system in the presence of some sensor noise which is evenly distributed among the distance sensors used to maintain a set distance from the ship hull. Noisy reading from sensors such as the IR devices discussed in section 3.2 are a common occurrence in underwater robotics applications and so the forward-facing IR sensors of the simulated robots are modelled with additive white Gaussian noise (AWGN) which is fortunately an available method of modelling noisy sensor reading in the Webots simulation software. AWGN is a basic noise model used to mimic the effects on signals caused by random processes that occur naturally - such as the temperature and clarity of the water or the intensity of ambient light - and is added to any noise that may be intrinsic to the sensor model. The noise is modelled with standard deviations of 5%, 10%, and 15% respectively, as shown in Figure 3.11.

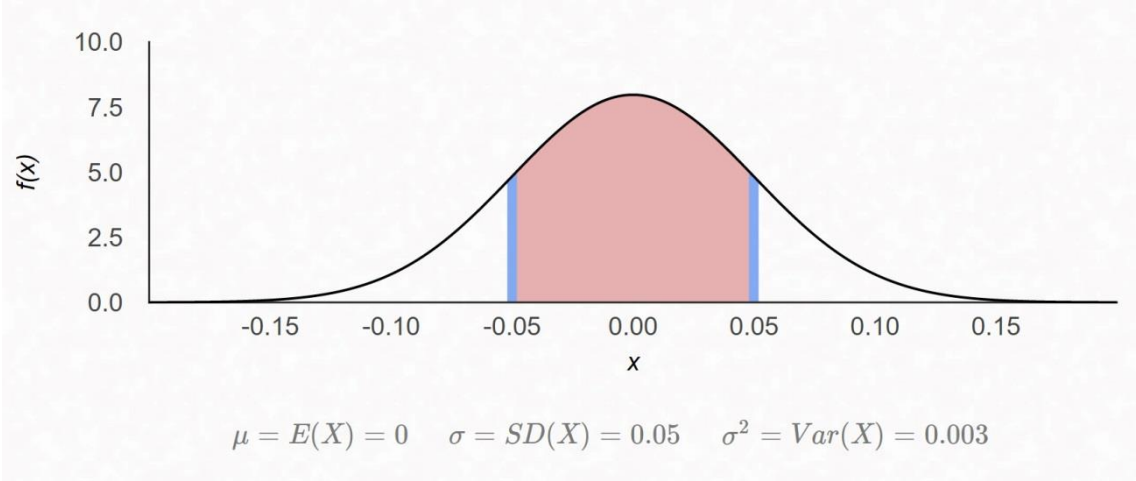



*Fig.3.11. Additive White Gaussian noise (AWGN) with a standard deviation (σ) of 5% added to the distance sensor values to examine how the system functions in the presence of noise. The highlighted red section identifies the range within which the majority (68.2%) of noise values will be generated. Standard deviations of 5%, 10%, and 15% are modelled in separate experiments, but following the same Gaussian distribution curve as shown.*

The third scenario tests the system performance with ideal sensor readings, but examines how the robots adapt when a percentage of the robot population completely fails and does not recover. Three experiments will be run per search method, where 5%, 10%, and 15% of the team of robot will be randomly selected to fail at different times during the experiment, at which point the functioning robots will instead treat the faulty robots as obstacles to be avoided. In such a scenario, the remaining robots must then either distribute the work evenly among the remaining robots by collectively filling the gaps that have formed, or delegate the work of each failed robot to its closest functioning neighbour.

The performance of the lawnmower and sweeping searches, in each of the three scenarios, is determined by the successful completion of the CAC task and three additional factors: field of view (FOV) overlap, FOV gaps, and completion time. One method of assessing the completeness of the CAC algorithms is to measure the total area of overlap and total area of gaps generated between the camera FOVs of each robot. If a single robot is tasked with inspecting an object but is operating with faulty sensors it may develop a false image of the target. However, if two or more units inspect a section of the same object and can reach consensus on their measurements, even if one is faulty this reduces the risk of taking erroneous readings or false positives as fact. So the more FOV overlap present, the higher the chance of observing the true state of a section of hull that is inspected. The quantity of FOV overlap and gaps generated are found by recording the global positions of each of the robots and the FOV measurements from the XY-plane, all of which are readily accessible through the simulator. These values are then applied to Eq. (3.3) where the total FOV overlap or gap can be calculated.

$$A_{fov} = \left(H_{fov} - \sqrt{\Delta x^2 + \Delta y^2}\right) \times \left(V_{fov} - \Delta z\right), \tag{3.3}$$

Where $\mathrm{A_{fov}}$ represents the area of overlap or resultant gap formed between two robots FOV. $\mathrm{H_{fov}}$ and $\mathrm{V_{fov}}$ represent the minimum xy-plane FOV measurements of the robot in question compared against its closest neighbour. Δx, Δy, and Δz represent the differences in position of the two robot cameras in the x, y and z coordinates, respectively.

The measurements and subsequent calculations from Eq. (3.3) are carried out for each robot and occur every 50cm that is traversed by the swarm. If the calculation yields a negative value, this indicates a gap has formed between the respective robot FOVs, while positive values represent an area of FOV overlap. Higher levels of overlap indicate a higher probability that the robots are observing the true state of the ship hull, contributing towards a more complete inspection. Gaps in FOV indicate sections of ship hull that have gone un-inspected and are thus counted as incomplete searches. Only approaches that do not generate gaps can be classed as complete. Figure 3.12 illustrates the concept of FOV overlap between two robots in the same arrangement that is used in the experiments.

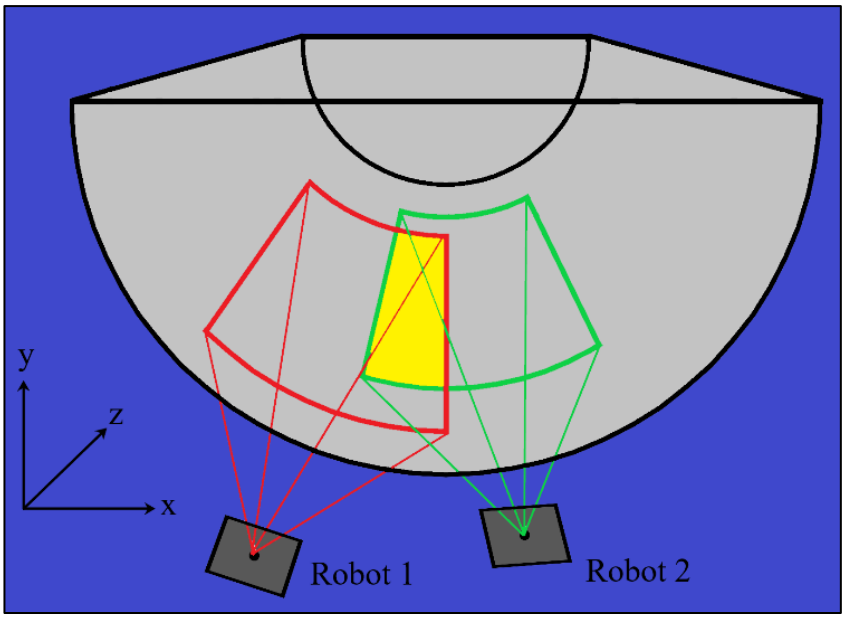



*Fig.3.12. Field of view overlap diagram. The image shows two robots angled towards a hollow cylinder, the field of view and camera frustums are drawn in red for robot 1 and green for robot 2. The yellow shaded section represents the overlap between the two robots' field of view.*

The final factor for measuring the performance of each approach is simulation completion time. Time is an important element of emergency ship hull repair as the longer a breach remains in disrepair, the higher the likelihood that flood boundaries within the ship will fail, leading to worse flooding and greater instability of the ship. The quicker the system can perform a complete search, the faster it can discover any potential hull breaches, and thus the more well suited it will be to forming part of the automated emergency ship hull repair system. These experiments should reveal the

approach which yields the fastest and most complete search of the ship hull, even in the presence of failed robot modules or erroneous sensor readings.

## Section 3.5 Results

In this section, the performance of the lawnmower and sweeping searches on a simulated generic cargo tanker are assessed in three separate scenarios: ideal conditions, sensors with 5%, 10%, and 15% additive noise, and 5%, 10%, and 15% population failure. Fifty separate simulations were carried out for each variable that was changed in each scenario for both approaches, resulting in a combined total of 700 simulations for the un-coordinated lawnmower search and the coordinated sweeping search. The results were compiled and compared in MATLAB to help identify the search method that yielded the quickest completion time and the most complete search. In all the figures shown forthwith, each bar represents the median result of the 50 simulations per variable change. Error bars are included indicating the maximum and minimum values obtained, except for Fig.3.13 in which the results are deterministic and therefore no variation in behaviour was observed.

### Section 3.5.1 Ideal Conditions

Figure 3.13 shows the comparison of results between the lawnmower and sweeping search in the ideal scenario, where all robot sensors operate without erroneous readings or noise, and none of the robot population fails throughout the simulation. Neither approach generates gaps throughout the experiment proving that complete area coverage has been achieved by both. It can be observed that the sweeping search achieves greater FOV overlap than the lawnmower approach, indicating that the sweeping search has a higher probability of observing the true state of the ship hull, providing a greater degree of certainty concerning the recorded sensor measurements. The largest distinction that can be observed between the two search methods is how the sweeping search takes less time to complete the same search while achieving a higher FOV overlap. These results are in agreement with other studies investigating the effect of turn minimising behaviour in robot teams for area coverage (Vandermeulen et al., 2019) further confirming the hypothesis that search methods with fewer turns typically result in fast completion times for area coverage. Under ideal conditions, the sweeping search seems to outperform the lawnmower search in terms of time to completion and FOV overlap.

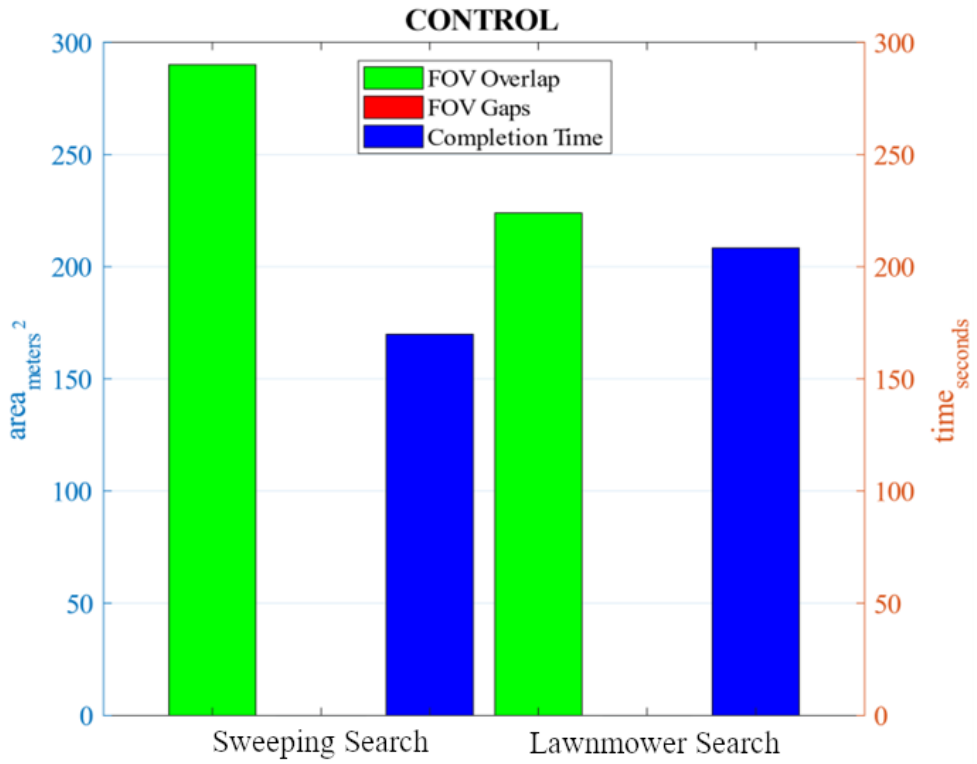



*Fig.3.13. Ideal scenario comparison for the lawnmower search and the sweeping search, where all agents operate with perfect sensor measurements and none of the population fails.*

### Section 3.5.2 Noisy Sensor Measurements

Figure 3.14 compares lawnmower and sweeping search performance in three scenarios where the forward-facing distance sensors of the robots return erroneous measurements due to the inclusion of additive white Gaussian noise (AWGN) with variances of 5%, 10% and 15%. Though errors are present, none of the population fully fails throughout these simulations so that each robot completes their individual search. From the results, it is clear that the lawnmower does not generate gaps, but the sweeping search does. This indicates that when significant noise is introduced to the distance sensors only the lawnmower approach achieves complete area coverage while the sweeping approach fails. The sweeping search yields greater FOV overlaps when less errors are present; however, it is quickly outperformed by the lawnmower approach once errors begin to build, and this performance gap widens when the magnitude of erroneous reading increases. In addition, the overlap generated by the lawnmower approach is more consistent, with low deviation across different percentages of error. The sweeping search may take less time to complete its search pattern than the lawnmower search, but because it has begun to generate gaps it can

no longer qualify as a complete search. This indicates that when erroneous sensor measurements are more prevalent, the lawnmower search is superior to the sweeping search. On completion of the initial experiments, several simulations were conducted with extreme erroneous sensor measurements of 30% which revealed that both systems quickly break formation and fail to complete the search. This demonstrated that while the lawnmower approach is more tolerant to sensor noise than the sweeping search, neither system is wholly immune.

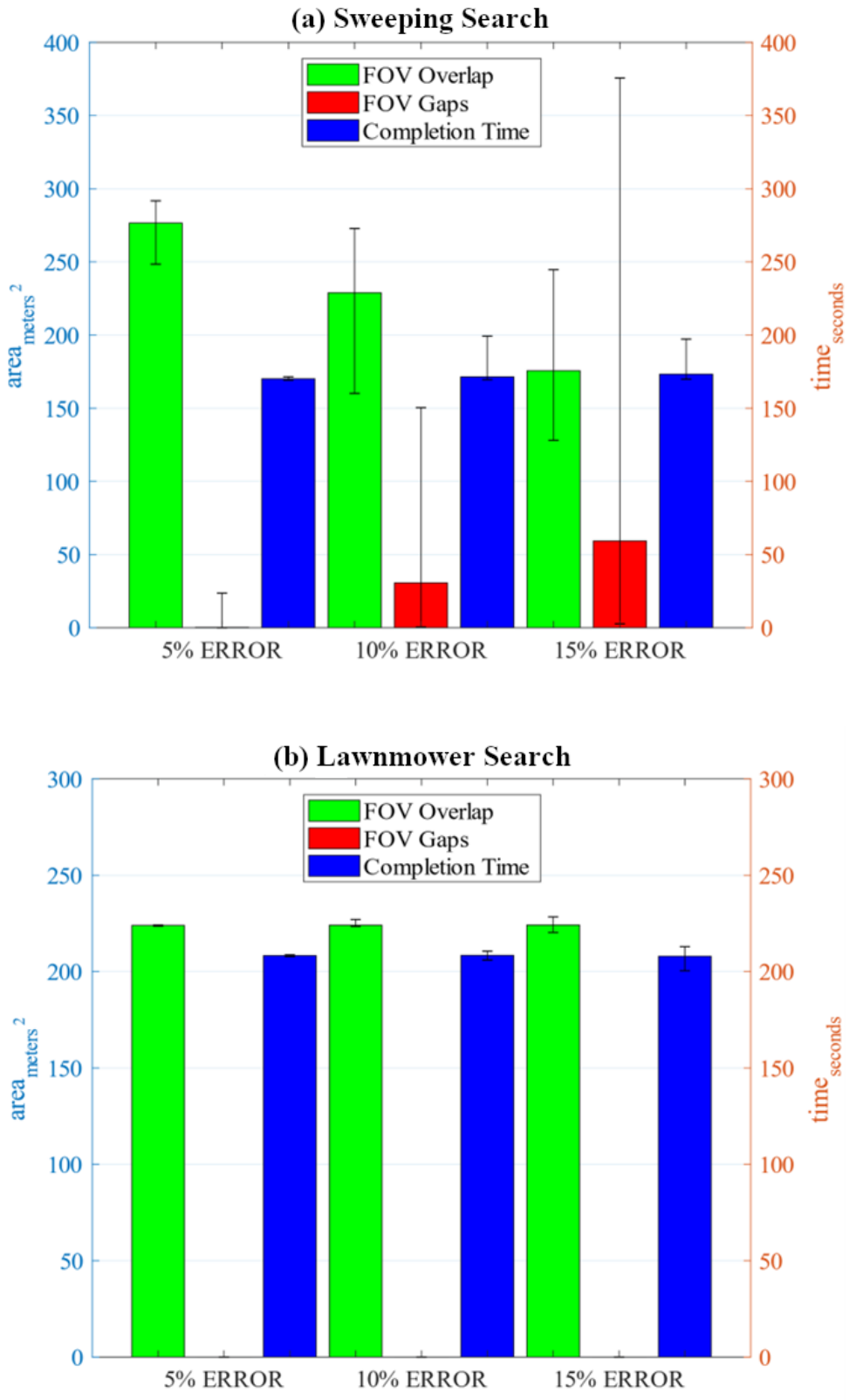



*Fig.3.14. Error Scenario Comparison; (a) the sweeping search (coordinated), and (b) the lawnmower search (uncoordinated). In these experiments, the robots operated with erroneous sensor measurements with variances of 5%, 10%, and 15%. None of the robots fully fail in this scenario but the noisy sensor readings have a negative effect on the robots' ability to complete their tasks.*

### Section 3.5.3 Partial Population Failure

Figure 3.15 compares the partial population failure scenarios for the lawnmower and sweeping searches. In these scenarios all robot sensors operate without erroneous measurements or noise, but a percentage of the population fails at a random time interval and does not recover, instead serving as obstacles that the remaining functioning robots must avoid. The robots are configured so that 5%, 10%, and 15% of the total population will fail in three respective separate scenarios. The results show that neither search method generates gaps in FOV which indicates that complete area coverage is still achieved in both cases. The lawnmower search achieves more cumulative overlap than the sweeping search, indicating that it has a higher probability of observing the true state of the ship hull. However, this comes at the cost of a significantly longer completion time than the sweeping search whose completion time is almost unaffected by decreases in population of up to 15%. In fact, the lawnmower approach takes approximately 5 times longer to complete its search than the sweeping search, but only yields 1.5 times the overall area of FOV overlap. This indicates that the uncoordinated lawnmower search is less efficient than the coordinated sweeping search at redistributing additional workload when part of the population fails, which is in agreement with section 2.5.1 where the benefits of distributed systems are discussed.

Full videos of these simulations showing the lawnmower and sweeping searches under ideal conditions can be accessed via the GitHub repository:

https://github.com/MattSHaire/Emergency-Ship-Hull-Repair.

Additionally, the code used to construct the Webots environment can be accessed via the same link, while experts of the programs used to control the robots can be examined in greater detail in Appendix A: Ship Hull Inspection Webots Simulation Code.

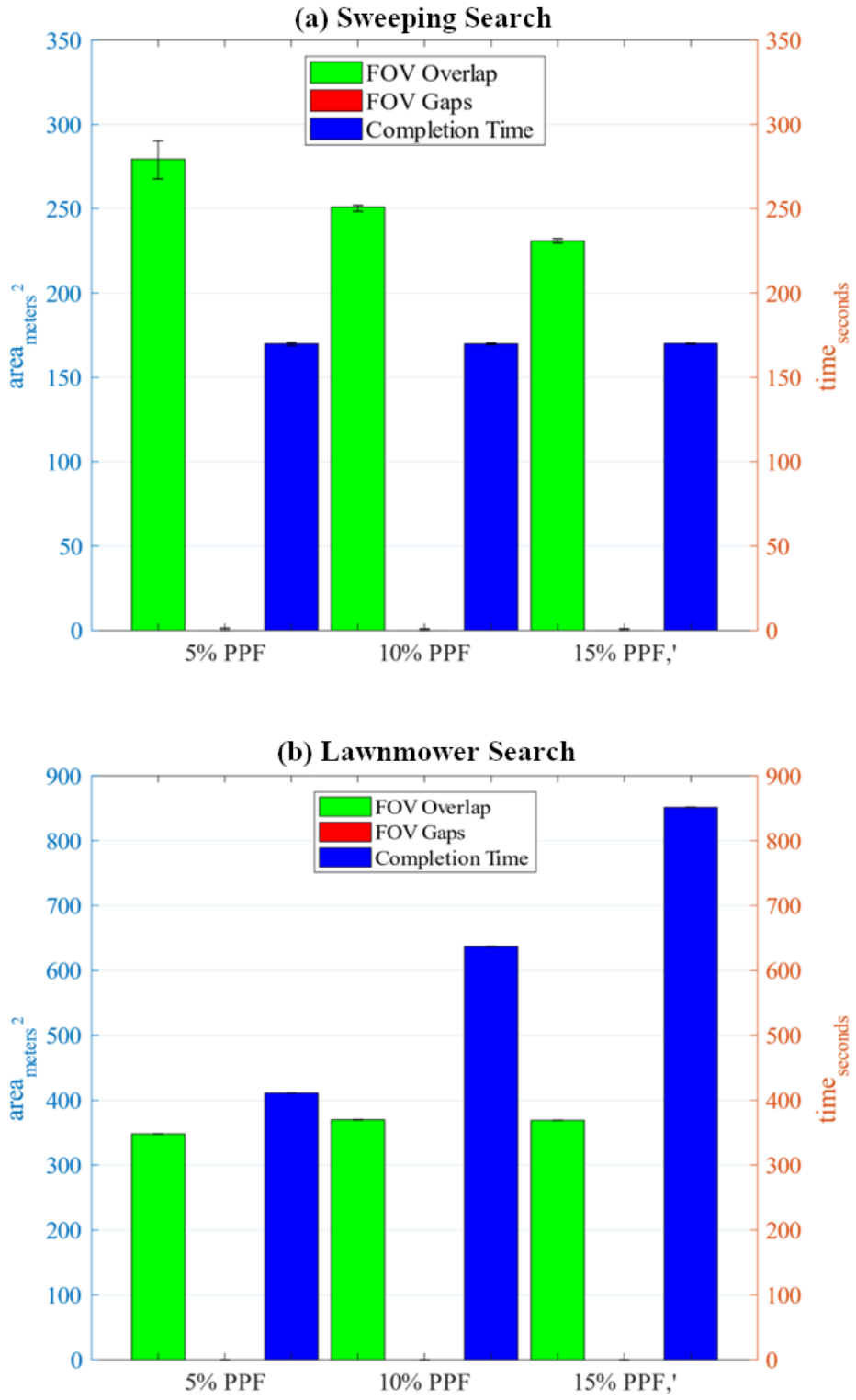



*Fig.3.15. Partial population failure scenario comparison; (a) the sweeping search (coordinated), and (b) the lawnmower search (uncoordinated). In these experiments, all the robots operated with ideal sensor measurements, but a percentage of the population completely failed and did not recover. Percentage failures of 5%, 10%, and 15% are shown which represent scenarios where one, two, or three robots completely fail at a random interval, and serve as obstacles from that point forward.*

### Section 3.5.4 Effect of Ship Size on Results

As mentioned in Section 3.4, these simulations were carried out in Webots with a bulk carrier ship hull measuring 100m×8m×10m serving as the object of interest. The algorithms deployed on the robots for maintaining a desired distance and orientation relative to the ship hull are designed to allow them to inspect various ship hull shapes without increasing the time of complete searches or reduce FOV overlap. However, using the same number of robots to inspect ship hulls of different lengths than our simulated model could significantly impact the results. For instance, if the circumference of the ship hull to be inspected was halved, such that the hull was narrower, and the same number of robots was used to perform inspection – this could decrease the completion time, but increase the risk of collisions between robots.

Similarly, if the circumference of the ship hull was double that of the simulated model; more robots would be required to achieve comparable completion times and FOV overlap to that of the results. Therefore, to achieve similar results recorded in these scenarios, Eq. (3.4) can be used to determine how many robots should be deployed based on the circumference of the ship hull, and the width of the robot FOV while the width of the robot's largest face is no greater than ¼ that of its FOV width.

$$N_R = \left\lceil \frac{W_H}{\min(H_{fov}, V_{fov})} \cdot 1.5 \right\rceil, \tag{3.4}$$

where $N_R$ is the recommended number for robots required for the insection of the ship hull, $W_H$ is the circumference of the ship hull in meters, and $H_{fov}$ and $V_{fov}$ are the horizontal and vertical dimensions of the robots FOV, respectively. The minimum of the FOV dimensions is used so that the robot can perform its inspection at any orientation. This result in multiplied by 1.5 to ensure adequate FOV overlap and is finally rounded to the smallest integer greater than or equal to the result to give the number of required robots. For example, the circumference of the simulated ship hull from the experiments was $26m$ and the $H_{fov} = V_{fov} = 2m$ which gives a result of 20 robots.

## Section 3.6 Discussion

It may seem as though the question of which method is superior under the three circumstances is rather clear, however there are additional considerations that must

be made for the sweeping search method. This is because as a coordinated method it relies on the accuracy of its sonar sensors on it to maintain contact with its neighbours. It was found that while many of the sensors in Webots quite accurately represent their real-life counterparts, the sonar sensors are regrettably less accurate. Rather than detecting objects that come within a given range of the sensor, in Webots sonar sensors are modelled as multiple laser rays emitting from a point, and objects are only detected when they intersect with these rays (Fig.3.16). This means objects that can fit between these rays can become more easily lost to the sensor, causing the robot to believe the object it was sensing has suddenly vanished. This is a scenario that frequently occurred in simulation when the robots turned at sharp angles while following the curvature of the ship hull. In these scenarios, the PID controller that works to equalize the distance measurements on each side of the robot quickly moves to regain equilibrium. When the robots fail to see each other in these scenarios they have a higher likelihood of colliding, which causes both agents to fall away from the arranged line, and ultimately fail to complete their searches.

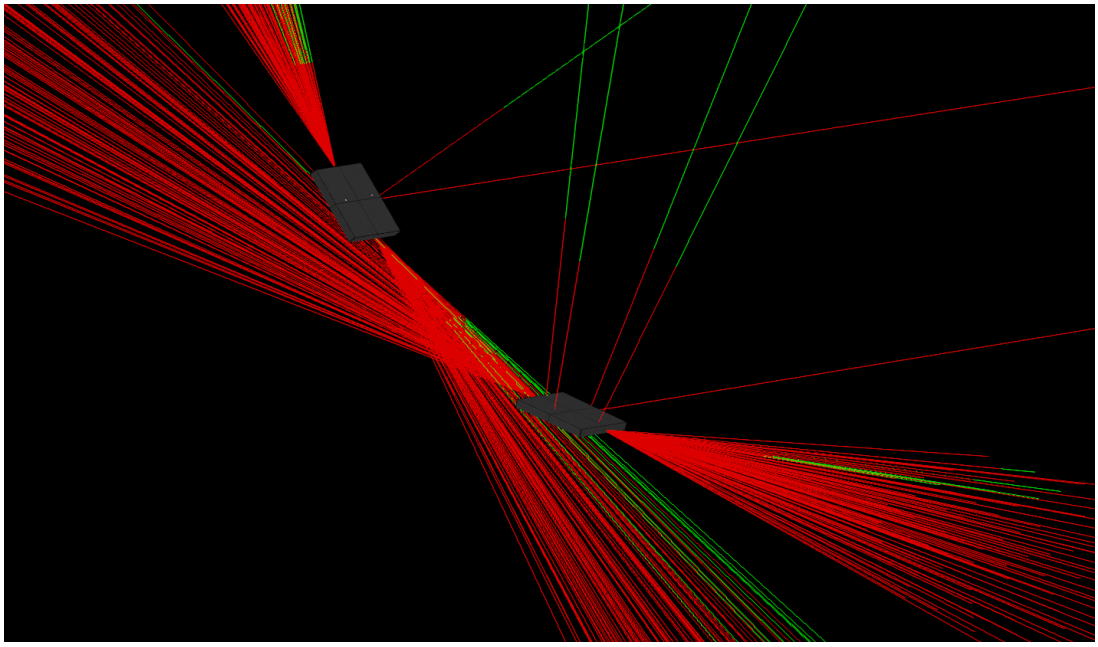


*Fig.3.16. Webots simulated representation of the sonar sensors in operation on two of the ship hull inspection robots. Multiple rays are used to represent the sonar sensors – red rays indicate a beam which has not detected an obstacle, but when a ray turns green this indicates it has passed through an obstacle. This image shows the common scenario where the robot on the upper left of the frame can see the other robot, but the robot of the lower right-hand side of the frame cannot see the robot on the upper left hand side of the frame.*

In the second set of experiments, where errors were introduced to the forward-facing distance sensors, the inaccuracies caused the robots to sway slightly, increasing the likelihood that neighbouring agents would fall between the already scattered sonar rays and become lost to the robot. To address this issue, a more accurate representation of the sonar sensors should be used in subsequent simulations to see how the system behaves without this unintended disturbance. It is reasonable to assume that a more accurate representation of sonar sensors would remove some of the instability that has caused gaps to form, and would further serve to reduce the reality gap between the simulated and real world systems. However, at present the lawnmower search appears to outperform the sweeping search in terms of robustness to sensor noise.

Despite these shortcomings, the coordinated sweeping search method was able to adapt well to losing a percentage of its population. In fact, when a higher percentage of the agents were removed the speed of completion appears to have slightly increased. This increase in speed could be due to how the area of the ship hull narrows towards the back of the vessel. When the area to be inspected becomes smaller, using larger populations of robots can lead to some overcrowding and begins serve as more of a hindrance than a benefit. Thus, it seems reducing the size of the population when approaching narrower section of the hull increases the speed of inspection by reducing overcrowding. Understanding this aspect could allow for adjustments to the search method so that the robot population size can be increased or reduced according to the collective space between robots rather than the predetermined maximum length between two points on the ship hull. This improvement could serve to increase the efficiency of the system by only recruiting robots from the available population which are deemed essential to the search, instructing others to form a second line of inspection or remain on standby. Such adjustments could also make implementation of this search method more readily applicable to ships with different hull shapes and sizes.

Comparison of the results from Fig.3.13 and Fig.3.15 (b) confirms that the completion time for the un-coordinated lawnmower search method is severely impacted by partial population failure as was predicted. When even a single agent (5% of the population) is removed, the time required to complete the inspection can increase by up to 100%.

This effect is compounded when a group of two or more consecutive agents fail near each other. This is due to the closest remaining functioning neighbour robot having to perform the work that would have been performed by the faulty robot, increasing the workload of the individual rather than the work being evenly distributed throughout the swarm as is the case in the sweeping search scenario. This reduces the scalability of the lawnmower approach, as larger swarms will perform less efficiently when the workload of failed agents is simply shifted to its closest neighbour.

The results show how the coordinated sweeping search method outperforms the un-coordinated lawnmower search method under ideal conditions and in the face of partial population failure, which is in agreement with the literature findings of Section 2.5 concerning coordinated motion and area coverage in dynamic and static 2D environments. Comparisons between these CAC experiments which took place in a 3D simulated environment and the 2D search space experiments from literature are possible thanks to the implementation of constraints to the robots working space. This allowed the swarm robots movements and pattern formations to mimic that of robots exploring 2D bounded arenas, despite carrying out an inspection of a curved surface without a solid boundary.

However, due to inaccuracies of the sonar sensors modelled in the Webots simulator as mentioned above, the sweeping search is presently more sensitive to sensor noise than the lawnmower search and leads to instabilities which render the inspection incomplete under certain conditions. The sweeping search may be superior in terms of time taken to complete the inspection, and robustness to partial population failure, but not to erroneous sensor readings. Provided the sonar sensors are modelled more accurately and the sweeping search method are modified to better adapt to sensor noise, it could in theory outperform the lawnmower search in all the scenarios. As it stands, the lawnmower search method is the only inspection which qualifies as complete in all scenarios and must be recognised as such.

# Chapter 4. Ship Hull Repair: Self-Assembly Algorithms

The research presented within this chapter pertains to the second stage of the emergency ship hull repair scenario outlined in Section 3.1 and is intended to follow successful completion of the ship hull inspection stage described in Section 3.3. This study examines the ability of the robot swarm to aggregate at a specified location and form a patch using a novel self-assembly technique which relies on direct optic communication. Self-assembly is a branch of robotics research which studies how distributed groups of robots can interact and arrange to form new configurations which are capable of more than the sum of the individual parts. As discussed in Section 2.6, the type of self-assembly detailed in these studies pertains to how a swarm of autonomous underwater modular robots can combine to create a patch of a given shape and size using their own bodies. The purpose of the resultant structure will be to cover and repair the ship hull damage which it has been created to address. The effect that increased robot traffic has on this robot assembly process is also studied by varying the population density across multiple simulations and scenarios.

The main contribution of this research is a method of self-assembly that allows modular robots to form repair patches, using their own bodies as material, which are large enough to coving a holes of various shapes and sizes in a ship hull. The results from the experiments are used to inform the design of an improved self-assembly approach which suggests a method of enhancing the initial approach by controlling the angle of approach the robots use when navigating their way to the damage, or by allowing more than one assembly location for the repair patch.

Much like the simulated repair robots of the ship hull inspection scenario, the simulated robots used in this study do not yet have a real-world counterpart and are instead restricted to more abstract descriptions of their abilities and morphology. Section 4.1 provides more insight into the use of direct communication for the purpose of self-assembly, and is followed by a description of the simulated robot morphology. The robots in these experiments are intended to possess much of the same abilities as those in Chapter 3 but their morphology and representation have been changed to allow for simpler simulation of larger robot populations. The necessity of these adjustments is provided in more detail in Section 4.1.2. Section 4.2 reveals the method of aggregation used to guide robots to the location of the ship hull damage, referred to

as the primary assembly point (PAP), and the self-assembly method which uses direct communication techniques to form a correctly sized repair patch. Section 4.3 discusses the experimental setup and explains how the success of each simulation is intended to be measured. The results of the experiments are provided in Section 4.4, and the chapter concludes with a discussion of these results and their implications in Section 4.5.

## Section 4.1 Simulated Robot Morphology

In Chapter 3, the robot morphology was designed primarily for assessing the ability of the robots to move relative to the ship hull and other robots using indirect communication methods. In this study however, the robots will require more direct forms of communication in order to carry out the aggregation behaviour and self-assembly procedures being tested. To this end, Section 4.1.1 lists the functions and physical capabilities the robot will require, in addition to those specified in Section 3.1.1, to carry out the self-assembly task. Section 4.1.2 presents the new simulated robot designed to meet these requirements and explains the reasons behind the changes which were deemed necessary to carry out the self-assembly behaviour.

### Section 4.1.1 Robot Specification

To perform the aggregation and self-assembly behaviours presented in this Chapter, at a minimum the AUVs require the ability to directly communicate with one another over short distances, and to connect with other robots to form larger structures. This is in addition to possessing an appropriate geometry which allows them to create water-tight seals between robot modules and the ability to effectively move underwater as described in Section 3.2.2. Direct communication between robots has been successfully achieved underwater using acoustic signalling (Paull, Huang, Seto, and Leonard, 2015), but this method operates at low bandwidth and is more subject to noise introduced from reflections from objects in close proximity in the water (Joordens and Jamshidi, 2010). Optic communication is an alternative method for short range underwater communication that uses light pulses (Schmickl et al., 2010). This approach overcomes some of the limitations observed in acoustic signalling but can be subject to other factors such as water clarity and ambient lighting conditions. Minimising the influence of environmental conditions on optic communication can be achieved by reducing the distance between agents. However, the shortcoming of both

systems may be better compensated for if a combination of the two communication methods is used instead (Lodovisi, Loreti, Bracciale, and Betti, 2018). A hybrid system could be well suited to multi-AUV systems that communicate under varied conditions, due to the ability to adapt the communication method based on the environmental conditions and transmission range.

As discussed in Section 2.6, swarm robots tasked with self-assembly are often dispersed within an environment and so require a method of aggregation to allow them to regroup at a common location to begin assembly. The purpose of the robots in this scenario is to address ship hull damage by using self-assembly to form a repair patch near the location of the damage. Therefore, environment mediated aggregation methods which rely on information from the environment may be more appropriate. In Section 2.6.1, it was explained how Arvin et al. (2014) used acoustic signalling systems to achieve such aggregation behaviours in robots. This is a technique which may be adaptable to underwater swarm robot scenarios since acoustic signalling has been shown to be effective at communicating over short and long distances in such environments.

There are a variety of self-assembly methods which have seen application in ground and air environments, as previously identified in Section 2.6.2, but relatively fewer have been applied to the underwater domain. As such, some of the self-assembly methods used for inspiration and guidance come from systems originally intended for a different environment than underwater, but could still be realistically implemented. The robots are intended to create a repair patch using their own bodies, a category of self-assembly methods referred to a morphogenesis, and the structure they form should be suited to the shape and size of the damage found on the ship hull. To achieve the proposed self-assembly behaviour, the robots require: a method of assessing the size and shape of the damage, sensing the presence of other robots, communicating their state to each other, a morphology which allows them to form watertight seals between units, and a method of forming physical connections between robots which are strong enough to withstand underwater currents and collisions.

### Section 4.1.2 Simulated Robot Design

In this study the robot morphology slightly diverges from the robots described in Section 3.2.2 by simplifying their representation for a 2D environment while implementing new direct communication and physical connection functions. These changes in simulation environment and robot representation were made to address issues surrounding simulation speed and accuracy. This study is primarily concerned with studying how well a swarm robot system can follow the proposed self-assembly protocol in the presence of high traffic scenarios where more than 20 robots are used. The complexity of the simulated robots and their sensors in Webots environment did not allow for efficient simulation of a significantly higher quantity of the robots tested in Chapter 3. With single runs of simulations taking days to complete, it was decided that using a simpler model in a simulation environment better suited to very high numbers of agents would be more beneficial for the initial tests of the algorithm. To this end, the experiments were conducted in Netlogo, the multi-agent programmable modelling environment, but underwater effects such as drag force and signal attenuation were omitted from the model.

The new simulated robots are assumed to possess much of the same abilities of the robots described in the ship hull inspection scenario from Chapter 3. These include the ability to maintain a set distance from the ship hull using propellers and distance sensors, detect the presence of other robots and obstacles using side-mounted sonar sensors, and assess the condition of the ship hull using a forward-facing camera. The new abilities available to these robots include direct communicate over short distances using sonar transmitters and receivers, connecting to other robots to form larger structures, disconnecting to reconfigure the resultant structure, and the ability to exchange information with neighbouring robots using optic communication (LEDs). The main distinctions between the previous and new simulated robots are their size, speed, and representation in the environment.

The robot modules described here are modelled at one tenth the size of the original simulated robots from Chapter 3, measuring 0.05m by 0.05m. Reducing the size of each robot module allowed for more accurate representation in the Netlogo simulator where the complexity of the agents modelled is more limited than Webots. However, to maintain this accurate representation it is necessary to also scale back the

maximum possible speed of each robot module to correspond with their reduced stature. Therefore, each robot is modelled with a maximum speed of 0.05 m/s which roughly equates to moving at one body length per simulated second of time. In the 2D Netlogo environment, these robots are represented as simple squares with different colours to represent their states. In the Webots simulator and on real world robots these states would instead be communicated via multi colours LEDs and corresponding colour sensitive photo-transistors.

Unlike the indirect communication methods used in the CAC algorithm experiments, the aggregation and self-assembly behaviours require more direct methods of information exchange. To allow the robots to gather at the location of the ship hull damage, the robots are assumed to possess an active omnidirectional low frequency acoustic signalling system which operates using sonar transmitters and receivers. This allows for any of the robots which discover the ship hull damage to send a global transmission about the location to the other robots within range on the same transmission/receiver frequency. The system is configured so that robots are only able to do one of these actions at a time; either they are transmitting because they have located the breach and are forming part of the repair patch using their bodies, or they are open to receiving signals because they are still in the ship hull inspection process, at which point they will abandon their search and move to participate in the repair at the specified location.

To perform the self-assembly behaviours, the robots use close range module-to-module interactions between connected agents to communicate the state each module is currently occupying – information which is essential for the self-assembly protocol to function as intended. This is achieved using multi-coloured LEDs on one robot and corresponding RGB colour sensors on the connected neighbouring robot. This form of optic communication has been shown to be as reliable as acoustic signalling at exchanging information at close range, as discussed in Section 2.6.2 of the literature review. The state of each of these robots determines whether their current connections are sufficient or if a re-configuration should be attempted to form a structure with a more appropriate shape or size. This form of direct communication is only achievable once the modules are aligned with each other so that the LEDs and phototransistors of two robots have an unobstructed line or sight. An effective method

of ensuring the robots being aligned is to allow this communication only once they have physically connected to one-another. There are benefits and drawback to both mechanical and magnetic links, as discussed in Section 2.6.2, but active magnetic links were selected as the connection method due to their ease of coupling and de-coupling and decreased likelihood of mechanical failure.

As described in Section 3.3 of the previous chapter, these robots are able to control their orientation relative to the ship hull using four forward-facing propellers. The robots use these propellers and corresponding distance sensors to maintain a distance of 2m from the ship hull, enabling them to implement algorithms which allow them to treat the ship hull as though it is a 2D plane and more easily interact with neighbouring robots. Robots then use their internal pumps to move closer or further away from other robots; working to align themselves so they may communicate their respective states directly using their corresponding LEDs and Phototransistors. These experiments were carried out in simulations that modelled static bodies of water so while the risk of each robot's orientation changing on approach to other robots is low, it is still possible. Because the risk of this occurrence in simulation was low, motorised controls to correct such a change in orientation were not implemented. Instead, the magnetic links used to couple robots together could be used to correct any minor tilts in orientation as illustrated in Figure 4.1.

As a tilted robot approaches a robot it intends to directly communicate with, it will attempt to align its most forward leading magnetic link with that of the other robot. As indicated in Figure 4.1(b), minor tilts can be corrected by the strength of the upper magnets coming together, nudging the robots orientation back into place. While this adjustment is sufficient to correct minor changes in orientation, corrections to major changes in robot orientation may require additional propellers in combination with an inertial measurement unit (IMU) to be included in future models to ensure greater control and stability.

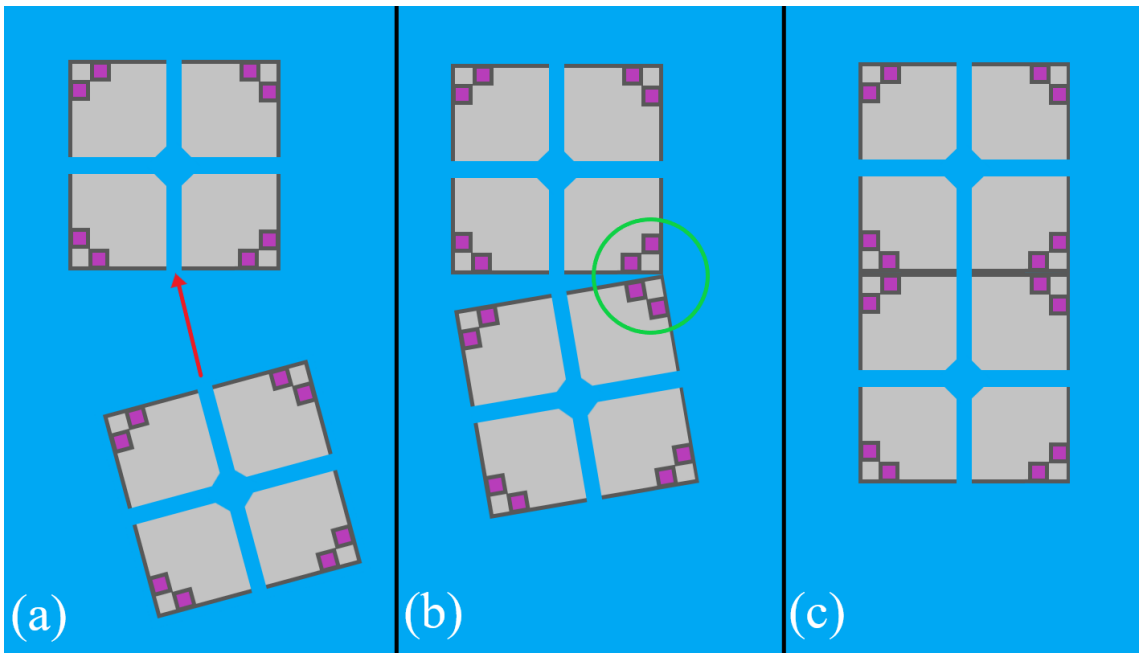



*Fig.4.1. Illustration of repair robots (light grey squares) correcting minor orientation tilit using magnetic links (purple squares on the corner of each robot face). (a) Shows one robot on approach to another at an unfavourable angle of orientation, indicated by the red arrow. (b) shows the moment the robots first make cotact, with the green circle highlighing the two magnetic links used to pull the robot back into orientation. (c) shows the final configuration of the two robots, now linked together by both magnetic links.*

In the Netlogo simulations the time taken for robots to identify ship hull damage, recognise when they are within sensor range of another robot, directly communicate with other robots, and their magnetic coupling/decoupling procedure are all modelled to be instantaneous. However, in more complex 3D simulations and real world experiments a time lag between these events in inevitable and the potential impact of these factors is discussed in Section 4.5.

## Section 4.2 Methodology

The self-assembly method presented in this section is intended for scenarios where a robot has already successfully identified hull breach damage. This robot serves as the primary assembly point (PAP) and sends out a signal which informs any robots within range of the transmission of its location. The first set of experiments examines scenarios where all the repair robots approach from a common direction as this increases the likelihood of high congestion events, and allows for the study how the

swarm robots adapt under such circumstances. The self-assembly protocol followed by the robots in this scenario instructs them to form a square structure which is wide enough and long enough to cover a circular breach, based on the diameter of the hole. The second set of experiments seeks to increase congestion further with the inclusion of obstacles for the robots to avoid when on-route to the PAP. This is included to examine how the system will perform under increasingly challenging condition in terms of time to completion and recorded collisions between robots.

The robot serving as the beacon for the PAP is working to maintain its position at the edge of the hole between the damaged and intact section of the hull. Much like the simulated environment in the ship hull inspection experiments, the ship is stationary and does not roll or drift which makes it easier for the robots to maintain their positions relative to the ship hull. However, in more realistic scenarios and simulations the ship hull is likely to roll and drift, requiring the robots to work to maintain their position by using connections to the ship hull such as physical tethers, magnets, or on-board image processing. The robot is already fitted with a forward-facing camera to recognise a damaged section of ship hull, and this same camera could be used to take pictures of the ship hull that the robot could use as a reference to where it wants to stay. Using this image to recognise when the ship hull is moving, the robot could calculate the direction of the drift and move in a similar direction to match the change in position, waiting for the moment when its current view of the ship hull and the saved image of its desired location are aligned once more. The robot broadcasts its location using the acoustic signalling system described in Section 4.1.2, while it is working to maintain its desired position, providing a rough estimate of its location to any robots within transmission range.

In all the ship hull repair scenarios, the robots are capable of indirect communication using side-mounted proximity sensors to detect the presence of other robots and obstacles and avoid or communicate as appropriate. They are also capable of two forms of direct communication, which they use to relay information about the general location of the breach and recognise the state of another robot forming part of the repair patch while also providing information about their own state. The first method of communication is an acoustic signalling system (Fig. 4.2) which uses short-ranged sonar transmitters and receivers to send and receive signals from robots located at the

hull breach indicating the PAP. The second method enables each robot to use their tri-coloured LEDs and corresponding RGB sensors to read the state of neighbouring robots, which indicates if the robots they are connecting to need other robots to connect to them, or if they should look for a different robot to attach to (Fig.4.3).

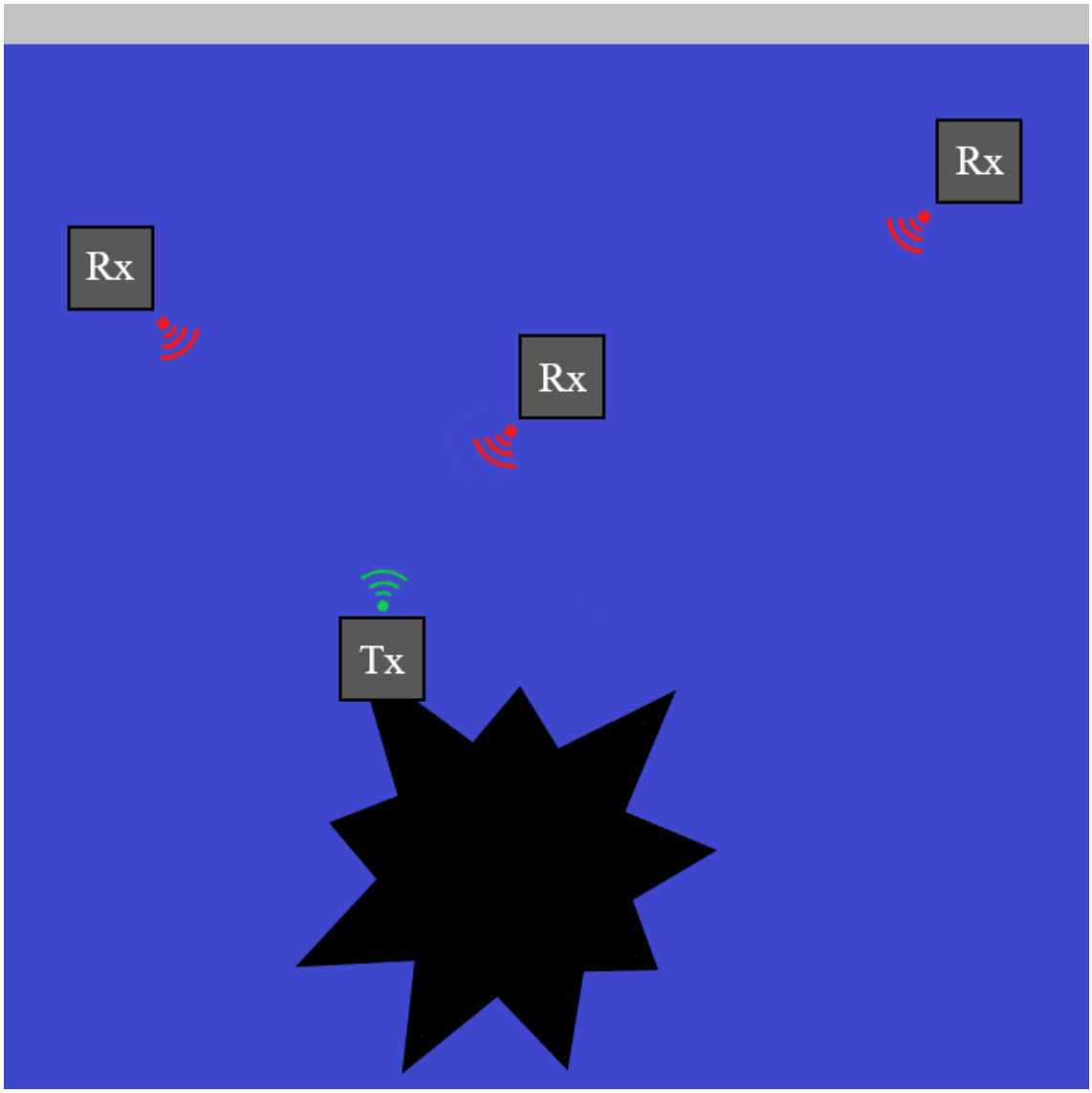



*Fig.4.2. Simulated robot modules using direct communication techniques to inform each other of the location of the hull breach and the primary assembly point (PAP).The Robot that has located the breach transmits (Tx) the signal using an omnidirectional low frequency sonar and the robots that have not located the breach receive (Rx) this signal moving towards the PAP.*

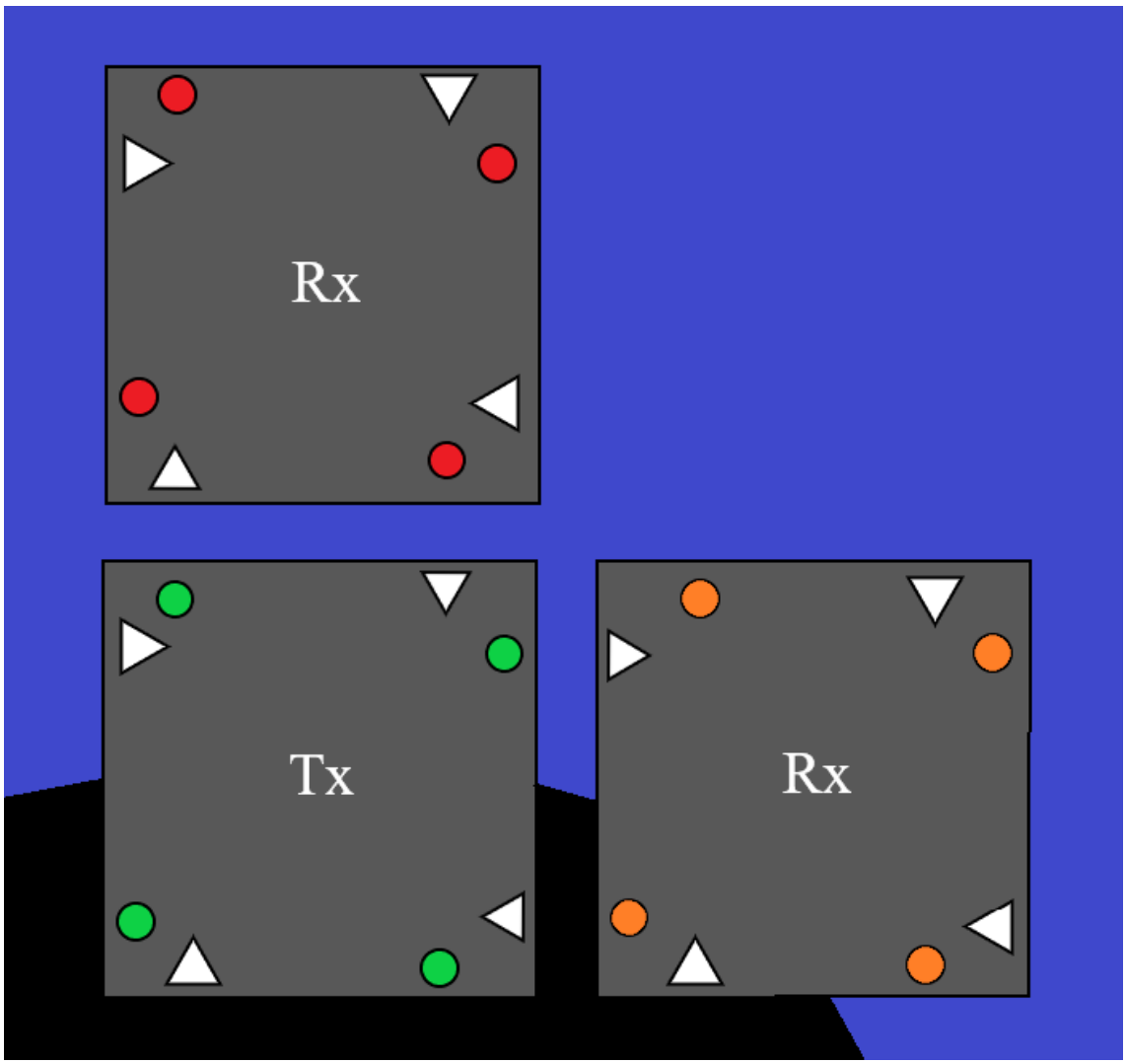



*Fig.4.3. Robot modules communicate their state with neighbours using LEDs (red, green, and orange circles) and RGB sensors (white triangles). Red LEDs indicate a robot that is in transit to the primary assembly point (PAP). Orange LEDs indicate a robot that is directly communicating with other robots and trying to find an appropriate place to attach to the structure. Green LEDs indicate a robot that has found an acceptable position, attached, and is now counted as a part of the repair patch.*

Inspired by the rules of cellular automata, each robot's state is determined primarily by the states of other connected robots that form a Von Neumann neighbourhood (Fig.4.4) and its position relative to the ship hull damage. The robot states are communicated to one another using LEDs, with states represented as different colours. When robot A(x, y) is in the red state and on route to the PAP it only takes into account the state of robot D(x, y-1) when deciding to transition to the orange state. When robot A(x, y) is in the orange state it uses the states of robots L(x-1, y) and R(x+1, y) to determine when if transitions to the green state. Finally, when robot A(x, y) is in the green state, it uses its position relative to the ship hull and the states of all the robots

in its Von Neumann neighbourhood to determine when it transitions to the purple state, indicating the repair structure is complete. Table 4.1 shows the partial truth table each robot uses to determine its state transitions based on the states of its neighbours, while Figure 4.5 shows how these transitions might unfold. Using LEDs to communicating robot states as different colours is a simple but effective method which can be used to inform other robots if there is a better position, they could occupy to better achieve the required shape and size of the repair structure.

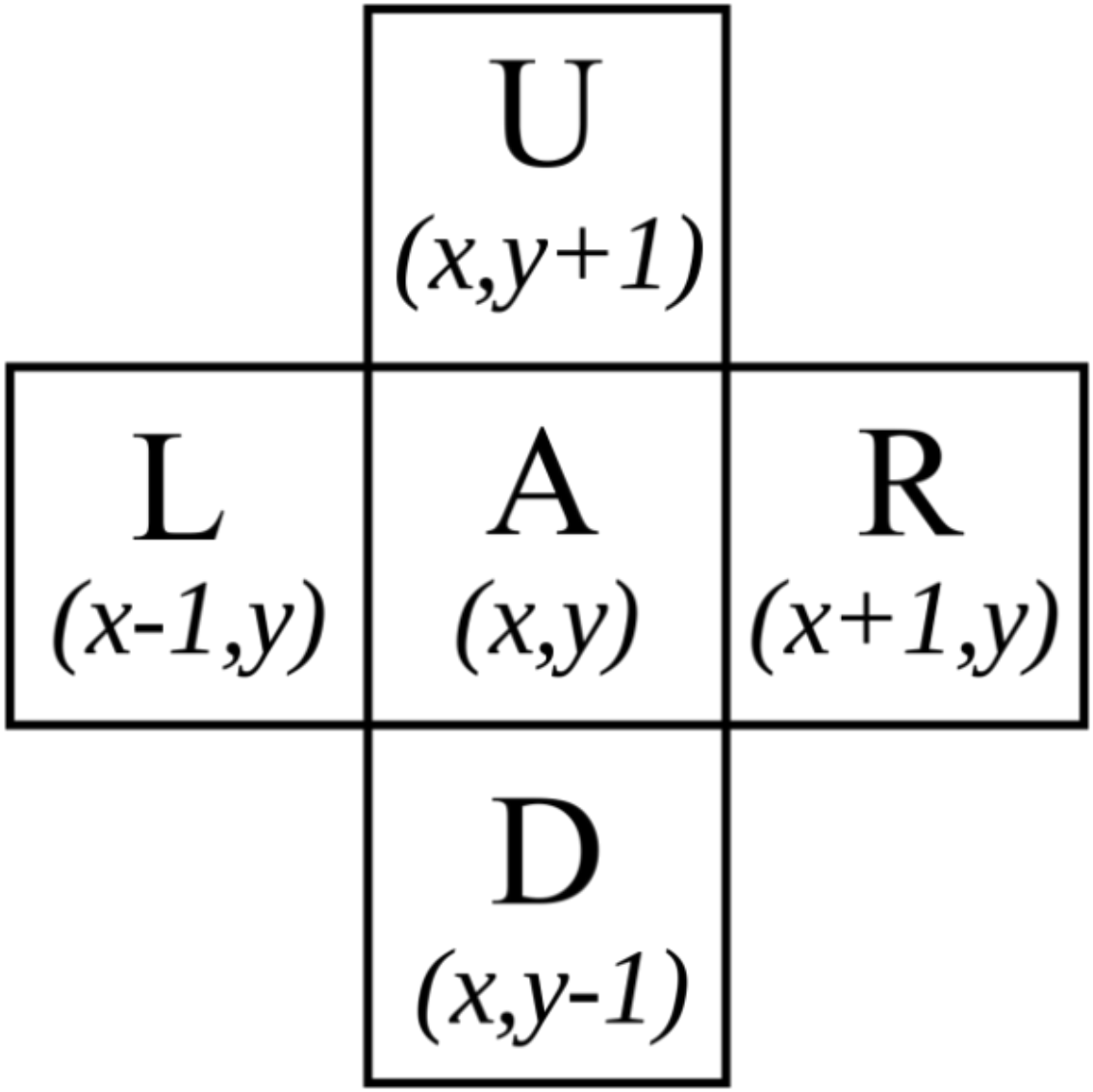
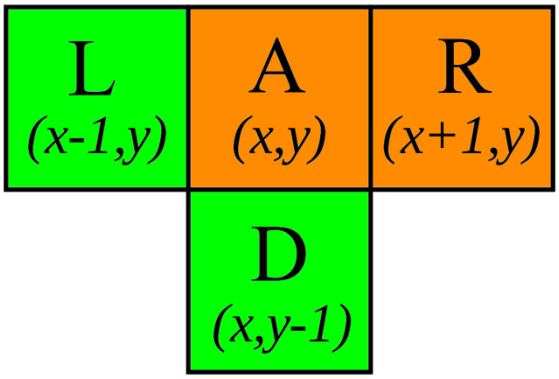



*Fig.4.4. Typical Von Neumann neighbourhood with notation adjusted to accurately represent the x-y plane is represented in Netlogo simulation software. A (x, y) represents the agent in question, L (x-1, y) is the agent to the left, R (x+1, y) is the agent to the right, U (x, y+1) is the agent upwards, and D (x, y-1) is the agent down from agent A's position.*

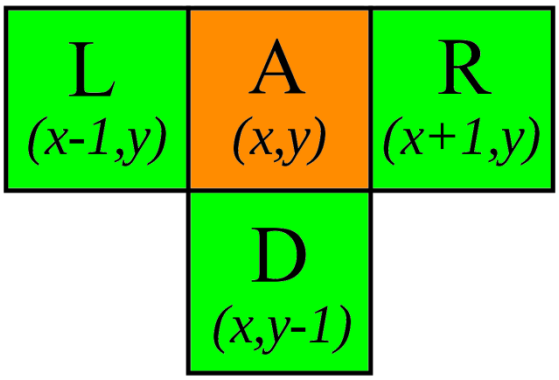
*Table.4.1. Partial truth table of the self-assembling repair robot A(x, y) when it is located at the PAP and in the orange state indicating that it is forming the central part of a block that is under construction. As discussed above, in this position it only uses the states of the neighbouring robots L(x - 1, y) and R(x + 1, y) to dictate its state transitions. Only when robot A(x, y) detects that these robots are in the green state does it decide to transition to the green state also. Number 1 represents the red state, number 2 represents the orange state, and number 3 represents the green state.*

| **L (x - 1, y) (t)** | **R(x + 1, y) (t)** | **A (x, y) (t + 1)** |
|---|---|---|
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 1 | 3 | 2 |
| 2 | 1 | 2 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |
| 3 | 1 | 2 |
| 3 | 2 | 2 |
| 3 | 3 | 3 |

(t)




(t + 1)




(t + 2)

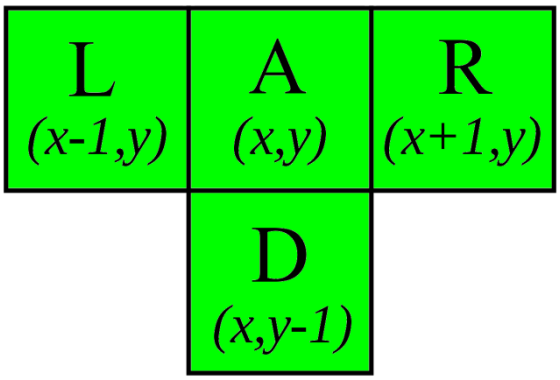



*Fig.4.5. this diagram shows a state transition example using the state transition truth table from Table 4.1. Robot A(x, y) at time (t) examines the states of robots L(x-1, y) and R(x+1, y) to determine whether it should turn green or remain orange. It observed that robot R(x+1, y) was not in the green state during inspection and so chooses to remain orange. At time (t+1) robot A(x, y) examines the states of robots L(x-1, y) and R(x+1, y) again to check if there has been a change, this time confirming that both robots are in the green state. At time step (t+2) robot A(x, y) has transitioned to green based on the states of its neighbours observed during the previous step.*

Once a robot module has successfully navigated to the location of a robot module close to the breach location, it can begin the CD self-assembly protocol. The protocol instructs each robot to begin attaching to other robots until they collectively form a block of robots that spans the full diameter of the breach. The length of this block is determined by the cameras which can recognise when the modules have formed a line of adequate length, which is achieved by checking if the modules that form each end of the block are centred over an intact section of ship hull, but also connected to a robot that is centred over the breach. Once the first block has fully formed, the block advances by one module body-length (0.05m) and the unattached modules begin forming a second block above the first, increasing the total area of the structure. This process then repeats until every module on the perimeter of the breach can confirm they are not directly above the breach but are still connected to a module that is. In this case the robots communicate to each other that a patch of appropriate size has been achieved and they enter their final state where they prepare to attach to the hull and seal the breach. The result is a square sheet formed of robot modules which is large enough to cover the hole.

At the beginning of this section it was stated that the simulated damage to be repaired would be a circular hole and here we revealed that the self-assembly algorithm is designed to form a square patch cable of coving this hole. However, the algorithm can be adapted to form square patches for a variety of hole shapes by using the forward facing camera to determine the maximum diameter of the hole and using this to determine the length of connected modules the robot must form. This length ensures the patch formed would be sufficient to cover the entire breach, provided enough chains of modules are connected together. However, the current approach is best suited for addressing circular holes, which were selected as they represent the most common form of battle damage sustained from a direct torpedo hit.

In order for the proposed Self-assembly algorithm to function effectively, the robots must be able to link and unlink with relative ease. As discussed in section 4.1.2 the robots are designed to use actively controlled magnetic links as they have a lower tendency of failure than more common mechanical linking methods in repeated coupling/decoupling scenarios. Should the magnetic links prove to be insufficient in withstanding the pressure from the surround fluid and forces of waves, a combination

of mechanical and magnetic links may be substituted to improve the integrity of the patch – even if this may increase the risk of failures of linking and unlinking from other robots. However, the ability of the swarm to maintain the integrity of the completed structure they form using magnetic links alone is a question that falls outside the scope of this scenario and is instead delegated to future works. For the purpose of this algorithm, it is assumed that the robots experience no failures to link or unlink during the self-assembly process.

The pseudocode in Fig.4.6 represents our self-assembly algorithm used in all of the experiments including those where additional obstacles are modelled. It shows the protocols for navigation, obstacle avoidance, and the state transitions each robot module undergoes to create the desired square structure.

**Algorithm 1** CD Self-Assembly Algorithm

```
1: begin program
2:
3:   while unattached to block do
4:       if obstacle ahead = false   then
5:           if agent ahead = false   then
6:               face reference point module.
7:               move forward by 0.05m.
8:           else
9:               if agent ahead = red   then
10:                  move backwards by 0.05m.
11:              else if agent ahead = green   then
12:                  attach to top of agent.
13:              else if left neighbour of agent ahead = green   then
14:                  attach to right side of block.
15:              else
16:                  attach to left side of block.
17:              end if
18:              end if
19:      else
20:          if space left of agent empty then
21:              turn left
```

```
22:                     move forward by 0.05m
23:             else if space right of agent empty then
24:                     turn right
25:                     move forward by 0.05m
26:             else
27:                     turn left
28:     end if
29: end while
30: while own state ≠ green do
31:     if hull breach in line of sight = false then
32:         own state = green.
33:     else
34:         if left neighbour state ≠ green then
35:             own state = orange.
36:         else
37:             if right neighbour state ≠ green then
38:                 own state = orange.
39:             else
40:                 own state = green.
41:             end if
42:         end if
43:     end if
44: end while
45: while own state ≠ purple do
46:     if all neighbour states = green then
47:         advance 0.05m to cover hull breach.
48:         if hull breach in line of sight = false then
49:             own state = purple.
50:         end if
51:     else
52:         if all neighbour states = purple then
53:             own state = purple.
54:         end if
55:     end if
56: end while
```

```
57:  while breach ≠ sealed do
58:      approach and seal hull breach.
59:  end while
60:
61: end program
```

*Fig.4.6. Pseudocode for the navigation, obstacle avoidance, and state transitions of our algorithm, instructing agents which module to attach to, and which state to occupy based on their displayed state.*

## Section 4.3 Experimental setup

As mentioned in Section 4.1, the experiments are conducted in the 2D simulation environment Netlogo using simplified robot morphology. This allowed for simulation of larger robot populations which could positively identify surrounding robots and obstacles more reliably than in the 3D Webots simulations of Chapter 3. The experimental setups are designed to test the ability of the robot swarm to self-assemble in scenarios of high congestion, where many robots are simultaneously vying for limited space to complete their task. The first setup is concerned with testing how varying the robot population density affect the time taken to complete repair structures of various sizes and how frequently collisions or errors occur. The second setup uses a similar configuration to the first, but includes additional obstacles placed at three different positions between the approaching robots and the PAP. This inclusion is to examine how the inclusion of additional obstacles affects the congestion observed in the experiments, and whether is hinders or benefits the system relative to the first experiments where no additional obstacles were present.

The simulated robot modules can freely move in the simulation space, but to reduce the widening reality gap related to the simpler morphology used, the robots are modelled with a boundary that cannot be crossed by other robots or obstacles. This embodiment of the robots means they are unable to move through, over, or under other robots and obstacles. In order to overcome obstacles and other robots blocking the robots path to the PAP, one of two methods must be used. If an obstacle is encountered ahead of the robot it will stop, examine the spaces to its left and right, and take the path less obstructed until the obstacle is no longer blocking its path. If instead another robot is encountered, it follows the rule which compels it to give way

to other robots that are either ahead or to the right of them. When all members of the swarm adhere to this rule of avoidance, it reduces the likelihood of collisions with robots and obstacles which would hinder the ability of the swarm to aggregate at the PAP and perform the assembly in an orderly manner.

All of the simulated environments use a rectangular arena, which represents a section of the ship hull to be inspected as shown in Fig.4.7. The light grey area represents the section of ship hull above the waterline, the blue area is the section of ship hull beneath the waterline, and the black area represents the hull breach. The robot modules are represented as small squares which display one of four colours, representing the colour of their LEDs which they use to communicate their respective states when in close proximity to other robots: red, orange, green, and purple. All robots begin as red squares indicating they have received a signal informing them of the PAP location and are in currently in transit to join the repair effort. Green blocks represent robots that are forming part of the repair structure and have settled on their final location. Orange blocks represent robot modules that are in the process of examining the states of other modules while moving to an appropriate place to attach, or connected to the structure, but forming part of an incomplete section. When a complete repair structure has been formed, the robots use their state and the states of their neighbours to propagate this information and become purple blocks, indicating that the process is complete and shown in Fig 4.7 (b).

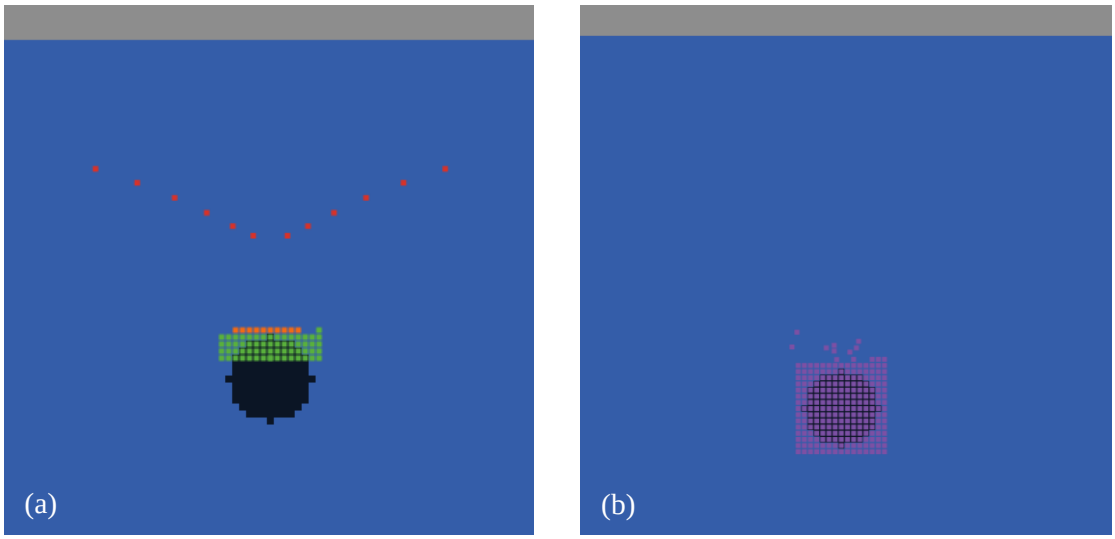



*Fig.4.7. Netlogo simulated environment, showing (a) robot modules carrying out the self-assembly protocol, and (b) robot modules that have successfully completed the repair patch. The colours of each block represent the LEDs on each face of the robot, which indicates their state: red for in transit, orange for looking for a place to attach, green for having already attached to a robot in an appropriate position, and purple indicating that a complete structure has been formed.*

All simulations begin with a group of robot modules already deployed in the water, ready to approach the ship hull breach. The initial number of robots deployed in the environment is deliberately insufficient to form complete repair patch, requiring that more robot modules are introduced as the simulation progresses. The number of robots added to the simulation each minute to assist with the repair is one of the controlled variables, and these deployments occur at steady rates ranging from 2 to 40 additional robots per minute. This rate of deployment is how increases and decreases in robot population density are implemented – breaking high congestion events into multiple separate instances allowing for a clearer vision of the effect these high congestion events have on the self-assembly process. In addition to the deployment rate, the number of robots currently on route to the PAP is also considered and the system prevents additional robots from being deployed until less than two full deployments of robots are approaching the PAP. It is hypothesised that controlling the number of robots deployed to form the repair patch in this way may reduce the risk of system failure due to overcrowding while still allowing high congestion events to occur. A series of 100 simulated experiments are performed for each variable changed,

including increases of decreases in robot deployment rate. The results from these experiments are graphed in Section 4.4, with each point represents the median value obtained all the simulations.

### Section 4.3.1 Robot Congestion Setup

These experiments address hull breaches less than or equal to 0.6 m in diameter, since this represents the upper bound of common torpedo diameters. In this scenario, a hull breach has led to a single compartment becoming fully flooded, but flood boundaries have been established within the ship to seal the room off from other sections of the ship. The constants of this experiment are the shape of the breach (circular), the maximum movement speed of the repair robots (0.05m/s), the speed of coupling and decoupling of modules (instant), and the speed of information exchange between the robots (instant). The variables of this experiment include the location of the hull breach (depth of 0.6m to 4.2m) and the size of the breach (diameter of 0.1m to 0.6m). Tables 4.2 and 4.3 show these constants and variables and list the range of values examined.

*Table.4.2. Constants of the high congestion experiments with their value listed*

| Constant | Value |
|---|---|
| Ship hull breach shape | Circular |
| Robot maximum speed (meters per second) | 0.05 |
| Coupling/decoupling speed | Instant |
| Communication speed | Instant |

*Table.4.3. Variables of the high congestion experiments, with their range of variables listed.*

| Variable | Values |
|---|---|
| Hull breach depth (meters) | 1.2, 2.4, and 3.6 |
| Hull breach diameter (meters) | 0.2, 0.4, and 0.6 |
| Robot deployment rate (per minute) | 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, and 40 |

A multi-robot system can be identified as suffering from high congestion when a significant fraction of the population is forced to change their speed or direction as a result of avoiding collisions with other robots. In the context of these experiments, a swarm is considered to be experiencing high congestion if more than 50% of the robots must perform robot avoidance on route to the PAP. By varying the number of robots deployed each minute and measuring how long it takes the swarm to complete the repair against the frequency of robot avoidance events (high congestion), it may be possible to identify the point at which the system performance begins to suffer due to overcrowding. This could be used to determine which of the deployment rates achieves the highest speed of completion without causing more than 50% of the population to avoid robot collisions (high congestion). Increasing the depth and diameter of the hull breach will change the angles of approach each robot follows when navigating to the PAP. This can be used as a secondary method of increasing the likelihood of high congestion forming and may reveal another point at which high congestion begins to occur due to robots using steep angles of approach, resulting in less space to manoeuvre when avoiding collisions with other robots.

### Section 4.3.2 Obstacle Avoidance Setup

The second set of experiments examine how the inclusion of additional obstacles between the approaching robots and the hull breach affects their ability to complete their self-assembly task relative to the results from the first scenario. In these experiments the hull breach diameter and depth are kept constant at 0.4m and 2.4m respectively, choosing to vary the size and location of the obstacles instead. All the obstacles have a circular shape and come in one of three different diameters: 0.2m for half the width of the breach, 0.4m for the same size as the breach, and 0.6m for one and a half times the size of the breach. In addition to this, each obstacle will occupy one of three separate locations between the starting point of the robots' journey and the PAP; above and left of the breach, directly above the breach, and above and right of the breach as illustrated in Fig.4.8. Tables 4.4 and 4.5 show the constants and variables of these obstacle avoidance experiments and list the range of values examined.

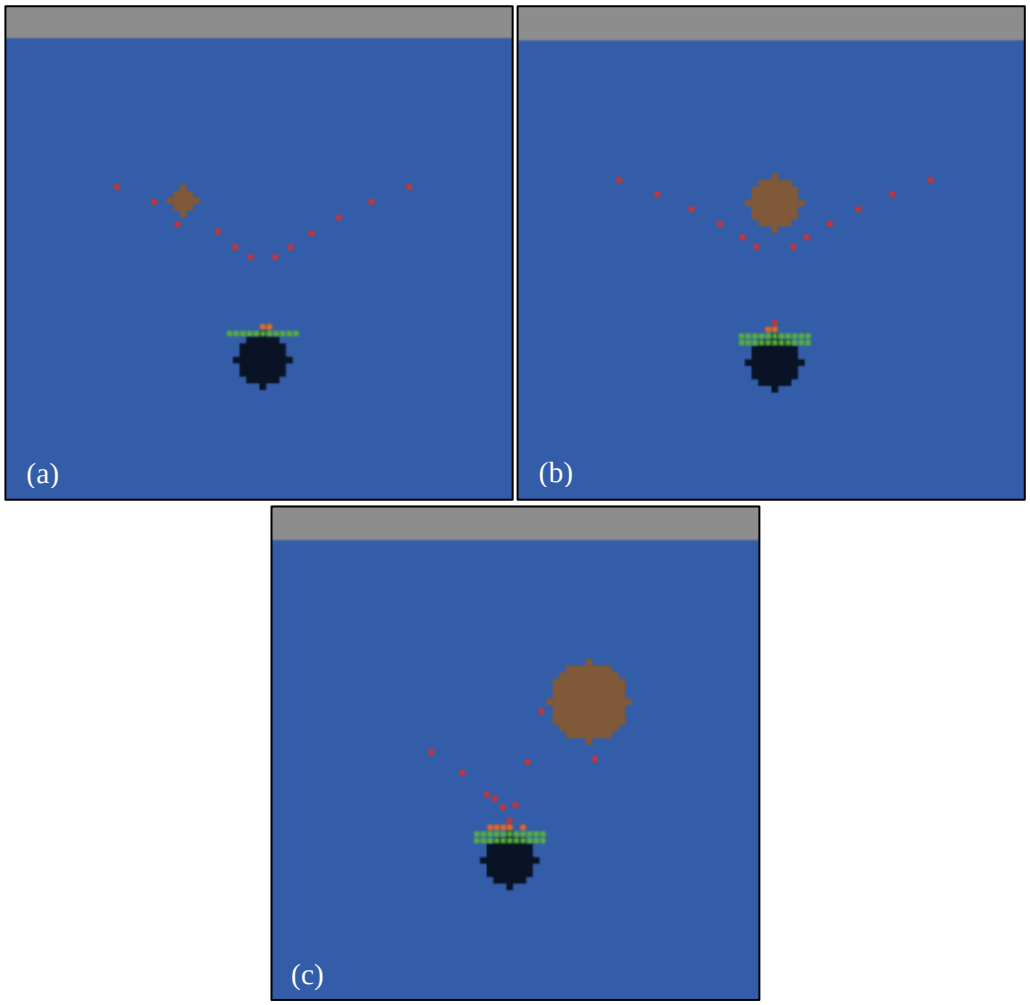



*Fig.4.8. Netlogo simulated environment, showing (a) small obstacle (0.2m diameter) placed above and to the left of the breach, (b) a medium obstacle (0.4m diameter) directly above the breach, and (c) a large obstacle (0.6m diameter) above and to the right of the breach.*

*Table.4.4. Constants of the obstacle avoidance experiments with their value listed.*

| **Constant** | **Value** |
|---|---|
| Obstacle shape | Circular |
| Ship hull breach shape | Circular |
| Hull breach diameter (meters) | 0.4 |
| Hull breach depth (meters) | 2.4 |
| Robot maximum speed (meters per second) | 0.05 |
| Coupling/decoupling speed | Instant |
| Communication speed | Instant |

*Table.4.5. Variables of the obstacle avoidance experiments with their range of variables listed.*

| Variable | Values |
|---|---|
| Obstacle location | Left, centre, and right |
| Obstacle diameter (meters) | 0.2, 0.4, and 0.6 |
| Robot deployment rate (per minute) | 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, and 40 |

The inclusion of additional obstacles may serve to slow the overall progress of the assembly, or prove a benefit in scenarios using higher deployment rates of robots by shifting the location of the congestion away from the PAP, allowing for faster assembly of the structure by reducing overcrowding. The inclusion of an obstacle may also cause some robots to alter their original course such that they encounter fewer robots along their new path, reducing the congestion experienced across the swarm as a whole.

## Section 4.4 Results

In this section, the results from the robot congestion and obstacle avoidance experiments, as described in Section 4.3, are presented and compared. A series of 100 simulated experiments is performed for each variable changed in the two scenarios, such as increases or decreases in robot deployment rate, and the size and location of the hull breaches and additional obstacles. Each subsection presents six of the most significant graphs, with the remainder delegated to Appendix C. Each point on these graphs represents the median value obtained from successful runs of each set of 100 simulations, excluding the scenarios where robots become stuck close to the PAP, causing a blockage to form and preventing any further assembly actions of robots. The cause of these blockage events and their implications are discussed within this section, while how this information can be used to improve the self-assembly approach is discussed in Section 4.5. The graphs are presented in sets of two according to varying breach depth, with the first graph in each set showing the time taken for each robot population to complete the self-assembly of a repair patch. The second graph of each set shows the percentage of each robot population which encountered another robot on route to the PAP, causing it to change it speed and direction, referred to as robot

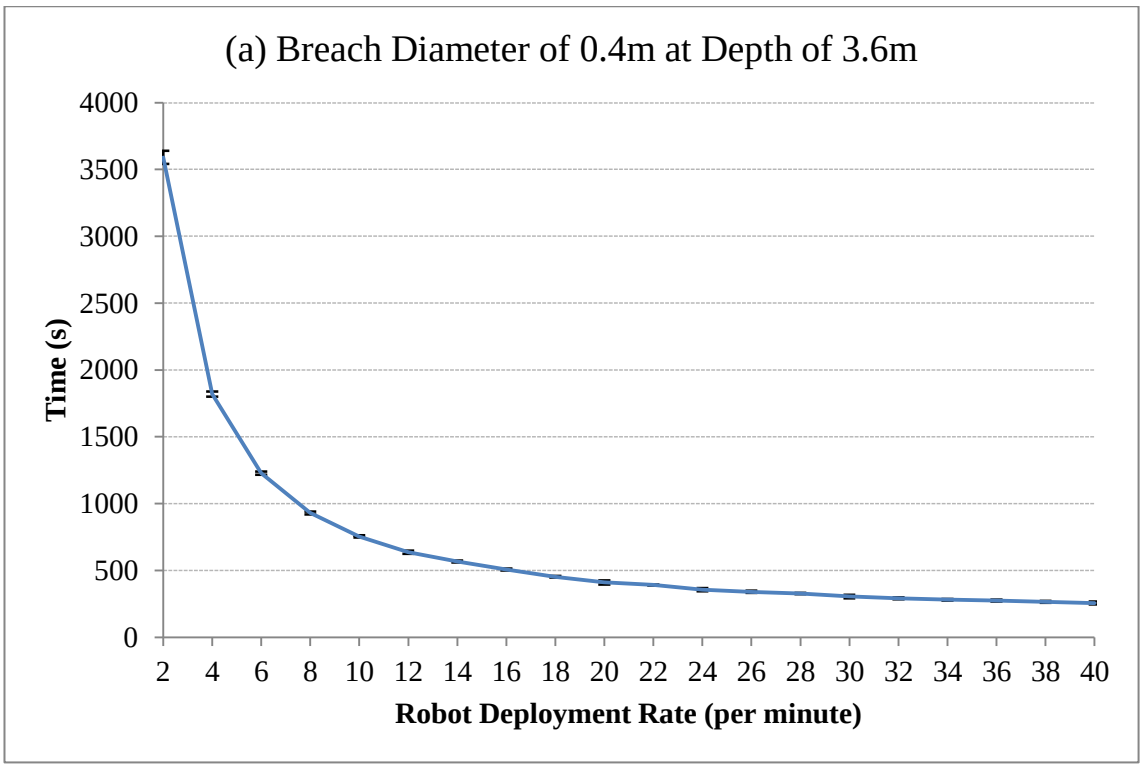
congestion percentage. In Section 4.4.2 the second graph of each set also includes the percentage of the robot population which have encountered an obstacle on route to the PAP, referred to as obstacle avoidance percentage, to measure how this affects the results.

### Section 4.4.1 Robot Congestion Results

The robot congestion experiments examined the swarm's ability to perform the self-assembly protocol on hull breaches of varying size and at varying depths beneath the waterline. The results of these experiments revealed truths that hold across all scenarios such that it is only necessary to show a sample of the results here to identify varying trends. As such, the graphs only show results from scenarios using a breach diameter of 0.4 meters. Table 4.6 shows samples from all the scenarios to demonstrate some of the correlations, but otherwise full results are delegated to Appendix C. Increasing the diameter of the breach predictably increases the time taken for each robot population to repair the breach – doubling the size of the breach doubles the time taken to perform the repair. Increasing the number of robots deployed each minute decreases the time taken to complete self-assembly, but also increases the number of robots performing robot avoidance. It can be observed from figure 4.9 (a) and (b) that as the percentage of the robot population experiencing congestion increases, the gains in competition speed start to decrease.

This correlation may at first seem to be the primary cause of the decreased performance; however this trend is not perfectly mirrored in figures 4.10 and 4.11 where the time taken to complete the self-assembly is closely correlated with that of figure 4.9 (a), but the percentage of robot congestion decreases when the depth of the breach decreases. This would indicate that the depth of the breach has a more significant impact on the number of robot avoidance events than initially considered. However, the reason for this may have less to do with breach depth and more to do with the angle of approach each robot takes on approaching the PAP. As the depth of the breach decreases, robots deployed at the top right and left of the arena begin to use shallower angles of approach than those deployed in the top middle. Shallower angles of approach seem to form paths that are more evenly spread across the arena than steep angles of approach, which concentrates the density of the robot population in the area directly above the PAP.

Our original measure of optimal robot deployment rate was to be determined by the robots which could complete the self-assembly procedure in the fastest time without pushing more than 50% of the total population to experience robot avoidance (high congestion). The results show that robot congestion varies according to the depth of the breach and the consequent angles of approach used by the swarm, so the optimal deployment rate varies also. So for swarms using the proposed self-assembly protocol to address breaches with a diameter of 0.4m, the optimal deployment rate (robot congestion < 50%) at 3.6m is 8 robots per minute (rob/m) which resulted in a completion time of 15.5 minutes, at 2.4m is 12 rob/m which resulted in a completion time of 10.33 minutes, and at 1.2m is 16 rob/m which resulted in a completion time of 7.7 minutes. These results have helped identify aspects of the approach which could be adjusted to improve the self-assembly protocol, such as changing the systems reliance on a PAP and controlling the robot angles of approach, which are discussed further in Section 4.5.



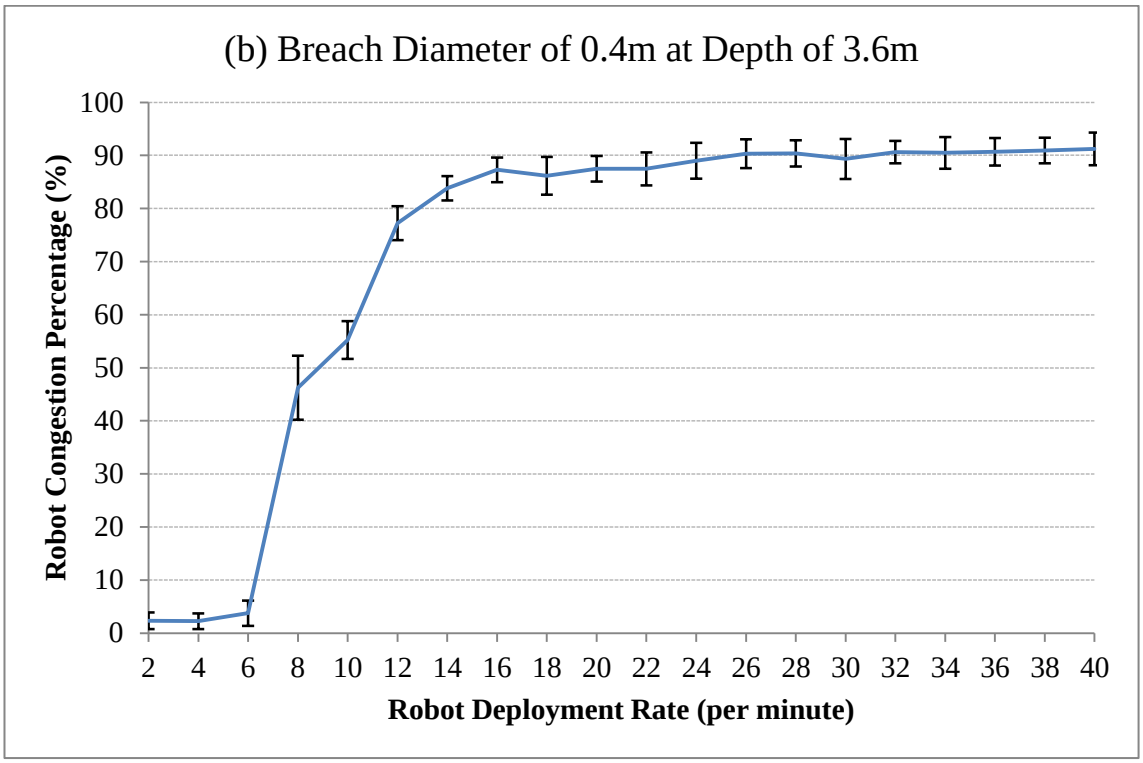
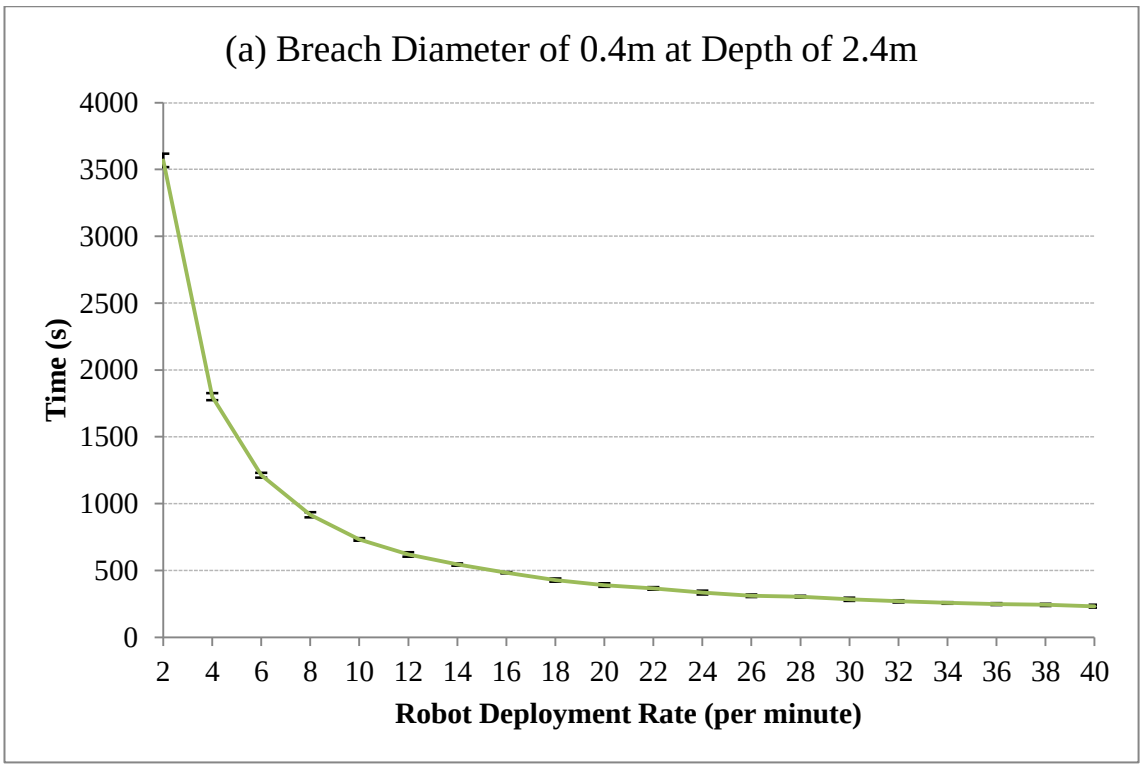




*Fig.4.9. Graphs of results from the robot congestion experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion percentage formed for varying deployment rates. Each result represents the median value obtained from 100 simulations, with error bars showing the standard deviation. This plot shows the results of the experiments using a breach diameter of 0.4 meters at a breach depth of 3.6 meters.*



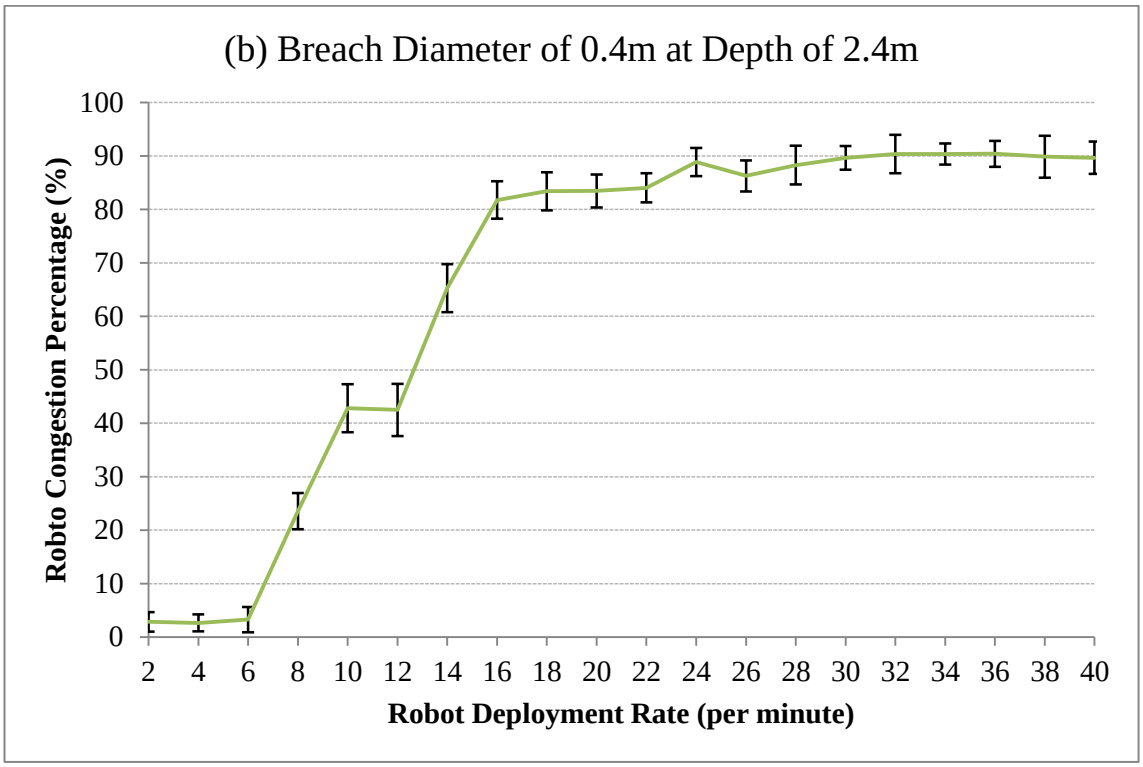
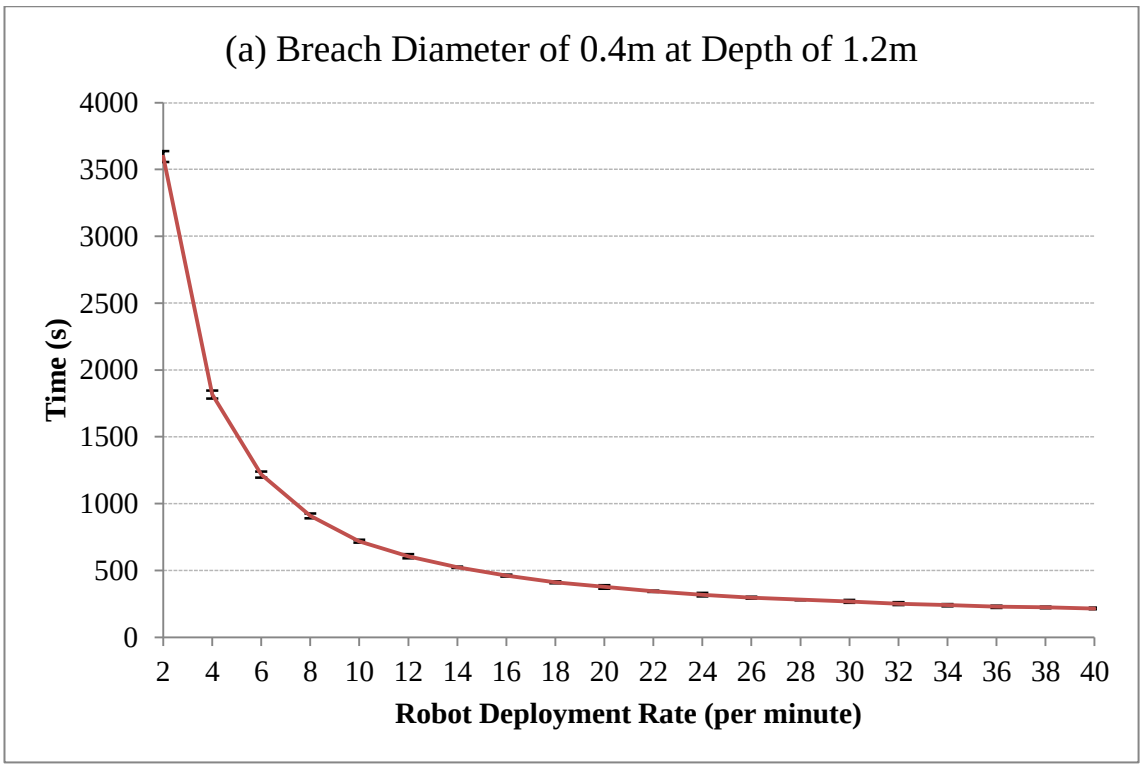




*Fig.4.10. Graphs of results from the robot congestion experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion percentage formed for varying deployment rates. This plot shows the results of the experiments using a breach diameter of 0.4 meters at a breach depth of 2.4 meters.*



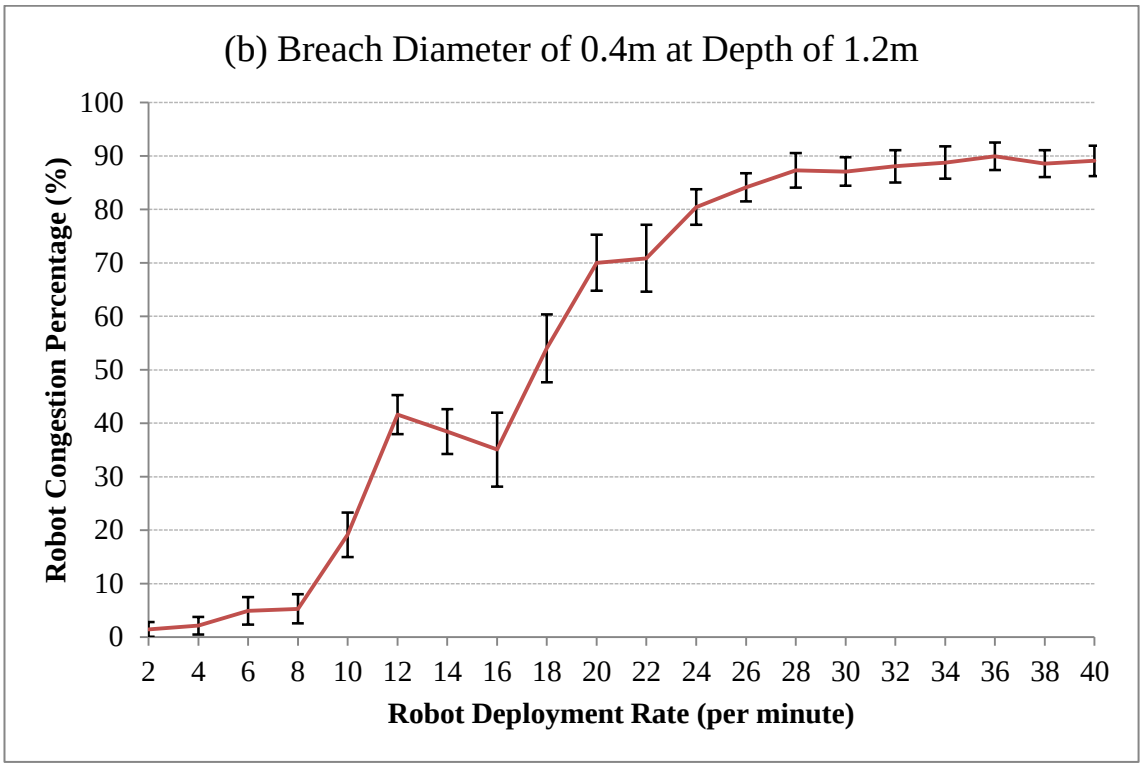




*Fig.4.11. Graphs of results from the robot congestion experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion percentage formed for varying deployment rates. This plot shows the results of the experiments using a breach diameter of 0.4 meters at a breach depth of 1.2 meters.*

*Table 4.6 Sample results from across all robot congestion scenarios.*

| Breach Diameter (meters | Breach Depth (meters) | Robot Deployment Rate (per minute) | Time Taken (seconds) | Robot Congestion (%) |
|---|---|---|---|---|
| 0.2 | 3.6 | 10 | 330 | 58 |
| 0.2 | 3.6 | 20 | 208 | 82 |
| 0.2 | 3.6 | 30 | 163 | 85 |
| 0.4 | 3.6 | 10 | 751 | 55 |
| 0.4 | 3.6 | 20 | 407 | 88 |
| 0.4 | 3.6 | 30 | 304 | 89 |
| 0.6 | 3.6 | 10 | 1372 | 55 |
| 0.6 | 3.6 | 20 | 726 | 88 |
| 0.6 | 3.6 | 30 | 512 | 91 |
| 0.2 | 2.4 | 10 | 308 | 44 |
| 0.2 | 2.4 | 20 | 184 | 82 |
| 0.2 | 2.4 | 30 | 138 | 83 |
| 0.4 | 2.4 | 10 | 732 | 43 |
| 0.4 | 2.4 | 20 | 389 | 83 |
| 0.4 | 2.4 | 30 | 285 | 90 |
| 0.6 | 2.4 | 10 | 1358 | 31 |
| 0.6 | 2.4 | 20 | 706 | 85 |
| 0.6 | 2.4 | 30 | 488 | 90 |
| 0.2 | 1.2 | 10 | 289 | 13 |
| 0.2 | 1.2 | 20 | 161 | 69 |
| 0.2 | 1.2 | 30 | 115 | 76 |
| 0.4 | 1.2 | 10 | 715 | 19 |
| 0.4 | 1.2 | 20 | 373 | 70 |
| 0.4 | 1.2 | 30 | 262 | 87 |
| 0.6 | 1.2 | 10 | 1348 | 14 |
| 0.6 | 1.2 | 20 | 696 | 64 |
| 0.6 | 1.2 | 30 | 467 | 88 |

### Section 4.4.2 Obstacle Avoidance Results

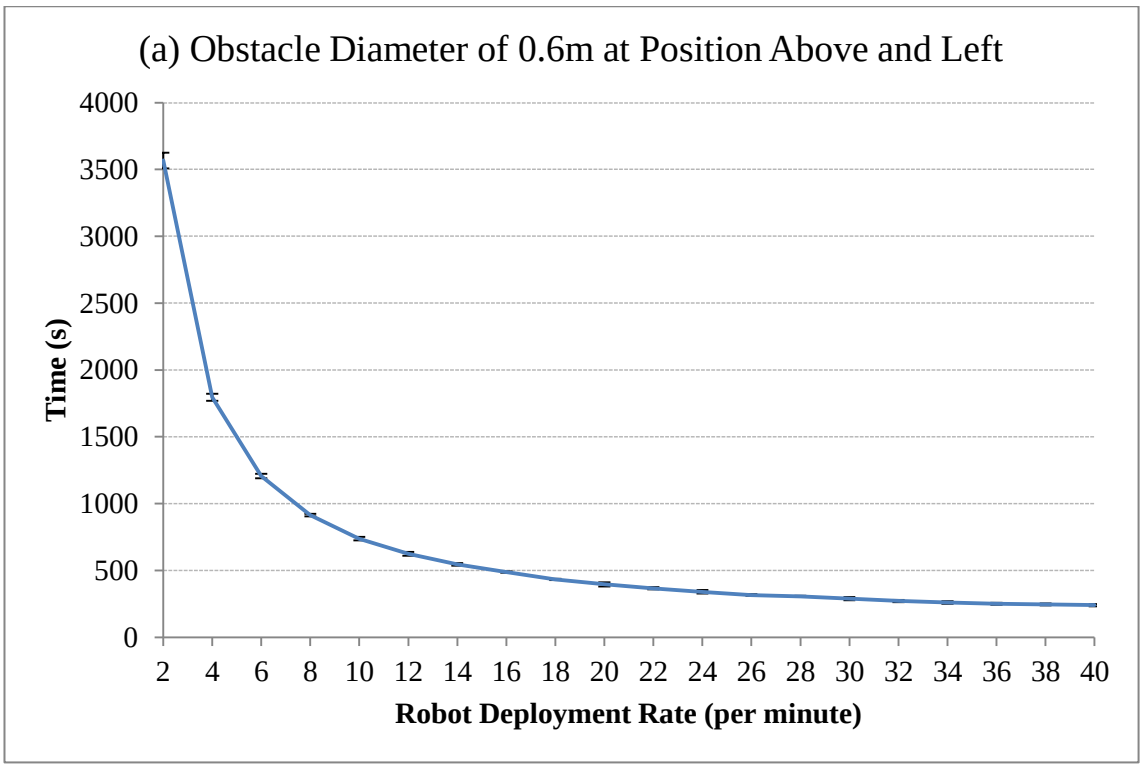
In the obstacle avoidance experiments the same self-assembly protocol is examined, but additional obstacles are introduced to the environment to measure if their inclusion affects the prevalence of robot congestion. The depth and diameter of the hull breach are kept the same at 2.4m and 0.4m respectively, but the size and location of the obstacles are varied. The results from these experiments reveal some interesting effects of including obstacles at the three locations: above and left of the

PAP, directly above the PAP, and above and right of the PAP. Obstacle diameters of 0.2m and 0.4m result in fewer obstacle avoidance events and so their effect is less pronounced than experiments using obstacle diameters of 0.6m. For this reason, the graphs show the results from the experiments using the largest obstacle diameter to make the effect of their inclusion clearer. Table 4.6 provides samples from all of the scenarios to help demonstrate trends, while full results of the smaller obstacle experiments are delegated to Appendix C.

Comparing the results from figures 4.12(a), 4.13(a), and 4.14 (a) to figure 4.10 (a) reveals that that inclusion of additional obstacles has not had a significant impact on the time taken to complete the self-assembly task. However, including the obstacles at the aforementioned locations has had a marked effect on the percentage of robot congestion experienced by the swarm. Figures 4.12 and 4.14 provide a mirror image of each other, showing how placing obstacles directly in the path of the robots approaching the PAP with the shallowest angles of approach has a significant effect on the amount of recorded robot congestion. Comparing Figure 4.13 (b) to Figure 4.10 (b) shows how the obstacle avoidance events occurring directly above the PAP has had a less significant effect on the recorded robot congestion percentage than when placed directly in the path of robots following shallow angles but has also increased the total amount of robot congestion. The inclusion of the obstacles did not noticeably change the number of robots than can be deployed per minute before experiencing high congestion (> 50%) when placed above the PAP. However, including obstacle above and left or above and right has effectively reduced the deployment rate from is 12 rob/m which resulted in a completion time of 10.33 minutes, to 8 rob/m which resulted in a completion time of 15.23 minutes.

Avoiding the obstacles placed on the left and the right seems to have caused the robots to take alternate routes to the PAP with a different angle of approach, with a knock-on effect of increasing the number of recorded robot avoidance events. This supports the view that controlling the angle of approach used by the robots on approach to the PAP may be leveraged to decrease robot congestion, which could allow for a greater number of robots to be deployed at one time to perform the self-assembly protocol, and ultimately decreasing the time taken to complete the ship hull repair. These considerations are discussed in more detail in the following Section 4.5.

The full program code used to conduct these experiments is included in Appendix B and full videos of the simulations showing the self-assembly protocol can be accessed via the dedicated GitHub repository at https://github.com/MattSHaire/Emergency-Ship-Hull-Repair.



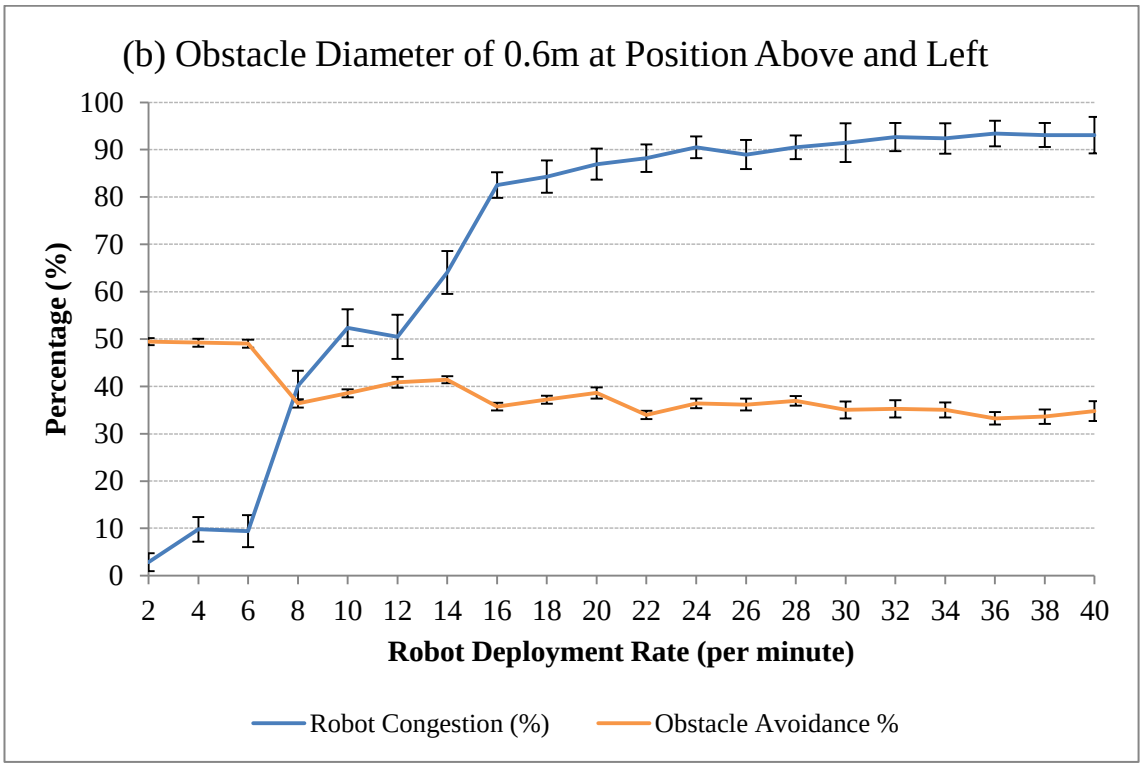
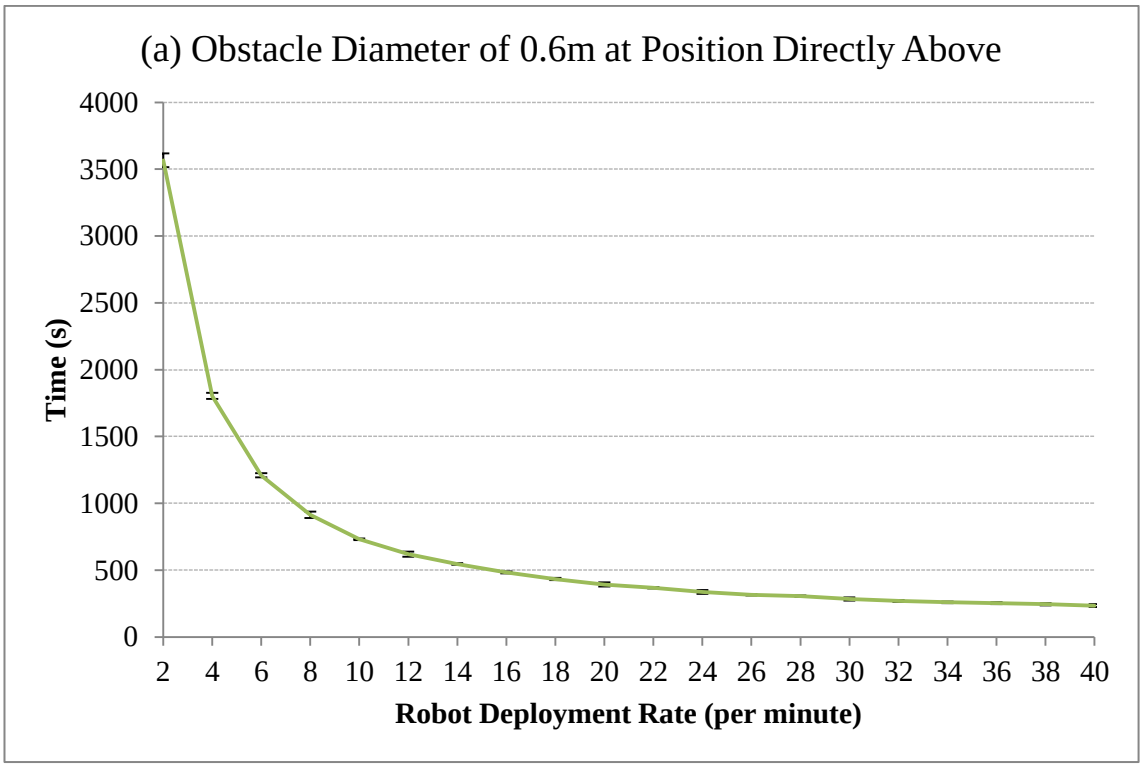




*Fig.4.12. Graphs of results from the obstacle avoidance experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion and obstacle avoidance percentages formed for varying deployment rates. Each result represents the median value obtained from 100 simulations, with error bars showing the standard deviation. This plot shows the results of the experiments using an obstacle diameter of 0.6 meters at the location above and left of the PAP.*



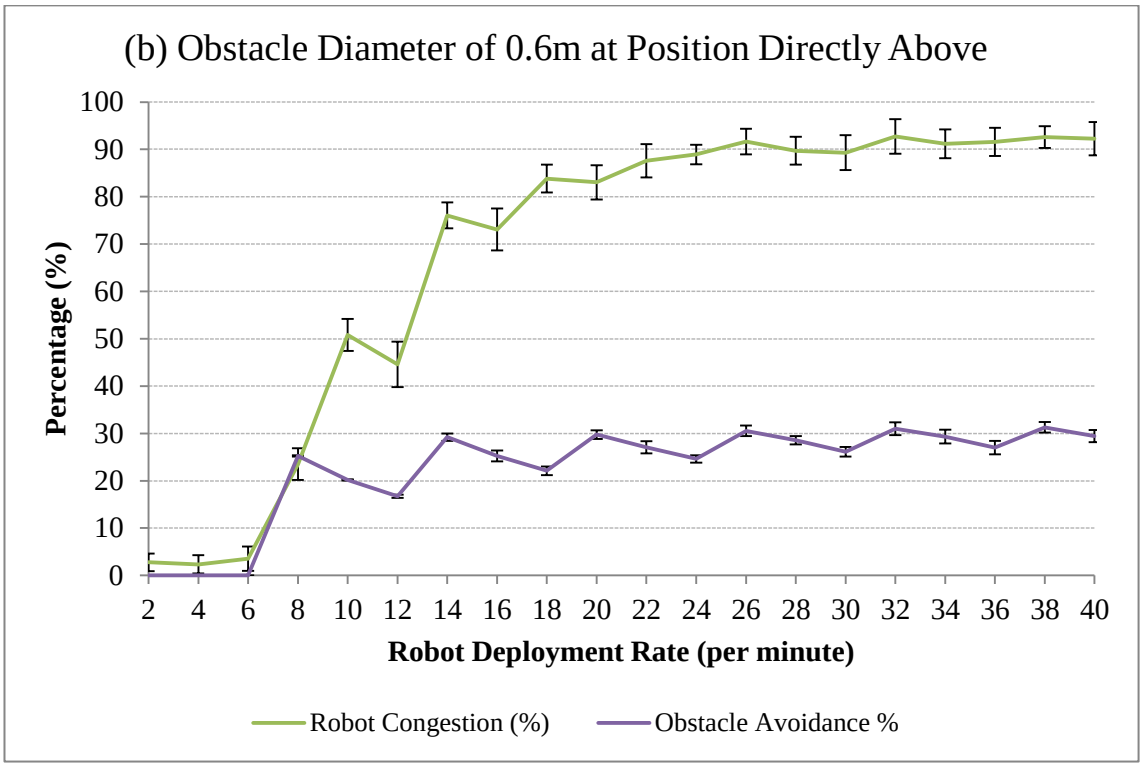
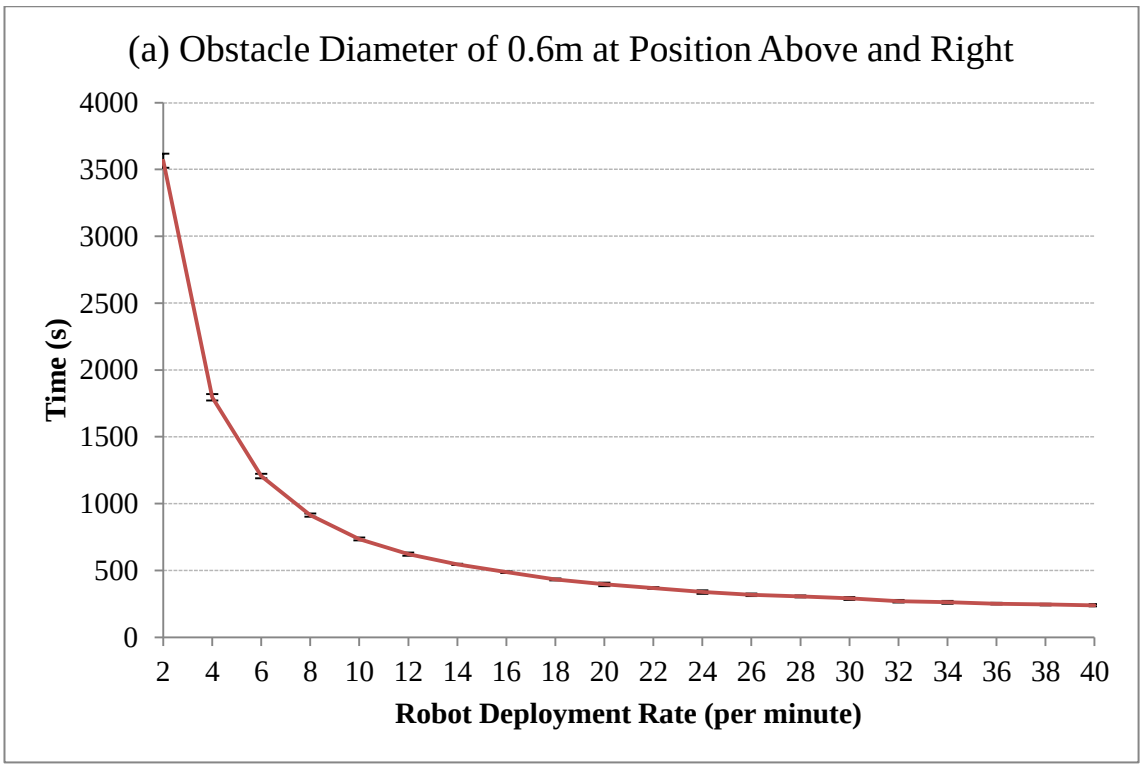




*Fig.4.13. Graphs of results from the obstacle avoidance experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion and obstacle avoidance percentages formed for varying deployment rates. This plot shows the results of the experiments using an obstacle diameter of 0.6 meters at the location directly above the PAP.*



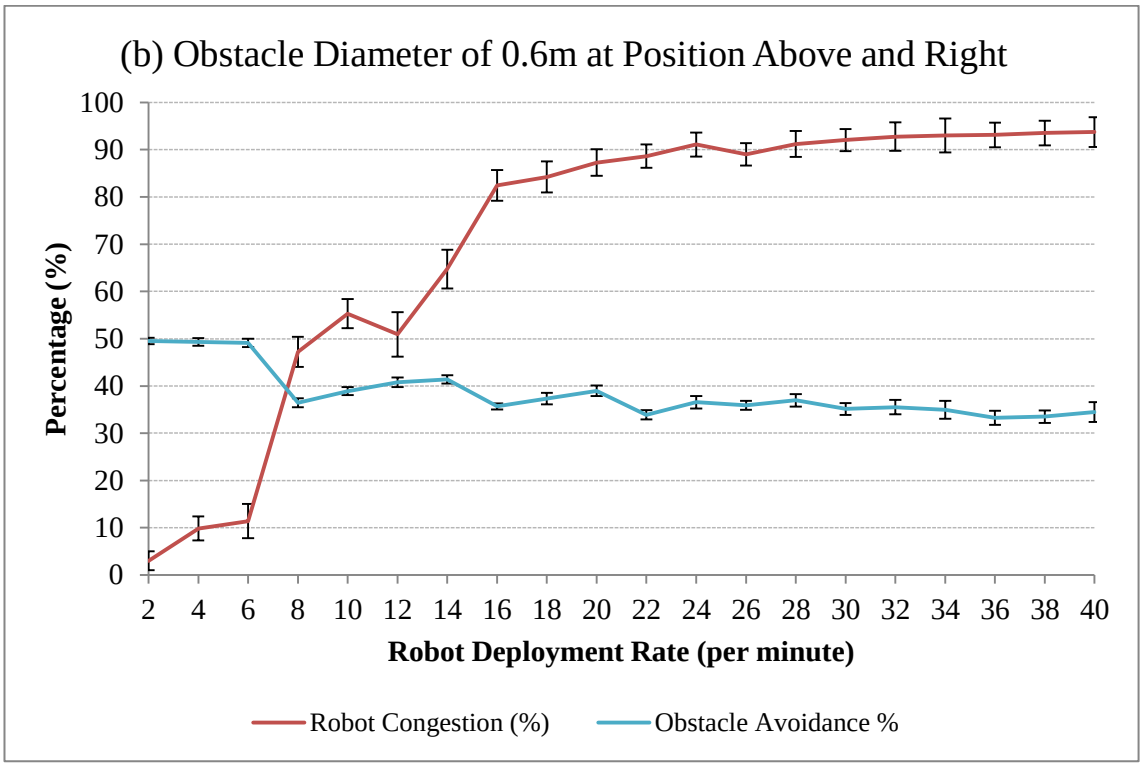




*Fig.4.14. Graphs of results from the obstacle avoidance experiments that shows (a) the time taken for each robot population to complete the self-assembly, and (b) the robot congestion and obstacle avoidance percentages formed for varying deployment rates. This plot shows the results of the experiments using an obstacle diameter of 0.6 meters at the location above and right of the PAP.*

*Table 4.7 Sample results from across all obstacle avoidance scenarios.*

| Obstacle Diameter (meters | Obstacle Location | Robot Deployment Rate (per minute) | Time Taken (seconds) | Robot Congestion (%) | Obstacle Avoidance (%) |
|---|---|---|---|---|---|
| 0.2 | left | 10 | 731 | 22 | 10 |
| 0.2 | left | 20 | 392 | 67 | 15 |
| 0.2 | left | 30 | 285 | 84 | 13 |
| 0.4 | left | 10 | 733 | 49 | 29 |
| 0.4 | left | 20 | 393 | 84 | 24 |
| 0.4 | left | 30 | 283 | 90 | 25 |
| 0.6 | left | 10 | 733 | 52 | 39 |
| 0.6 | left | 20 | 393 | 87 | 38 |
| 0.6 | left | 30 | 288 | 91 | 35 |
| 0.2 | centre | 10 | 731 | 14 | 0 |
| 0.2 | centre | 20 | 391 | 52 | 10 |
| 0.2 | centre | 30 | 283 | 86 | 13 |
| 0.4 | centre | 10 | 730 | 32 | 20 |
| 0.4 | centre | 20 | 389 | 81 | 20 |
| 0.4 | centre | 30 | 282 | 88 | 20 |
| 0.6 | centre | 10 | 733 | 51 | 20 |
| 0.6 | centre | 20 | 394 | 83 | 30 |
| 0.6 | centre | 30 | 284 | 89 | 26 |
| 0.2 | right | 10 | 732 | 22 | 10 |
| 0.2 | right | 20 | 391 | 67 | 15 |
| 0.2 | right | 30 | 282 | 83 | 13 |
| 0.4 | right | 10 | 734 | 51 | 29 |
| 0.4 | right | 20 | 396 | 85 | 24 |
| 0.4 | right | 30 | 288 | 91 | 25 |
| 0.6 | right | 10 | 736 | 55 | 39 |
| 0.6 | right | 20 | 396 | 87 | 39 |
| 0.6 | right | 30 | 290 | 92 | 35 |

## Section 4.5 Discussion

The results from the Section 4.4 identified trends that hold across breach sizes and depths , as shown in tables 4.6 and 4.7, and confirm the hypothesis that increasing the number of robots deployed per minute decreases the time taken for the swarm to aggregate at the PAP and perform the self-assembly, but sees diminished returns due to increased robot congestion. The results also helped reveal that occurrences of robot congestion events can be disrupted by changing the angle of approach the robots use

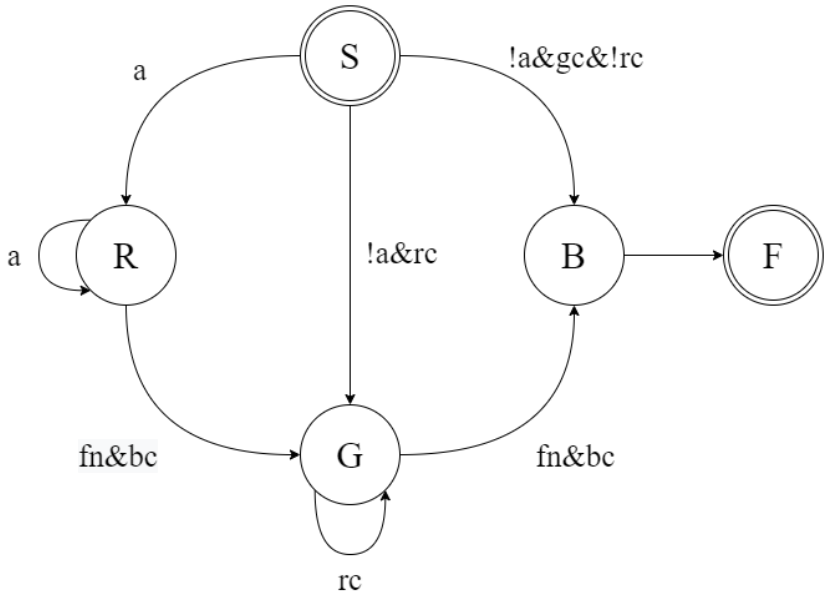
when approach the breach by way of additional obstacles. The diminished returns observed in time taken to complete the repair are primarily due to the overcrowding of robots on approach to the PAP. Using a single assembly point may be appropriate for smaller swarm sizes, as observed from the results which show how fewer robot avoidance events occur when the robot deployment rate is held below 8 robots per minute, but unfit for larger swarms. As the number of robots swarming about the PAP increases, so too does the likelihood of robots becoming stuck near this point and becoming unable to manoeuvre away – leading to blockages which could lead the system to fail. This is in agreement with existing literature which argues that increased congestion in restricted arenas can have a detrimental effect on performance. The following section discusses how we can use this analysis to inform the design guidance of such systems, overcoming the issues identified, and leading to a more reliable method of self-assembly.

There are a variety of ways one could address the issues of overcrowding in these experiments based on the data gathered in Section 4.4. One such method of overcoming the limitation imposed by using a single assembly point would be to introduce multiple alternative assembly points which robots could navigate to if the PAP is too congested. This would allow the robots to better distribute themselves around the hull breach and reduce the number of robot vying for space about a single point, creating a more scalable approach. An extension of this adjustment would be to replace the concept of assembly points with a more general assembly area (GAA), where the robots have more control over their angles of approach and instructing them instead to attach to other robots in the general vicinity to form the necessary patch. However, such a significant change to the approach would warrant a new algorithm that could account for not using a single point, but still using the states of neighbouring robots and hull breach location to determine its next actions. Such an approach is outlined below.

This modified method proposes an alternative self-assembly protocol designed for robots approaching the general location of the hull breach from different directions with varying angle of approach as this better represents the robot distribution likely to follow the ship hull inspection stage of repair presented in Chapter 3. Further to avoiding overcrowding about a single point, this approach is also intended to improve

the efficiency of the approach by making the shape of the repair patch more specific. Rather than assembling to form a predefined square structure of adequate size, this approach instructs the robots to form a structure closer to the shape of the hull damage. The shape of the resultant patch is dictated by the shape of the hull breach such that the hole in the ship serves as a rough template for the robots to copy. This approach is intended as a more efficient use of the robots forming the patch, reducing the number of redundant robots on the perimeter of the hull breach as observed in the previous self-assembly approach.

For this approach to function correctly, the abilities of the robots to communicate and sense their environment must remain unchanged from those outlined in Section 4.1. The robots should still be able to sense the presence of other robots using side-mounted proximity sensors and communicate over short distances using omnidirectional low frequency sonar sysetms, as shown in Fig.4.5. Most improtantly, they should still exchange information about their respective states when physically connected to other robots using their tri-coloured LEDs and corresponding RGB sensors, as shown Fig.4.6. The biggest distinction is the method by which the robots self-assemble to form a structure which mimics the shape of the damage to be repaired. However, this self-assembly approach is still intended to function provided a robot has already located the hull breach, but unlike the previous method the protocol functions well regardless which position the robot occupies relative to the hull damage. The first robot broadcasts its position which gives an indication of GAA which can be anywhere above the hull breach, but most likely on the perimeter of the damage. Figure 4.15 shows a finite state machine of the new approach describing how the robot transitions between the various states to form the repair structure.




| States | | Transitions | |
|---|---|---|---|
| **S** | Approach the general assembly area | **a** | Robot is above the hull breach |
| **R** | Activate red LED | **rc** | Connected to a robot with a red state |
| **G** | Activate green LED | **gc** | Connected to a robot with a green state |
| **B** | Activate blue LED | **bc** | Connected to a robot with a blue state |
| **F** | Finished self-assembly | **fn** | Full neighbourhood of robots confirmed |

*Fig.4.15. Finite state machine and transition table for the Improved self-assembly protocol. This approach forms a repair structure that mimics the shape of the ship hull breach, reducing the number of unessential robots.*

All robots on approach to the GAA begin in the (S) state and have no LEDs active, indicating that they are performing the aggregation behaviour which come before the self-assembly behaviour. When the robot nears the assembly area there are three potential events it will follow. If it enters the GAA and detects that it is above the hull breach (a) it will transition to the (R) state, activating its red LED and begin searching for other robots to connect to in its vicinity. If it enters the GAA but encounters a robot before it detects it is above the hull breach it will examine the state of that robot using its phototransistor. If the robot it detected is displaying a red LED (R), this indicates it has encountered a robot on the outer most edge breach (!a & rc) and will physically attach to this robot, transitioning to the (G) state, activating its green LED. If the robot

encounters a robot before encountering the hull breach in the GAA that is displaying a green LED (!a & gc & !rc), it will physically attach to it and transition to the (B) state, activating its blue LED. If the robot encounters robots in the (B) state around the GAA, it will not attach and instead look for another robot to attach to in the (G) or (R) state.

Once physically connected to a robot and within the GAA, the robots next actions are determined by the number of robots in its Von Neumann neighbourhood and their respective states. If the robot in question is occupying the (R) state, it will continue to hold this state until it has detected a full neighbourhood and one of its neighbours is displaying the blue state (fn & bc), at which point it will transition to the (G) state. Robots occupying the (G) state will continue to occupy this state until the same conditions are met (fn & bc), at which point it will transition to the (B) state. Once all robots have reached the (B) state, it will work to hold its position and state, transitioning to the (F) state to indicate it has fulfilled its function for the self-assembly procedure.

The procedure described above provides an alternative method of carrying out self-assembly while avoiding the pitfalls of overcrowding and handicaps imposed by unfavourable angles of approach which occur when relying on a single point of assembly. Future work could investigate the effectiveness of this new approach in comparison to the original aggregation and self-assembly protocol but in the meantime, the current approach may suffice with respect to how the complete emergency ship hull repair approach could be expected to unfold. The proposed self-assembly protocol that relies on a single point of assembly and varied angles of approach has proved capable of forming a repair patch of adequate shape and size to address hull breaches as originally outlined in Section 4.1. The optimal number of robots to be deployed each minute using this approach has been identified in Section 4.3 and shown to vary according to the breach depth and size.

This chapter marks the end of the current research into emergency ship hull repair using swarm of autonomous underwater robots. Chapter 5 instead moves into the realm of nature-inspired swarm robotics, with focus given to obstacle avoidance in large swarm performing foraging behaviours as discussed in Section 2.7 of the literature review.

# Chapter 5. Collective Foraging Using Nature Inspired Swarm Robots

Foraging, as discussed in Section 2.7, is a behaviour that allows agents in an arena to seek out and retrieve objects of interest and can be described from the individual's perspective as a sequence of tasks: exploration of an environment surrounding a depot, identifying objects and areas of interest, returning objects to the depot, communicating its discovery with others, and returning to areas of interest to collect more objects (Dorigo and Di Caro, 1999). How cooperative teams of agents can work together to achieve more efficient and robust collective foraging strategies is a subject which can prove very beneficial for engineers designing multi robot systems. Distance-quality trade-offs and foraging strategies which can scale well with different swarm sizes are of particular interest.

The study presented in this Chapter is an extension of a study on pheromone-based collective foraging (Font Llenas et al. 2018) and shows how one can achieve a sophisticated collective foraging strategy with minimalist agents using a virtual pheromone system of stigmergic communication and simple wall avoidance behaviour (Talamali et al., 2020). The controllers implemented on the individual robots use simplified binary pheromone sensors but prove to be capable of reproducing classical foraging experiments that used more capable agents that utilise their ability to sense pheromone concentrations and follow gradients. The wall avoidance behaviour is implemented using a similar binary sensor allowing robots to avoid collisions with the bounded arena. A key feature of the controller is a parameter which can be tuned to adjust the selectivity of individual agents comparing the distance of an object of interest from a central depot and quality of the object. The system is examined in the ARGOS simulator (Pinciroli et al., 2018) and verified using a physical swarm of up to 200 robots using the Augmented Reality Kilobots (ARK) system to implement virtual pheromone trails, sensors, and actuators (Reina at al., 2017).

The author's main contribution to research from this study is the implementation of obstacle avoidance behaviour with low computational overhead on a large swarm of robots tasked with collective foraging in environments. This inclusion of obstacle avoidance behaviour solved a major issue where the physical robots would previously

become stuck against the walls of their bounded arena and other robots. By implementing obstacle avoidance, the robots are able to avoid falling into these un-recoverable states, which improves the performance of the swarm, and creates a system more capable of emulating the collective foraging behaviours observed in biological counterparts such as ant colonies.

Section 5.1 presents the morphology of the simulated and physical robots used in the experiments with details of how the virtual sensors and actuators implemented to augment their existing abilities. Section 5.2 discussed the methodology used to create the optimal resource collection model, with details on how it is validated and verified through simulation and experimentation provided in Section 5.3. The results of these simulations and experiments are presented in Section 5.4 and Section 5.5 discusses these results, their implications, and how the wall avoidance behaviour can be adapted to implement robot avoidance which has been shown to reduce collisions and enable more efficient foraging strategies.

## Section 5.1 Problem Definition and Robot Morphology

This study investigates the problem of resource collection in an unknown environment by a swarm of robots with limited computational and memory capabilities. S number of robots occupy an environment which features a central depot of radius 10cm, surrounded by n circular source areas of radius 10cm, each referred to as Ai where i ∈ {1,..., n}. Each of these areas offers an unlimited number of items of varying quality Qi, which is a numerical indication of the importance of the resource with higher values indicating items of greater importance; this system is intended to mimic the nutritional value of items in animal foraging. When a robot enters a source area, it immediately collects one virtual item and returns it to the central circular depot. In this model the handling time of the resource item and time taken to discover items within the source areas are both assumed to be instantaneous. Robot are modelled with the ability to only carry one item at a time between the source areas and central depot and the robot speed when carrying an item remains the same as when not carrying an item. Using higher levels of abstraction for these aspects helps keep the collective motion aspect and allocation of robots to source areas as the focus of the study. The main purpose of this study is to discern how indirect communication, in the form of virtual pheromones, can be leveraged by the robot swarm to balance the trade-off between

quality of resource items and the distance between the central depot and source areas – resulting in more optimal foraging strategies.

In order to carry out the behaviours investigated in this study, the robots must possess capabilities which allow them to operate in an unknown environment. With limited computational and memory abilities, the robots will be incapable of memorising the locations of the source areas and must instead rely on pheromone trails to rediscover these sources. To accomplish this, the robots must be capable of stigmergic communication (see Section 2.7) by applying and reading temporary marks in the environment. The robots are also assumed to always know the direction to the central depot relative to their position, similar to the path integration abilities observed in social insects (Collett and Collett, 2002; Bregy et al., 2008; Heinze et al., 2018). Finally, the robots will be able to detect when obstacles are in front of them but will only use this ability for the purpose of avoiding collisions. This ability to sense objects is not utilised to detect other robots and robots are not allowed to directly communicate amongst each other as this would complicate the results of this study which is primarily concerned with the capabilities of indirect communication only.

The following sections discuss the morphology of the robots selected to serve as foraging agents in both the simulations and physical experiments (5.1.1), how the capabilities of these robots are expanded by using augmented reality to model additional virtual sensors and actuators (5.1.2), and how these sensors are used to implement pheromone trails following and wall avoidance behaviours (5.1.3).

### Section 5.1.1 Kilobots

The robot selected to conduct the simulated and physical experiments is the Kilobot Fig. (5.1): a minimalistic robot which has become a popular choice for swarm robotics research concerning agents with limited physical and cognitive abilities (Rubenstein et al. 2014). Kilobots are designed to move on flat surfaces using a pair of vibration motors to perform a slip-stick differential drive motion. This system allows the Kilobot to move at an approximate speed of 1 cm/s and rotate at a rate of approximately $40°/s$. Though small, Kilobots come with a number of capabilities such as the ability to directly communicate with other Kilobots within 10cm using its infrared (IR) transceiver, display its internal state to others by using its RGB LED, detect changes in

Lux intensity using an on-board ambient-light sensor, and is able to receive messages from an overhead control board (OHC). The OHC is a key component which allows for the augmentation of Kilobots with virtual sensors and actuators (see Section 5.1.2) which enables the agents to perform more complex tasks.

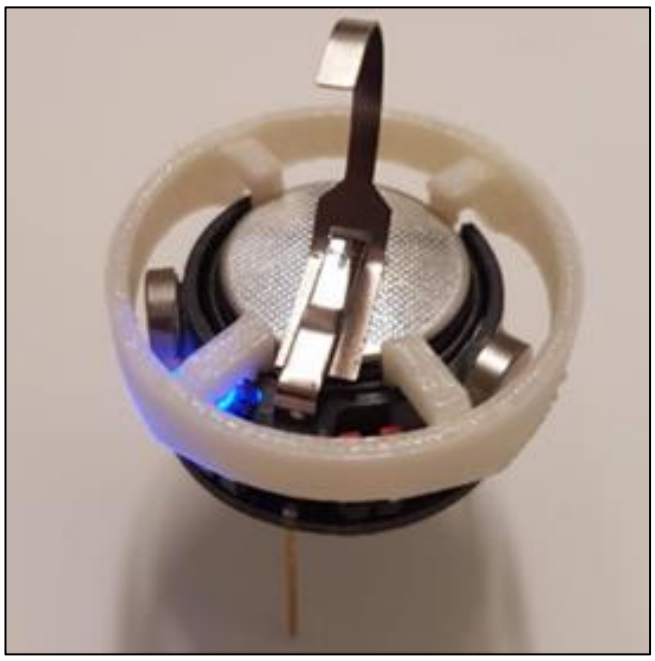


*Fig.5.1. a picture of a Kilobot with a 3D printed ring originally designed for the study of Pratissoli et al. (2019) and reused in Talamali et al. (2020). The addition of the ring significantly improves the ability of the ARK system to track Kilobots and detect their LED colours.*

Simulation of both the Kilobots and the ARK system were made possible thanks to the work from Pinciroli et al. (2018) who successfully developed a plugin for the popular ARGoS simulator. Their plugin accurately captures the behaviour of real Kilobot allowing for faster simulation of behaviours in very large swarms prior to real-world experiments. In simulation, the process of resetting each robot to new start locations and resetting its memory is a very fast simple process, but this is not the case for physical swarms. In real-world experiments, programming each Kilobot individually is a very time-consuming process which significantly slows the rate at which experiments can be run by increasing the time taken to reset each Kilobot. The OHC can be used to overcome this bottleneck by allowing users to quickly program multiple Kilobots through wireless IR communication, significantly reducing the time taken to reset the swarm for each experiment. The Kilobot is a low-cost and easy-to-operate platform, and its simplicity makes it an ideal choice for these experiments which are concerned

with how optimal foraging strategies can be achieved using agents with limited capabilities.

### Section 5.1.2 Augmented Reality Kilobots (ARK)

Kilobots are simple robots with only limited capabilities and while previous experiments have shown that they are capable of recreating common swarm behaviours such as exploration, they require additional functionality to perform more complex behaviours such as stigmergic communication. To extend the abilities of the Kilobot and allow for the modelling of complex behaviours, Reina et al. (2017) and Valentini et al. (2018) implemented open-source technology that could augment the Kilobots capabilities with virtual sensors and actuators, referred to as Augmented Reality for Kilobots (ARK) and the Kilogrid respectively. The ARK system was selected as the augmentation method for the Kilobots in this study because of its low installation cost and ability to perform other tasks such as motor calibration, unique ID assignment and video recording of experiments.

The ARK system uses an overhead camera array to track the Kilobots, an IR-OHC to communicate with the Kilobots, and a computer which serves as a base control station (BCS) to simulate the virtual environment. The virtual sensor information for each Kilobot is computed on the BCS and communicated to the specific robots with addressed messages via the OHC. The virtual actuator information is computed on-board by the Kilobots and is communicated to the BCS using colour-coded messages which it displays using its LEDs. The position and colour of the LED is captured by the overhead cameras and sent to the BCS which processes the information and updates the virtual environment appropriately. The BCS is also responsible for updating the temporal dynamics of the virtual environment such that entropy of pheromone concentration can be accurately modelled. Using the ARK system this way allows each Kilobot to receive personalised information about its virtual sensors in real-time and autonomously decides when to modify the virtual environment using virtual actuators.

The ARK system is used to allow robots to apply and detect virtual pheromone which evaporates and diffuses over time – mimicking the behaviour of stigmergic communication observed in ant species (see Section 2.7). To achieve this, each Kilobot

is equipped with five additional virtual sensors and one virtual actuator. As specified in Talamali et al. (2020) each Kilobot is equipped with the following:

| **Sensor / Actuator** | **Description** |
|---|---|
| Area sensor | The Kilobots can detect if they are within the depot or a source area and distinguish between the two. |
| Item quality sensor | When the Kilobots enter a source area they can assess the quality of the item available for retrieval. When the Kilobots enter the depot, they can recognise the quality of the items that have been collected up to that instant. |
| Depot direction sensor | The Kilobots always knows the direction of the depot relative to their own position. |
| Wall sensor | The Kilobots can sense if there is a wall (obstacle) at a distance of $\sim$ 5cm in front of themselves; note that this ability is not used in this study to detect the presence of other robots (Fig.5.2). |
| Pheromone gland actuator | The Kilobots can deposit a drop of pheromone at their location - they express this behaviour by blinking their top-mounted LED blue. |
| Pheromone antennae | The Kilobots can sense the presence of pheromone at a distance of $\sim$ 3.5cm from their centre in front of themselves. |

The Kilobot swarm operates in a bounded arena which ARK represents as a discrete 2D matrix made up of cells measuring $6.7 \times 6.7mm^2$. ARK stores information about deposited pheromone presence and concentration in each respective cell forming a pheromone matrix which is updated at each time-step of length $\Delta t = 0.5s$ to reflect the evaporation and diffusion of pheromone. The pheromone matrix can be altered at each time step by robots depositing pheromone, where each drop represents of an increment of $\phi = 250$ in the cell under the robot's centre. The concentration of pheromone in each matrix cell $m_{(i,j)}$ is determined by Eq. (5.1) which was first introduced in Talamali et al. (2020):

$$m_{(i,j)} = m_{(i,j)}\left[e^{\log(0.5)\epsilon\Delta t} - 4\gamma\Delta t\right] + \left[m_{(i,j\pm1)} + m_{(i\pm1,j)}\right]\gamma\Delta t \quad (5.1)$$

where the parameters $\epsilon = 0.1$ and $\gamma = 0.02$ are the evaporation and diffusion rates, respectively. Eq. (5.1) is a discrete realisation of Fick's law of diffusion (Fick, 1855), where the exponential term in introduced to take into account pheromone evaporation, which is consistent with biological studies (Garnier et al. 2013).

### Section 5.1.3 Stigmergic Communication and Wall Avoidance

The virtual sensors and actuators of the augmented Kilobots enable them to perform the two behaviours which are essential to completing the foraging task – stigmergic communication and wall avoidance (Fig. 5.2 and 5.3 respectively). The Kilobots are capable of sensing the presence of virtual pheromone in front of themselves at a distance of $\sim 3.5cm$ in four $45°$ wide sectors for a total detection arc of $180°$.The virtual sensors signify the presence or absence of pheromone as binary values and are not capable of discerning the quantity or concentration of the pheromone they are sensing. When a Kilobot who is exploring detects pheromone, this event triggers a change in behaviour causing the Kilobot to abandon its search and instead move towards the detected pheromone as indicated in Fig. 5.2.

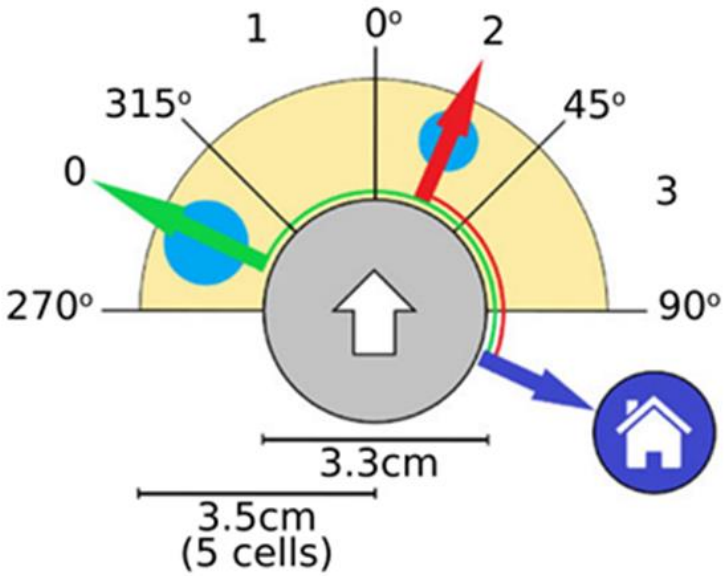



*Fig.5.2. This diagram illustrates the Kilobot pheromone detection system via the ARK as introduced in Talamali et al. (2020).*

Each Kilobot can detect the presence of pheromone of any of its four forward facing $45°$ wide sectors. In Fig. 5.2, the virtual pheromones are represented as blue circles

and because their presence is represented by the Kilobot as binary values, its virtual sensor readings would effectively read as [1, 0, 1, 0]. In the illustration, the robot has encountered pheromone in two of its detection sectors and must choose one direction to follow. It does this by comparing the angles between each sector in which pheromone is present and the direction of the depot (the depot being represented by a white house within a dark blue circle) and choosing to follow the sector which has the largest angle of difference in direction (represented as red and green arrows). The pheromone in sector 0 has a greater angle of difference than the pheromone in sector 2 and so the robot chooses to move in the direction of the pheromone in sector 0.

The Kilobots are also able to detect the presence of obstacles in front of themselves at a distance of $\sim 5cm$ in four $45°$ wide sectors for a total detection arc of $180°$. Similar to the pheromone detection system, the presence of obstacles in each sector is represented as binary values and is used to indicate when the robot has encountered a wall, but not other robots. This is used by the robot to avoid collisions with obstacles that fall between itself and either the depot or the source area.

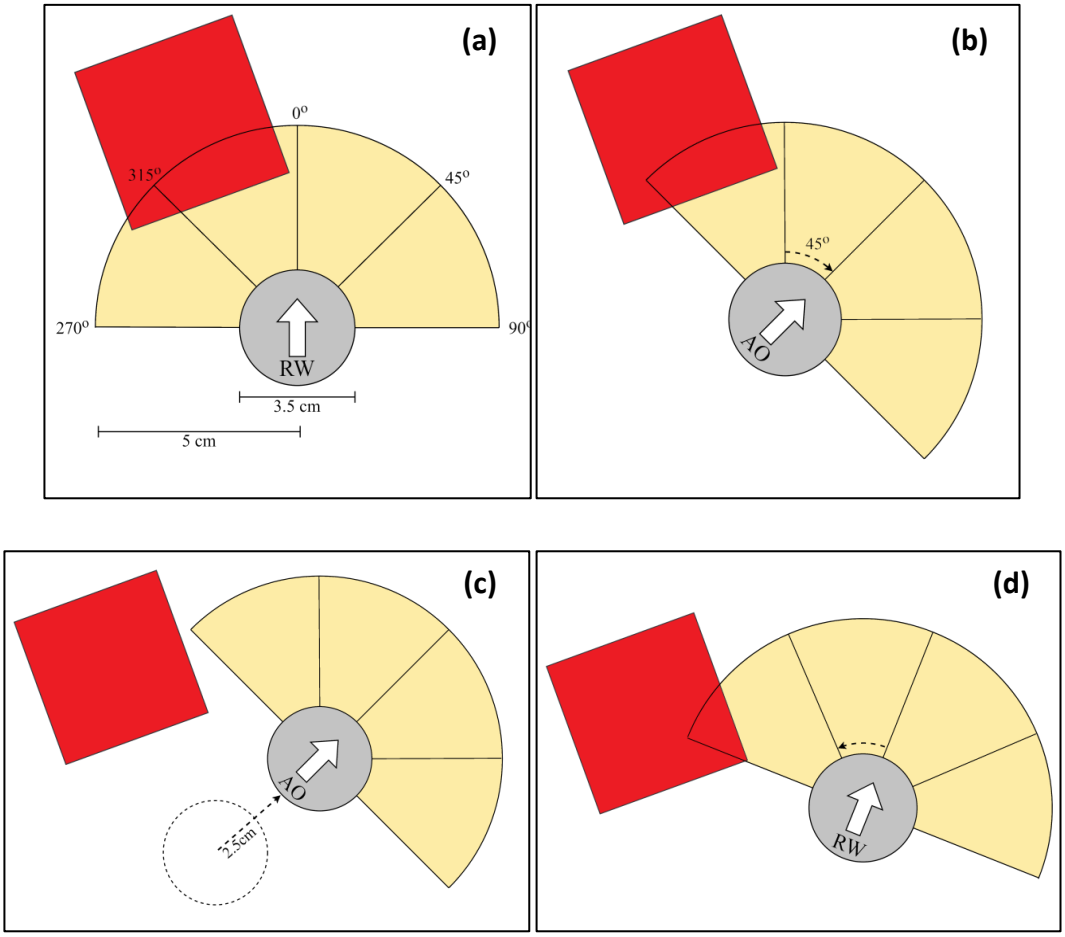


*Fig.5.3. This diagram illustrates the Kilobot wall avoidance system via the ARK.*

When a wall is detected by the two central sectors in the range $[-45°, 45°]$ of the robot's heading the robot will turn left or right at an angle of $22.5°$ in the opposite direction of the sensed obstacle until an obstacle is no longer detected in the two central sectors, at which point the robot will then move in a straight line for $2.5s$ and then return to its previous task of item retrieval, or exploration. Obstacles are detected as grey squares (represented in Fig 5.3 as a red square) and similar to the detection of pheromone, the Kilobot represents this obstacle presence as binary values, such that its virtual sensor readings would be [1, 1, 0, 0] in (a). In the illustration, the robot has encountered an obstacle in one of its central sectors (AO state) while exploring the environment (RW state) and must choose an appropriate direction to turn and avoid a collision. It does this by comparing the presence of obstacles in the two central sectors in the range: $[-45°, 45°]$ of its current heading. There is an obstacle detected in sector 1 and not sector 2 and so the robot chooses to rotate right in steps of $22.5°$ until no obstacle is detected in the two central sectors (b), and then moves in a straight line for $2.5s$ (c), after which it will return to its previous task of exploration (d). Both stigmergic communication and wall avoidance behaviours are better characterised by the probabilistic finite state machine (PFSM) used to describe the individual behaviour of the Kilobots (Fig.5.4) and discussed in more detail in the following Section 5.2 on the methodology.

## Section 5.2 Methodology

Section 5.2.1 explains how the desired foraging behaviour can be described at the microscopic level using a probabilistic finite state machine (PFSM) to represent the individual robot behaviours, as illustrated in Fig. 5.4. The main structure of the behaviour is based on the control scheme designed by Font Llenas et al. (2018), but has been enriched by the inclusion of a new obstacle avoidance state (indicated as AO in Fig. 5.4). The abilities of the individual have been further enhanced by including an additional form of indirect communication which enables adaptability to different item qualities (Section 5.2.1), and by allowing for probabilistic transitions and tuneable pheromone functions (Section 5.2.2).

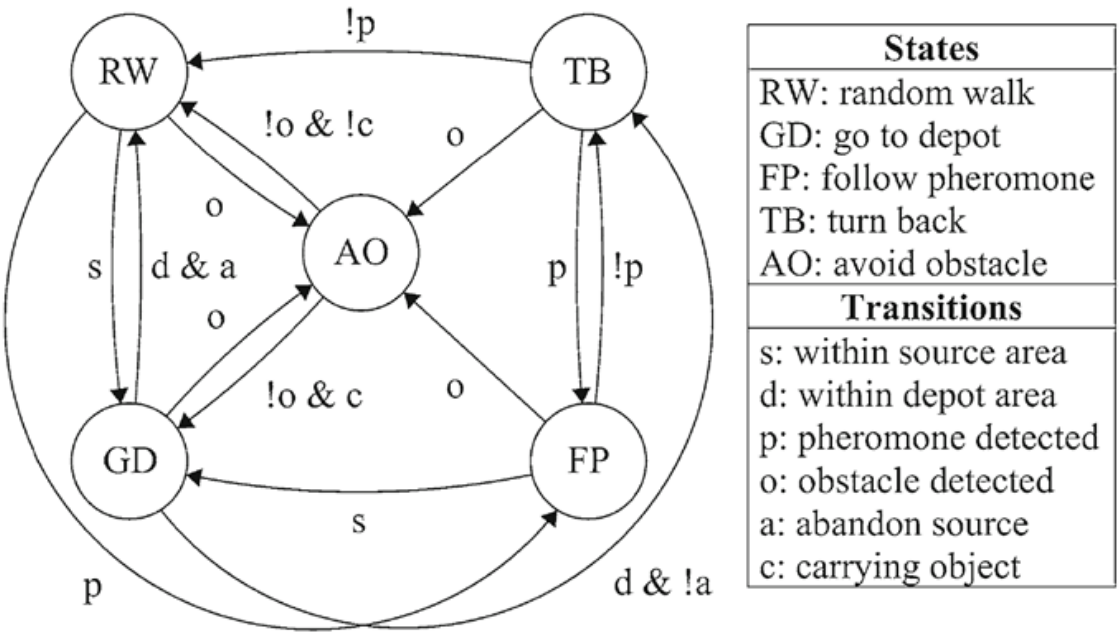



*Fig. 5.4 Probabilistic finite state machine (PFSM) of the individual robot behaviour from Talamali et al. (2020). Circles represent states and arrows represent transitions. The labels for each state and transition are listed in the table adjoined to the diagram.*

### Section 5.2.1 Individual Behaviour

In all the examined scenarios, the robots do not possess a-priori knowledge of the environment surrounding the depot such as the number of items, their location, or item qualities. As such, the first action of the robots is to begin exploring the environment to discover item source areas (RW state in Fig.5.4). The Kilobots' limited capabilities (Section 5.1.1) prohibit the use of more complex coordinated search patterns such as those discussed in Section 2.5, and instead force the robots to rely on using an isotropic random walk to explore. Though less efficient than coordinated exploration, random walks provide a simple but effective method of searching for targets in unknown environments (Dimidov et al. 2016) most suitable to simple robot swarms.

When executing the random walk, the robots will alternate between travelling in a straight motion for 10 seconds, and performing a uniformly random rotation between$[-\pi, \pi]$; a pattern which repeats until the robot encounters an event which triggers a transition to a new state. One such event is when a robot encounters a source area, which causes the robot to (virtually) pick up an item and begin transporting it to the central depot (GD state in Fig. 5.4). As outlined in Section 5.1.4, The Kilobots are assumed to possess limited memory and are only ever able to

remember the direction towards a single location in the search space, and in these scenarios this location is the relative direction to the depot. This assumption of limited memory is consistent with the behaviour observed in several ant species which use path integration to return to their nest (Collett and Collett 2002; Bregy et al. 2008; Heinze et al. 2018).

The robots use their memory of the depot location to always find their way back when returning items they have collected. To memorise other points of interest in the environment surrounding the depot, the robots use their virtual pheromone as a form of stigmergic communication, creating form of collective memory. When a robot is returning to the depot from a source location, it deposits virtual pheromone to allow itself and other robots to rediscover the source area. Robots in the GD state perform a probabilistic function Eq. (5.2) every four seconds which determines the rate of pheromone deposition based on the quality of the collected item, as described in Section 5.2.2. When the robot reaches the depot, it deposits its item and executes another probabilistic function (see Eq. (5.3)), based on a comparison between the quality of its recently deposited item and the highest quality item any robot has deposited so far, as described in Section 5.2.3. This function determines whether the robot will turn to follow the pheromone trail it recently created (TB state in Fig. 5.4.), or to abandon the previously discovered source area and resume exploration though random walk (RW state).

When any of the Kilobot sense virtual pheromone via their virtual antennae, composed of the four sectors as described in Section 5.1.3, they immediately shift to follow the trail (FP state in Fig. 5.4.), moving in the direction of the triggered antennae sector until they reach either the end of the trail or an intersect between two of more trails. If a robot detects pheromone in more than one direction, e.g. both left and right sectors as in the illustration of Fig. 5.2, the robot will compare the sensed-pheromone directions with the directions to the depot (red and green angles in Fig. 5.2) and moves in the direction of the pheromone most opposed in direction to the depot (green arrow in Fig. 5.2). This decision relies on the assumptions that robots only deposit pheromone in their straight path from a source area to the depot and that they always have access to the depot vector. As detailed in Section 5.1.3 and illustrated in Fig. 5.3, the robot behaviour has been enriched through the inclusion of an obstacle avoidance

state (AO state in Fig. 5.4), allowing the robot to detect when it is in close proximity to obstacles. However, in these experiments this ability is only utilised to detect when it is in proximity to walls.

**Section 5.2.2 Adaptability to Different Item Qualities**

As mentioned in Section 5.2.1, the robots do not possess prior information about the environment including the item qualities available to be collected. When the system is initialised, a maximum item quality has not yet been established for robots to compare the quality of their collected items to. As such, they are left to initially assume the quality of the item they have collected is the global maximum. However, the value of this global maximum quality ($Q_{max}$) can be updated over time according to the highest quality item returned to the depot by any robot, which can be used to tune the robot behaviour to what was initially an unknown quality range. This indirect communication method, where the robots compare the quality of their items with the global maximum, occurs within the depot.

Each time a robot enters the depot, it compares the value of the item it is carrying with the highest item quality collected up to that point. If the quality of the item is it carrying is greater than that of the current maximum, it will update $Q_{max}$ to reflect that its item quality is the highest collected. If the quality of its collected item is less than the global maximum, this will increase the probability that the robot will abandon its current source in order to seek the source of the high quality items. This mechanism of quality comparison parallels the behaviour of social animals where individuals can assess the nutrient quality of the swarm's reserves and compare it to their own collected items (Dussutour and Simpson 2009; Arganda et al. 2014).

In these studies, unlimited item sources are utilised to investigate the steady state regime; however, in cases of limited sources (i.e. with a limited number of items) the robots may update their quality range by only observing the latest collected items. In this way, the swarm is predicted to be able to flexibly adapt to appearances or depletions of available sources.

**Section 5.2.3 Adaptable Behaviour from Tuneable Functions**

When robots are returning items from a source area to the depot, they lay a pheromone trail. These trails of pheromones serve as temporary paths between source areas and the depot, indirectly communicating the locations of discovered source areas to other robots. The contribution of each robot in creating and maintaining these paths proliferates to create a form of collective memory, allowing the swarm to recall the locations of source areas in the environment. This system is what allows robots which cannot internally store source locations to overcome their individual limitations and return to previously discovered source areas. These trails are created and detected via the virtual pheromone glands and antennae of the robots as described in Section 5.1.2. Similar to the approach of Font Llenas et al. (2018), robots returning with items probabilistically decide every four seconds whether or not to lay the next drop of pheromone. In their approach, the probability of pheromone deposition was dictated by a linear function relating the item quality to the global maximum, i.e.$P_\phi(Q_i) = \frac{Q_i}{Q_{max}}$, which allowed the swarm to give priority to higher-quality source areas. In this study, a tuneable function is implemented to allow robots to regulate their selectivity on quality using a single parameter α ≥ 0. The probability that the robot will deposit its next drop of pheromone is given by Eq.5.2 which was introduced in Talamali et al. (2020):

$$P_\phi(Q_i) = e^{\alpha(Q_i - Q_{max})Q_i^{-1}} \tag{5.2}$$

Each individual robot has access to α and can alter this value to vary the global response. Values of $\alpha > 1$ cause the function to have an exponential shape on $Q_i$, resulting in highly selective behaviour in favour of the highest quality sources. A value of $\alpha \approx 1$ leads to an approximately linear response which is similar to the function investigated in (Font Llenas et al. 2018) such that Eq. (5.2) can be used as a generalisation of the previous specific function. Finally, decreasing α when α < 1 gradually flattens out the function to a constant value, so that when the limit of $\alpha = 0$ becomes constant $P_\phi(Q_i) = 1$; this results in constant pheromone trails irrespective of the quality of items within the source areas.

Up to this point only the quality of the items has been considered when choosing the probability of pheromone drop deposition. However, in social insects it has been

observed that the distance of the different source areas from the nest is also a factor which can be used to better determine which route will generate the greatest energy gain [i.e. foraging ants (Shaffer et al. 2013) and house hunting honeybees (Seeley et al. 2012)]. For instance, in scenarios where a high quality object is significantly further away than a closer lower quality item, the closer item may net the higher energy gain and be considered a superior option, or if the path to the best source is overcrowded a less crowded path to a different source may be a better option. To balance this distance-quality trade-off, the individual robot capabilities are expanded to include a decay function $P_d(t_i)$ in Eq. (5.3) which is called when the robot enters the depot to deposit an item, helping them to decide whether to continue exploiting the same source or abandon it and begin searching for new sources. The travel time $t_i$ is measured by the robots as the time spent between the item collection (from the source $A_i$) and the item deposition (in the depot). The function $P_d(t_i)$, similarly to $P_\phi(Q_i)$ of Eq. (5.2), is modulated by the parameter α as:

$$P_d(t_i) = (\alpha + 1)^{-2} e^{\frac{t_i - t_{max}}{(\alpha+1)\sqrt{t_i}}} \tag{5.3}$$

where $t_{max}$ is a parameter indicating robot's prior knowledge on the maximum acceptable time to return from a source. The $t_{max}$ parameter could be adaptively tuned (similarly to $Q_{max}$ in Sect. 5.2.2), however this aspect falls beyond the scope of this study and instead the value of $t_{max}$ is fixed at $100s$. Using a fixed value of $t_{max}$ can be considered reasonable as agents in both biological and artificial systems will only consider sources areas that are within a certain maximum distance of the depot which is decided a priori. This distance can be decided for instance using the length of time a robot can remain operational before needing to return to a depot to recharge, or the distance a robot may travel before encountering a wall of a bounded arena.

Eq. (5.2) and Eq. (5.3) are linked by the parameter α which the robots can regulate to alter the swarm behaviour. Increasing $\alpha > 1$ has the combined effect of increasing discriminability on quality $Q_i$ and flattening $P_d(t_i) \approx 0$ for any distance; In this scenario the swarm chooses to ignore the distance of source areas but is highly selective based on higher-quality source areas. Conversely, small values of $\alpha < 1$ flattens out quality differences $P_\phi(Q_i) \approx 1$ and accentuates differences on travel time with an exponential abandonment $P_d(t_i)$ on high travel times; this leads to a system

where the only discriminating factor on source selection is distance due to a combination of evaporation and abandonment of sources which are further away. Finally, intermediate values of $\alpha \approx 1$ give a quasi-linear response of $P_\phi(Q_i)$ and sublinear $P_d(t_i) > 0$ which allows the swarm to balance the distance-quality trade-off similarly to what has been reported in Font Llenas et al. (2018).

## Section 5.3 Experimental Setup and Model Prediction

In this section, the parameters used to assess whether the swarm of robots are achieving an optimal approach to resource collection are explained. The mathematical models presented in each subsection tie these qualities together and are inspired by general aspects of optimal foraging theory (Kacelnik 1984; Houston and McNamara 2014). The model is used to determine the effectiveness of the system by comparing the benefits gained from the resources gathered with the cost incurred from transporting these items to the central depot. Section 5.3.1 introduces the three main components of the model and the resultant equation used to predict the performance of the system in each experiment. Section 5.3.2 presents the various environment configurations used to assess the system and confirm the optimality of the resource collation model.

### Section 5.3.1 Model of Optimal Resource Collection

The three main components of the model are the quality of the items retrieved, the number of robots dedicated to each available source area, and the time taken the travel between the respective source areas and the central depot. The number of robots allocated to a source area is modelled as $\rho_j$ (with $j \in \{1, \cdots, n\}$) which represents the fraction of the total robot population currently on the trail between the central depot and source area $A_j$. The robots that are actively transporting items between any of the $n$ source areas are referred to as workers and their fraction of the robots currently on a trail is denoted by $\rho_w = \sum_{j=1}^{n} \rho_j$. The remaining robots that are exploring the arena are called explorers and their fraction is denoted as $\rho_e = 1 - \rho_w$.

The time taken for robots to travel between the central depot and their allocated source area is determined by the distance between the source and the depot, and the level of congestion on their respective trail; highly congested trails typically lead to increased collisions between robots which results in longer travel times. Talamali et al.

(2020) combined these qualities in Eq. (5.4) to create the swarm yield variable $R$ which represents the net gain of the system. This value can be used to clearly correlate the performance of the system with how it allocates robots to different sources under varying environmental conditions – helping to determine which approach represents the most optimal foraging strategy.

$$R = \sum_{j=1}^{n} \frac{q_j \beta_j \rho_j S}{\tilde{d}_j^2}, \qquad (5.4)$$

$$with\ \tilde{d}_j = d_j + v_o T_{C,j}(\rho_j S)$$

where S is the swarm size, $q_j = \frac{Q_i}{Q_{max}}$ is the normalised quality of source area $A_j$, $\rho_j$ is the fraction of robots on the trail between central depot and source area $A_j$, $\beta_j$ is a fitting parameter characterising the relationship between the number of collected items from source $A_j$ and the number of robots on the trail to $A_j$. The parameter $\tilde{d}_j$ represents the sum of parameter $d_j$, which is the distance between source area $A_j$ and the central depot, $v_o = 1cm/s$ which is the Kilobot's speed (fixed parameter), and the function $T_{C,j}(\rho_j S)$ which models the additional travel time arising from traffic congestion. This equation models traffic congestion as an increase of the travel distance $d_j$ by accumulating the additional length of $v_o T_{C,j}(\rho_j S)$.

The design parameters used to obtain the function $T_{C,j}(\rho_j S)$ were collected from physics-based simulation data as described in Appendix A of Talamali et al. (2020), which provides full details of how these parameters were derived. The details surrounding the derivation of traffic congestion model are purposefully omitted from this Chapter as this aspect is primarily a contribution of the co-author Salah Talamali and does not represent the main contribution presented in this study, i.e. obstacle avoidance behaviour implementation.

The experiments undertaken in this study consider cases of resource collection in environments with $n = 2$ source areas, in order to study the basic properties of the yield function in Eq. (5.4). The aim of the swarm robot system is to optimally allocate the population of robots between the two source areas to maximise the yield R. To simplify the assessment of the basic properties, all robots are assumed to be actively involved in resource collection (i.e. all robots are workers and none are explorers

($\rho_w = 1, \rho_e = 0$); where the fraction ρ1 = ρ is the robots collecting items from source $A_1$, and the fraction ρ2 = 1 − ρ represents robots collecting items from source $A_2$. With these assumptions, the yield function can instead be given as:

$$R(\rho) = R_1(\rho) + R_2(\rho)$$

$$where\ R_1(\rho) = \frac{q_1 \beta_1 \rho S}{\tilde{d}_1^2},\ \ R_2(\rho) = \frac{q_2 \beta_2 (1-\rho) S}{\tilde{d}_2^2} \tag{5.5}$$

The purpose of this function is to examine how the swarm allocates its resources, so the dependency of R on ρ is explicitly mentioned in Eq. (5.5); this helps in determining the optimal value of 0 ≤ ρ ≤ 1 that will maximise the yield. In Talamali et al. (2020) they found that increasing ρ, where ρ ∈ [0, 1] lead to the following outcomes:

| Variation of ρ | Result |
|---|---|
| Monotonic increase of $R(\rho)$ until $\rho = 1$ | Workers converge on global maximum - all workers allocated to source area A1 |
| Monotonic decrease of $R(\rho)$ until $\rho = 0$ | Workers converge on global maximum - all workers allocated to source area A2 |
| $0 < \rho < 1$ | Workers are split between 2 local maxima (one of which is also the global maximum). |

In the following section, the equations for swarm yield are used to predict the system performance in each of the experiments. The experimental setups used to study the effects of varying environment parameters, such as item quality and source distance, on the yield function are also described. The first scenario looks at source areas of equal distance and different item quality, the second scenario uses equal item qualities and varied source area distances, and the final scenario predicts the critical swarm size in an equal distance and item quality scenario.

### 5.3.2 Equal Distances and Varying Qualities

As indicated in the previous section, overcrowding on the trails between the depot and source areas can lead to congestion which negatively affects the swarm yield. When source areas are both equally close to the depot the risk of overcrowding increases and moving the source areas further away from the depot decreases the risk of

overcrowding, such that when the source areas are sufficiently distanced from the depot, overcrowding effects are negligible. The first setup examines both scenarios with $n = 2$ source areas which are equally far from the depot or equally near the depot, and where the item quality of source area $A_1$ is held constant at $q_1 = 1$, and the item quality of source area $A_2$ is varied at $q_2 \in \{0.5, 0.75, 1\}$. The results from these experiments (Fig. 5.5) reveal that when the sources are relatively far (Fig. 5.5 (a)), it is optimal to allocate all workers to the better-quality source area, whereas for source areas in close proximity (Fig. 5.5 (b)) the yield is maximised if the trail between the higher-quality option and depot does not become overcrowded.

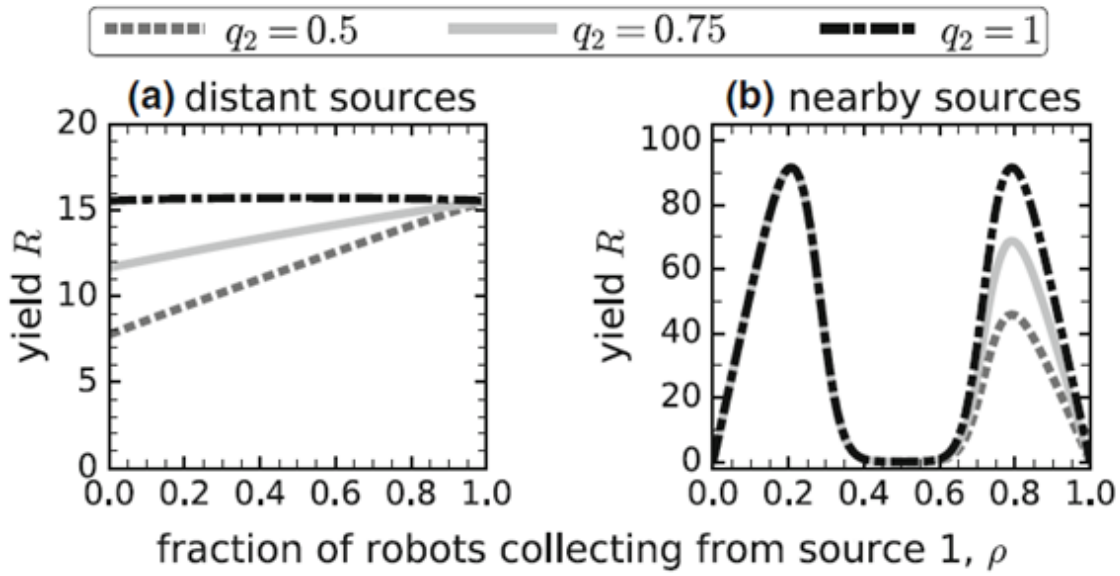



*Fig. 5.5 Model predictions of yield R depending on worker allocation ρ for: (a) equally distant sources where $d_1 = d_2 = 3.5m$; and (b) equally nearby sources where $d_1 = d_2 = 0.6m$. The parameter values selected for this experiment were:$\beta_1 = \beta_2 = \bar{\beta} = 0.965$, $T0_1 = T0_2 = \overline{T0} = 0.029$, $\kappa_1 = \kappa_2 = \bar{\kappa} = 2.321$, and $S = 200$.*

In scenarios where the qualities of the items available in each source area are different it may seem intuitive to allocate all workers to collect from the higher quality source. However, allocating workers in this way tends to lead to increases risk of robot collisions and overcrowding which we know increases congestion and reduces the overall yield of the system. This is especially prevalent in scenarios where the source areas are equally near to the depot and the risk of overcrowding is already elevated. This means there is a limit to the efficiency of the robot swarm collecting the items

which is dependent on the total number of workers, their size, and the space available of the trail between the source areas and depot.

Fig. 5.5 (a) shows that for sufficiently large distances between source area and depot, where the risk of overcrowding is significantly lower, it is indeed optimal to allocate all workers to the source area containing higher-quality items. If the quality of the items available in both source areas is equal, then the yield from exploiting each source is marginally larger if both source areas are exploited equally. However, in cases where both source areas are near the depot (Fig. 5.5 (b)) the optimal strategy changes. Here the best strategy is not to equally exploit both resources, but instead to minimise traffic congestion on the trail between the depot and the sources which have the highest quality items (low ρ in Fig.5.5 (b)). The system may achieve this by allocating a higher fraction of the workers to the lower quality item source area. Interestingly, this remains the best strategy for maximising yield even when the quality of the items available from both source areas is equal.

### 5.3.3 Equal Qualities and Varying Distances

The second experimental setup examines cases where both of the available source areas contain items with equal quality, but the distance between the source areas and the depot are different – this will help determine how the yield R is affected by varying the distance. In Fig. 5.6 the graph plots the corresponding yield function for equal qualities $q_1 = q_2 = 1$, a fixed swarm size of S = 200 robots, a fixed distance of the first source area $A_1$ where $d_1 = 0.6m$, and varying the distance of the second source area $A_2$ where $d_2 \in \{0.3m, 0.6m, 0.9m\}$. Fig 5.6 shows how overcrowding of the trail affects the swarm yield R and reveals that the optimal strategy is to allocate the majority of the robots to which ever source area is the furthest from the central depot. This effectively reduces the congestion experience on the closer source area, resulting in fewer collisions which could slow the progress of the workers, and increasing the yield.

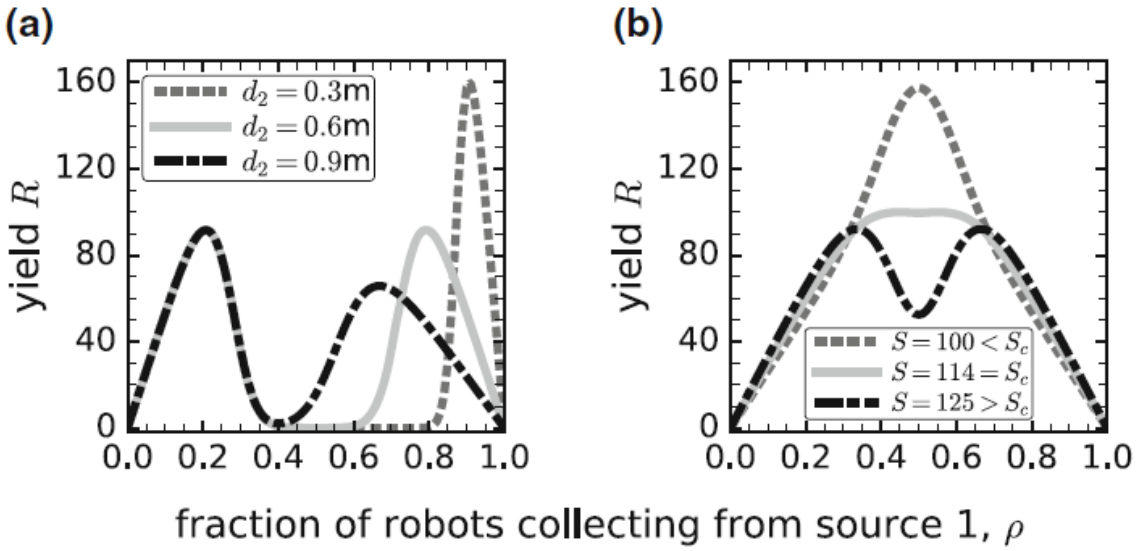



*Fig. 5.6 Model predictions of swarm yield R depending on workers allocation ρ for equal qualities* $q_1 = q_2 = 1$. *The parameter values selected for this experiment are the same as the first setup at:* $\beta_1 = \beta_2 = \bar{\beta} = 0.965,\ T0_1 = T0_2 = \overline{T0} = 0.029,\ \kappa_1 = \kappa_2 = \bar{\kappa} = 2.321$, *and* $S = 200$.

Fig.5.6 (a) shows the effect of varying the distance of the second source area on R, and Fig.5.6 (b) shows the effect of varying the swarm size S with respect to the critical swarm size $S_c$. The effect of overcrowding has a significant effect on the efficiency of the swarm, where it is only possible to attain the maximum yield when a limited number of workers (10–20%) collect from the nearest source area in order to reduce congestion on that trail. The critical swarm size Sc characterises the effect of overcrowding, i.e. when the swarm is sufficiently large ($S > S_c$) it is optimal to keep at least one path with less than 50% workers; otherwise, the effect of overcrowding begins to decrease the income of resources on both paths. The expression used to obtain the value of the critical swarm size and detailed analyses are provided in Appendix C of Talamali et al. (2020).

From the models, it is possible to determine the optimal foraging strategy for different robot population sizes relative to the critical swarm size, assuming the source areas are equally distant from the depot, and their item qualities are the same. If the number of robots exceeds the critical swarm size ($S > S_c$) the optimal strategy is to allocate more robots to one of either available sources, as collection from either source would give the same reward and incur the same cost. The main aim of this strategy is to avoid overcrowding both paths to increase the yield of robots on the less populated trail. In

scenarios where $S < S_c$, the maximum possible yield for swarms where $S > S_c$ is smaller due to the increased prevalence of overcrowding which leads to less efficient foraging. This highlights the importance of controlling the number of workers for the purpose of maximising the global intake; a strategy which is implemented in a decentralised fashion by ants (Charbonneau et al. 2015; Pagliara et al. 2018), and recently investigated in the context of swarm robotics (Mayya et al. 2019).

## Section 5.4 Results

The performance of the proposed system was primarily studied by using physics-based simulations of a variety of experimental conditions. Experiments using up to 200 physical Kilobots were conducted to validate the Kilobot behaviour in a bounded arena that matched the central depot, multi-source area simulated environment with no obstacles. The physics-based simulations were conducted with ARGoS (Pinciroli et al. 2012, 2018) which is a state-of-the-art swarm robotics simulator that accurately and efficiently simulates the Kilobots and the ARK system via a dedicated plug-in (Pinciroli et al. 2018).

The physical robot experiments were run with fully charged Kilobots whose motors have been automatically calibrated through ARK (Reina et al. 2017). The results presented within this section mainly pertain to the simulations as this is where the wall avoidance behaviour and adaptability function were implemented with respect to additional obstacles in the environment. In the physical experiments, the wall avoidance behaviour mainly serves to prevent Kilbots becoming stuck on the boundary of the arena which was a common occurrence before the introduction of this capability and affected the outcomes of the previous experiments. Details of the physical set-up, experimental results and analysis can be found in Talamali et al. (2020). Section 5.4.1 presents a set of the simulation results that highlight the benefits of having introduced a virtual wall sensor, adaptability to unknown environmental scenarios, and behaviour modulation to balance the distance-quality trade-off. The robot simulation code is open source and available online at:
https://github.com/DiODeProject/PheromoneKilobotSwarmIntell

Videos of the physical experiments, augmented by superimposing the virtual environment, are available online at:
https://www.youtube.com/playlist?list=PLCGKY9OHLZwMaGeB6cxVfxmHwhBFqKF7a

### Section 5.4.1 Tuneable and Adaptive Swarm Response

In this subsection, we report the results from the simulated and real robot experiments to provide evidence of the swarm behaviour obtained using obstacle avoidance, adaptability, and individual function modulation.

The inclusion of obstacle avoidance behaviour allows the robots to navigate in more complex environment where there are obstructions between the depot and the source areas. A set of simulated experiments were conducted in order to demonstrate this ability and prove the swarms' capability of selecting the most optimal foraging strategy given various choices. Fig. 5.8 (b) shows a screenshot of the experiments which we inspired by the well-known study of Goss et al. (1989) which showed that ants are able to exploit the shorter path in double-bridge experiments – scenarios in which there are multiple paths between a source and depot with different lengths. In the experiments reported herein, the robots possess less cognitive capability than individual ants and are unable to distinguish between difference pheromone intensities, follow gradients, or make decisions based on differences in pheromone concentration. However, the results show that the robots still display a preference for the shortest path available, which demonstrates their ability to determine the most optimal strategy even with limited information.

It is important to note that this outcome was not limited to conditions where the rate of pheromone evaporation was too high to maintain the longer path but establish the shorter path. The robots still showed preference for the short path in scenarios where both the longer path and shorter path were complete and viable. Following the initial experiment, the environment was modified to block the shorter path and only allow the longer path to remain as an option. As shown in Fig. 5.8 (a), the robots were still able to exploit the best available path. Double-bridge experimental setups have been emulated in other swarm robotics studies such as Montes de Oca et al. (2010) and Scheidler et al. (2016) though the swarm behaviour and desired goals under investigation were different that this study.

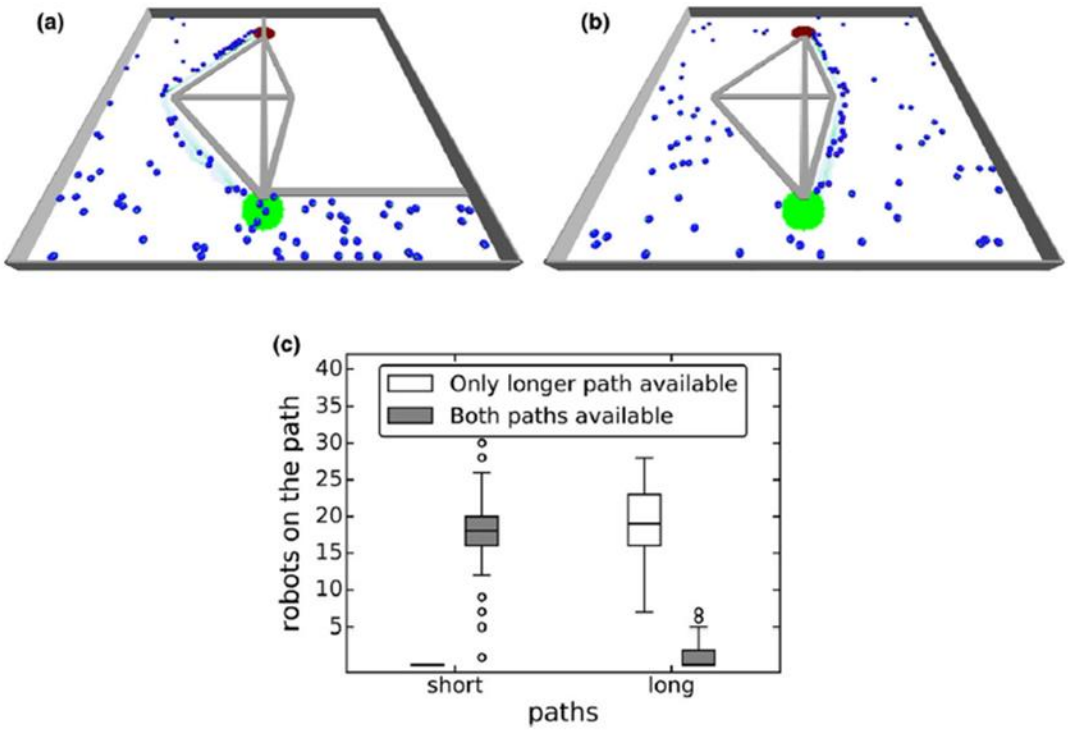



*Fig. 5.8. The Final distribution of Kilobots from the simulated 50 Kilobot swarm experiment inspired by the ants' double-bridge experiment by Goss et al. (1989), originally published in Talamali et al. (2020), in which two paths (a long 1.8m path and a shorter 1.4m path) connect a single source to a single depot.*

When the swarm only had access to the longest path (Fig. 5.8 (a)) the Kilobots using the path to collect items reinforced the concentration of pheromone using repeated trips for their collections. However, when both paths were available (Fig. 5.8 (b)), the Kilobots disregarded the longer path, showing a high preference for the shorter path between the source and depot for their collections. The number of robots on the two paths at the end of one simulated hour is shown in (Fig. 5.8 (c)). The boxes represent the 1$^{st}$ to 3$^{rd}$ quartile range of data from 100 simulations, with the median results represented as a horizontal line within the box, and whiskers extending to represent 1.5 times the interquartile range. The individual Kilobots cannot follow a pheromone gradient nor detect any difference in pheromone concentration. Despite their limited individual capabilities, under certain conditions the robot swarm can reproduce behaviour similar to foraging ant colonies, which instead rely on much higher cognitive abilities at the individual level.

These results indicate that for experimental conditions similar to those conducted here, individual agents with simpler cognitive abilities are sufficient for reproducing

the emergent behaviour observed in more complex social insect colonies such as ants. However, ants and more capable agents with the ability to distinguish between different pheromone concentrations are likely to be more flexible with the ability to optimise path lengths in a wider variety of dynamic environments than the robot system presented herein. In fact, it is likely that changing variables such as the density of robots in the environment or significantly varying the path lengths may degrade the performance of the Kilobots, but this remains to be confirmed.

The next set of experiments set out to prove the ability of the swarm to adapt to any range of qualities, as described in in section 5.2.2, showing a response that is sensitive to ratios between qualities $\left(\frac{Q_2}{Q_1}\right)$ rather than absolute values. These experiments examined three scenarios (Fig. 5.8) with $n = 2$ sources where the ratio between the two item qualities remained the same $\left(\frac{Q_2}{Q_1} = 0.4\right)$, but the absolute values of the qualities were $\left(\frac{Q_2}{Q_1} = \frac{6}{15}\ ,\ \frac{Q_2}{Q_1} = \frac{4}{10}\ ,\ \frac{Q_2}{Q_1} = \frac{2}{5}\right)$. These results compare the new adaptive strategy based on quality ratio (white box plots) and a strategy that only considers absolute quality values (grey box plots).

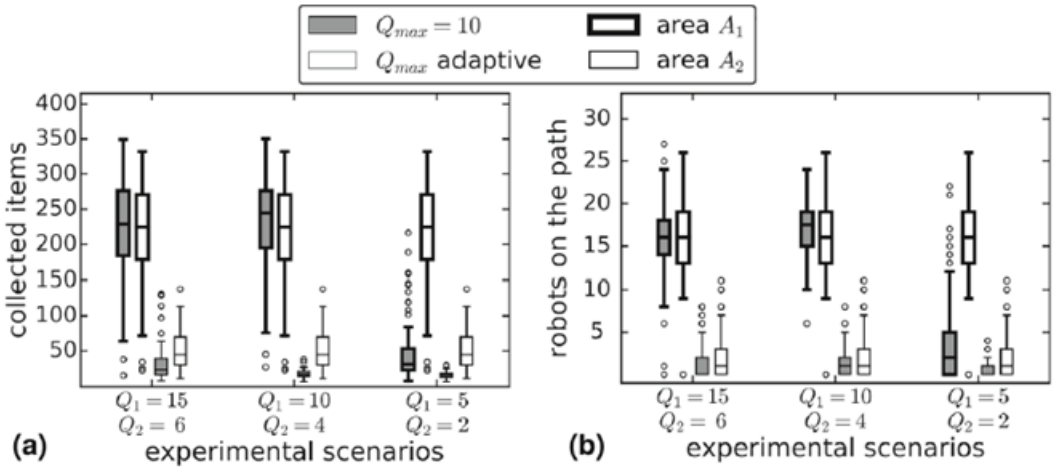



*Fig. 5.9. Simulation results showing the robot swarms ability to adapt to different item qualities based on ratios rather than absolute values, originally published in Talamali et al. (2020).*

The key of figure 5.9 indicates how the data of the aforementioned experiments are represented. Grey boxes represent the experiments where the maximum quality used by the swarm to tune its response is set prior to be 10 and remains so even if the actual maximum quality recorded in the environment is higher or lower than 10. The

white boxes represent the experiments where the maximum quality use by the swarm to tune its response is updated according to the highest quality item returned to the depot, creating an adaptive response. Boxes with a bold outline represent information pertaining to area A1, while boxes with a normal outline represent information pertaining to the second area A2. For instance, the bold grey box on the right hand side of Fig 5.9 (b) indicates the small number of robots dedicated to the path between the depot and area A1 when the swarm is using a static maximum quality of 10.

Fig. 5.9 (a) and (b) show that the adaptive strategy allows the swarm to adapt to any condition to maximise the number of items collected while the constant maximum quality of 10 strategy was only able to maximise its items collected when the predetermined quality range matched that of the environment's range. This sensitivity to relative quality of food source rather than absolute quality has also been documented in foraging ant species (Wendt et al. 2018). Fig. 5.9 (a) shows the number of item collected, and Fig. 5.9 (b) shows the number of robots on each path at the end of the simulation. The experiments used a single depot and $n = 2$ source areas, where the superior source $A_1$ and inferior source $A_2$ were equal distance from the depot $(d_1 = d_2 = 1m)$. The ratio of the item qualities in $A_1$ and $A_2$ were kept constant at $\left(\frac{Q_2}{Q_1} = 0.4\right)$, but the absolute values were varied as indicated on the x-axis of both graphs. All experiments were conducted with swarms of $S = 50$ Kilobots and an intermediate value of $\alpha = 0.85$ in Eq.(5.2) and Eq.(5.3). Similar to Fig. 5.7, the boxes range from the 1st to 3rd quartile of the data from 100 simulations with the median indicated by a horizontal line; the whiskers extend to 1.5 times the interquartile range. The constant range strategy (dark boxplots) only yields good results if the predefined range matches the actual range of the environment (central experiment). Whereas the adaptive strategy allows the swarm to exploit resources as a function of their relative qualities in a range adapted to the environment.

As discussed in section 5.2.2, robots can modulate their behaviour to give priority to source areas which are closer (low α) or contain higher quality items (high α) – an ability which can results in different collective responses depending on the environment. These dynamics were investigated in experiments using $S = 50$ simulated Kilobots operating in an environment with $n = 2$ source areas, where $A_1$

contained higher quality items ($Q_1 = 10$) and $A_2$ contained lower quality items ($Q_2 = 4$). The distance of $A_1$ from the depot remained constant at $d_1 = 1m$, while the distance of $A_2$ was varied at $d_2 \in [0.5m, 1m]$. The relatively small swarm size of $S = 50$ was selected due to the results reported by (Font Llenas et al. 2018) for a similar scenario, where it was shown that large swarms do not discriminate between sources where there are enough robots to maximally exploit both areas.

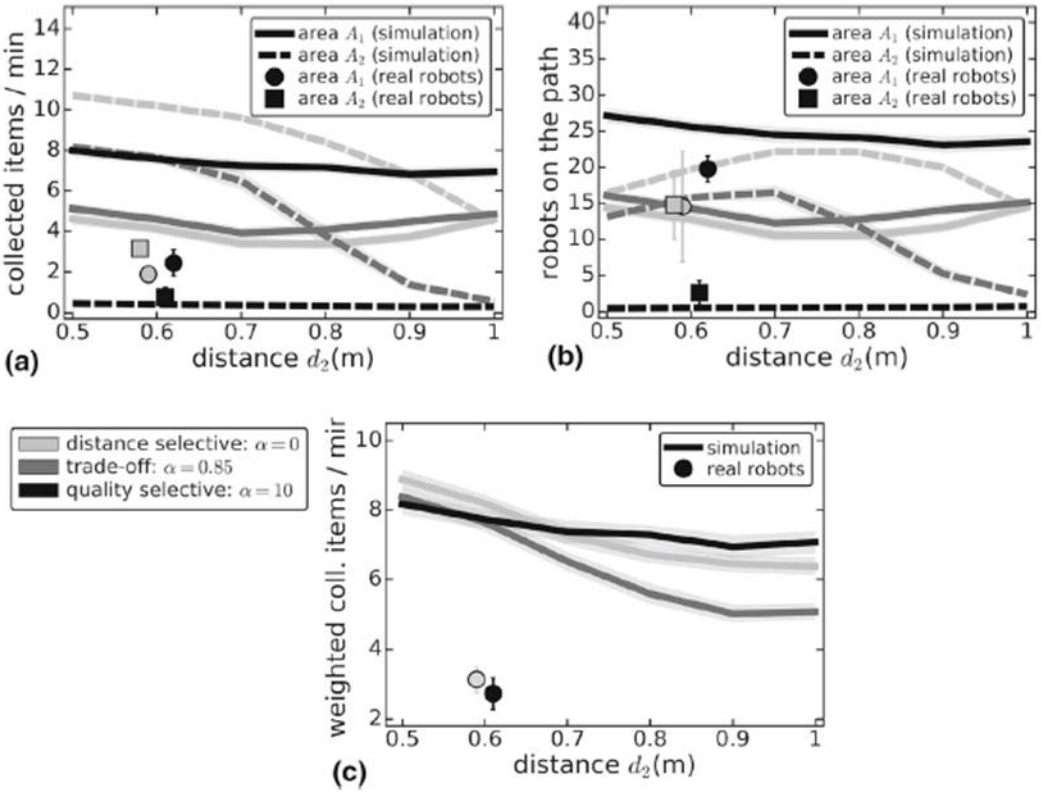



*Fig. 5.10. Effect of modulating parameter α from Eq. (5.2) and (5.3) to favour nearer source areas (α = 0), to favour the best-quality sources (α = 10), or to balance the distance-quality trade-off (0 < α < 10).*

Fig. 5.10 is made up of three graphs to show what effect tuning the selectivity of the robots has on (a) the number of items collected per minute, (b) the number of robots on the path between the depot and area A1 or the path between the depot and area A2, and (c) the weighted collected items per minute; while varying the distance of the second source area. The key to the left of Fig. 5.10 (c), shows how α is represented in the graphs where light grey coloured points and lines represent setting the tuneable parameter to α = 0 (distance selective behaviour), dark grey coloured points and lines

represent setting the tuneable parameter α = 0.85 (optimal distance-quality trade-off), and black coloured points and lines represent setting the tenable parameter to α = 10 (quality selective). In addition, each graph of Fig.5.10 contains a second key to indicate the distribution of robots in the environment for simulation and physical experiments. Solid lines represent information pertaining to area A1 of the simulation, while dashed lines represent information pertaining to area A2 of the simulation. For instance, in Fig.5.10 (b) the dashed, dark grey line indicates the number of robots on the path between the depot and area A2 in simulation, when the tuneable parameter α is set to 0.85 so robots can weigh the benefit of quality versus distance of the source. The solid circles represent information pertaining to area A1 of the physical experiments, and solid squares represent information pertaining to area A2. For example, the solid black circle of Fig.5.10 (b) indicates that a much higher portion of robots were on the path between the depot and area A1 than the depot and area A2 in the physical experiments.

The results represent hour-long simulated and physical robots experiments for scenarios with $n = 2$ sources. The initial exploration phase is excluded, with mean values indicating the last 30 minutes only. Physical robots' results are indicated as solid symbols with vertical bars indicating the 95% confidence intervals of 3 runs for each condition. Lines represent the mean of 100 simulations with the shaded areas representing 95% confidence intervals. Source $A_1$ had quality $Q_1 = 10$ and was located at distance $d_1 = 1m$; source $A_2$ had quality $Q_2 = 4$ and varying distance $d_2 \in [0.5m, 1m]$. (a) shows the rate of items collected per minute, (b) shows the mean number of robots on each path, and (c) is the rate of item collected per minute weighted by the normalised quality $q_1 = 1.0$ and $q_2 = 0.5$.

Using α = 0 promoted distance selectivity, where the simulated swarm had the highest item collection per minute in (Fig. 5.10 (a)) from the closest source ($A_2$) to which the majority of the workers were deployed in (Fig. 5.10 (b)). Using α = 10 promoted quality selectivity, where the simulated swarm had the highest item collection per minute in (Fig. 5.10 (a)) from the highest quality source ($A_1$) to which the majority of the robots were deployed in (Fig. 5.10 (b)). Finally, intermediate value of α = 0.85, led to a distance-quality trade-off where the swarm exploited the nearest inferior-quality source only if it was much closer than the farther superior-quality source. Three

experiments were conducted with 50 physical robots for each of the two limit cases of quality-selective α = 10 (solid black symbols) and of distance-selective α = 0 (solid light-grey symbols). Videos of these experiments are available as online supplementary material at:
https://www.youtube.com/playlist?list=PLCGKY9OHLZwMaGeB6cxVfxmHwhBFqKF7a

The results from the physical experiments showed the Kilobots to be less efficient at resource collection than their simulated counterparts. This reality gap is likely due to a difference in the motion speed of the physical and simulated robots. The simulation was accurately tuned on the movement speed of fully charged Kilobots (Pinciroli et al. 2018), but did not take into account that the robot's speed reduced over time as its battery charge diminished. Despite this difference, the results still demonstrated that in both cases the two strategies ($\alpha \in [0,10]$) favoured either the closest or best-quality source area respectively, as shown in simulation.

## Section 5.5 Discussion

The results from these experiments demonstrate how complex collective foraging strategies can emerge from simple individual agents. Both the simulated and real robots possessed a minimal cognitive architecture which utilised only a simply binary detector for pheromone trails and obstacles, and maintenance of a home vector which informed them of the relative direction of the depot. Therefore, the individual robots have only a fraction of the capabilities of a real ant. However, when imbued with the ability to deposit pheromone at a rate determined by a single tuneable parameter, they become capable of qualitatively reproduce classical results such as the shortest path exploitation observed in lab ant colonies (Goss et al. 1989), and achieve distance-quality trade-off of foraging. The experiments also examined the effect of resource distribution on the optimal distribution of foragers over source areas. Other studies have considered the effect of colony size on recruitment strategy (Planqué et al. 2010; Pagliara et al. 2018; Mayya et al. 2019), the approach within this study instead assumes the recruitment strategy and investigates the optimal distribution.

The agent controllers are able to approximate the optimal distribution for relatively small swarm sizes, although large swarms depart from optimality. Large swarms lead to crowded environments which require strategies to clear paths in order to reduce

traffic congestion. Two possible strategies to limit traffic congestion were examined in this study: modifying the abandonment strategy or enriching the individual behaviour with collision-reactive states. Should an agent decide to abandon its current source, the robots simply resumed exploration however the effects of this abandonment strategy are limited as robots then quickly rediscover a path (which may be already congested). A better abandonment strategy could improve the results of the abandonment behaviour introduced in this work. For instance, the robots could be instructed to remain at the depot for a period of time before resuming exploration, similar to ants (Pagliara et al. 2018). This would allow time for the pheromone trail it has abandoned to weaken and reduce the risk of rediscovering the path it abandoned. In addition, it is possible to maintain a steady flow of traffic which remains undisrupted even in relatively crowded conditions by individual ants changing their behaviour as a function of collisions with other ants (Dussutour et al. 2004; Poissonnier et al. 2019). Inspired by these results, the robot behaviour could be enriched with new collision-dependent states using the same collision detection system employed for wall avoidance behaviour outlined in Section 5.1.3.

The results from the experiments are complementary to other approaches using minimal controllers for collective behaviour in the swarm robotics field (Gauci et al. 2014; Özdemir et al. 2018). Using simple controllers increases their transferability to various robotics platforms because the functions require minimal hardware to carry out the necessary functions. Simple behaviours also contribute to shrinking the reality gap, preserving consistent dynamics when moving from simulation to physical systems – as demonstrated in section 5.4.1 where the same control software was used to produce qualitatively similar results. The ability of simple robots to generate complex collective dynamics should be of interest to biologist and offer practical utility to engineers.

# Chapter 6. Conclusions and Future Work

At the begging of this thesis, the author set out to answer whether collaborative multi-robot teams could outperform uncoordinated multi-robot teams, and how such systems may be applied to address real world issues that present a challenge to conventional single-robot systems and human teams. In this concluding Chapter, the significance of the findings, the contribution to knowledge they represent, and the new questions that should be considered following these discoveries are presented. Section 6.1 presents the main conclusions to the research in Chapters 3, 4, and 5, while Section 6.2 presents some of the more prominent questions that remain to be explored following the findings of the research.

## Section 6.1 Main conclusions

Chapters 3 and 4 set out to investigate the feasibility of a multi-robot system intended to replace human teams in carrying out emergency ship hull repair of vessels that suffer a hull breach at sea.

Chapter 3 focused on the inspection process, and proved through simulation that the cooperative search algorithm was more effective at achieving complete area coverage of the ship hull in less time than the same multi-robot system using an uncoordinated search algorithm. Additionally, a robot sensor arrangement and accompanying control scheme was presented that instructed the robots to maintain a set distance from a 3D object, allowing them to treat their environment more akin to a 2D plane. This demonstrated a method of implementing simpler search algorithms on swarm robots while maintaining their ability to carry out inspection of a 3D surface underwater.

Chapter 4 examined how modular multi-robot teams may be used to seal circular hull breaches of various sizes by self-assembling in a decentralised manner to form a repair patch of appropriate shape and size. The results from the experiments also informed an improved self-assembly method where robot congestion may be reduced by controlling the angle of approach the robots use when navigating their way to the damage, or by allowing more than one assembly location for the repair patch – ultimately improving the speed of structure formation.

Chapter 5 investigated the use of nature inspired multi-robot teams in foraging scenarios. Specifically, the study demonstrated how the researchers' implementation

of stigmergic communication, plastic behaviours, and the newly implemented obstacle avoidance behaviour enabled very simple robots to perform comparatively well to more complex robot teams tasked with the same objectives. The obstacle avoidance behaviour solved a major issue of physical robots becoming stuck against the walls of their bounded arena and other robots which improved the performance of the swarm, and created a system more capable of emulating the collective foraging behaviours observed in biological swarms such as ants.

### Section 6.1.1 Ship Hull Inspection: Complete Area Coverage

Previous studies have demonstrated that collaborative multi-robot teams may outperform uncoordinated multi-robot teams under certain conditions but efforts to examine this supposition in a wider range of scenarios have been limited. Using the emergency ship hull repair scenario introduced in Chapter 3, the author set out to examine if collaborative multi-robot systems could indeed outperform their uncoordinated counterparts, and how this could be leveraged to improve on existing solutions. The two main aspects of emergency ship hull repair that could benefit the most from the use of collaborative multi-robot teams were inspection and repair. Chapter 3 focused on the inspection aspect by proposing an individual robot behaviour that encourages collaboration between robots when performing a search, with the intent of developing a system which was faster and more resilient to sensor errors and individual robot failure that uncoordinated teams.

A bespoke simulated robot model whose constituent parts were based on existing technologies employed in modern autonomous robots was developed, including more recent developments in underwater propulsion such as modular hydraulic propulsion. To assess whether a coordinated team of these robots could indeed outperform an uncoordinated team of the same robots, a series of simulated experiments based around autonomous ship hull inspection were carried out in the open source robot simulator Webots. In these scenarios both robot teams were tasked with inspecting the mid-section of the ship hull and were assessed on which team could accomplish the task the fastest without leaving gaps in their inspection. The system resilience was tested in later scenarios, by introducing errors to the robot distance sensors or by choosing a percentage of the robot team to fail at a random time. These test

conditions were intended to reveal which of the teams would perform better in the presence of significant sensor noise, or the partial population failure.

The results of these experiments revealed that the coordinated team of robots managed to outperform the un-coordinated robot team under ideal conditions and in the face of partial population failure, while the uncoordinated team was more resilient to significant sensor noise than the coordinated team which heavily relied on these readings to coordinate their efforts. These results identified some shortcoming of the Webots simulation environment such as inaccurate sonar sensor models, and helped identify some ways in which the coordinated search algorithm may be improved. Ultimately, the results confirmed the author's hypothesis that a coordinated robot teams could indeed outperform the same collection of robots performing the same search task without coordinating with one another. However, it should be noted that significant sensor errors can act as the tipping point – moving these systems to become unstable and perform worse than teams of robots who do not rely on these sensors for their behaviour.

One of the most significant contributions to knowledge of this research is the introduction of a complete area coverage algorithm and novel underwater inspection method introduced as part of the ship hull repair scenario. The robots' method of controlling its distance and orientation relative to the ship hull using distance sensors allowed the robots to treat the surface of a 3D object as a 2D plane. This allowed for the implementation of less complex search algorithms comparable to those employed in swarm systems operating in 2D bounded arenas. This reduction in complexity allows for its implementation on less capable robots that would otherwise lack the computation ability to perform path planning, localization or mapping in complex 3D environments. In addition, the method of ship hull inspection using a swarm of autonomous robots presents an alternative approach for emergency ship hull inspection that could save humans from performing such high-risk tasks.

### Section 6.1.2 Ship Hull Repair: Aggregation and Self-Assembly

The next step in developing the emergency ship hull repair method was to investigate how the coordinated team of robots could progress from inspection of the hull to repair of damage. Chapter 4 investigated this repair aspect by proposing an individual

robot behaviour that could be used by a team of robots to self-assemble into a repair patch for sealing hull breaches in a decentralised manner. Decentralised methods of organization are advantageous compared to centralised methods as they do not require the governance of a single entity which removes a critical point of failure and increases the robustness and scalability of the approach. The method of self-assembly proposed was reliant on the use of local communication between robots using, which would demonstrate a method of complex organisation that did not require external guidance.

To assess the effectiveness of the self-assembly protocol, a series of experiments which would examine how modular robot teams of different population sizes could form a repair patch of an appropriate size and shape over hull breaches of varying sizes was devised. The experiments were carried out in Netlogo, a simulator well suited to studies involving very large numbers of robots, and using a simplified model of the same robots utilised in Chapter 3. The experiments assessed the performance of robot teams of different sizes and their ability to form repair patches about a single assembly point – a task which grew increasingly more difficult for teams to perform as the size of robot population grew. This would allow the researchers to identify the point at which the size of the robot population densities became more of a detriment to team performance than an improvement, and assess any shortcomings of the approach.

The results of the experiments show that the robots' teams could indeed use the self-assembly protocol to form repair patches of appropriate shapes and sizes for hull breaches of different sizes and at different locations under certain conditions. One of the most important conditions necessary to ensure the robots could best perform self-assembly, was the location of the hull breach and the associated assembly point. If the space around the assembly point was too small to accommodate multiple robots, team performance would suffer. This vulnerability could be addressed by increasing the number of assembly point the robots could choose to being assembling, allowing them to assemble from any approach vector. These shortfalls of the approach were addressed in the closing section of Chapter 4 with the proposal of a modified version of the self-assembly method which would allow the robots to approach from any direction.

The main contributions of this research was the introduction of a self-assembly algorithm which could be used by teams of homogeneous modular robots to successfully create complex formations using only simple local communication. The algorithm was applied to the emergency ship hull repair scenario to show how such self-organisation techniques could be applied to solve real world challenges. The self-assembly algorithm was investigated in a 2D environment but if applied to the robots used in Chapter 3, which are capable of representing 3D surface as a 2D plane, it would be possible to implement this technique on that same system. This represents another step towards realising an approach to emergency ship hull repair using a coordinated multi-robot team.

### Section 6.1.3 Nature Inspired Swarms: Foraging and Obstacle Avoidance

Following the investigation into how co-operative multi-robot teams could be applied to ship hull inspection and repair, a separate collaborative effort was made towards researching how stigmergic communication and plastic behaviours could enable very simple robots to perform as competently as more complex robot teams in foraging scenarios. Chapter 5 presented the efforts the author made towards implementing obstacle avoidance behaviour in a large team of Kilobots and improve the swarm's ability to navigate unknown environments, while the collaborators of this research focused on expanding the stigmergic communication and plastic behaviours presented in their earlier study concerning foraging Kilobots in a multi-source environment.

To assess the performance of these modifications, several simulated and physical experiments were devised to test the swarm's ability to forage efficiently in a bounded arena with a central depot and different source areas of items of varying quality. The robots used in this study, referred to as Kilobots, are very simple agents, but were imbued with greater ability by the collaborators implementation of the augmented reality Kilobot (ARK) system. This enabled the Kilobots to perform more complex actions such as obstacle avoidance, pheromone deposition, and pheromone sensing. The obstacle avoidance behaviour was implemented in both simulated and real-world Kilobots, but was primarily assessed in simulation using a setup inspired by the double bridge experiment – where the robots would have an option of choosing between a longer of shorter path between the source and depot with the aim of increasing efficiency. The Kilobot swarm's virtual pheromone-based communication system and

ability to choose the most optimal foraging strategy based on relative item quality and source area distances were examined in simulated and real-world experiments.

The results of the double-bridge inspired simulations, and the multi-source real world experiments that used physical Kilobots, showed that the obstacle avoidance behaviour successfully enabled the robots to avoid becoming stuck on the boundaries of obstacles. The results also proved the swarm capable of using simplified stigmergic communication to favour the shortest path between the central depot and the source, performing comparatively well to more complex multi-robot systems. The simulated and real-world multi-source experiment results demonstrated the system's ability to use the relative ratios of quality and distance to identify the foraging strategy which maximised yield and to gravitate towards that approach.

Prior to the inclusion of the obstacle avoidance behaviour, individual Kilobots would frequently become stuck on the edges of the bounded arena, reducing the overall effectiveness of the swarm. Without the obstacle avoidance behaviour, the swarm could not be expected to perform effectively in environments with obstacles, a common feature of real-world environments, as increasing the obstacles would likewise increase the number of collisions and robots becoming stuck. The most notable contribution of the author to this collaborative study was the inclusion of this behaviour which better prepared the robots to function in more complex environments, and helped make their performance comparable to more capable artificial and biological multi-agent systems.

## Section 6.2 Future Work

The studies presented within this text represent but a fraction of the potential of truly cooperative multi-robot systems, and while much has been revealed from this research, there is far more which remains to be unveiled. This final section presents a collection of questions which could prove beneficial in advancing the field of swarm robotics, as pertains to emergency ship hull repair using autonomous underwater robots, and nature inspired foraging multi-robot systems.

### Section 6.2.1 Complete Emergency Ship Hull Repair

Using a coordinated team of robots to carry out ship hull inspection and to repair hull breaches as proposed in Chapters 3 and 4 represent significant steps toward realising a

full solution to emergency ship hull repair. The results of the research show these approaches may be plausible, but there remain questions which need to be addressed before such techniques should be feasibly implemented on real robots. For instance, in both the inspection and self-assembly scenarios the robots are operating in a static body of water, but even the calmest oceans are significantly more dynamic than this. Fluid dynamics such as waves and underwater currents, and obstructions such as sand, seaweed are all common features of underwater environments which these robots must be equipped to handle. A ship that has suffered hull damage as a result of a battle will not typically halt their course to repair while still in the midst of combat, and so the system could be adapted to service a ship which is still in transit. Even if these aspects are addressed and the team of robot successfully form a patch of appropriate shape and size to cover a hull breach, the method by which the robots adhere to the hull has not yet been decided.

Future studies relating to this application should investigate aspects such as: the ability of the robots to maintain their stability in the presence of additional external forces such as waves, the wakes formed by obstacles, and underwater currents. The inspection, self-assembly, and repair processes ought to be adapted so that they may perform these actions on vessels which are still in transit, allowing for repairs to occur even in the midst of combat which would help to restore stability without exposing the ship to greater risk of attack. Changing the shape of the robots to make them more hydrodynamic would significantly increase the performance individuals and likely would benefit the swarm as whole. The obstacle avoidance behaviour of these robots should be developed further to enable individuals to anticipate and avoid additional moving obstacles. An appropriate method of underwater adhesion, which would allow the robots to seal a hull breach, needs to be selected to complete the emergency ship hull repair process. Studies such as these would greatly contribute to completing this novel approach to emergency ship hull repair and help make the final leap from concept to reality.

### Section 6.2.2 Robot Avoidance in Foraging Swarm Robots

In Chapter 5, the ARK system was used to imbue Kilobots with obstacle avoidance behaviour which proved to be a useful tool for preventing the robot become stuck at the boundary of walls. However, wall avoidance is only a single example of how this

ability could be leveraged to improve the performance of the swarm. This same ability to detect walls in the forward sections of the Kilobot could be used to detect the presence of other obstacles, including other Kilobots. Previous studies on multi-agent systems (Dussutour et al. 2004; Poissonnier et al. 2019) have shown how the traffic congestion in environments with large numbers of agents can be limited by using collision-reactive behaviours. Introducing a collision-dependant state to the individual robot's behaviour, such that they may form multiple traffic lanes between sources and depots, could prove to increase the efficiency of the swarm without having altered their morphology, and with minimal additional computational overhead. The stigmergic communication and adaptive quality-sensitive behaviour established in this work demonstrated how simple individuals with limited capabilities, were able to achieve similar levels of performance to more complex biological multi-agent systems, and Introducing robot-avoidance and queue forming behaviours to such a system could prove to increase the efficiency of the system even further.

Scholars interested in pursuing swarm robotics research should note the following: Every addition to an individual's behaviour invites a measure of change to the whole system which may be difficult to predict. How one might design an individual behaviour which reliably and predictably proliferates into a desired collective behaviour is one of the driving forces behind swarm robotics research today. That is why it is the author's firm belief that developing a general design pattern for swarm robot systems is one of the most important pursuits open to swarm robotics researchers today. When general design patterns for swarm robot system are finally achieved, cooperative multi-robot systems such as our emergency ship hull repair robots will no longer require years of novel research and development. Instead, scientists and engineers will finally have a reliable, evidence-based method to enable them to transform individual ideas into collective realities, opening the floodgates to new innovations which could change the world.

# References


[1] Aguirre, F., Vargas, S., Valdes, D., & Tornero, J. (2017). State of the art of parameters for mechanical design of an autonomous underwater vehicle. International Journal of Oceans and Oceanography, 11(1), 89-103.

[2] Antonelli, G., Arrichiello, F., & Chiaverini, S. (2010). Flocking for multi-robot systems via the null-space-based behavioural control. Swarm Intelligence, 4(1), 37.

[3] Arganda, S., Nicolis, S. C., Perochain, A., Péchabadens, C., Latil, G., & Dussutour, A. (2014). Collective choice in ants: The role of protein and carbohydrates ratios. Journal of Insect Physiology, 69, 19–26.

[4] Arkin, R. C. (1989). Motor schema—based mobile robot navigation. The International journal of robotics research, 8(4), 92-112.

[5] Arkin, R. C., Balch, T., & Nitz, E. (1993, May). Communication of behavioural state in multi-agent retrieval tasks. In [1993] Proceedings IEEE International Conference on Robotics and Automation (pp. 588-594). IEEE.

[6] Arkin, R., & Bekey, G. (1997). Robot colonies-editorial. Autonomous Robots, 4(1).

[7] Arkin, R. (1998). Behaviour-based robotics. MIT Press.

[8] Arvin, F., Turgut, A. E., Bazyari, F., Arikan, K. B., Bellotto, N., & Yue, S. (2014). Cue-based aggregation with a mobile robot swarm: a novel fuzzy-based method. Adaptive Behaviour, 22(3), 189-206.

[9] Arvin, F., Turgut, A. E., Bellotto, N., & Yue, S. (2014, October). Comparison of different cue-based swarm aggregation strategies. In International Conference in Swarm Intelligence (pp. 1-8). Springer, Cham.

[10] Arvin, F., Yue, S., & Xiong, C. (2015). Colias-ϕ: An autonomous micro robot for artificial pheromone communication. International Journal of Mechanical Engineering and Robotics Research, 4(4), 349–353.

[11] Åström, K. J., Hägglund, T., & Astrom, K. J. (2006). Advanced PID control (Vol. 461). Research Triangle Park, NC: ISA-The Instrumentation, Systems, and Automation Society.

[12] Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. IEEE robotics & automation magazine, 13(3), 108-117.

[13] Bailey, T., Nieto, J., & Nebot, E. (2006, May). Consistency of the FastSLAM algorithm. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. (pp. 424-429). IEEE.

[14] Balch, T., & Arkin, R. (1994). Communication in reactive multiagent robotic systems. Autonomous Robots, 1(1), 27–52.

[15] Balch, T., & Arkin, R. (1998). Behaviour-based formation control for multirobot teams. IEEE Transactions on Robotics and Automation, 14(6), 926–939.

[16] Balch, T., & Hybinette, M. (2000, April). Social potentials for scalable multi-robot formations. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 1, pp. 73-80). IEEE.

[17] Barca, J., & Sekercioglu, Y. (2013). Swarm robotics reviewed. Robotica, 31(3), 345–359.
[18] Batalin, M. A., & Sukhatme, G. S. (2002). Spreading out: A local approach to multi-robot coverage. In Distributed Autonomous Robotic Systems 5 (pp. 373-382). Springer, Tokyo.
[19] Bayindir, L. (2016). A review of swarm robotics tasks. Neurocomputing, 172, 292–321.
[20] Beckers, R., Deneubourg, J. L., Goss, S., & Pasteels, J. M. (1990). Collective decision making through food recruitment. Insectes Sociaux, 37(3), 258–267.
[21] Belke, C. H., & Paik, J. (2017). Mori: a modular origami robot. IEEE/ASME Transactions on Mechatronics, 22(5), 2153-2164.
[22] Beni, G. (1988). The concept of cellular robotic system. Proceedings IEEE International Symposium on Intelligent Control 1988, 57–62.
[23] Beni, G., and J. Wang. "Swarm Intelligence." In Proceedings Seventh Annual Meeting of the Robotics Society of Japan, 425-428. Tokyo: RSJ Press, 1989.
[24] Beni, G., and J. Wang. "Theoretical Problems for the Realization of Distributed Robotic Systems." In Proceedings 1991 IEEE International Conference on Robotic and Automation, 1914-1920. Los Alamitos, CA: IEEE Computer Society Press, 1991.
[25] Beni, G., and S. Hackwood. "Stationary Waves in Cyclic Swarms." In Proceedings 1992 IEEE Int. Symp. on Intelligent Control, 234-242. Los Alamitos, CA: IEEE Computer Society Press, 1992.
[26] Beni, G. (2005). From Swarm Intelligence to Swarm Robotics. In Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers (Vol. 3342, pp. 1–9). Berlin, Heidelberg: Springer Berlin Heidelberg.
[27] Bererton, C., & Khosla, P. K. (2001). Towards a team of robots with repair capabilities: a visual docking system. In Experimental Robotics VII (pp. 333-342). Springer, Berlin, Heidelberg.
[28] Boeing, A., & Bräunl, T. (2012, December). Leveraging multiple simulators for crossing the reality gap. In 2012 12th International Conference on Control Automation Robotics & Vision (ICARCV) (pp. 1113-1119). IEEE.
[29] Bonabeau, E., Theraulaz, G., Deneubourg, J., Aron, S., & Camazine, S. (1997). [Review of Self-organization in social insects]. Trends in Ecology & Evolution, 12(5), 188–193.
[30] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm intelligence: from natural to artificial systems. New York: Oxford University Press.
[31] Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., ... & Mondada, F. (2010, October). The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4187-4193). IEEE.
[32] Borenstein, J., Everett, H. R., & Feng, L. (1996). Navigating mobile robots: Systems and techniques. AK Peters, Ltd.

[33] Bosien, A., Turau, V., & Zambonelli, F. (2012). Approaches to fast sequential inventory and path following in RFID-enriched environments. International Journal of Radio Frequency Identification Technology and Applications, 4(1), 28–48.
[34] Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence, 7(1), 1–41.
[35] Bregy, P., Sommer, S., & Wehner, R. (2008). Nest-mark orientation versus vector navigation in desert ants. Journal of Experimental Biology, 211(12), 1868–1873.
[36] Brooks, R. (1986). A robust layered control system for a mobile robot. IEEE journal on robotics and automation, 2(1), 14-23.
[37] Brooks, R. A. (1990). Elephants don't play chess. Robotics and autonomous systems, 6(1-2), 3-15.
[38] Burgard, W., Moors, M., Stachniss, C., & Schneider, F. E. (2005). Coordinated multi-robot exploration. IEEE Transactions on robotics, 21(3), 376-386.
[39] Camazine, S., Deneubourg, J.L., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau. (2003). Self-Organisation in Biological Systems (Vol. 7). Princeton University Press, NJ, USA.
[40] Campo, A., & Dorigo, M. (2007, September). Efficient multi-foraging in swarm robotics. In European Conference on Artificial Life (pp. 696-705). Springer, Berlin, Heidelberg.
[41] Campo, A., Gutiérrez, Á., Nouyan, S., Pinciroli, C., Longchamp, V., Garnier, S., et al. (2010). Artificial pheromone for path selection by a foraging swarm of robots. Biological Cybernetics, 103(5), 339–352.
[42] Castano, A., Behar, A., & Will, P. M. (2002). The Conro modules for reconfigurable robots. IEEE/ASME transactions on mechatronics, 7(4), 403-409.
[43] Çelikkanat, H., & Şahin, E. (2010). Steering self-organized robot flocks through externally guided individuals. Neural Computing and Applications, 19(6), 849-865.
[44] Center, N. (2013). Navy Damage Controlman - NAVEDTRA 14057 (Non-resident Training Course). [S.l.]: LULU COM.
[45] Charbonneau, D., Hillis, N., & Dornhaus, A. (2015). 'Lazy' in nature: Ant colony time budgets show high 'inactivity' in the field as well as in the lab. Insectes Sociaux, 62(1), 31–35.
[46] Cohen, W. W. (1996). Adaptive mapping and navigation by teams of simple robots. Robotics and autonomous systems, 18(4), 411-434.
[47] Collett, T. S., & Collett, M. (2002). Memory use in insect visual navigation. Nature Reviews Neuroscience, 3(7), 542–552.
[48] Couceiro, M. S., Rocha, R. P., & Ferreira, N. M. (2011, November). A novel multi-robot exploration approach based on particle swarm optimization algorithms. In 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (pp. 327-332). IEEE.
[49] de Sá, A. O., Nedjah, N., & de Macedo Mourelle, L. (2016). Distributed efficient localization in swarm robotic systems using swarm intelligence algorithms. Neurocomputing, 172, 322-336.

[50] Deng, X., Jiang, D., Wang, J., Li, M., & Chen, Q. (2015, November). Study on the 3D printed robotic fish with autonomous obstacle avoidance behaviour based on the adaptive neuro-fuzzy control. In IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society (pp. 000007-000012). IEEE.

[51] Detrain, C., & Deneubourg, J. L. (2008). Collective decision-making and foraging patterns in ants and honeybees. Advances in insect physiology, 35, 123-173.

[52] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. Ieee Transactions On Systems Man And Cybernetics Part B-Cybernetics, 26(1), 29–41.

[53] Doyle, M. J., Xu, X., Gu, Y., Perez-Diaz, F., Parrott, C., & Groß, R. (2016, May). Modular hydraulic propulsion: A robot that moves by routing fluid through itself. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5189-5196). IEEE.

[54] Ducatelle, F., Förster, A., Di Caro, G. A., & Gambardella, L. M. (2009). Supporting navigation in multi-robot systems through delay tolerant network communication. IFAC Proceedings Volumes, 42(22), 25-30.

[55] Ducatelle, F., Di Caro, G. A., Pinciroli, C., Mondada, F., & Gambardella, L. M. (2011). Communication assisted navigation in robotic swarms: self-organization and cooperation. In Proceedings of the 2011 IEEE/RSJ international conference on intelligent robots and systems (IROS 2011) (pp. 4981–4988). IEEE.

[56] Ducatelle, F., Di Caro, G. A., Pinciroli, C., & Gambardella, L. M. (2011). Self-organized cooperation between robotic swarms. Swarm Intelligence, 5(2), 73–96.

[57] Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. IEEE robotics & automation magazine, 13(2), 99-110.

[58] Dussutour, A., Fourcassié, V., Helbing, D., & Deneubourg, J. L. (2004). Optimal traffic organization in ants under crowded conditions. Nature, 428(6978), 70–73.

[59] Dussutour, A., & Simpson, S. J. (2009). Communal nutrition in ants. Current Biology, 19(9), 740–744.

[60] Evans, J., Redmond, P., Plakas, C., Hamilton, K., & Lane, D. (2003, September). Autonomous docking for Intervention-AUVs using sonar and video-based real-time 3D pose estimation. In Oceans 2003. Celebrating the Past... Teaming Toward the Future (IEEE Cat. No. 03CH37492) (Vol. 4, pp. 2201-2210). IEEE.

[61] Falconi, R., Sabattini, L., Secchi, C., Fantuzzi, C., & Melchiorri, C. (2015). Edge-weighted consensus-based formation control strategy with collision avoidance. Robotica, 33(2), 332-347.

[62] Farr, N., Bowen, A., Ware, J., Pontbriand, C., & Tivey, M. (2010, May). An integrated, underwater optical/acoustic communications system. In OCEANS'10 IEEE SYDNEY (pp. 1-6). IEEE.

[63] Feldman, J. A., & Sproull, R. F. (1977). Decision theory and artificial intelligence II: The hungry monkey. Cognitive Science, 1(2), 158-192.

[64] Ferrante, E., Turgut, A. E., Mathews, N., Birattari, M., & Dorigo, M. (2010, September). Flocking in stationary and non-stationary environments: a novel

communication strategy for heading alignment. In International conference on parallel problem solving from nature (pp. 331-340). Springer, Berlin, Heidelberg.

[65] Ferrante, E., Turgut, A. E., Stranieri, A., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). A self-adaptive communication strategy for flocking in stationary and non-stationary environments. Natural Computing, 13(2), 225-245.

[66] Fetecau, R. C., & Meskas, J. (2013). A nonlocal kinetic model for predator–prey interactions. Swarm Intelligence, 7(4), 279-305.

[67] Fick, A. (1855). Ueber diffusion. Annalen der Physik, 170(1), 59–86.

[68] Floreano, D., & Mattiussi, C. (2008). Bio-inspired artificial intelligence theories, methods, and technologies . Cambridge, MA: MIT Press.

[69] Font Llenas, A., Talamali, M. S., Xu, X., Marshall, J. A. R., & Reina, A. (2018). Quality-sensitive foraging by a robot swarm through virtual pheromone trails. InM. Dorigo, M. Birattari, C. Blum, A. Christensen, A. Reina, & V. Trianni (Eds.), Swarm Intelligence (ANTS 2018), LNCS (Vol. 11172, pp. 135–149). Berlin: Springer.

[70] Fox, D., Burgard, W., Kruppa, H., & Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. Autonomous robots, 8(3), 325-344

[71] Fujisawa, R., Dobata, S., Kubota, D., Imamura, H., & Matsuno, F. (2008). Dependency by concentration of pheromone trail for multiple robots. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, & A. F. T. Winfield (Eds.), SAnt colony optimization and swarm intelligence (ANTS 2008), LNCS (Vol. 5217, pp. 283–290). Berlin: Springer.

[72] Fujisawa, R., Dobata, S., Sugawara, K., & Matsuno, F. (2014). Designing pheromone communication in swarm robotics: Group foraging behaviour mediated by chemical substance. Swarm Intelligence, 8(3), 227–246.

[73] Fukuda, T., & Nakagawa, S. (1988). Approach to the dynamically reconfigurable robotic system. Journal of Intelligent and Robotic Systems, 1(1), 55–72.

[74] Fukuda, T., & Kawauchi, Y. (1990, May). Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator. In Proceedings., IEEE International Conference on Robotics and Automation (pp. 662-667). IEEE.

[75] Fukuda, T., Husband, T., & Ueyama, T. (1994). Cellular robotics and micro robotic systems (Vol. 10). World Scientific.

[76] Gamroth, C. A. (2010). Automatic detection and tracking in underwater environments with marine snow (Doctoral dissertation, University of British Columbia).

[77] Garnier, S., Jost, C., Jeanson, R., Gautrais, J., Asadpour, M., Caprari, G., & Theraulaz, G. (2005, September). Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In European conference on artificial life (pp. 169-178). Springer, Berlin, Heidelberg.

[78] Garnier, S., Tâche, F., Combe, M., Grimal, A., & Theraulaz, G. (2007). Alice in pheromone land: An experimental setup for the study of ant-like robots. In Proceedings of the 2007 IEEE swarm intelligence symposium (SIS 2007) (pp. 37–44). IEEE.

[79] Garnier, S., Gautrais, J., Asadpour, M., Jost, C., & Theraulaz, G. (2009). Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. Adaptive Behaviour, 17(2), 109-133.

[80] Garnier, S., Combe, M., Jost, C., & Theraulaz, G. (2013). Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test bed. PLoS Computational Biology, 9(3), e1002903.

[81] Gauci, M., Chen, J., Li,W., Dodd, T. J., & Groß, R. (2014). Self-organized aggregation without computation. The International Journal of Robotics Research, 33(8), 1145–1161.

[82] Goldberg, D., & Mataric, M. J. (2000). Robust behaviour-based control for distributed multi-robot collection tasks. University of Southern California Los Angeles United States.

[83] Goss, S., Deneubourg, J. L., & Pasteels, J. M. (1989). Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76(12), 579–581.

[84] Goss, S., Deneubourg, J. L., Bourgine, P.,&Varela, E. (1992).Harvesting by a group of robots. In 1st European conference on artificial Life (pp. 195–204). MIT Press.

[85] Griffith, S., Goldwater, D., & Jacobson, J. M. (2005). Self-replication from random parts. nature, 437(7059), 636-636.

[86] Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. IEEE transactions on robotics, 22(6), 1115-1130.

[87] Groß, R., & Dorigo, M. (2008). Self-assembly at the macroscopic scale. Proceedings of the IEEE, 96(9), 1490-1508.

[88] Groß, R., & Dorigo, M. (2009). Towards group transport by swarms of robots. International Journal of Bio-Inspired Computation, 1(1-2), 1-13.

[89] Hackwood, S., and G. Beni. "Self-Organizing Sensors by Deterministic Annealing." In Proceedings 1991 IEEE/RSJ International Conference on Intelligent Robot and Systems, IROS'91, 1177-1183. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[90] Hackwood, S., and G. Beni. "Self-Organization of Sensors for Swarm Intelligence." In Proceedings IEEE 1992 International Conference on Robotics

[91] Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2019a). Ship hull inspection using a swarm of autonomous underwater robots: a Search algorithm. In 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 114-115). IEEE.

[92] Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2019b). Ship hull repair using a swarm of autonomous underwater robots: A self-assembly algorithm. In 2019 European Conference on Mobile Robots (ECMR) (pp. 1-6). IEEE.

[93] Hamann, H., Schmickl, T., Wörn, H., & Crailsheim, K. (2012). Analysis of emergent symmetry breaking in collective decision making. Neural Computing and Applications, 21(2), 207-218.

[94] Hayes, A. T., & Dormiani-Tabatabaei, P. (2002, May). Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In

Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292) (Vol. 4, pp. 3900-3905). IEEE.

[95] Hecker, J. P., Letendre, K., Stolleis, K., Washington, D., & Moses, M. E. (2012). Formica ex Machina: Ant swarm foraging from physical to virtual and back again. In M. Dorigo, et al. (Eds.), Swarm intelligence (ANTS 2012), LNCS (Vol. 7461, pp. 252–259). Berlin: Springer. Heinze, S., Narendra, A., & Cheung, A. (2018). Principles of insect path integration. Current Biology, 28(17), R1043–R1058.

[96] Herianto, Kurabayashi, D. (2009). Realization of an artificial pheromone system in random data carriers using RFID tags for autonomous navigation. In Proceedings of the 2009 IEEE/RSJ international conference on robotics and automation (ICRA 2009) (pp. 2288–2293). IEEE.

[97] Herianto, Sakakibara T., & Kurabayashi, D. (2007). Artificial pheromone system using RFID for navigation of autonomous robots. Journal of Bionic Engineering, 4(4), 245–253.

[98] Hoff, N. R., Sagoff, A., Wood, R. J., & Nagpal, R. (2010, December). Two foraging algorithms for robot swarms using only local communication. In 2010 IEEE International Conference on Robotics and Biomimetics (pp. 123-130). IEEE.

[99] Hoff, N., Wood, R., & Nagpal, R. (2012). Distributed colony-level algorithm switching for robot swarm foraging. In A. Martinoli, et al. (Eds.), Distributed autonomous robotic systems (DARS 2010), STAR (Vol. 83, pp. 417–430). Berlin: Springer.

[100] Hölldobler, B., & Wilson, E. O. (1990). The Ants. Cambridge: Harvard University Press.

[101] Houston, A. I.,&McNamara, J.M. (2014). Foraging currencies, metabolism and behavioural routines. Journal of Animal Ecology, 83(1), 30–40.

[102] Hover, F. S., Eustice, R. M., Kim, A., Englot, B., Johannsson, H., Kaess, M., & Leonard, J. J. (2012). Advanced perception, navigation and planning for autonomous in-water ship hull inspection. The International Journal of Robotics Research, 31(12), 1445-1464.

[103] Hosokawa, K., Shimoyama, I., & Miura, H. (1994). Dynamics of self-assembling systems: Analogy with chemical kinetics. Artificial Life, 1(4), 413-427.

[104] Howard, A., Matarić, M. J., & Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In Distributed Autonomous Robotic Systems 5 (pp. 299-308). Springer, Tokyo.

[105] Ijspeert, A. J., Martinoli, A., Billard, A., & Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. Autonomous Robots, 11(2), 149-171.

[106] Jevtic, A., Gutiérrez, A., Andina, D., & Jamshidi, M. (2011). Distributed bee's algorithm for task allocation in swarm of robots. IEEE Systems Journal, 6(2), 296-304.

[107] Joordens, M. A., & Jamshidi, M. (2010). Consensus control for a system of underwater swarm robots. IEEE Systems Journal, 4(1), 65-73.

[108] Kacelnik,A. (1984).Central place foraging in Starlings (Sturnus vulgaris). I. patches residence time. The Journal of Animal Ecology, 53(1), 283.

[109] Kazadi, S., Lee, J. R., & Lee, J. (2007, October). Artificial physics, swarm engineering, and the Hamiltonian method. In World congress on engineering and computer science (pp. 623-632).

[110] Kazadi, S., Lee, J. R., & Lee, J. (2009). Model independence in swarm robotics. Int. J. Intelligent Computing and Cybernetics, 2(4), 672-694.

[111] Khaliq, A. A., Di Rocco, M., & Saffiotti, A. (2014). Stigmergic algorithms for multiple minimalistic robots on an RFID floor. Swarm Intelligence, 8(3), 199–225.

[112] Khodayari, M. H., & Balochian, S. (2015). Modelling and control of autonomous underwater vehicle (AUV) in heading and depth attitude via self-adaptive fuzzy PID controller. Journal of Marine Science and Technology, 20(3), 559-578.

[113] Kopman, V., Cavaliere, N., & Porfiri, M. (2011). MASUV-1: A miniature underwater vehicle with multidirectional thrust vectoring for safe animal interactions. IEEE/ASME transactions on mechatronics, 17(3), 563-571.

[114] Krieger, M. J., & Billeter, J. B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. Robotics and Autonomous Systems, 30(1-2), 65-84.

[115] Kube, C., & Hong Zhang. (1993). Collective Robotics: From Social Insects to Robots. Adaptive Behaviour, 2(2), 189–218.

[116] Kumar, V., & Sahin, F. (2003, October). Cognitive maps in swarm robots for the mine detection application. In SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483) (Vol. 4, pp. 3364-3369). IEEE.

[117] Levander, O. (2017). Autonomous ships on the high seas. Spectrum, IEEE, 54(2), 26-31.

[118] Levi, P., & Kernbach, S. (Eds.). (2010). Symbiotic multi-robot organisms: reliability, adaptability, evolution (Vol. 7). Springer Science & Business Media.

[119] Liu, W., & Winfield, A. F. (2010, September). Autonomous morphogenesis in self-assembling robots using IR-based sensing and local communications. In International Conference on Swarm Intelligence (pp. 107-118). Springer, Berlin, Heidelberg.

[120] Liu, Y., & Nejat, G. (2013). Robotic urban search and rescue: A survey from the control perspective. Journal of Intelligent & Robotic Systems, 72(2), 147-165.

[121] Lodovisi, C., Loreti, P., Bracciale, L., & Betti, S. (2018). Performance analysis of hybrid optical–acoustic AUV swarms for marine monitoring. Future Internet, 10(7), 65.

[122] Ludwig, L., & Gini, M. (2006). Robotic swarm dispersion using wireless intensity signals. In Distributed Autonomous Robotic Systems 7 (pp. 135-144). Springer, Tokyo.

[123] Mabrouk, M. H., & McInnes, C. R. (2008). Solving the potential field local minimum problem using internal agent states. Robotics and Autonomous Systems, 56(12), 1050-1060.
[124] Madhavan, R., Fregene, K., & Parker, L. E. (2004). Distributed cooperative outdoor multirobot localization and mapping. Autonomous Robots, 17(1), 23-39.
[125] Maes, P., Mataric, M. J., Meyer, J. A., Pollack, J., & Wilson, S. W. (1996). A study of territoriality: The role of critical mass in adaptive task division.
[126] Mallios, A., Ridao, P., Ribas, D., Maurelli, F., & Petillot, Y. (2010, October). EKF-SLAM for AUV navigation under probabilistic sonar scan-matching. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4404-4411). IEEE.
[127] Mamei, M., & Zambonelli, F. (2005). Physical deployment of digital pheromones through RFID technology. In Proceedings of the 2005 IEEE swarm intelligence symposium (SIS 2005) (pp. 281–288). IEEE.
[128] Mamei, M., & Zambonelli, F. (2007). Pervasive pheromone-based interaction with RFID tags. ACM Transactions on Autonomous and Adaptive Systems, 2(2), 4.
[129] Mao, G., Fidan, B., & Anderson, B. D. (2007). Wireless sensor network localization techniques. Computer networks, 51(10), 2529-2553.
[130] Martinelli, A., Pont, F., & Siegwart, R. (2005, April). Multi-robot localization using relative observations. In Proceedings of the 2005 IEEE international conference on robotics and automation (pp. 2797-2802). IEEE.
[131] Martinoli, A. (1999). Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies (Doctoral dissertation, Verlag nicht ermittelbar).
[132] Martinoli, A., Ijspeert, A. J., & Gambardella, L. M. (1999, September). A probabilistic model for understanding and comparing collective aggregation mechanisms. In European Conference on Artificial Life (pp. 575-584). Springer, Berlin, Heidelberg
[133] Mathews, N., Christensen, A. L., O'Grady, R., Rétornaz, P., Bonani, M., Mondada, F., & Dorigo, M. (2011, September). Enhanced directional self-assembly based on active recruitment and guidance. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4762-4769). IEEE.
[134] Mayet, R., Roberz, J., Schmickl, T., & Crailsheim, K. (2010). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In M. Dorigo, et al. (Eds.), Swarm intelligence (ANTS 2010), LNCS (Vol. 6234, pp. 84–94). Berlin: Springer.
[135] Mayya, S., Pierpaoli, P., & Egerstedt, M. (2019). Voluntary retreat for decentralized interference reduction in robot swarms. In Proceedings of the 2019 IEEE/RSJ international conference on robotics and automation (ICRA 2019) (pp. 9667–9673). IEEE.
[136] McLurkin, J., & Smith, J. (2004). Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In in 7th International Symposium on Distributed Autonomous Robotic Systems (DARS).

[137] McPhail, S. (2009). Autosub6000: A deep diving long range AUV. Journal of Bionic Engineering, 6(1), 55-62.

[138] Mermoud, G., Upadhyay, U., Evans, W. C., & Martinoli, A. (2014). Top-down vs. bottom-up model-based methodologies for distributed control: a comparative experimental study. In Experimental Robotics (pp. 615-629). Springer, Berlin, Heidelberg.

[139] Mikkelsen, S. B., Jespersen, R., & Ngo, T. D. (2013). Probabilistic communication based potential force for robot formations: A practical approach. In Distributed Autonomous Robotic Systems (pp. 243-253). Springer, Berlin, Heidelberg.

[140] Moeslinger, C., Schmickl, T., & Crailsheim, K. (2010, September). Emergent flocking with low-end swarm robots. In International Conference on Swarm Intelligence (pp. 424-431). Springer, Berlin, Heidelberg.

[141] Mogilner, A., & Edelstein-Keshet, L. (1999). A non-local model for a swarm. Journal of mathematical biology, 38(6), 534-570.

[142] Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. W., Floreano, D., Deneubourg, J. L., ... & Dorigo, M. (2004). SWARM-BOT: A new distributed robotic concept. Autonomous robots, 17(2-3), 193-221.

[143] Mondada, F., Bonani, M., Guignard, A., Magnenat, S., Studer, C., & Floreano, D. (2005, September). Superliner physical performances in a SWARM-BOT. In European Conference on Artificial Life (pp. 282-291). Springer, Berlin, Heidelberg.

[144] Montague, P. R., Dayan, P., Person, C., & Sejnowski, T. J. (1995). Bee foraging in uncertain environments using predictive hebbian learning. Nature, 377(6551), 725-728.

[145] Montes de Oca, M., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., & Dorigo, M. (2010).Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. Swarm Intelligence, 5(3–4), 305–327

[146] Moravec, H., & Elfes, A. (1985, March). High resolution maps from wide angle sonar. In Proceedings. 1985 IEEE international conference on robotics and automation (Vol. 2, pp. 116-121). IEEE.

[147] Morlok, R., & Gini, M. (2007). Dispersing robots in an unknown environment. In Distributed Autonomous Robotic Systems 6 (pp. 253-262). Springer, Tokyo.

[148] Mullins, J., Meyer, B., & Hu, A. P. (2012, September). Collective robot navigation using diffusion limited aggregation. In International Conference on Parallel Problem Solving from Nature (pp. 266-276). Springer, Berlin, Heidelberg.

[149] Muniganti, P., & Pujol, A. O. (2010, May). A survey on mathematical models of swarm robotics. In Workshop of physical agents (pp. 29-30).

[150] Murata, S., Kakomura, K., & Kurokawa, H. (2006, October). Docking experiments of a modular robot by visual feedback. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 625-630). IEEE.

[151] Navarro, I., & Matía, F. (2013). An Introduction to Swarm Robotics. ISRN Robotics, 2013, 1–10.

[152] Navarro, I., & Matía, F. (2013). A survey of collective movement of mobile robots. International Journal of Advanced Robotic Systems, 10(1), 73.
[153] Nedjah, N., & Junior, L. (2019). Review of methodologies and tasks in swarm robotics towards standardization. Swarm and Evolutionary Computation, 50.
[154] Nonacs, P., & Dill, L. M. (1990). Mortality risk vs. food quality trade-offs in a common currency: Ant patch preferences. Ecology, 71(5), 1886–1892.
[155] Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2009). Teamwork in self-organized robot colonies. IEEE Transactions on Evolutionary Computation, 13(4), 695-711.
[156] O'Grady, R., Christensen, A. L., & Dorigo, M. (2009). SWARMORPH: multirobot morphogenesis using directional self-assembly. IEEE Transactions on Robotics, 25(3), 738-743.
[157] O'Grady, R., Groß, R., Christensen, A. L., & Dorigo, M. (2010). Self-assembly strategies in a group of autonomous mobile robots. Autonomous Robots, 28(4), 439-455.
[158] Okubo, A. (1986). Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. Advances in biophysics, 22, 1-94.
[159] Olsson, O., Brown, J. S., & Helf, K. L. (2008). A guide to central place effects in foraging. Theoretical population biology, 74(1), 22-33.
[160] Özdemir, A., Gauci, M.,Bonnet, S.,&Groß, R. (2018). Finding consensus without computation. IEEE Robotics and Automation Letters, 3(3), 1346–1353.
[161] Pagliara, R., Gordon, D. M., & Leonard, N. E. (2018). Regulation of harvester ant foraging as a closed-loop excitable system. PLOS Computational Biology, 14(12), e1006200.
[162] Paull, L., Saeedi, S., Seto, M., & Li, H. (2013). AUV navigation and localization: A review. IEEE Journal of Oceanic Engineering, 39(1), 131-149.
[163] Paull, L., Huang, G., Seto, M., & Leonard, J. J. (2015, May). Communication-constrained multi-AUV cooperative SLAM. In 2015 IEEE international conference on robotics and automation (ICRA) (pp. 509-516). IEEE.
[164] Payton, D. W., Daily, M., Estowski, R., Howard, M., & Lee, C. (2001). Pheromone robotics. Autonomous Robots, 11(3), 319–324.
[165] Pedlosky, J. (2013). Geophysical Fluid Dynamics. Springer Publishing.
[166] Penders, J., & Alboul, L. (2012). Emerging robot swarm traffic. International Journal of Intelligent Computing and Cybernetics, 5(3), 312–339.
[167] Pinciroli, C., Talamali, M. S., Reina, A., Marshall, J. A. R., & Trianni, V. (2018). Simulating Kilobots within ARGoS: Models and experimental validation. In M. Dorigo, M. Birattari, C. Blum, A. Christensen, A. Reina, & V. Trianni (Eds.), Swarm intelligence (ANTS 2018), LNCS (Vol. 11172, pp. 176–187). Berlin: Springer.
[168] Planqué, R., Van Den Berg, J. B., & Franks, N. R. (2010). Recruitment strategies and colony size in ants. PLoS One, 5(8), e11664.
[169] Poissonnier, L. A., Motsch, S., Gautrais, J., Buhl, J., & Dussutour, A. (2019). Still flowing, experimental investigation of ant traffic under crowded conditions. eLife (in press).

[170] Poduri, S., & Sukhatme, G. S. (2004, April). Constrained coverage for mobile sensor networks. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 (Vol. 1, pp. 165-171). IEEE.

[171] Pratte, M., Gervet, J., & Theraulaz, G. (1990). Behavioural Profiles in Polistes Dominulus (Christ) Wasp Societies: a Quantitative Study. Behaviour, 113(3-4), 223–249.

[172] Press, P. (1945). Handbook of Damage Control NAVPERS 16191.

[173] Purnamadjaja, A. H., & Russell, R. A. (2007). Guiding robots' behaviours using pheromone communication. Autonomous Robots, 23(2), 113–130.

[174] Quan, Q. (2017). Introduction to multicopter design and control (pp. 150-160). Beijing: Springer.

[175] Ranganathan, T., Thondiyath, A., & Kumar, S. P. S. (2015, February). Design and analysis of an underwater quadrotor-AQUAD. In 2015 IEEE Underwater Technology (UT) (pp. 1-5). IEEE.

[176] Rashid, A. T., Frasca, M., Ali, A. A., Rizzo, A., & Fortuna, L. (2015). Multi-robot localization and orientation estimation using robotic cluster matching algorithm. Robotics and Autonomous Systems, 63, 108-121.

[177] Reif, J. H., & Wang, H. (1999). Social potential fields: A distributed behavioural control for autonomous robots. Robotics and Autonomous Systems, 27(3), 171-194.

[178] Reina, A., Miletitch, R., Dorigo, M., & Trianni, V. (2015). A quantitative micro–macro link for collective decisions: the shortest path discovery/selection example. Swarm Intelligence, 9(2-3), 75-102.

[179] Reina, A., Valentini, G., Fernández-Oto, C., Dorigo, M., & Trianni, V. (2015). A design pattern for decentralised decision making. PloS one, 10(10).

[180] Reina, A., Salvaro, M., Francesca, G., Garattoni, L., Pinciroli, C., Dorigo, M., & Birattari, M. (2015). Augmented reality for robots: Virtual sensing technology applied to a swarm of e-pucks. In Proceedings of the 2015 NASA/ESA conference on adaptive hardware and systems (AHS 2015) (pp. 1–6). IEEE.

[181] Reina, A., Cope, A. J., Nikolaidis, E., Marshall, J. A. R., & Sabo, C. (2017). ARK: Augmented reality for Kilobots. IEEE Robotics and Automation Letters, 2(3), 1755–1761.

[182] Reynolds, C. W. (1987, August). Flocks, herds and schools: A distributed behavioural model. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (pp. 25-34).

[183] Robinson, E. J. H., Jackson, D. E., Holcombe, M., & Ratnieks, F. L. W. (2005). 'No entry' signal in ant foraging. Nature, 438(7067), 442–442.

[184] Roumeliotis, S. I., & Bekey, G. A. (2002). Distributed multirobot localization. IEEE transactions on robotics and automation, 18(5), 781-795.

[185] Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. Science, 345(6198), 795-799.

[186] Rubino, F., Nisticò, A., Tucci, F., & Carlone, P. (2020). Marine application of fibre reinforced composites: a review. Journal of Marine Science and Engineering, 8(1), 26.

[187] Rybski, P. E., Larson, A., Veeraraghavan, H., LaPoint, M., & Gini, M. (2004). Communication strategies in multi-robot search and retrieval: Experiences with mindart. In Proc. 7th Int. Symp. Distributed Autonomous Robotic Systems.

[188] Sahin, E., Labella, T. H., Trianni, V., Deneubourg, J. L., Rasse, P., Floreano, D., ... & Dorigo, M. (2002, October). SWARM-BOT: Pattern formation in a swarm of self-assembling mobile robots. In IEEE International Conference on Systems, Man and Cybernetics (Vol. 4, pp. 6-pp). IEEE.

[189] Şahin, E. (2005). Swarm Robotics: From Sources of Inspiration to Domains of Application. In Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers (Vol. 3342, pp. 10–20). Berlin, Heidelberg: Springer Berlin Heidelberg.

[190] Sallam, G., & Baroudi, U. (2015, October). COVER: a cooperative virtual force robot deployment technique. In 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (pp. 1308-1315). IEEE.

[191] Sarhadi, P., Noei, A. R., & Khosravi, A. (2016). Model reference adaptive PID control with anti-windup compensator for an autonomous underwater vehicle. Robotics and Autonomous Systems, 83, 87-93.

[192] Scheidler, A., Brutschy, A., Ferrante, E., & Dorigo, M. (2016). The k-unanimity rule for self-organized decision-making in swarms of robots. IEEE Transactions on Cybernetics, 46(5), 1175–1188.

[193] Schmickl, T., Möslinger, C., & Crailsheim, K. (2006, September). Collective perception in a robot swarm. In International Workshop on Swarm Robotics (pp. 144-157). Springer, Berlin, Heidelberg.

[194] Schmickl, T., & Hamann, H. (2011). BEECLUST: A swarm algorithm derived from honeybees. Bio-inspired Computing and Communication Networks. CRC Press (March 2011).

[195] Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., ... & Manfredi, L. (2011, October). CoCoRo--The Self-Aware Underwater Swarm. In 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (pp. 120-126). IEEE.

[196] Shaffer, Z., Sasaki, T., & Pratt, S. C. (2013). Linear recruitment leads to allocation and flexibility in collective foraging by ants. Animal Behaviour, 86(5), 967–975.

[197] Smogeli, Ø. N. (2006). Control of marine propellers: from normal to extreme conditions.

[198] Solomon, N. (2004). U.S. Patent Application No. 10/421,593.

[199] Soysal, O., & Sahin, E. (2005, June). Probabilistic aggregation strategies in swarm robotic systems. In Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. (pp. 325-332). IEEE.

[200] Spears, W. M., Spears, D. F., Hamann, J. C., & Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. Autonomous Robots, 17(2-3), 137-162.

[201] Statista. 2020. Global Merchant Fleet - Number Of Ships By Type 2019 | Statista. [online] Available at: https://www.statista.com/statistics/264024/number-of-merchant-ships-worldwide-by-type/ [Accessed 1 October 2020].

[202] Stickland, T. R., Britton, N. F., & Franks, N. R. (1999). Models of information flow in ant foraging: The benefits of both attractive and repulsive signals. In C. Detrain, J. L. Deneubourg, & J.M. Pasteels (Eds.), Information processing in social insects (pp. 83–100). Basel: Birkhäuser.

[203] Sugawara, K., Kazama, T., & Watanabe, T. (2004). Foraging behaviour of interacting robots with virtual pheromone. In Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS 2004) (Vol. 3, pp. 3074–3079). IEEE.

[204] Sumpter, D., & Pratt, S. (2003). A modelling framework for understanding social insect foraging. Behavioural Ecology and Socio-biology, 53(3), 131-144.

[205] Svennebring, J., & Koenig, S. (2004). Building terrain-covering ant robots: A feasibility study. Autonomous Robots, 16(3), 313–332.

[206] Talamali, M. S., Bose, T., Haire, M., Xu, X., Marshall, J. A., & Reina, A. (2020). Sophisticated collective foraging with minimalist agents: a swarm robotics test. Swarm Intelligence, 14(1), 25-56.

[207] Tan, H. P., Diamant, R., Seah, W. K., & Waldmeyer, M. (2011). A survey of techniques and challenges in underwater localization. Ocean Engineering, 38(14-15), 1663-1676.

[208] Teo, K., Ong, K. W., & Lai, H. C. (2009, October). Obstacle detection, avoidance and anti-collision for MEREDITH AUV. In OCEANS 2009 (pp. 1-10). IEEE.

[209] Thrun, S. (2002). Robotic mapping: A survey. Exploring artificial intelligence in the new millennium, 1(1-35), 1.

[210] Traniello, J. F. (1989). Foraging strategies of ants. Annual review of entomology, 34(1), 191-210.

[211] Turgut, A. E., Çelikkanat, H., Gökçe, F., & Şahin, E. (2008). Self-organized flocking in mobile robot swarms. Swarm Intelligence, 2(2-4), 97-120.

[212] Ugur, E., Turgut, A. E., & Sahin, E. (2007, November). Dispersion of a swarm of robots based on realistic wireless intensity signals. In 2007 22nd international symposium on computer and information sciences (pp. 1-6). IEEE.

[213] Valentini, G., Antoun, A., Trabattoni, M., Wiandt, B., Tamura, Y., Hocquard, E., et al. (2018). Kilogrid: A novel experimental environment for the Kilobot robot. Swarm Intelligence, 12(3), 245–266.

[214] Vandermeulen, I., Groß, R., & Kolling, A. (2019, May). Turn-minimizing multirobot coverage. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 1014-1020). IEEE.

[215] Vanualailai, J., & Sharma, B. N. (2010). A Lagrangian-based swarming behaviour in the absence of obstacles. In Proceedings of the Workshop on Mathematical Control Theory (pp. 119-135).

[216] Vásárhelyi, G., Virágh, C., Somorjai, G., Tarcai, N., Szörényi, T., Nepusz, T., & Vicsek, T. (2014, September). Outdoor flocking and formation flight with autonomous aerial robots. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3866-3873). IEEE.

[217] Vaughan, R. T., Støy, K., Sukhatme, G. S., & Matarić, M. J. (2000, June). Whistling in the dark: cooperative trail following in uncertain localization space. In Proceedings of the fourth international conference on Autonomous agents (pp. 187-194).

[218] Visioli, A. (2006). Practical PID control. Springer Science & Business Media.

[219] Wang, Z., Hang, G., Li, J., Wang, Y., & Xiao, K. (2008). A micro-robot fish with embedded SMA wire actuated flexible biomimetic fin. Sensors and Actuators A: Physical, 144(2), 354-360.

[220] Wang, Y., Liang, A., & Guan, H. (2011, April). Frontier-based multi-robot map exploration using particle swarm optimization. In 2011 IEEE symposium on Swarm intelligence (pp. 1-6). IEEE.

[221] Wei, H., Chen, Y., Tan, J., & Wang, T. (2010). Sambot: A self-assembly modular robot system. IEEE/ASME Transactions on Mechatronics, 16(4), 745-757.

[222] Werger, B.B., Matari´c, M.J. (1996). Robotic “food” chains: Externalization of state and program for minimal agent foraging. In From animals to animats 4. Proceedings of the 4th international conference on simulation of adaptive behaviour (SAB 96) (pp. 625–634). MIT Press.

[223] White, P. J., Kopanski, K., & Lipson, H. (2004, April). Stochastic self-reconfigurable cellular robotics. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 (Vol. 3, pp. 2888-2893). IEEE.

[224] White, P., Zykov, V., Bongard, J. C., & Lipson, H. (2005, June). Three Dimensional Stochastic Reconfiguration of Modular Robots. In Robotics: Science and Systems (pp. 161-168).

[225] Witkowski, U., El Habbal, M. A. M., Herbrechtsmeier, S., Tanoto, A., Penders, J., Alboul, L., & Gazi, V. (2008). Ad-hoc network communication infrastructure for multi-robot systems in disaster scenarios. In Proceedings of IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance (RISE 2008), Benicassim, Spain.

[226] Wolfram, S. (1983). Cellular automata. Cellular Automata Modelling of Chemical Systems, 9.

[227] Woods, S. A., Bauer, R. J., & Seto, M. L. (2012). Automated ballast tank control system for autonomous underwater vehicles. IEEE Journal of Oceanic Engineering, 37(4), 727-739.

[228] Yamakita, M., Taniguchi, Y., & Shukuya, Y. (2003, September). Analysis of formation control of cooperative transportation of mother ship by SMC. In 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422) (Vol. 1, pp. 951-956). IEEE.

[229] Yim, M., Duff, D. G., & Roufas, K. D. (2000, April). PolyBot: a modular reconfigurable robot. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 1, pp. 514-520). IEEE.

[230] Yun, S., Lee, J., Chung, W., Kim, E., & Kim, S. (2009). A soft computing approach to localization in wireless sensor networks. Expert Systems with Applications, 36(4), 7552-7561.

[231] Zarzhitsky, D., Spears, D. F., & Spears, W. M. (2005, August). Distributed robotics approach to chemical plume tracing. In 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4034-4039). IEEE.

[232] Zhang, G., Fricke, G. K., & Garg, D. P. (2011). Spill detection and perimeter surveillance via distributed swarming agents. IEEE/asme Transactions on Mechatronics, 18(1), 121-129.

# Appendix A: Ship Hull Inspection Webots Simulation Code

Appendix A features the code used to carry out the experiments from Chapters 3. Webots was the program used for the Search algorithm simulations. The code used to perform the Partial Population Failure (PPF) experiments for the sweeping search is:

SHI_A1_PPF_10.wbt, SHIR_A1_SUP_PPF_10.c, and SHIR_A1_ROB_PPF.c

The c programs displayed here are those used by the robots to perform ship hull inspection according to the design presented in the main text of Chapter 3. These can also be accessed using the GitHub repository: https://github.com/MattSHaire/Emergency-Ship-Hull-Repair. The code used to perform the Partial Population Failure (PPF) experiments for the lawnmower search is:

SHI_A2_PPF_10.wbt, SHIR_A2_SUP_PPF_10.c, and SHIR_A2_CON.c

These files can be executed on the Webots Desktop App, which can be accessed from the following link: https://www.cyberbotics.com/download/download.

## SHIR_A1_ROB_PPF.c

```
/*
 * File:         SHIR_A1_ROB_PPF.c
 * Date:         03/04/2018
 * Description:   This controller tells each of the robots how to behave in order
 *               to achieve complete area coverage of the ship hull during inspection.
 * Author:       Matthew Haire
 */

/* Webot specific libraries */
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/supervisor.h>
#include <webots/gps.h>
#include <webots/inertial_unit.h>
#include <webots/emitter.h>
#include <webots/receiver.h>
#include <webots/distance_sensor.h>

/* Standard C libraries */
#include <time.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
```

```
/* Macros used throughout program */
#define TIME_STEP 16.0
#define WATERLINE 19.0
#define MAXDIST 2.0
#define SPACE 2.0
#define ASSEMBLY_TIME 10.0

/* PID Controller Structure */
struct SPID
{
 double pGain, iGain, dGain; // proportional, integral, and differential gains
 double iState; // integrator state
 double dState; // last position input
} SPID_X, SPID_Y, SPID_Z;

/* Update Robot position function */
double UpdatePos(struct SPID pid, double current_pos, double desired_pos);

/* Main program */
int main(int argc, char **argv)
{
 wb_robot_init();
 /* Distance sensors */
 const WbDeviceTag ds_ft = wb_robot_get_device("ds_ft");
 wb_distance_sensor_enable(ds_ft, TIME_STEP);
 const WbDeviceTag ds_fb = wb_robot_get_device("ds_fb");
 wb_distance_sensor_enable(ds_fb, TIME_STEP);
 const WbDeviceTag ds_fl = wb_robot_get_device("ds_fl");
 wb_distance_sensor_enable(ds_fl, TIME_STEP);
 const WbDeviceTag ds_fr = wb_robot_get_device("ds_fr");
 wb_distance_sensor_enable(ds_fr, TIME_STEP);
 const WbDeviceTag ps_t = wb_robot_get_device("ps_t");
 wb_distance_sensor_enable(ps_t, TIME_STEP);
 const WbDeviceTag ps_b = wb_robot_get_device("ps_b");
 wb_distance_sensor_enable(ps_b, TIME_STEP);
 const WbDeviceTag ps_l = wb_robot_get_device("ps_l");
 wb_distance_sensor_enable(ps_l, TIME_STEP);
 const WbDeviceTag ps_r = wb_robot_get_device("ps_r");
 wb_distance_sensor_enable(ps_r, TIME_STEP);
 /* GPS */
 const WbDeviceTag gps  = wb_robot_get_device("gps");
 wb_gps_enable(gps, TIME_STEP);
 /* Motors */
 const WbDeviceTag rmxft = wb_robot_get_device("rmxft");
 wb_motor_set_position(rmxft, INFINITY);
 wb_motor_set_velocity(rmxft, 0);
 const WbDeviceTag rmxfb = wb_robot_get_device("rmxfb");
 wb_motor_set_position(rmxfb, INFINITY);
 wb_motor_set_velocity(rmxfb, 0);
 const WbDeviceTag rmxfl = wb_robot_get_device("rmxfl");
 wb_motor_set_position(rmxfl, INFINITY);
 wb_motor_set_velocity(rmxfl, 0);
 const WbDeviceTag rmxfr = wb_robot_get_device("rmxfr");
```

```
 wb_motor_set_position(rmxfr, INFINITY);
 wb_motor_set_velocity(rmxfr, 0);
 const WbDeviceTag rmy = wb_robot_get_device("rmy");
 wb_motor_set_position(rmy, INFINITY);
 wb_motor_set_velocity(rmy, 0);
 const WbDeviceTag rmz = wb_robot_get_device("rmz");
 wb_motor_set_position(rmz, INFINITY);
 wb_motor_set_velocity(rmz, 0);

 /* PID initial Controller variables */
 SPID_X.pGain = 1;    // increase speed of response
 SPID_X.iGain = 100;   // reduce error between actual and desired sensor values
 SPID_X.dGain = -85;   // Remove oscillation, but increase cumulative error (threatens stability
of system)
 SPID_X.iState = 0;
 SPID_X.dState = 0.001;
 SPID_Y = SPID_X;
 SPID_Z = SPID_X;

 /* Local variables */
 double x_pos = 0.0, y_pos = 0.0, z_pos = 0.0;
 double goal_x = 24.0, goal_y = 18.25, end_goal = -20.0;
 //double lost_x = 0.0, lost_y = 0.0, lost_z = 0.0;
 double ps_error = 0.0;
 bool ASSEMBLY_COMPLETE = 0, SEARCH_COMPLETE = 0, WAIT = 0;
 // set random fail time
 srand(time(NULL));
 int fail_time = (rand() % 10) +1;

 /* Main loop */
 while (wb_robot_step(TIME_STEP) != -1)
 {
  x_pos = wb_gps_get_values(gps)[0];
  y_pos = wb_gps_get_values(gps)[1];
  z_pos = wb_gps_get_values(gps)[2];

   ASSEMBLY_COMPLETE = 1; // SKIP ASSEMBLY STAGE

   if(!ASSEMBLY_COMPLETE)
   {
    /* ASSEMBLY STAGE */
    // Set motor velocity of lost agents to 0.0
    if(wb_distance_sensor_get_value(ds_ft) >= 3.0 && wb_distance_sensor_get_value(ds_fb)
>= 3.0 && wb_distance_sensor_get_value(ds_fl) >= 3.0 &&
wb_distance_sensor_get_value(ds_fr) >= 3.0)
    {

    // change this section so that if the robot loses it tethers with other robots it defaults to
the position where it lost contact and waits.
      wb_motor_set_velocity(rmxft, 0.0);
      wb_motor_set_velocity(rmxfb, 0.0);
      wb_motor_set_velocity(rmxfl, 0.0);
      wb_motor_set_velocity(rmxfr, 0.0);
```

```
        wb_motor_set_velocity(rmy, 0.0);
        wb_motor_set_velocity(rmz, 0.0);
      }
        // Stay centred
        wb_motor_set_velocity(rmz, -(UpdatePos(SPID_Z, x_pos, goal_x)));
        //get difference in distance between agents using sensors
        ps_error = wb_distance_sensor_get_value(ps_t) -
wb_distance_sensor_get_value(ps_b);

         // Maintain distance of 2.0m from ship hull
         wb_motor_set_velocity(rmxft, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_ft), MAXDIST))/2);
         wb_motor_set_velocity(rmxfb, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fb), MAXDIST))/2);
         wb_motor_set_velocity(rmxfl, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fl), MAXDIST))/2);
         wb_motor_set_velocity(rmxfr, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fr), MAXDIST))/2);

        //set desired positions
        if(wb_distance_sensor_get_value(ps_t) >= 3.0 && y_pos >= WATERLINE && z_pos > 0.0)
        {
         wb_motor_set_velocity(rmy, (UpdatePos(SPID_Y, y_pos, WATERLINE + 0.5))*2); // Stay
close to the
        }
        else if(wb_distance_sensor_get_value(ps_b) >= 3.0 && y_pos >= WATERLINE && z_pos
< 0.0)
        {
         wb_motor_set_velocity(rmy, -(UpdatePos(SPID_Y, y_pos, WATERLINE + 0.5))*2); //
Stay close to the Waterline
        }
        else
        {
         wb_motor_set_velocity(rmy, -(UpdatePos(SPID_Y, ps_error, 0.0)));
        }
     //timer for assembly complete
     if(wb_robot_get_time() >= ASSEMBLY_TIME) ASSEMBLY_COMPLETE = 1;
  }
  else
  {
   /* SEARCH STAGE */
   // search complete
   if(x_pos <= end_goal || wb_robot_get_time() >= fail_time) SEARCH_COMPLETE = 1;

   if(!SEARCH_COMPLETE)
   {
    // Move at a steady speed towards the end position
    wb_motor_set_velocity(rmz, 5.0);
    //get difference in distance between agents using sensors
    ps_error = wb_distance_sensor_get_value(ps_t) - wb_distance_sensor_get_value(ps_b);

        // Maintain distance of 2.0m from ship hull
```

```
          wb_motor_set_velocity(rmxft, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_ft), MAXDIST))/2);
          wb_motor_set_velocity(rmxfb, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fb), MAXDIST))/2);
          wb_motor_set_velocity(rmxfl, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fl), MAXDIST))/2);
          wb_motor_set_velocity(rmxfr, (UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fr), MAXDIST))/2);

      // set desired positions
      if(wb_distance_sensor_get_value(ps_t) >= 3.0 && y_pos >= WATERLINE && z_pos > 0.0)
      {
        wb_motor_set_velocity(rmy, (UpdatePos(SPID_Y, y_pos, WATERLINE + 0.5))*2); // Stay
close to the Waterline
      }
      else if(wb_distance_sensor_get_value(ps_b) >= 3.0 && y_pos >= WATERLINE && z_pos <
0.0)
      {
        wb_motor_set_velocity(rmy, -(UpdatePos(SPID_Y, y_pos, WATERLINE + 0.5))*2); // Stay
close to the Waterline
      }
      else
      {
        wb_motor_set_velocity(rmy, -(UpdatePos(SPID_Y, ps_error, 0.0)));
      }
    }
    else
    {
      if(!WAIT)
      {
        /* EXIT STAGE */
        goal_x = x_pos - 0.5;
        goal_y = y_pos;

        /* print fail_time to file */
        FILE * fp;
        fp = fopen("SHIR_A1_PPF_10_FT.txt", "a");
        fprintf(fp, "\nFail Time = %3d", fail_time);
        fclose(fp);

        WAIT = 1;
      }
      else
      {
        // Maintain positions until end of simulation
        wb_motor_set_velocity(rmz, -UpdatePos(SPID_Z, x_pos, goal_x));
        wb_motor_set_velocity(rmy, UpdatePos(SPID_Y, y_pos, goal_y));
        wb_motor_set_velocity(rmxft, UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_ft), MAXDIST));
        wb_motor_set_velocity(rmxfb, UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fb), MAXDIST));
        wb_motor_set_velocity(rmxfl, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fl),
MAXDIST));
```

```
        wb_motor_set_velocity(rmxfr, UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fr), MAXDIST));
      }
     }
    }
  }
 wb_robot_cleanup();
 return(0);
}

double UpdatePos(struct SPID pid, double current_pos, double desired_pos)
{
 double error = desired_pos - current_pos;
 double pTerm, iTerm, dTerm;
 pTerm = pid.pGain * error;
 pid.iState += error;
 iTerm = pid.iGain * pid.iState;
 dTerm = pid.dGain  * (error - pid.dState);
 pid.dState = error;
 double result = pTerm + iTerm + dTerm;
 if(result >= 10.0) return(10.0);
 else if(result <= -10.0) return(-10.0);
 else return(result);
}
```

## SHIR_A2_CON.c

```
/*
* File:         SHIR_A2_CON.c
* Date:         25/03/2019
* Description:   This controller tells each of the robots how to behave in order
*               to achieve complete area coverage of the ship hull during inspection.
* Author:       Matthew Haire
*/

/* Webot specific libraries */
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/supervisor.h>
#include <webots/gps.h>
#include <webots/inertial_unit.h>
#include <webots/emitter.h>
#include <webots/receiver.h>
#include <webots/distance_sensor.h>

/* Standard C libraries */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

/* Macros used throughout program */
```

```
#define TIME_STEP 16.0
#define WATERLINE 19.0
#define MAXDIST 2.00
#define ASSEMBLY_TIME 2.00

/* PID Controller Structure */
struct SPID
{
 double pGain, iGain, dGain; // proportional, integral, and differential gains
 double iState; // integrator state
 double dState; // last position input
} SPID_X, SPID_Y, SPID_Z;

/* Update Robot position function */
double UpdatePos(struct SPID pid, double current_pos, double desired_pos);

/* Main program */
int main(int argc, char **argv)
{
 wb_robot_init();
 /* Distance sensors */
 const WbDeviceTag ds_ft = wb_robot_get_device("ds_ft");
 wb_distance_sensor_enable(ds_ft, TIME_STEP);
 const WbDeviceTag ds_fb = wb_robot_get_device("ds_fb");
 wb_distance_sensor_enable(ds_fb, TIME_STEP);
 const WbDeviceTag ds_fl = wb_robot_get_device("ds_fl");
 wb_distance_sensor_enable(ds_fl, TIME_STEP);
 const WbDeviceTag ds_fr = wb_robot_get_device("ds_fr");
 wb_distance_sensor_enable(ds_fr, TIME_STEP);
 /*
 const WbDeviceTag ps_t = wb_robot_get_device("ps_t");
 wb_distance_sensor_enable(ps_t, TIME_STEP);
 const WbDeviceTag ps_b = wb_robot_get_device("ps_b");
 wb_distance_sensor_enable(ps_b, TIME_STEP);
 const WbDeviceTag ps_l = wb_robot_get_device("ps_l");
 wb_distance_sensor_enable(ps_l, TIME_STEP);
 const WbDeviceTag ps_r = wb_robot_get_device("ps_r");
 wb_distance_sensor_enable(ps_r, TIME_STEP);
 */
 /* GPS */
 const WbDeviceTag gps  = wb_robot_get_device("gps");
 wb_gps_enable(gps, TIME_STEP);
 /* Motors */
 const WbDeviceTag rmxft = wb_robot_get_device("rmxft");
 wb_motor_set_position(rmxft, INFINITY);
 wb_motor_set_velocity(rmxft, 0);
 const WbDeviceTag rmxfb = wb_robot_get_device("rmxfb");
 wb_motor_set_position(rmxfb, INFINITY);
 wb_motor_set_velocity(rmxfb, 0);
 const WbDeviceTag rmxfl = wb_robot_get_device("rmxfl");
 wb_motor_set_position(rmxfl, INFINITY);
 wb_motor_set_velocity(rmxfl, 0);
 const WbDeviceTag rmxfr = wb_robot_get_device("rmxfr");
```

```
  wb_motor_set_position(rmxfr, INFINITY);
  wb_motor_set_velocity(rmxfr, 0);
  const WbDeviceTag rmy = wb_robot_get_device("rmy");
  wb_motor_set_position(rmy, INFINITY);
  wb_motor_set_velocity(rmy, 0);
  const WbDeviceTag rmz = wb_robot_get_device("rmz");
  wb_motor_set_position(rmz, INFINITY);
  wb_motor_set_velocity(rmz, 0);

  /* PID initial Controller variables */
  SPID_X.pGain = 1;   // increase speed of responce
  SPID_X.iGain = 100; // reduce the error caused by gravity pulling object down
  SPID_X.dGain = -85;   // Remove oscillation
  SPID_X.iState = 0;
  SPID_X.dState = 0.001;
  SPID_Y = SPID_X;
  SPID_Y.pGain = 2;
  SPID_Z = SPID_Y;

  /* Local variables */
  double x_pos = 0.0, y_pos = 0.0, z_pos = 0.0;
  double goal_x = 0.0, goal_y = 0.0;
  bool SEARCH_COMPLETE = 0, DOWN = 1, UP = 0;

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1)
  {
    x_pos = wb_gps_get_values(gps)[0];
    y_pos = wb_gps_get_values(gps)[1];
    z_pos = wb_gps_get_values(gps)[2];

    // This approach does not require an assembly protocol
    if(wb_robot_get_time() <= TIME_STEP)
    {
      goal_x = x_pos;
    }

    if(!SEARCH_COMPLETE)
    {
      // If agent goes out of bounds, deactivate.
      if(x_pos <= -21.0 || x_pos >= 26.0 || y_pos <= 12.0 || y_pos >= 21.0 || z_pos >= 9.0 ||
z_pos <= -7.0)
      {
        wb_motor_set_velocity(rmxft, 0.0);
        wb_motor_set_velocity(rmxfb, 0.0);
        wb_motor_set_velocity(rmxfl, 0.0);
        wb_motor_set_velocity(rmxfr, 0.0);
        wb_motor_set_velocity(rmy, 0.0);
        wb_motor_set_velocity(rmz, 0.0);
        break;
      }
      else
      {
```

```
    //printf("DOWN: %d UP: %d\n", DOWN, UP);
    // Maintain distance of 2.0m from ship hull
    wb_motor_set_velocity(rmxft, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_ft),
MAXDIST)/2);
    wb_motor_set_velocity(rmxfb, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fb),
MAXDIST)/2);
    wb_motor_set_velocity(rmxfl, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fl),
MAXDIST)/2);
    wb_motor_set_velocity(rmxfr, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fr),
MAXDIST)/2);
    // MOVE DOWN
    if(DOWN && !UP)
    {
     wb_motor_set_velocity(rmy, -5.0);
     wb_motor_set_velocity(rmz, -UpdatePos(SPID_Z, x_pos, goal_x));
     if(z_pos < 0.0 && y_pos >= WATERLINE)
     {
      UP = 1;
      goal_x = (x_pos + 1.0);
     }
    }
    // MOVE FORWARD
    if(DOWN && UP)
    {
     wb_motor_set_velocity(rmy, -UpdatePos(SPID_Y, y_pos, WATERLINE));
     wb_motor_set_velocity(rmz, -5.0);
     if(x_pos >= goal_x)
     {
      DOWN = 0;
      goal_x = x_pos;
     }
    }
    // MOVE UP
    if(!DOWN && UP)
    {
     wb_motor_set_velocity(rmy, 5.0);
     wb_motor_set_velocity(rmz, -UpdatePos(SPID_Z, x_pos, goal_x));
     if(z_pos > 0.0 && y_pos >= WATERLINE)
     {
      UP = 0;
      DOWN = 0;
      goal_x = (x_pos + 1.0);
     }
    }
    // MOVE FORWARD
    if(!DOWN && !UP)
    {
     wb_motor_set_velocity(rmy, UpdatePos(SPID_Y, y_pos, WATERLINE));
     wb_motor_set_velocity(rmz, -5.0);
     if(x_pos >= goal_x)
     {
      DOWN = 1;
      goal_x = x_pos;
```

```
        }
       }
       // SEARCH COMPLETE
       //This section requires editing for PPF experiments
       if(x_pos >= 27.0)
       {
        SEARCH_COMPLETE = 1;
        goal_x = x_pos + 1.0;
        goal_y = WATERLINE;
       }
      }
     }
     else
     {
      break;
      // ASSEMBLE AT THE FRONT OF THE SHIP
      //wb_motor_set_velocity(rmxft, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_ft),
MAXDIST));
      //wb_motor_set_velocity(rmxfb, UpdatePos(SPID_X,
wb_distance_sensor_get_value(ds_fb), MAXDIST));
      //wb_motor_set_velocity(rmxfl, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fl),
MAXDIST));
      //wb_motor_set_velocity(rmxfr, UpdatePos(SPID_X, wb_distance_sensor_get_value(ds_fr),
MAXDIST));
      //wb_motor_set_velocity(rmy, UpdatePos(SPID_Y, y_pos, goal_y));
      //wb_motor_set_velocity(rmz, -UpdatePos(SPID_Z, x_pos, goal_x));
     }
    }
    wb_robot_cleanup();
    return(0);
}
double UpdatePos(struct SPID pid, double current_pos, double desired_pos)
{
    double error = desired_pos - current_pos;
    double pTerm, iTerm, dTerm;
    pTerm = pid.pGain * error;
    pid.iState += error;
    iTerm = pid.iGain * pid.iState;
    dTerm = pid.dGain  * (error - pid.dState);
    pid.dState = error;
    double result = pTerm + iTerm + dTerm;
    return(result);
}
```

# Appendix B: Ship Hull Repair Netlogo Simulation Code

Appendix B features the code used to carry out the experiments from Chapters 4. Netlogo was the simulation suite used for the Self-assembly algorithm simulations. The two sets of code displayed here are: ESHR SA Experiment 1 and ESHR SA Experiment 2. These can be files can be downloaded from the GitHub repository (https://github.com/MattSHaire/Emergency-Ship-Hull-Repair) and executed on the Netlogo Desktop App or uploaded and executed in a browser using Netlogo Web which can be accessed from https://www.netlogoweb.org/

## ESHR SA Experiment 1

```
;; EMERGENCY SHIP HULL REPAIR
;; SELF ASSEMBLING AGENTS APPROACH VERSION 3.0
;; BY MATTHEW HAIRE
;; LAST EDITED:    03 May 2019

globals
[
  seal  ;; check if breach is sealed or not
  speed ;; movement speed of turtles
  goalx ;; goal x coordinate
  goaly ;; goal y coordinate
  sproutx ;; x coordinates of turtle for creation
  increments ;; while loop variable for robots
  spacing ;; spacing for robots
  turtles_attached ;; robots that form part of breach
  turtles_unattached ;; robots that remain unattached
]

turtles-own
[
  active  ;; state of the turtle - either in transit or in position
  goalpos ;; goal position of turtle - either centre, left or right or breach
  agent_ahead ;; reporter for color of agent ahead of turtle
]

to setup
  clear-all
  reset-ticks
  set seal 0
  set speed 1
  set turtles_attached 0
  set turtles_unattached 0
  setup-environment
  setup-turtles
  vid:start-recorder
```

```
end

to setup-environment
 resize-world -50 50 -50 50
 set-patch-size 5
 set goalx breachx
 set goaly (breachy + breachsize + 1)
 ask patches
 [
  set pcolor 5
  if pycor <= (min-pycor + (max-pycor * 1.9))
  [
   set pcolor 105
  ]
  ask patch breachx breachy
  [
   set pcolor 101
   ask patches in-radius breachsize
   [
    set pcolor 101
   ]
  ]
 ]
end

to setup-turtles

 set increments 0
 set sproutx 0
 set spacing (96 / robotpop)

 while [increments < (robotpop / 2)]
 [
  set sproutx (sproutx - spacing)
  ask patch sproutx 48
  [
   Sprout 1
  ]
  set increments (increments + 1)
 ]

 set sproutx 0
 while [increments < robotpop]
 [
  set sproutx (sproutx + spacing)
  ask patch sproutx 48
  [
   Sprout 1
  ]
```

```
   set increments (increments + 1)
  ]

  ask turtles
   [
    set active 1
    set goalpos "centre"
    set shape "square"
    set size 1
    set color red
    set agent_ahead "null"
   ]
end

to start
  if seal = 1 OR ticks > 1000
  [
   set turtles_attached count turtles with [xcor >= (breachx - breachsize - 1) AND xcor
<= (breachx + breachsize + 1) AND ycor >= (breachy - breachsize - 1) AND ycor <=
(breachy + breachsize + 1)]

   if breachsize = 6 [set turtles_unattached count turtles - turtles_attached]
   if breachsize = 5 [set turtles_unattached count turtles - turtles_attached]
   if breachsize = 4 [set turtles_unattached count turtles - turtles_attached]
   if breachsize = 3 [set turtles_unattached count turtles - turtles_attached]
   if breachsize = 2 [set turtles_unattached count turtles - turtles_attached]
   if breachsize = 1 [set turtles_unattached count turtles - turtles_attached]

   show count turtles
   show turtles_unattached
   show seal
   stop
  ]
  turtle-actions
  spawn-turtles
  advance-line
  tick
end

to turtle-actions
  ask turtles
  [
   if active = 1
   [
    if goalpos = "left"
    [
     ifelse pycor > (goaly + 1)
     [
      setxy pxcor (goaly + 1)
```

```
  ]
  [
    ifelse not any? (turtles-on patch-left-and-ahead 90 1)
    [
      set agent_ahead "null"
      setxy pxcor goaly
      set color orange
      set active 0
      if pxcor > goalx
      [
        set goalpos "right"
      ]
    ]
    [
      set heading 270
      forward speed
    ]
    if pxcor < (goalx - breachsize - 1)
    [
      set goalpos "lost"
      set active 1
    ]
  ]
]
if goalpos = "right"
[
  ifelse pycor > (goaly + 1)
  [
    setxy pxcor (goaly + 1)
  ]
  [
    ifelse not any? (turtles-on patch-right-and-ahead 90 1)
    [
      set agent_ahead "null"
      setxy pxcor goaly
      set color orange
      set active 0
      if pxcor < goalx
      [
        set goalpos "left"
      ]
    ]
    [
      set heading 90
      forward speed
    ]
    if pxcor > (goalx + breachsize + 1)
    [
      set goalpos "lost"
```

```
        set active 1
      ]
    ]
  ]

  if goalpos = "lost"
  [
    facexy pxcor (goaly + 6)
    ifelse not any? (turtles-on patch-ahead 1.5)
    [
    set agent_ahead "null"
    ifelse pycor >= (goaly + 6)
     [
      set goalpos "centre"
      facexy goalx goaly
      set color red
     ]
     [
      forward speed
     ]
    ]
    [
     if any? (turtles-on patch-ahead 1.5) with [color = red]
     [
      set agent_ahead "red"
      back speed ;; previously 1
     ]
     if any? (turtles-on patch-ahead 1.5) with [color = orange]
     [
      set agent_ahead "orange"
      back speed ;; previously 1
     ]
     if any? (turtles-on patch-ahead 1.5) with [color = green]
     [
      set agent_ahead "green"
      back speed ;; previously 1
     ]
    ]
  ]

  if goalpos = "centre"
  [
   facexy goalx goaly
   ifelse not any? (turtles-on patch-ahead 1.5)
   [
    set agent_ahead "null"
    ifelse (pxcor < (goalx + 0.5) AND pxcor > (goalx - 0.5) AND pycor > (goaly - 0.5)
AND pycor < (goaly + 0.5))
     [
```

```
          setxy goalx goaly
          set heading 180
          set color orange
          set active 0
        ]
        [
          forward speed ;; previously 1
        ]
      ]
      [
        if any? (turtles-on patch-ahead 1.5) with [color = red]
        [
          set agent_ahead "red"
          back speed ;; previously 1
        ]
        if any? (turtles-on patch-ahead 1.5) with [color = orange]
        [
          set agent_ahead "orange"
          setxy pxcor pycor
          if not any? (turtles-on patch (goalx - 1) goaly) OR any? (turtles-on patch (goalx -
1) goaly) with [color = orange]
          [
            set heading 270
            ifelse any? (turtles-on patch-ahead 1) OR any? (turtles-on patch-ahead 2)
            [
              set heading 90
              set goalpos "right"
            ]
            [
              set heading 270
              set goalpos "left"
            ]
          ]
          if (not any? (turtles-on patch (goalx + 1) goaly) OR any? (turtles-on patch (goalx
+ 1) goaly) with [color = orange]) AND (goalpos != "left")
          [
            set heading 90
            set goalpos "right"
          ]
        ]
        if any? (turtles-on patch-ahead 1.5) with [color = green]
        [
          set agent_ahead "green"
          ifelse not any? (turtles-on patch goalx goaly)
          [
            setxy goalx goaly
            set heading 180
            set color orange
            set active 0
```

```
          ]
          [
            back (speed / 2)
          ]
        ]
      ]
    ]
   ]
   if active = 0
   [
    ifelse goalpos = "centre"
    [
     if any? ((turtles-on patch (goalx - 1) goaly) with [color = green]) AND any? ((turtles-
on patch (goalx + 1) goaly) with [color = green])
      [
       set color green
      ]
    ]
    [
     ifelse xcor < (goalx - breachsize) OR xcor > (goalx + breachsize)
     [
      set color green
     ]
     [
      if any? (turtles-on patch (pxcor - 1) pycor) with [color = green] OR any? (turtles-
on patch (pxcor + 1) pycor) with [color = green]
       [
        set color green
       ]
      ]
    ]
   ]
 ]
end

to spawn-turtles
 if (ticks > 1) AND (remainder ticks Deployrate) = 0 [
 set increments 0
 set sproutx 0
 while [increments < (robotpop / 2)]
 [
  set sproutx (sproutx - spacing)
  ask patch sproutx 48
  [
   Sprout 1
  ]
  set increments (increments + 1)
 ]
```

```
 set sproutx 0
 while [increments < robotpop]
 [
  set sproutx (sproutx + spacing)
  ask patch sproutx 48
  [
   Sprout 1
  ]
  set increments (increments + 1)
 ]

  ask turtles with [pycor = 48]
  [
   set active 1
   set goalpos "centre"
   set shape "square"
   set size 1
   set color red
   set agent_ahead "null"
  ]
 ]
end

to advance-line
 if any? (turtles-on patch breachx (breachy - breachsize - 1)) AND any? (turtles-on
patch goalx goaly) with [color = green]
 [
  ask turtles
  [
   set color violet
   set active 3
  ]
  set seal 1
 ]
 if any? (turtles-on patch goalx goaly) with [color = green]
 [
  ask turtles with [ycor < (goaly + 0.5) AND goalpos != "lost"]
  [
   set heading 180
   forward speed
   set active 3
  ]
 ]
end
```

## ESHR SA Experiment 2

```
;; EMERGENCY SHIP HULL REPAIR
;; SELF ASSEMBLING AGENTS APPROACH VERSION 3.1
;; BY MATTHEW HAIRE
```

```
;; LAST EDITED:    03 May 2019

globals
[
  seal  ;; check if breach is sealed or not
  speed ;; movement speed of turtles
  goalx ;; goal x coordinate
  goaly ;; goal y coordinate
  sproutx ;; x coordinates of turtle for creation
  increments ;; while loop variable for robots
  spacing ;; spacing for robots
  sub_breach
  sub_agent
  total_ingress
  Q
  Area
  turtles_attached
  turtles_unattached
]

turtles-own
[
  active  ;; state of the turtle - either in transit or in position
  goalpos ;; goal position of turtle - either centre, left or right or breach
  agent_ahead ;; reporter for color of agent ahead of turtle
]

to setup
  clear-all
  reset-ticks
  set seal 0
  set speed 1
  set turtles_attached 0
  set turtles_unattached 0
  setup-environment
  setup-turtles
end

to setup-environment
  resize-world -50 50 -50 50
  set-patch-size 5
  set goalx breachx
  set goaly (breachy + breachsize + 1)
  set sub_agent 0
  set total_ingress 0
  set Q 0
  set Area (pi * breachsize ^ 2 * 0.00694)
  ask patches
  [
```

```
  set pcolor 5
  if pycor <= (min-pycor + (max-pycor * 1.9))
  [
    set pcolor 105
  ]
  ask patch breachx breachy
  [
    set pcolor 101
    ask patches in-radius breachsize
    [
      set pcolor 101
    ]
  ]
 ]
end

to setup-turtles

 set increments 0
 set sproutx 0
 set spacing (96 / robotpop)

 while [increments < (robotpop / 2)]
 [
  set sproutx (sproutx - spacing)
  ask patch sproutx 48
  [
    Sprout 1
  ]
  set increments (increments + 1)
 ]

 set sproutx 0
 while [increments < robotpop]
 [
  set sproutx (sproutx + spacing)
  ask patch sproutx 48
  [
    Sprout 1
  ]
  set increments (increments + 1)
 ]

 ask turtles
  [
    set active 1
    set goalpos "centre"
    set shape "square"
    set size 1
```

```
    set color red
    set agent_ahead "null"
   ]
end

to start
 if (ticks > 1) AND (remainder ticks mov_spd) = 0
 [
  water-ingress-calc
 ]
 if seal = 1 OR total_ingress >= 14870
 [
  set turtles_attached count turtles with [xcor >= (breachx - breachsize - 1) AND xcor
<= (breachx + breachsize + 1) AND ycor >= (breachy - breachsize - 1) AND ycor <=
(breachy + breachsize + 1)]

  if breachsize = 6 [set turtles_unattached count turtles - turtles_attached]
  if breachsize = 5 [set turtles_unattached count turtles - turtles_attached]
  if breachsize = 4 [set turtles_unattached count turtles - turtles_attached]
  if breachsize = 3 [set turtles_unattached count turtles - turtles_attached]
  if breachsize = 2 [set turtles_unattached count turtles - turtles_attached]
  if breachsize = 1 [set turtles_unattached count turtles - turtles_attached]

  show count turtles
  show turtles_unattached
  show total_ingress
  show seal
  stop
 ]
 water-ingress-calc
 turtle-actions
 spawn-turtles
 advance-line
 tick
end

to turtle-actions
 ask turtles
 [
  if active = 1
  [
   if goalpos = "left"
   [
    ifelse pycor > (goaly + 1)
    [
     setxy pxcor (goaly + 1)
    ]
    [
     ifelse not any? (turtles-on patch-left-and-ahead 90 1)
```

```
    [
      set agent_ahead "null"
      setxy pxcor goaly
      set color orange
      set active 0
      if pxcor > goalx
      [
        set goalpos "right"
      ]
    ]
    [
      set heading 270
      forward speed
    ]
    if pxcor < (goalx - breachsize - 1)
    [
      set goalpos "lost"
      set active 1
    ]
  ]
]
if goalpos = "right"
[
  ifelse pycor > (goaly + 1)
  [
    setxy pxcor (goaly + 1)
  ]
  [
    ifelse not any? (turtles-on patch-right-and-ahead 90 1)
    [
      set agent_ahead "null"
      setxy pxcor goaly
      set color orange
      set active 0
      if pxcor < goalx
      [
        set goalpos "left"
      ]
    ]
    [
      set heading 90
      forward speed
    ]
    if pxcor > (goalx + breachsize + 1)
    [
      set goalpos "lost"
      set active 1
    ]
  ]
```

```
    ]

    if goalpos = "lost"
    [
       facexy pxcor (goaly + 6)
       ifelse not any? (turtles-on patch-ahead 1.5)
       [
       set agent_ahead "null"
       ifelse pycor >= (goaly + 6)
        [
          set goalpos "centre"
          facexy goalx goaly
          set color red
        ]
        [
          forward speed
        ]
       ]
       [
        if any? (turtles-on patch-ahead 1.5) with [color = red]
        [
          set agent_ahead "red"
          back speed ;; previously 1
        ]
        if any? (turtles-on patch-ahead 1.5) with [color = orange]
        [
          set agent_ahead "orange"
          back speed ;; previously 1
        ]
        if any? (turtles-on patch-ahead 1.5) with [color = green]
        [
          set agent_ahead "green"
          back speed ;; previously 1
        ]
       ]
    ]

    if goalpos = "centre"
    [
      facexy goalx goaly
      ifelse not any? (turtles-on patch-ahead 1.5)
      [
        set agent_ahead "null"
        ifelse (pxcor < (goalx + 0.5) AND pxcor > (goalx - 0.5) AND pycor > (goaly - 0.5)
AND pycor < (goaly + 0.5))
        [
          setxy goalx goaly
          set heading 180
          set color orange
```

```
      set active 0
     ]
     [
      forward speed ;; previously 1
     ]
    ]
    [
     if any? (turtles-on patch-ahead 1.5) with [color = red]
     [
      set agent_ahead "red"
      back speed
     ]
     if any? (turtles-on patch-ahead 1.5) with [color = orange]
     [
      set agent_ahead "orange"
      setxy pxcor pycor
      if not any? (turtles-on patch (goalx - 1) goaly) OR any? (turtles-on patch (goalx -
1) goaly) with [color = orange]
      [
       set heading 270
       ifelse any? (turtles-on patch-ahead 1) OR any? (turtles-on patch-ahead 2)
       [
        set heading 90
        set goalpos "right"
       ]
       [
        set heading 270
        set goalpos "left"
       ]
      ]
      if (not any? (turtles-on patch (goalx + 1) goaly) OR any? (turtles-on patch (goalx
+ 1) goaly) with [color = orange]) AND (goalpos != "left")
      [
       set heading 90
       set goalpos "right"
      ]
     ]
     if any? (turtles-on patch-ahead 1.5) with [color = green]
     [
      set agent_ahead "green"
      ifelse not any? (turtles-on patch goalx goaly)
      [
       setxy goalx goaly
       set heading 180
       set color orange
       set active 0
      ]
      [
       back (speed / 2)
```

```
          ]
        ]
      ]
    ]
  ]
  if active = 0
  [
    ifelse goalpos = "centre"
    [
      if any? ((turtles-on patch (goalx - 1) goaly) with [color = green]) AND any? ((turtles-
on patch (goalx + 1) goaly) with [color = green])
      [
        set color green
      ]
    ]
    [
      ifelse xcor < (goalx - breachsize) OR xcor > (goalx + breachsize)
      [
        set color green
      ]
      [
        if any? (turtles-on patch (pxcor - 1) pycor) with [color = green] OR any? (turtles-
on patch (pxcor + 1) pycor) with [color = green]
        [
          set color green
        ]
      ]
    ]
  ]
 ]
end

to spawn-turtles
 if (ticks > 1) AND (remainder ticks Deployrate) = 0 [
 set increments 0
 set sproutx 0
 while [increments < (robotpop / 2)]
 [
  set sproutx (sproutx - spacing)
  ask patch sproutx 48
  [
   Sprout 1
  ]
  set increments (increments + 1)
 ]

 set sproutx 0
 while [increments < robotpop]
 [
```

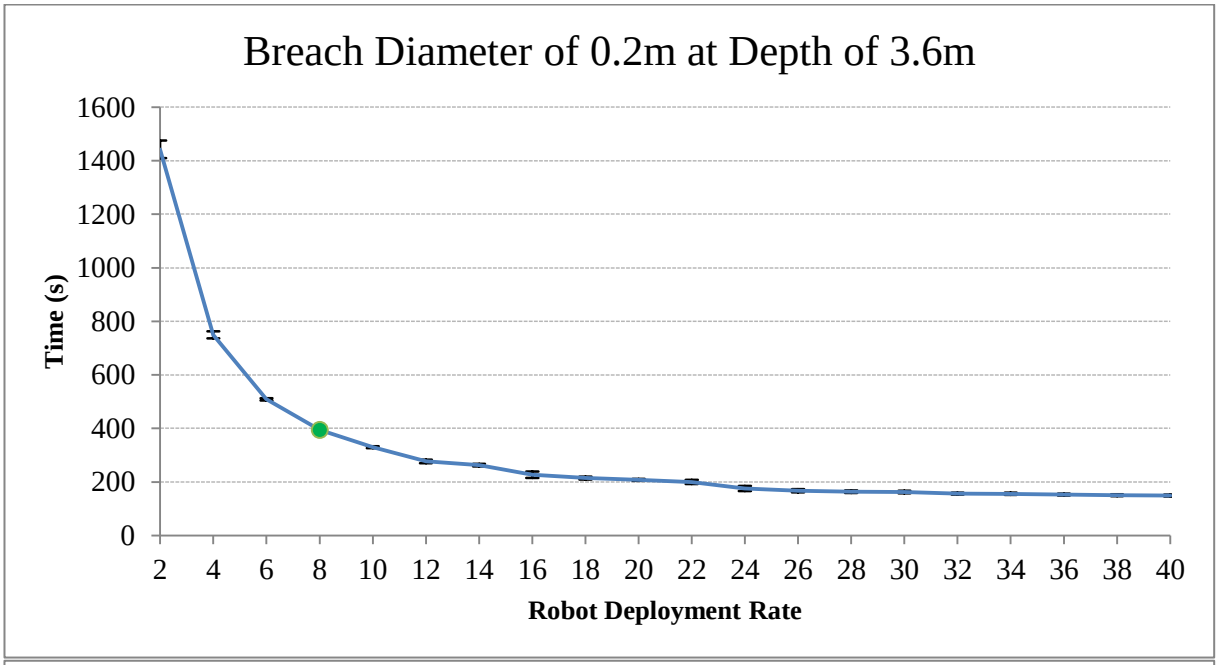
```
   set sproutx (sproutx + spacing)
   ask patch sproutx 48
   [
    Sprout 1
   ]
   set increments (increments + 1)
  ]

   ask turtles with [pycor = 48]
   [
    set active 1
    set goalpos "centre"
    set shape "square"
    set size 1
    set color red
    set agent_ahead "null"
   ]
  ]
end

to advance-line
  if any? (turtles-on patch breachx (breachy - breachsize - 1)) AND any? (turtles-on
patch goalx goaly) with [color = green]
  [
   ask turtles
   [
    set color violet
    set active 3
   ]
   set seal 1
  ]
  if any? (turtles-on patch goalx goaly) with [color = green]
  [
   ask turtles with [ycor < (goaly + 0.5)]
   [
    set heading 180
    forward speed
    set active 3
   ]
  ]
end

to water-ingress-calc
  if breachy = 33 [set Q (Area * sqrt (64.348 * 1))]
  if breachy = 21 [set Q (Area * sqrt (64.348 * 2))]
  if breachy = 9 [set Q (Area * sqrt (64.348 * 3))]
  if breachy = -3 [set Q (Area * sqrt (64.348 * 4))]
  if breachy = -15 [set Q (Area * sqrt (64.348 * 5))]
  if breachy = -27 [set Q (Area * sqrt (64.348 * 6))]
```

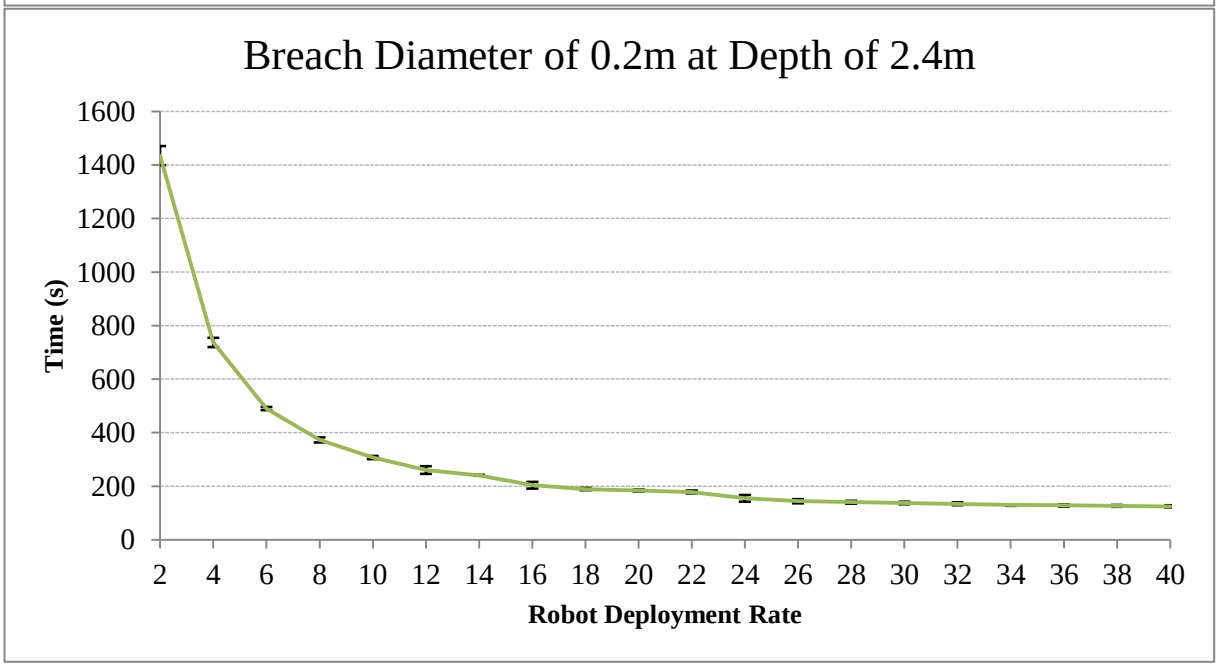
```
  if breachy = -39 [set Q (Area * sqrt (64.348 * 7))]

  set sub_breach (count patches with [pcolor = 101])
  set sub_agent (count turtles-on patches with [pcolor = 101])
  set Q (Q * (sub_breach - sub_agent) / sub_breach)
  set Q (Q / 60 * 6.23)
  set total_ingress (total_ingress + Q)
end
```

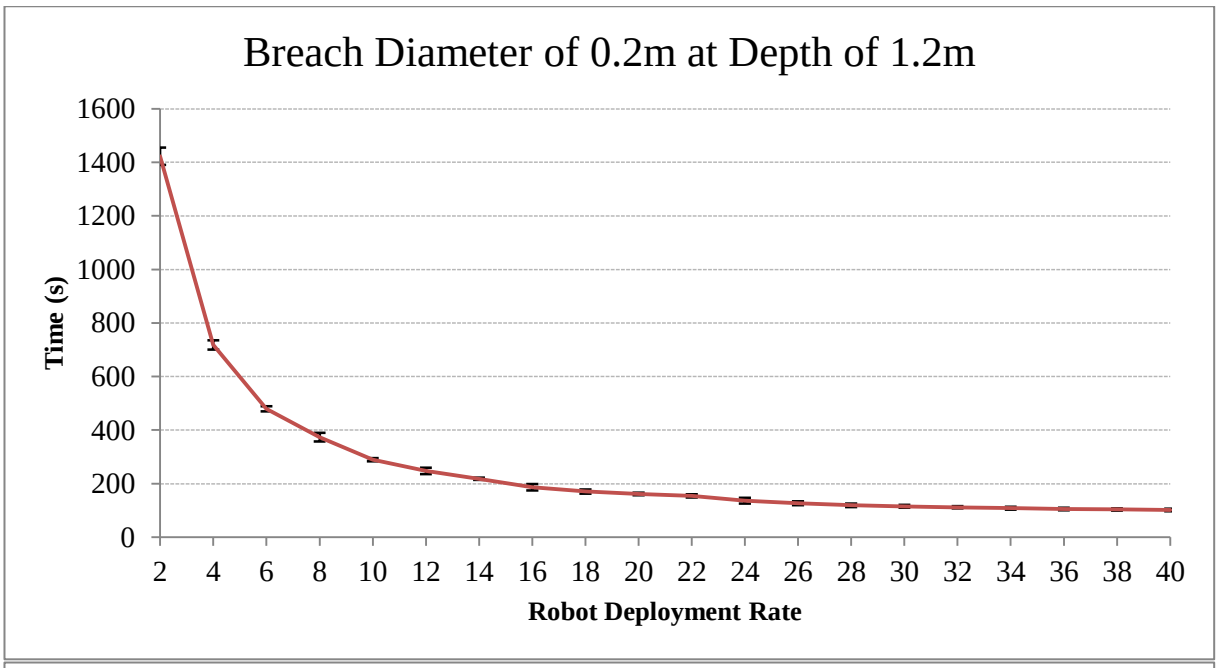
# Appendix C: Additional Results from Ship Hull Repair Experiments

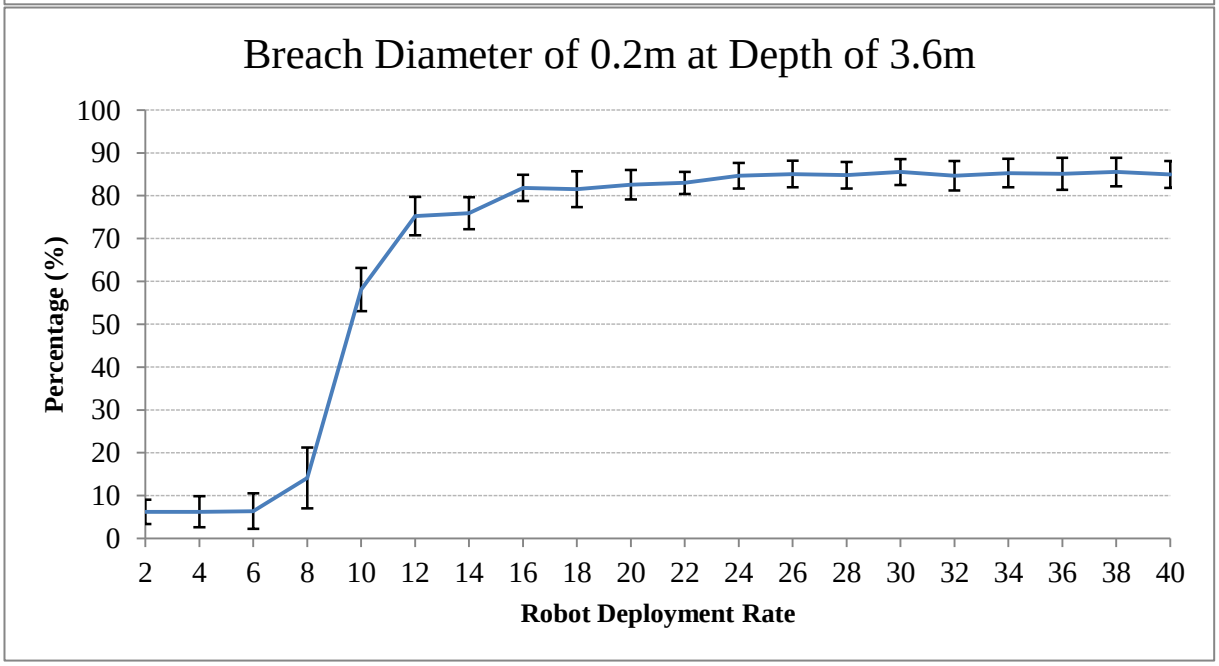
Appendix C features the graphed results from the emergency ship hull repair experiments of Chapter 4 which examined self-assembly of a repair patch under varying conditions. The graphs here primarily relate to the experiments which examined breach diameters of 0.2m and 0.6m when no obstacles were present and the experiments where the breach diameter remained constant but an obstacle was included whose diameter and position would vary.

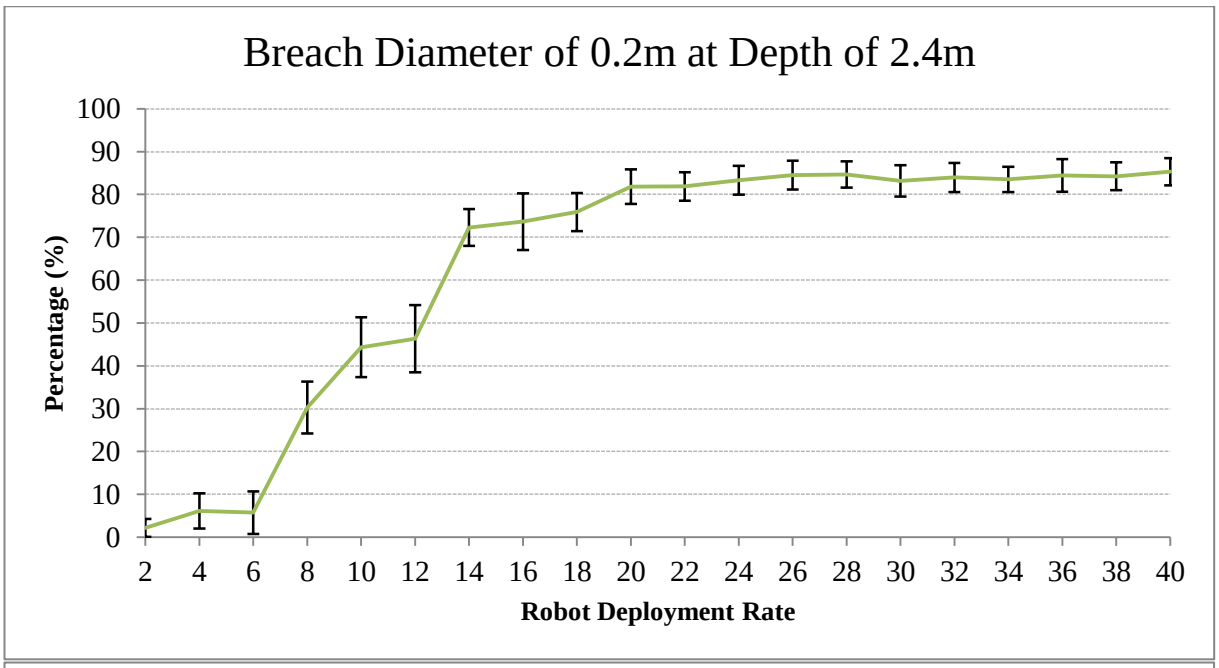
## Breach Diameter 0.2m (No Obstacles)

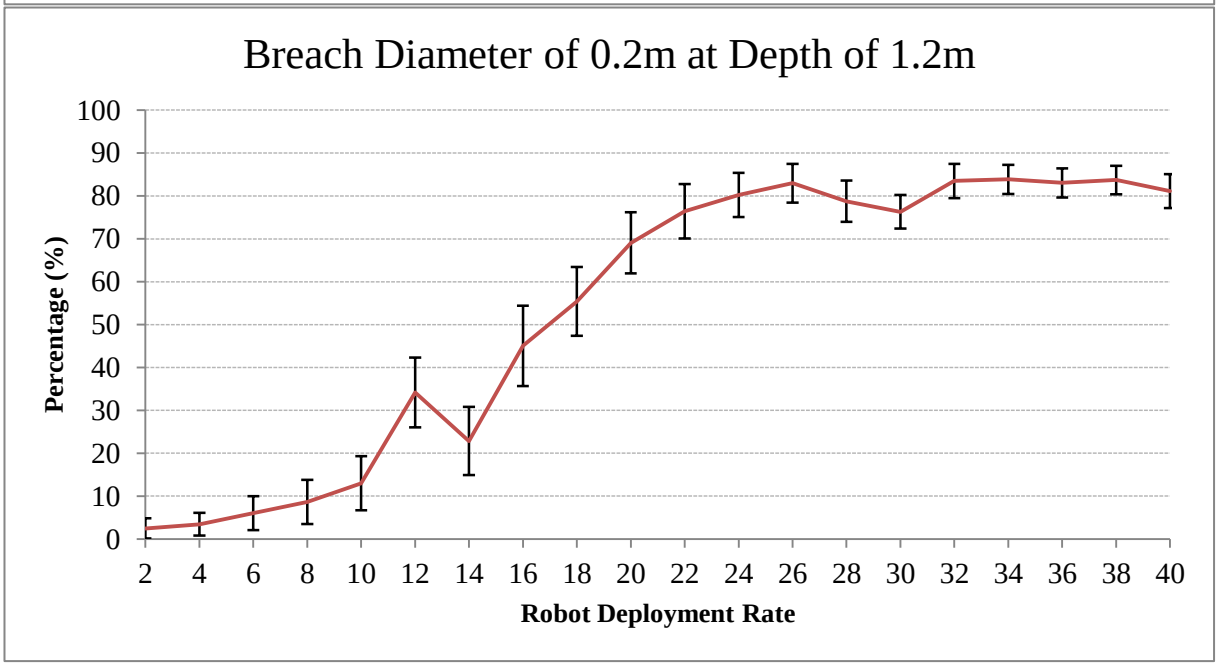


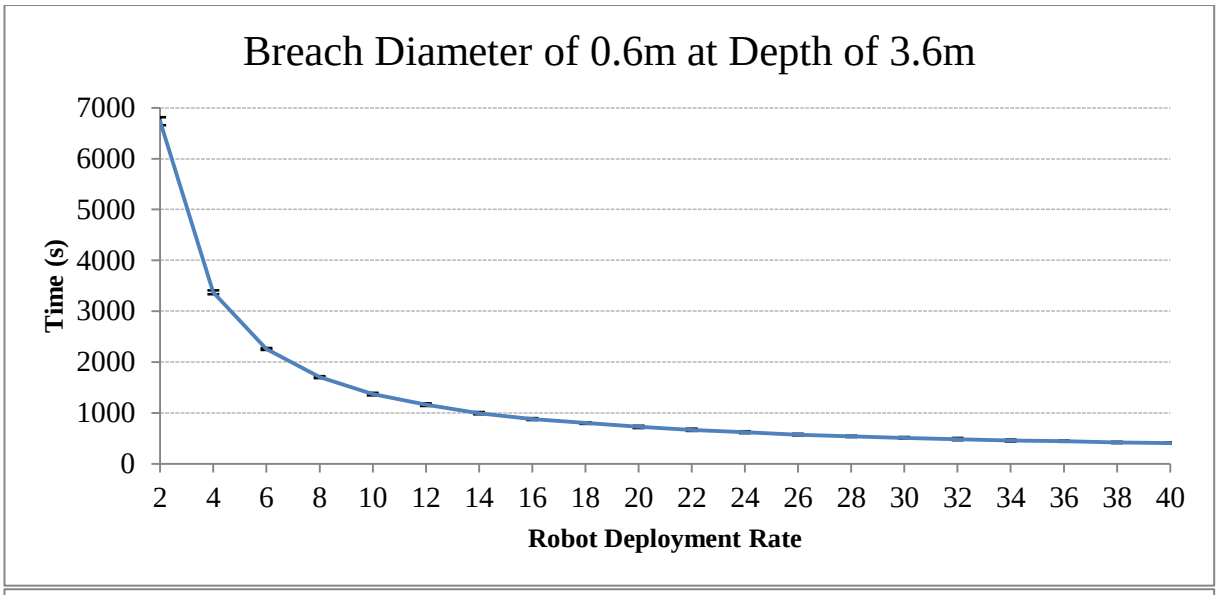

Breach Diameter of 0.2m at Depth of 1.2m
Time (s)
1600
1400
1200
1000
800
600
400
200
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate

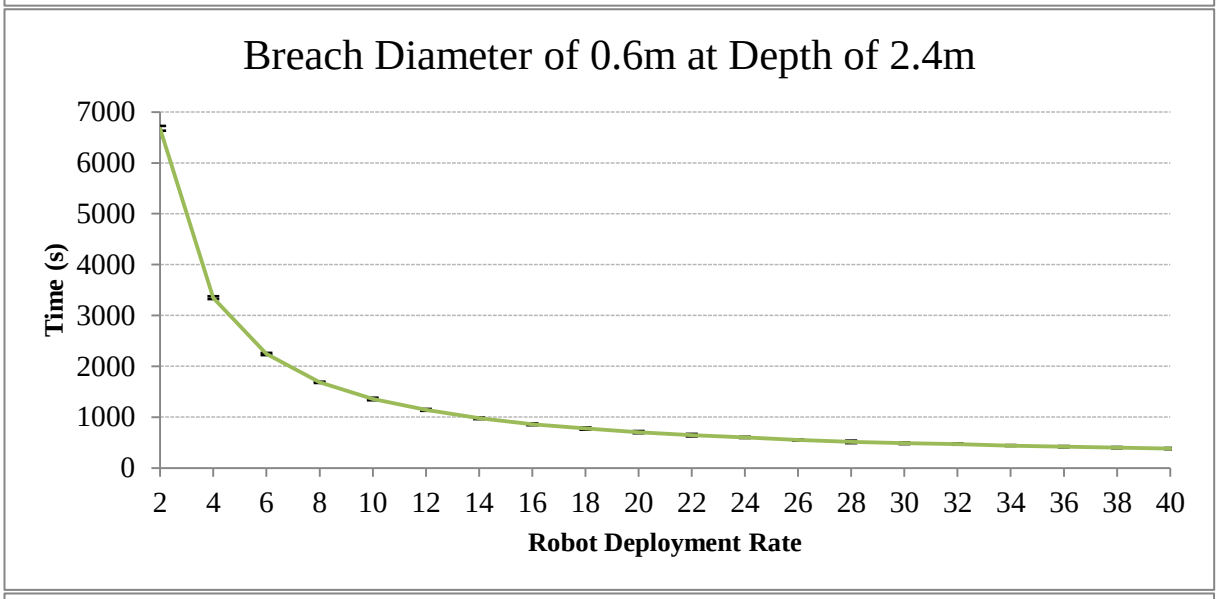

Breach Diameter of 0.2m at Depth of 3.6m
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate

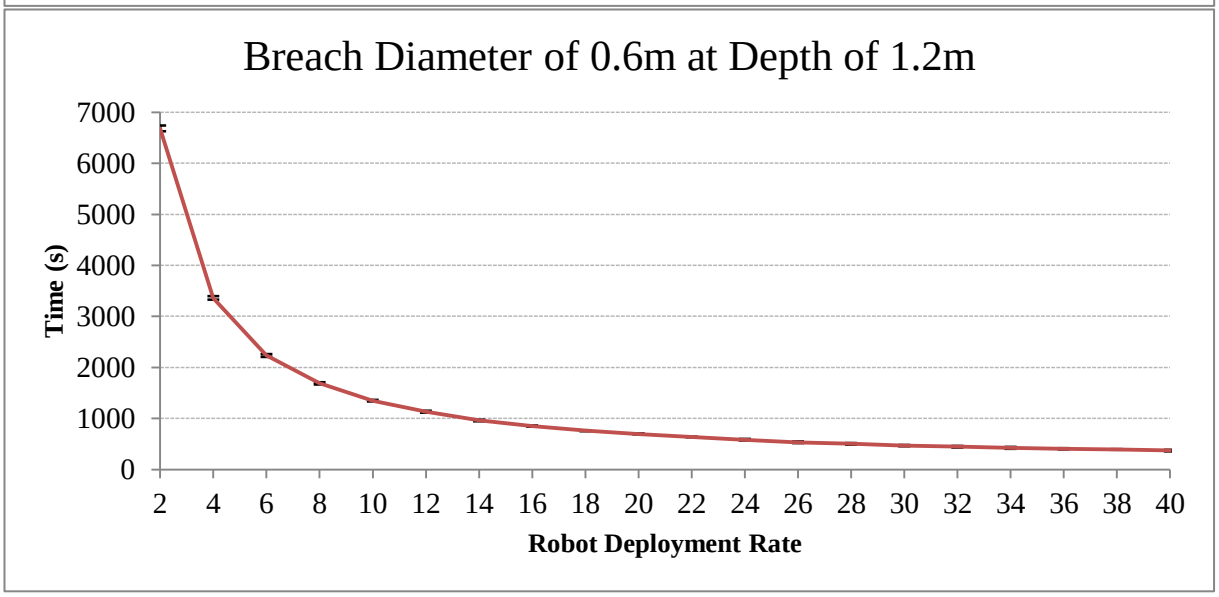

Breach Diameter of 0.2m at Depth of 2.4m
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate

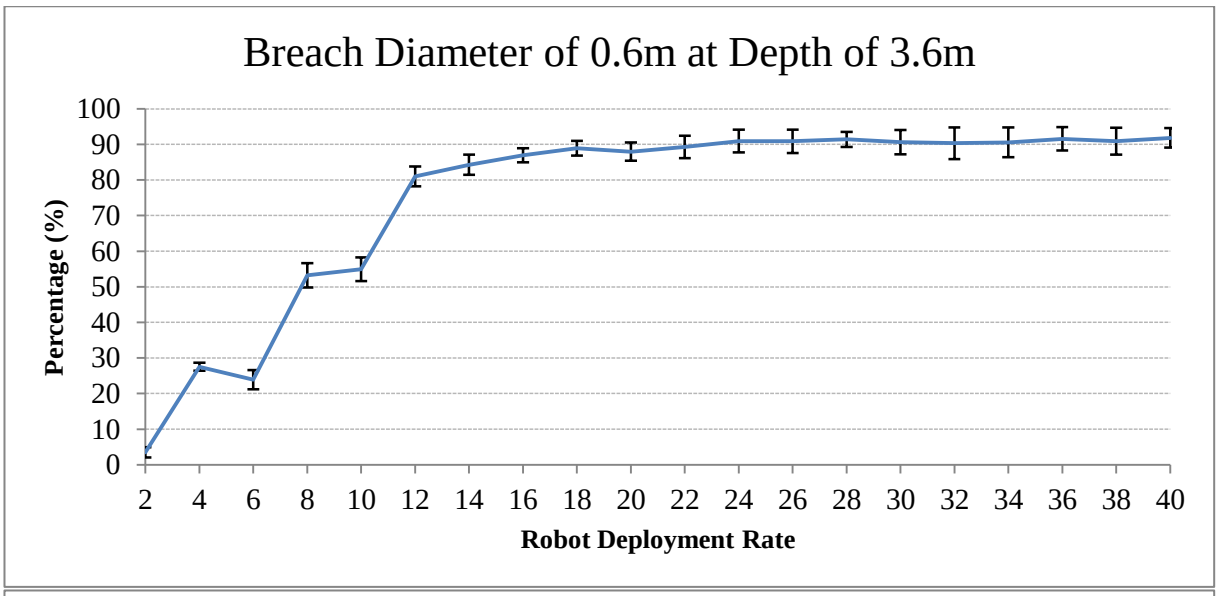

Breach Diameter of 0.2m at Depth of 1.2m
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate

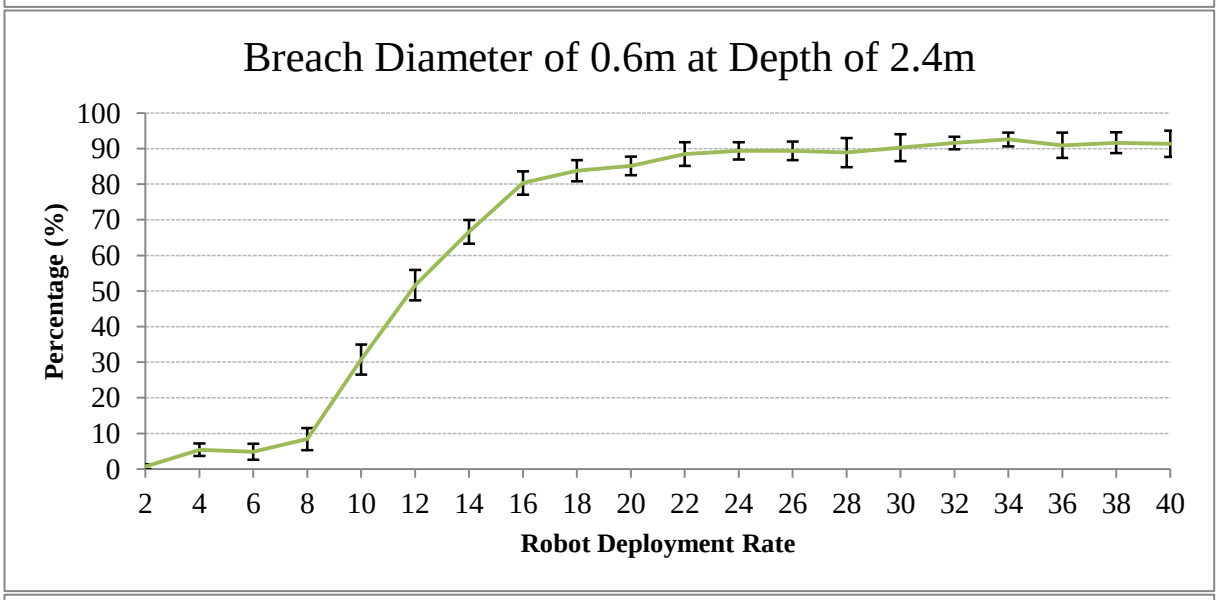
**Breach Diameter 0.6m (No Obstacles)**

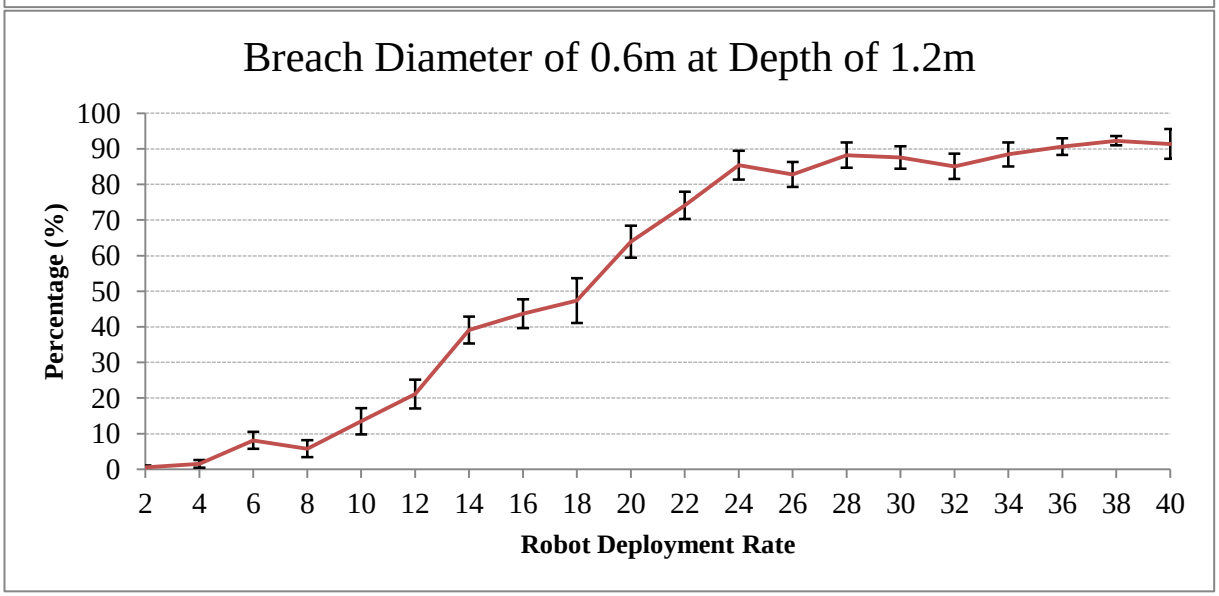


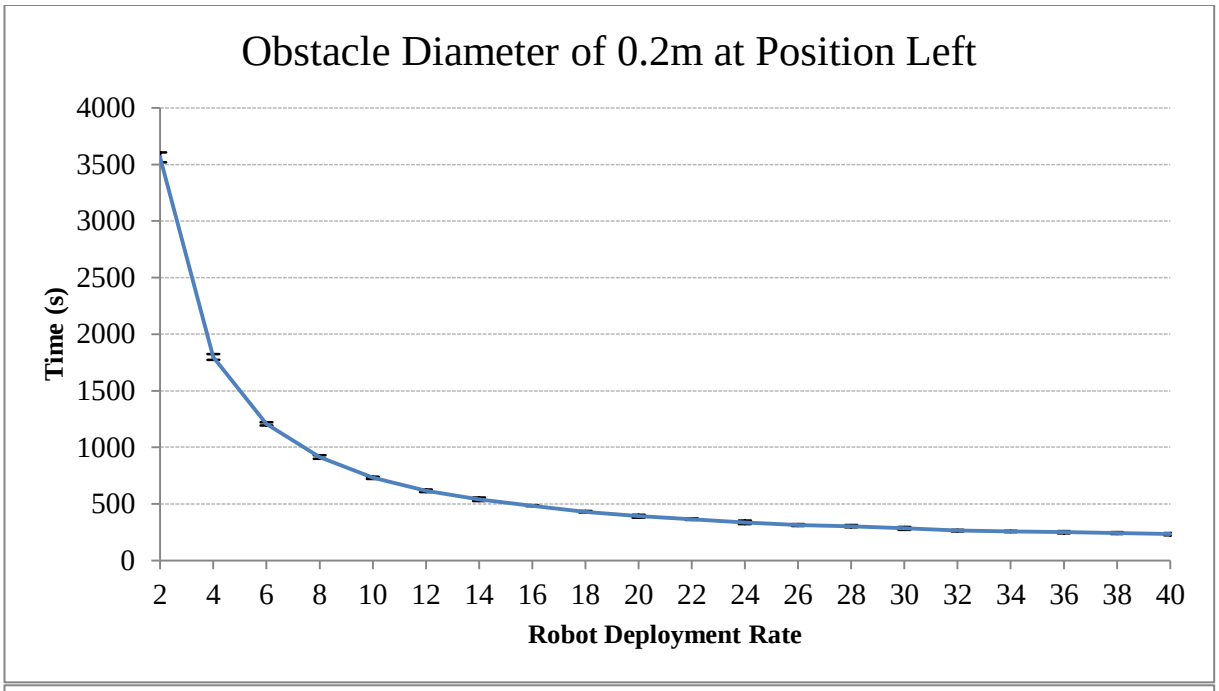



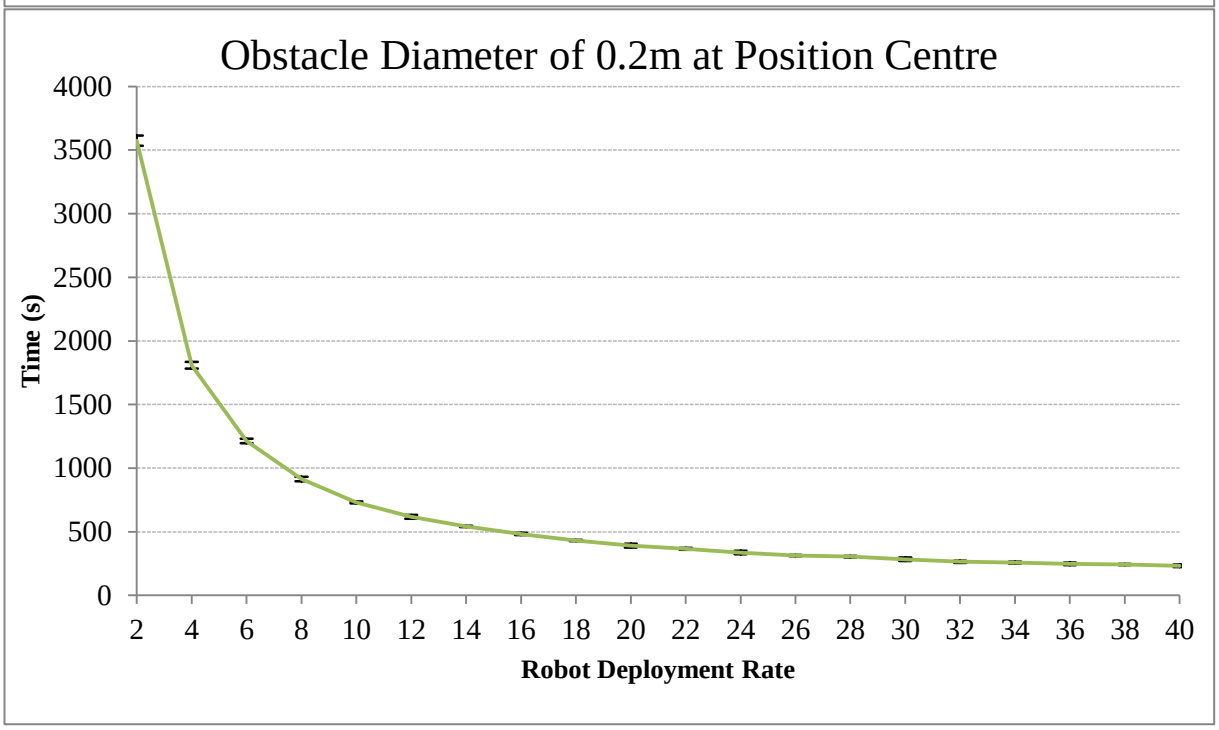

Breach Diameter of 0.6m at Depth of 3.6m
Percentage (%)
Robot Deployment Rate

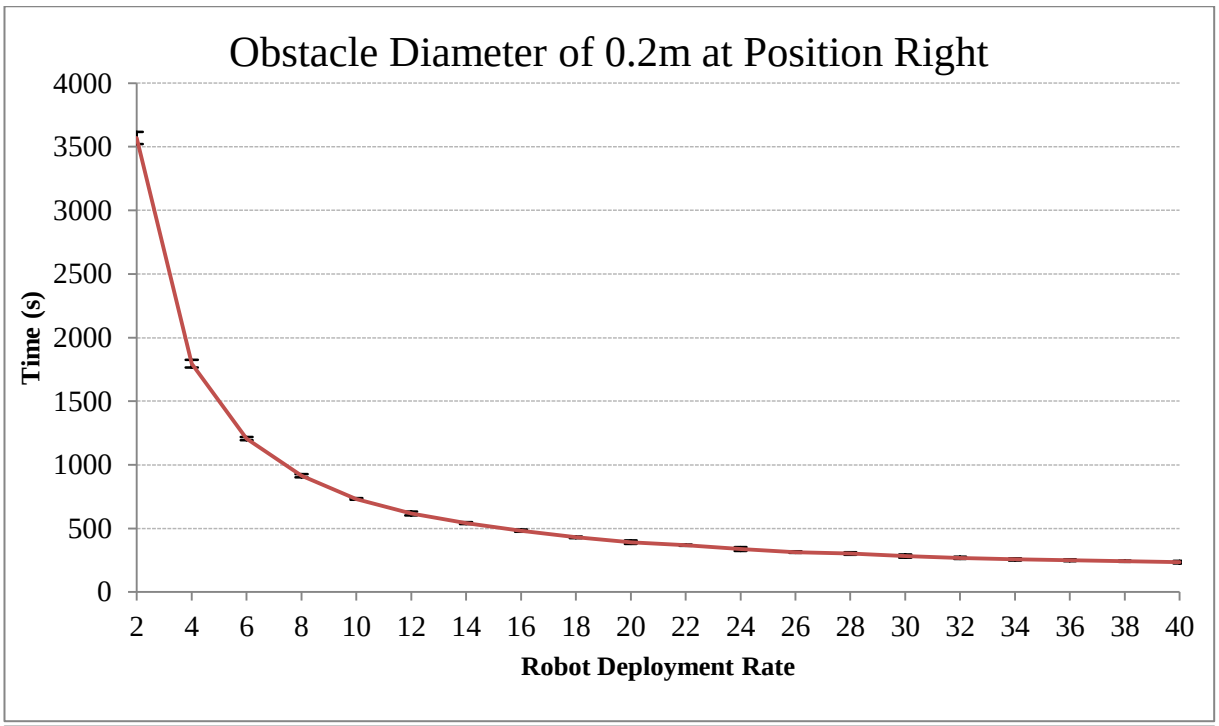

Breach Diameter of 0.6m at Depth of 2.4m
Percentage (%)
Robot Deployment Rate

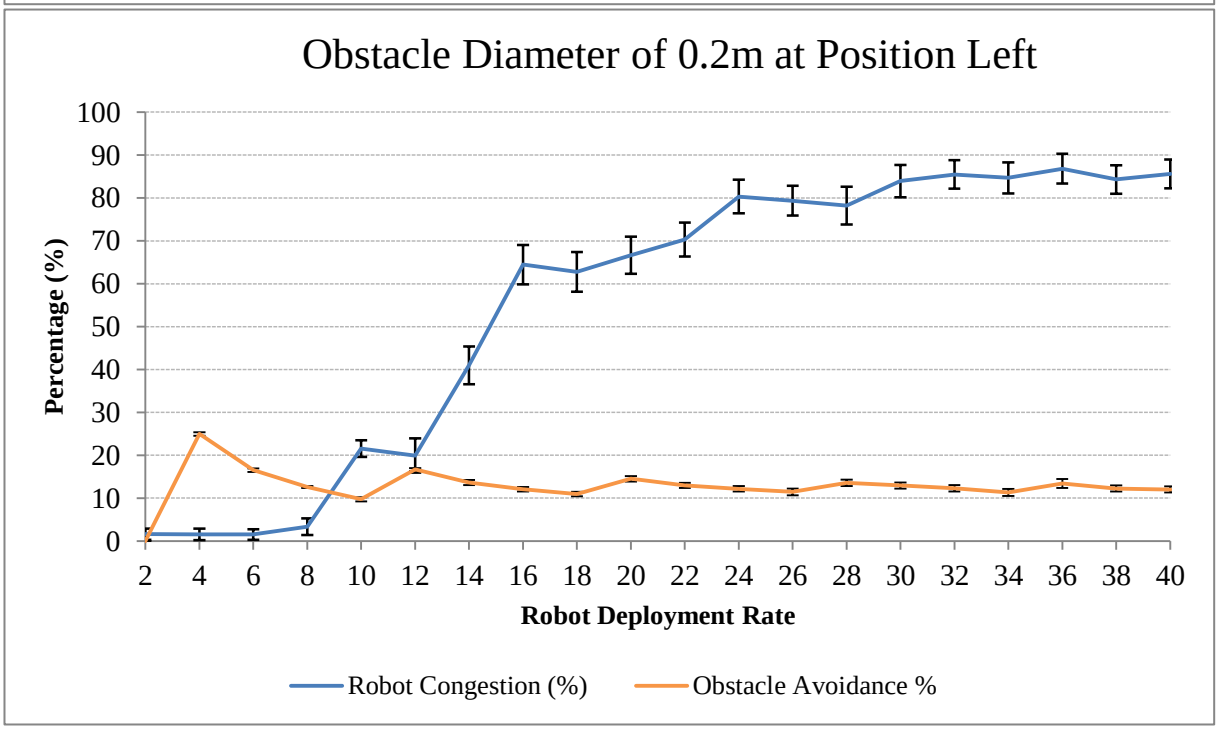

Breach Diameter of 0.6m at Depth of 1.2m
Percentage (%)
Robot Deployment Rate

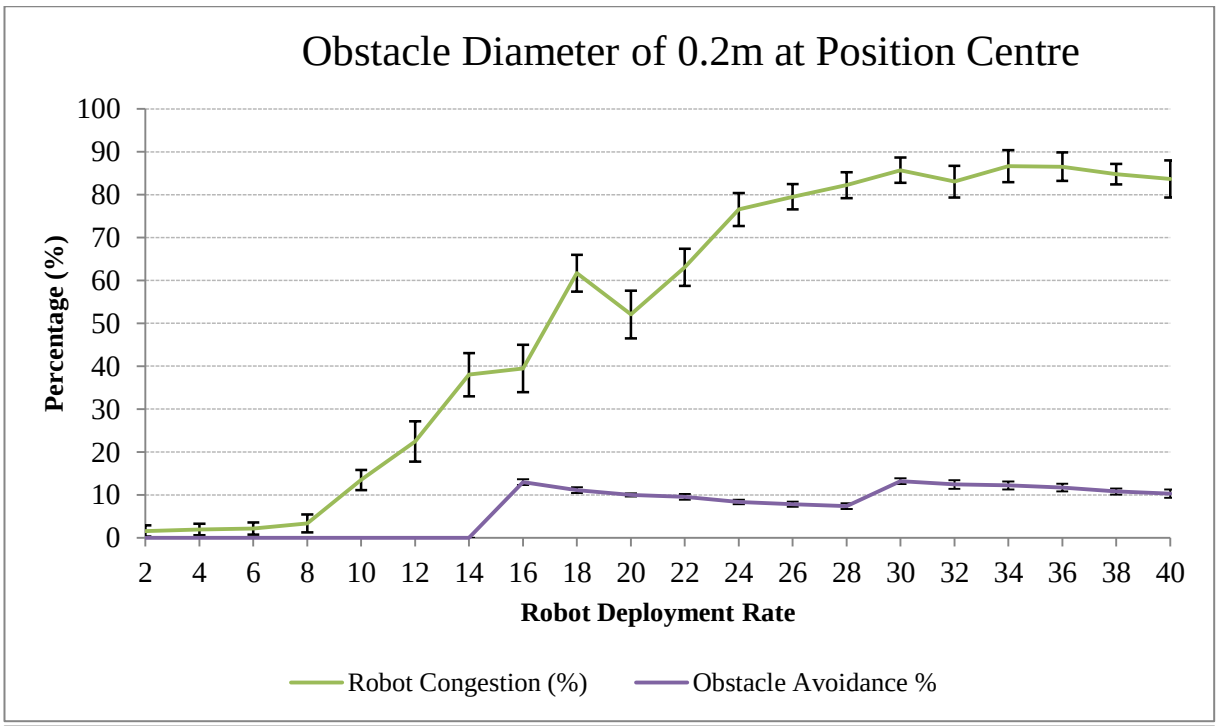
**Obstacle Diameter 0.2m (Breach Diameter 0.4m)**

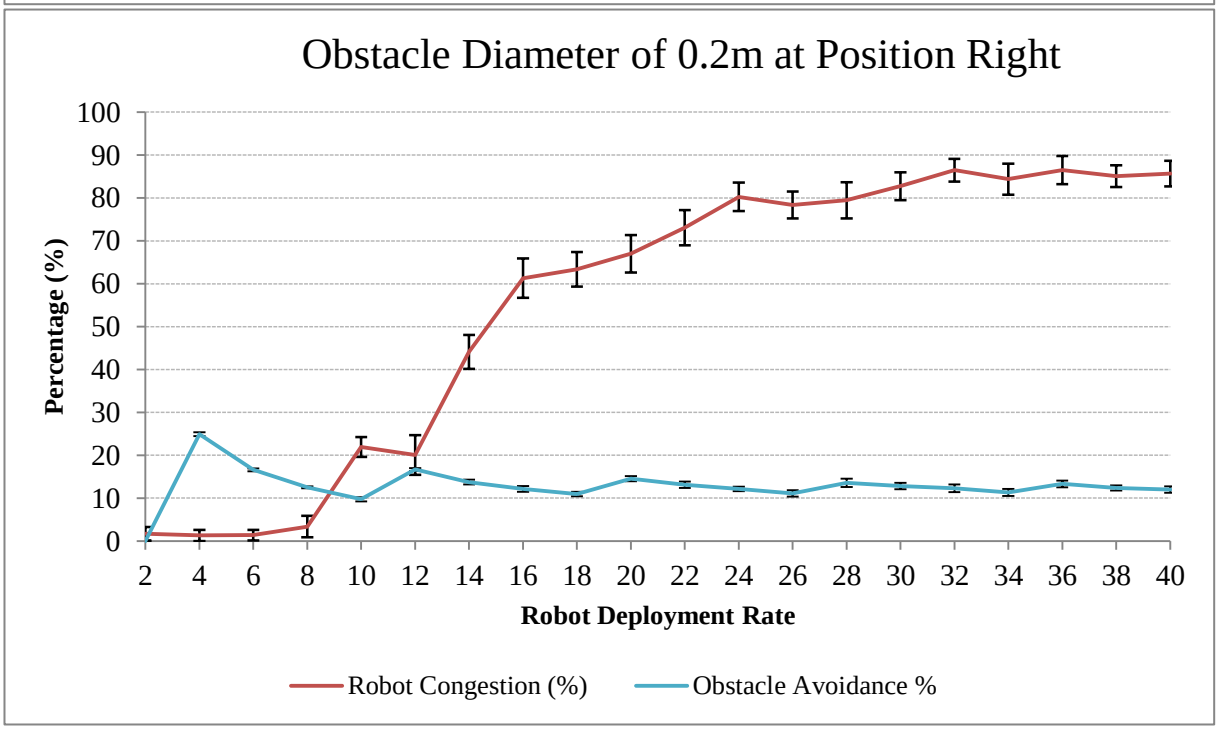


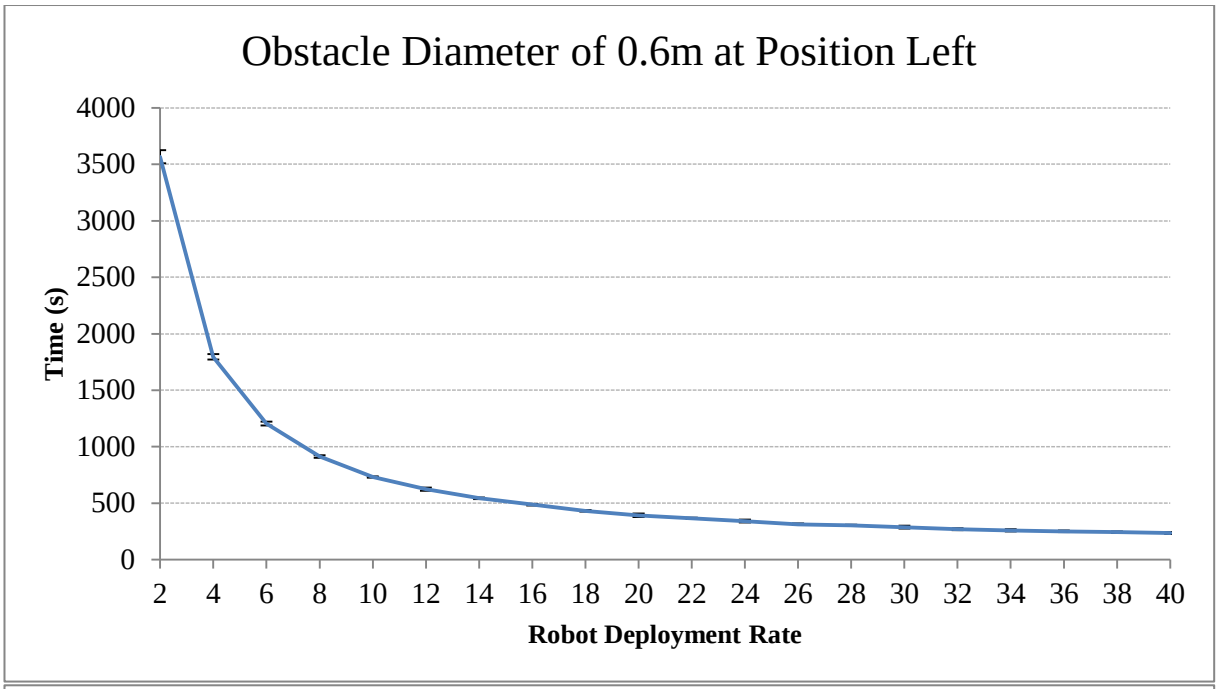

Obstacle Diameter of 0.2m at Position Right
Time (s)
4000
3500
3000
2500
2000
1500
1000
500
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate

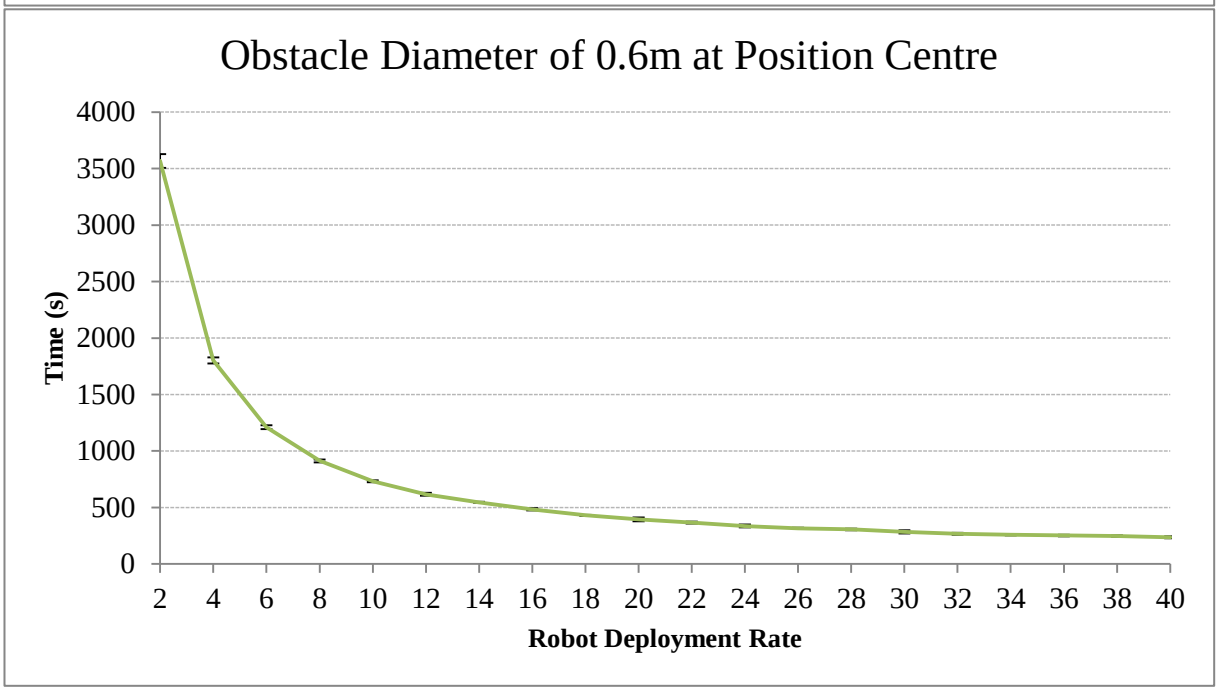

Obstacle Diameter of 0.2m at Position Left
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate
Robot Congestion (%)
Obstacle Avoidance %

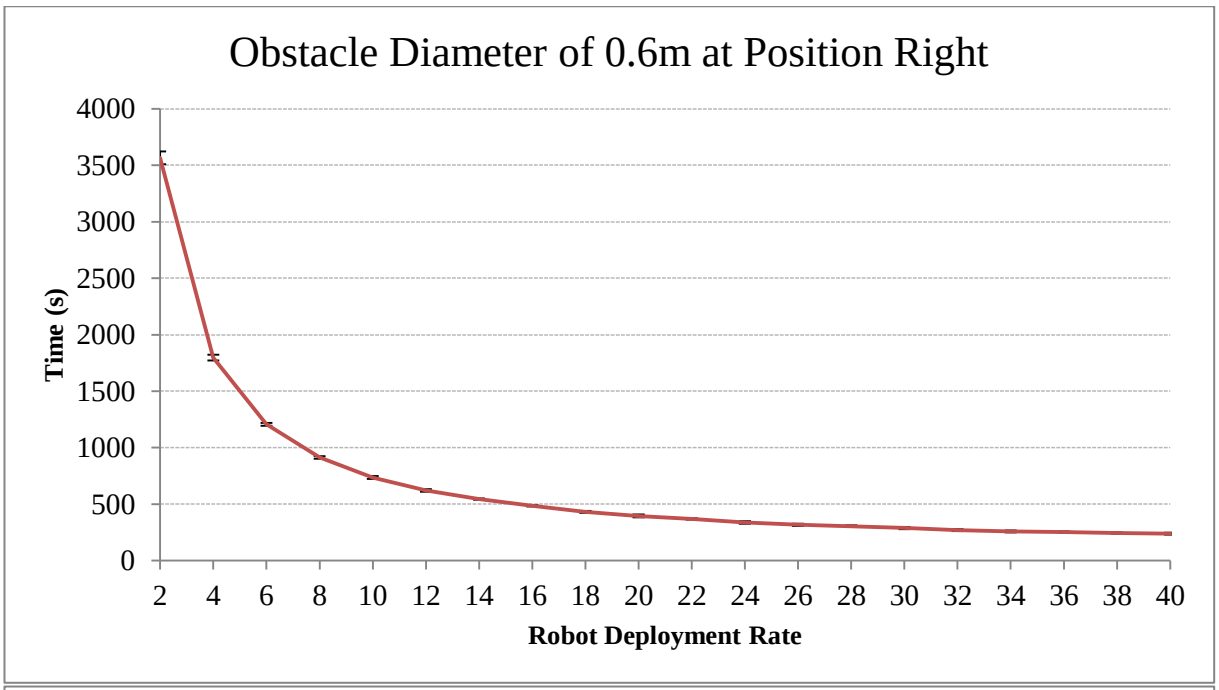

Obstacle Diameter of 0.2m at Position Centre
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate
Robot Congestion (%)
Obstacle Avoidance %

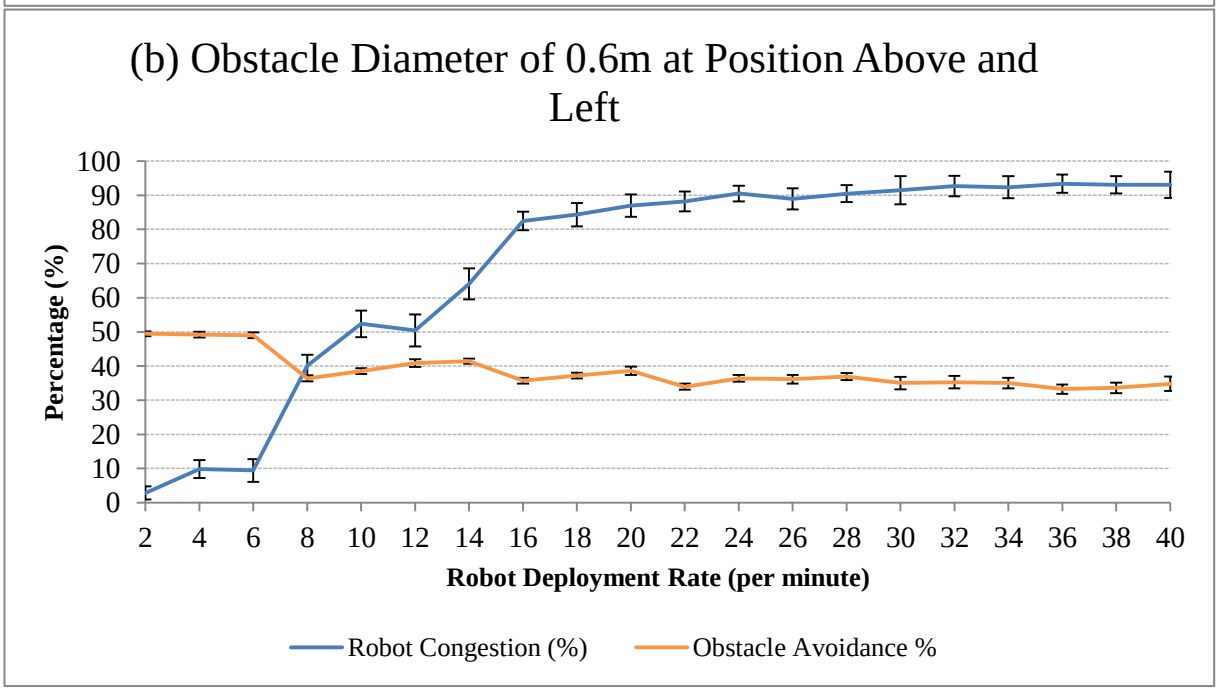

Obstacle Diameter of 0.2m at Position Right
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate
Robot Congestion (%)
Obstacle Avoidance %

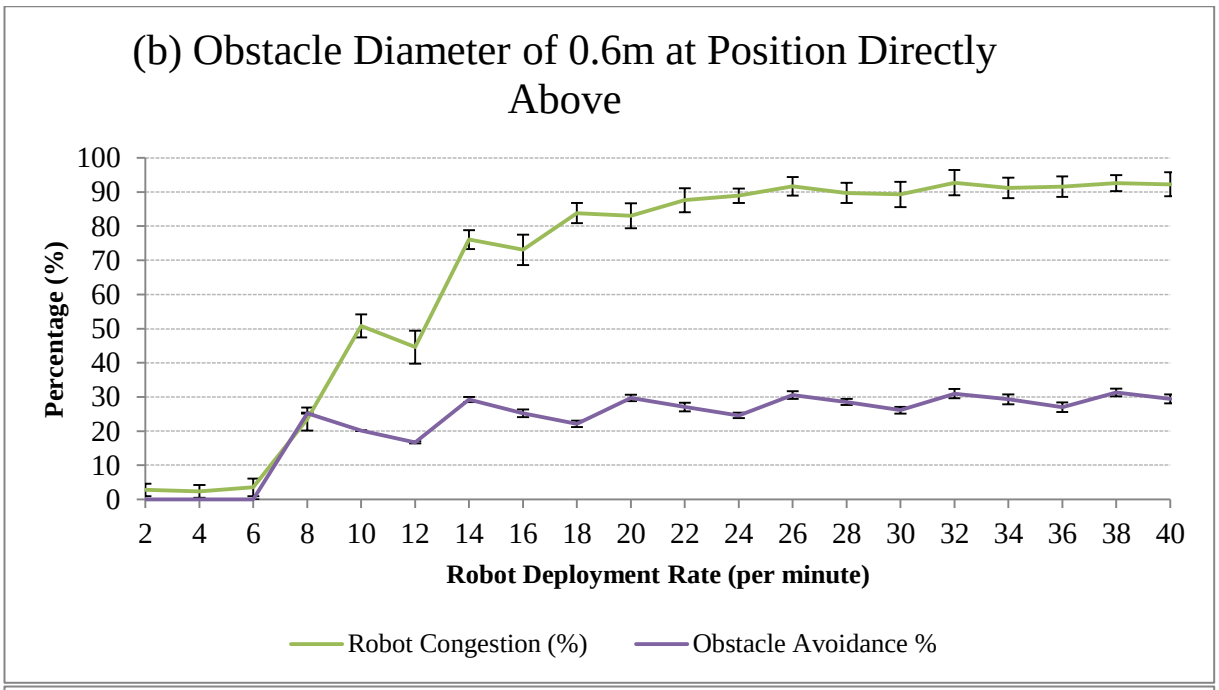
**Obstacle Diameter 0.6m (Breach Diameter 0.4m)**

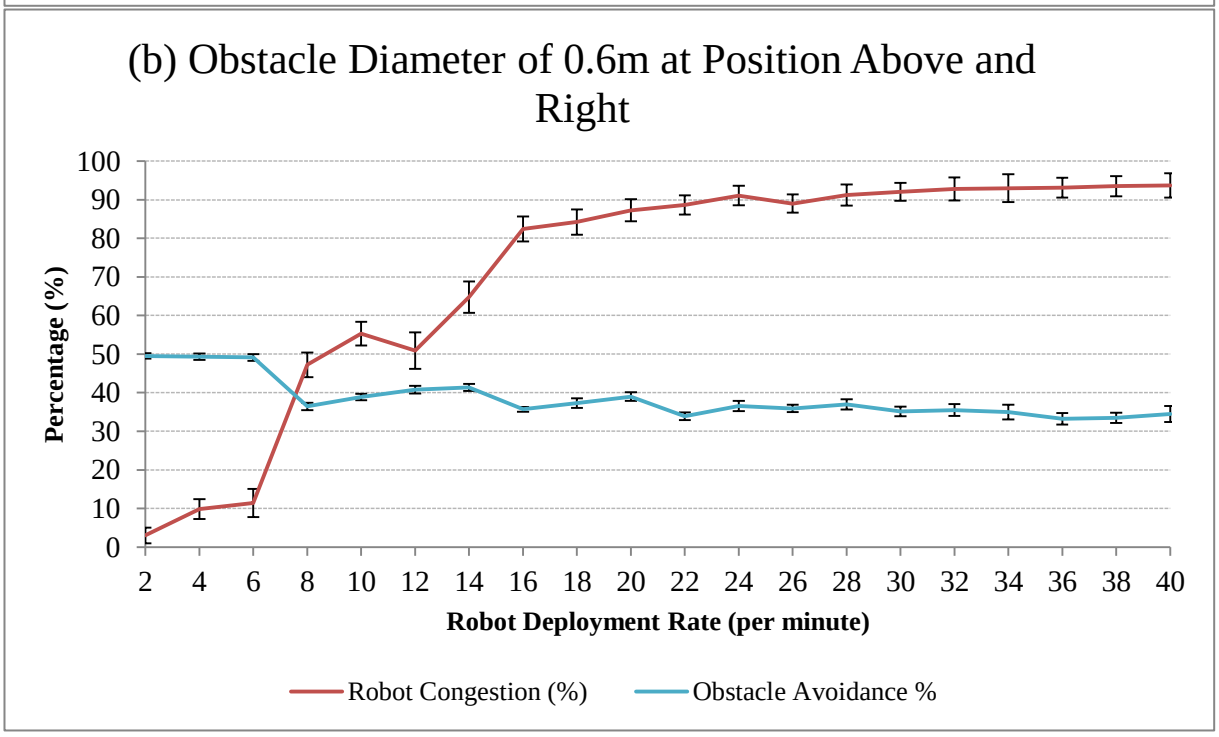



Obstacle Diameter of 0.6m at Position Right
Time (s)
4000
3500
3000
2500
2000
1500
1000
500
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate


(b) Obstacle Diameter of 0.6m at Position Above and Left
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate (per minute)
Robot Congestion (%)
Obstacle Avoidance %


(b) Obstacle Diameter of 0.6m at Position Directly Above
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate (per minute)
Robot Congestion (%)
Obstacle Avoidance %


(b) Obstacle Diameter of 0.6m at Position Above and Right
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Robot Deployment Rate (per minute)
Robot Congestion (%)
Obstacle Avoidance %

# Appendix D: Publications and Research Outputs

As referenced within the main text, this thesis was born out of years of research into swarm robotics which resulted in findings significant enough to warrant two conference papers, a journal paper, symposium oral presentations and a symposium poster – which was fortunate enough to win the runner-up prize for best poster. These are listed below:

## Conference Papers

Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2019, September). Ship hull inspection using a swarm of autonomous underwater robots: a Search algorithm. In 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 114-115). IEEE.

Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2019, September). Ship hull repair using a swarm of autonomous underwater robots: A self-assembly algorithm. In 2019 European Conference on Mobile Robots (ECMR) (pp. 1-6). IEEE.

## Journal Papers

Talamali, M. S., Bose, T., Haire, M., Xu, X., Marshall, J. A., & Reina, A. (2020). Sophisticated collective foraging with minimalist agents: a swarm robotics test. Swarm Intelligence, 14(1), 25-56.

## Posters

Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2017). Bio-inspired artificial nest-site selection swarm simulation and potential applications. Poster. Sheffield Hallam University MERI Symposium 2017. (Runner-Up Best Poster Award)

## Oral Presentations

Haire, M., Xu, X., Alboul, L., Penders, J., & Zhang, H. (2018). Emergency Ship Hull Repair Using a Swarm of Autonomous Underwater Robots. Poster. Sheffield Hallam University MERI Symposium 2018.