Distributed Autonomy and Formation Control of a Drifting Swarm of Autonomous Underwater Vehicles

by

Nicholas Rahardiyan Rypkema

BE(Hons) Electrical Engineering, The University of Queensland (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

and the

WOODS HOLE OCEANOGRAPHIC INSTITUTION

September 2015

Author .....
Department of Electrical Engineering and Computer Science
August 19, 2015

Certified by .....
Henrik Schmidt
Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Accepted by .....
Henrik Schmidt
Professor of Mechanical and Ocean Engineering
Chair, Joint Committee for Applied Ocean Science and Engineering

Distributed Autonomy and Formation Control of a Drifting Swarm of Autonomous Underwater Vehicles

by

Nicholas Rahardiyan Rypkema

BE(Hons) Electrical Engineering, The University of Queensland (2009)

Submitted to the Department of Electrical Engineering and Computer Science
on August 19, 2015, in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

Abstract

Recent advances in autonomous underwater vehicle (AUV) technology have led to their widespread acceptance and adoption for use in scientific, commercial, and defence applications in the underwater domain. At the same time, research progress in swarm robotics has seen swarm intelligence algorithms in use with greater effect on real-world robots in the field. A group of AUVs utilizing swarm intelligence concepts has the potential to address issues more effectively than a single AUV, and such a group can potentially open up new areas of application. Examples include the monitoring and tracking of highly dynamic oceanographic phenomena such as phytoplankton blooms and the use of an AUV swarm as a virtual acoustic receiver for sea-bottom seismic surveying or the monitoring of naturally occurring acoustic radiation from cracking ice. However, the limitations of the undersea environment places unique constraints on the use of existing swarm robotics approaches with AUVs. In particular, algorithms must be distributed and robust in the face of localization error and degraded communications.

This work presents an investigation into one particular swarm strategy for a group of AUVs, termed formation control, with consideration to the constraints of the underwater domain. Four formation control algorithms, each developed and tested within the MOOS-IvP framework, are presented. In addition, a 'formation quality' metric is introduced. This metric is used in conjunction with a measure of formation energy expenditure to compare the efficacy of each behaviour during construction of a desired formation, and formation maintenance while it drifts in ocean currents. This metric is also used to compare robustness of each algorithm in the presence of vehicle failure and changing communication rate.

Thesis Supervisor: Henrik Schmidt
Title: Professor of Mechanical and Ocean Engineering

Acknowledgements

First and foremost, I would like to thank Professor Henrik Schmidt, without whose guidance and support this work would not have been possible; I will forever be grateful for the opportunity to work at MIT.

Thank you as well to my old mentors at DSTO Sydney for introducing me to the world of marine robotics; this path would never have been opened to me without your help.

To my colleagues in the Laboratory for Autonomous Marine Sensing Systems, thank you for making this journey a little less rocky.

A special thanks to Dehann, whose assistance and friendship helped me through many demanding classes and problem sets.

A very special thanks to Jenn, whose support and encouragement made getting through this thesis so much easier; my time at MIT has been made immeasurably better by your friendship.

Above all else, thank you to my family - to my brother Henrik, and my parents Fred and Tatie, thank you for always being there to support me; I owe all that I have achieved to you.

This research was supported by APS under contract number N66001-11-C-4115 and award numbers N66001-13-C-4006 and N66001-14-C-4031.

Contents

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

Chapter 1

Introduction

The past decade has seen rapid progress in autonomous underwater vehicle (AUV) technology, with a growing interest and adoption of such vehicles for scientific, commercial, and defence applications in the underwater domain. These vehicles have the potential to fundamentally impact data gathering approaches in these fields. In addition, the emergence of swarm robotics research during this decade, where the concepts of swarm intelligence are applied to multi-robot systems, is allowing scientists and other users of robotics systems to leverage the use of multiple coordinated robots in achieving desired goals. As AUV technology continues to mature, the development of distributed cooperative and swarm autonomy for AUVs can potentially address issues more effectively than a single AUV typically can; for example, monitoring and determining the location of toxic ocean spills, tracking algae or phytoplankton blooms or other oceanographic chemical and biological phenomena, or the use of a swarm as a virtual sensor array for seismic surveying. A swarm of AUVs using ocean currents for propulsion has several attractive characteristics for undersea monitoring; such a swarm would be able to exploit ocean currents to monitor ocean phenomena for long periods of time while contributing very little ambient noise; the characteristics of the swarm would contribute to its endurance by being fault tolerant, as well as allowing it to flexibly survey areas of complex geometry containing obstacles; and the swarm would be able to monitor areas of large expanse, enabling effective investigation of the structure of complicated ocean phenomena. In addition, the characteristics of the underwater environment necessarily force unique constraints on the swarm robotics problem (namely in terms of communication and localization), and the development of autonomous behaviours that can successfully overcome these constraints has the potential to contribute greatly to the field of swarm robotics.

Recognizing the potential advantages of the use of a large, coordinated group of AUVs for oceanographic data gathering, the principal objective of this work is to investigate approaches to swarm navigation and formation control in the underwater realm. The communications

constraints of such an environment, with data rates on the order of 100 bits per second to 5000 bits per second, and the fairly frequent loss of packets, poses significant challenges on the ability to form and maintain multi-vehicle AUV formations. Although the existing literature on multi-robot formation control is sizeable, the unique constraints of the underwater domain implies that it is not simply a matter of applying existing above-ground techniques to the underwater realm - its constraints necessitate the development and testing of formation control behaviours that can operate successfully with significant restrictions on inter-vehicle communications. An investigation into this issue is the main contribution of this thesis.

1.1 Contributions

This thesis presents three contributions - the primary contribution is the development of four different formation control behaviours which are implemented in the MOOS-IvP autonomy infrastructure, and which allow a group of AUVs to form lattice or pattern formations that are rotationally and translationally invariant. The second is the proposal of a formation quality metric. This metric quantifies how well the behaviour is able to maintain the desired swarm formation, and is used in conjunction with formation energy expenditure (how efficiently the behaviour uses energy to maintain formation), and formation robustness (how well the formation is maintained in the presence of node failure and varying communication rate), to compare the efficacy of each behaviour. Finally, the current MOOS-IvP AUV simulation test-bed is augmented to allow the accurate simulation of ocean currents, as well as acoustic communication between multiple vehicles.

1.1.1 Lattice/Pattern Formation Behaviours in MOOS-IvP

The formation of lattices or patterns by a swarm of autonomous underwater vehicles is the principal objective of this work. We envision the use of such a formation of AUVs for two specific applications - by equipping each vehicle with an acoustic receiver, the formation can be employed as a 'virtual' acoustic array, allowing its use as the receiver for sea-bottom seismic surveying, or the monitoring of naturally occurring acoustic and seismic radiation from ice; secondly, by equipping each vehicle with biological sensors, biological phenomena which are very spatially and temporally dynamic (such as phytoplankton blooms) can be monitored with greater accuracy over large regions. Ideally, behaviours would allow the swarm to autonomously maintain a formation without requiring a comprehensive navigation and communication infrastructure, using a paradigm similar to that of a school of fish, where each vehicle navigates and maintains formation solely by communicating acoustically with its nearest neighbours. In addition, we wish to maximize the length of such missions, explicitly by making use of ocean currents as a means of propelling the swarm formation. In order to

increase the endurance of missions using the swarm, the swarm is expected to utilize average ocean currents for propulsion, with the vehicles using their motors only as a means of maintaining formation geometry. Finally, given the communications constraints of the underwater environment, pattern formation cannot be performed in a centralized (or even decentralized) manner - all behaviours must be completely distributed.

Taking into account these ambitions, as well as the constraints of the underwater environment, four pattern formation behaviours were developed in an attempt to meet these goals and challenges. A brief overview of each is provided here.

Attraction/Repulsion Atomic Model (BHV_AttractionRepulsion)

I present a lattice formation behaviour based on the idea of interatomic forces. As the forces between atoms naturally produce a hexagonal lattice, applying similar forces between vehicles is an obvious method for producing a hexagonal lattice formation of AUVs. This idea has been well researched in past swarm robotics literature, under the broad title of physics-based formation control, but my implementation uses a direct optimization technique to determine the force minimum, a novel extension of past work.

Pairwise Neighbour Referencing (BHV_PairwiseNeighbourReferencing)

This approach requires a predetermined plan of the formation pattern, as well as communication of unique vehicle IDs. The plan details the shape and scale of the formation, with the relative positions of each vehicle within the formation. Every vehicle utilizes this plan in order to reference itself against pairs of its neighbours, using the angle and distance between itself and each pair to determine its relative optimal location. This is perhaps one of the simplest approaches to pattern formation, but is robust, translationally and rotationally invariant, allows any arbitrary shape to be formed, and as far as I can tell, has not been described in previous literature.

Rigid Neighbour Registration (BHV_RigidNeighbourRegistration)

This behaviour was inspired in part by the iterative closest point (ICP) algorithm and as an approach to improving on the previous behaviour. ICP is widely used to align two point clouds without knowing the correspondence between individual points in each cloud; however, if the correspondences are user defined, the alignment between the clouds, in which we wish to minimize the mean-squared error in distance between all points (by selecting an optimal rotation and translation combination), has a closed-form solution. This error minimization problem is known as the orthogonal Procrustes, or the rigid point set registration problem, and is the transformation step in the ICP algorithm. In terms of pattern/lattice formation,

given our predetermined plan and unique vehicle IDs that correspond to positions in our plan, this behaviour transforms the plan to the actual vehicle positions in a distributed manner, so as to create the desired pattern.

Assignment Registration (BHV_AssignmentRegistration)

In an attempt to improve on the Rigid Neighbour Registration behaviour, I present a novel approach that attempts to both perform the point correspondence between the plan and vehicles, and the rigid point set registration. Vehicles are assigned to points in the plan dynamically via the Hungarian algorithm so as to minimize distance travelled, and as before, minimization of the mean-squared error in distance is used to determine the optimal rotation and translation of the local plan. By removing the requirement of unique vehicle IDs, I hope to reduce the communication requirement between vehicles.

1.1.2 Formation Comparison Metrics

A goal of this work is to compare the efficacy of each behaviour, with respect to our scenario - we wish to form a lattice formation with minimal energy expenditure by each vehicle in the swarm, and once formed, we want the formation to maintain its shape, utilizing surface currents as effectively as possible for swarm travel (expending energy only to maintain shape). To perform this comparison, I developed a metric to quantify the quality of the formation, using a similar approach used in the Assignment Registration behaviour above. Given the location of each vehicle, and the lattice formation we wish to achieve, I essentially assign each vehicle to a point in the desired formation so as to minimize Euclidean distance using the Hungarian algorithm, and then determine the optimal rotation and translation of the desired formation to minimize the mean-squared error between each point. The value of this error is used as the quality comparison metric.

This metric is used as a comparison against energy expenditure (both during the forming of the formation, and during the maintaining of the formation), as well as against vehicle failure and communication rate to determine formation robustness.

1.1.3 Comparison Scenarios, Behaviour Testing, and the MOOS-IvP Ocean Simulation Test-Bed

Each behaviour is tested in four comparison scenarios, where each scenario represents a different ocean current field - the first is a field with no currents; the second is in a field of three current channels of different velocity; the third is in a vortex current field; and the fourth is in a current field representing realistic ocean currents. These scenarios and testing of each

behaviour for comparison is performed in the MOOS-IvP Ocean Simulation test-bed.

The MOOS-IvP Ocean Simulation test-bed provides a platform to test autonomous behaviours using accurate ocean current dynamics, acoustic communications, and vehicle dynamics, and allows the user to visualize these behaviours in rates faster than real-time. My contribution to improving this test-bed is in the efficient integration of ocean current simulations provided by the MIT MSEAS group, allowing us to utilize the test-bed for scenarios with highly realistic ocean currents.

1.2 Summary

This chapter has presented the reader with an overview on the motivations, objectives, and contributions of the work undertaken in this thesis. I have described the behaviours I have implemented for the goal of forming desired lattice or pattern formations for a group of AUVs, the metrics by which I evaluate each behaviour on its ability to produce the desired formation, and the scenarios and test-bed used to undertake these comparisons.

The remainder of this thesis is organized into six chapters. Chapter 2 provides the reader with an outline of background topics pertinent to my research, as well as a survey of related literature in swarm robotics. Chapter 3 provides a detailed explanation of the four lattice/pattern formation behaviours implemented for this thesis. Chapter 4 gives the reader an overview of the methodology, infrastructure, and comparison metrics used to evaluate each behaviour in simulation. Chapter 5 provides the results and an analysis of the comparison metrics. Finally, chapter 6 details a discussion of the results of the analysis, conclusions drawn from this research, and future work to be undertaken.

This page intentionally left blank.

Chapter 2

Background and Related Work

To give the reader a good understanding of the work undertaken in this thesis, we provide a gentle introduction to background topics related to this research, with the assumption of little prior knowledge from the reader. We begin with an overview of autonomous underwater vehicles, the platform that we envision this work to be applied to; this is followed by an introduction to MOOS-IvP, the software architecture I have used to implement my work; we then provide a broad outline of swarm robotics and swarm intelligence research, with a more detailed review of prior work in swarm formation control; and finally, we specify the numerous assumptions along with the scope of this work, and summarize.

2.1 Autonomous Underwater Vehicles

Autonomous underwater vehicles (AUVs) are untethered underwater mobile robotic systems that carry a sensor payload that is typically used for measuring water properties and the ocean bathymetry and environment. In recent years, AUVs have become an increasingly valuable and popular tool for oceanographic research, and the dramatic improvement in their capabilities and reliability over the past two decades has resulted in a growing interest in their use for scientific, commercial, and military applications. The commercialisation of AUVs has also meant that they now potentially provide a cost effective alternative to traditional ship-based oceanographic sampling and measurement. Depending on the application, AUVs can provide a number of advantages over traditional methods. For example, with regards to scientific oceanography, ship-based methods can be limited in their sampling rate and their ability to measure highly dynamic and unpredictable spatial and temporal ocean phenomena. In contrast, AUVs have demonstrated their utility in gathering time-series oceanographic data by repeated water column surveys, and their autonomy opens up the prospect of adaptive behaviours that may allow the effective tracking of dynamic ocean phenomena. In the military sphere, the autonomy of AUVs potentially allows their use in areas which are restricted to,

or dangerous for personnel, and their minimal profile presents their possible application in covert operations.

The majority of currently available AUVs can be classified into four categories depending on their intended application; these categories are shallow water survey AUVs, mid-water AUV's, deep-water AUVs, and glider AUVs. Shallow water AUVs are designed to operate in coastal and littoral areas, and at depths less than 500 meters. They are typically fairly small (usually around 1-2 meters) since they do not have to withstand high water pressures, and have a high drag to thrust ratio, which allow them to manoeuvre in areas with high currents. These vehicles are typically used to survey areas fairly quickly, and at low resolution, and so their operating speeds are relatively high. Mid-water AUVs are usually rated for depths of up to 2.5 kilometres, and are typically used to perform mid-water column surveys or surveys in shallower areas. In order to handle the pressure at these depths, these vehicles are usually quite bulky, which in turn means that they need more thrust and power, adding to their size (which range from 4-6 meters in length). As the currents at these depths are generally low, AUVs in this class can have a small drag to thrust ratio. Depending on their application, their operating speed can vary from less than one knot to several knots. Deep-water AUVs are designed to be used at depths greater than 2.5 kilometres, and are typically large in volume in order to withstand the high oceanographic pressures at such depths. Since this class of AUVs are usually used close to the bottom of the ocean floor for high resolution surveys, they are designed to manoeuvre at low speeds. As such, their design is typically quite different to that of shallow water survey and mid-water AUVs, the majority of which have a tubular, torpedo-shaped hull and use a single propeller in conjunction with elevators and fins. In contrast, deep-water AUVs usually have multi-hull designs with multiple thrusters. Glider AUVs operate in a significantly different manner to that of the previous AUV classes, in that they are propelled through the water via changes in buoyancy and water temperature in conjunction with wings to convert vertical motion into forward motion. This propulsion method typically achieves a much higher efficiency than conventional electric thrusters, increasing their range to the order of thousands of kilometres, at the expense of horizontal and vertical manoeuvrability. These vehicles usually operate in the upper water column, and are typically rated for depths of less than 1 kilometre.

In more recent years, an additional class of AUVs has emerged - the long-range AUV. These AUVs have been specifically designed in order to perform sampling of oceanographic processes that evolve over periods of days or even weeks, and as such, are built with the goal of maximizing endurance. In order to do so, they are designed with novel improvements to previous AUV classes, such as active buoyancy control, allowing them to sample processes at a desired depth without expending energy on propulsion. Two such examples of these AUVs

are the Tethys [1] AUV built at MBARI, and which has active buoyancy control and utilizes custom energy saving strategies (such as power-down of motor controllers and non-essential systems), and the Folaga [2] AUV built by Graaltech, which similarly has active buoyancy control as well as an actuation mechanism that allows it to propel as a glider. We envision the use of this class of AUVs for long endurance multi AUV oceanographic sampling, and their emergence can be interpreted as an endorsement for the benefit of using swarms of AUVs for this purpose.

Despite the enormous developmental gains in autonomous underwater vehicle technology over the past few decades, there remains a number of challenges facing their applicability and usefulness. This includes challenges in power consumption, navigation and positional accuracy, underwater communications, and intelligent and adaptive autonomy. As the focus of this thesis is the investigation of autonomous swarm behaviours, the architecture that I use to develop and implement such behaviours, as well as existing examples of swarm robotics, are expanded upon here.

2.2 MOOS-IvP

MOOS-IvP [3] is an open-source software infrastructure used for the development of autonomous behaviours for unmanned marine vehicles. It is composed of two open-source projects - MOOS (the Mission Oriented Operating Suite), which provides core middleware capabilities in a publish-subscribe architecture, whereby MOOS applications (MOOSApps) asynchronously publish and subscribe to information from a central database (the MOOSDB); and the IvP (Interval Programming) Helm, a foundational MOOS application that provides vehicle behaviour arbitration via multi-objective optimization, deciding upon an optimal output (typically vehicle heading, speed and depth) by evaluating competing behaviours. The IvP portion of this infrastructure was developed by the Laboratory of Autonomous Marine Sensing Systems (LAMSS) at MIT, a group of which I am part, and consequently this is the software of choice for the implementation of my behaviours.

In the context of this work, the organization of the MOOS-IvP architecture is as follows; a single centralized 'shoreside' MOOS community is run, which houses a number of MOOSApps related to the extraction or calculation of ocean currents, the evaluation of the formation quality metric, and formation visualization; in addition, each vehicle in the swarm has its own MOOS community, running MOOSApps related to simulating vehicle dynamics, calculating energy consumption, and an IvP helm to process the formation behaviours. To simulate acoustic communication, vehicle MOOS communities pass information related to vehicle state to the shoreside MOOS community, the acoustic communication is simulated

centrally (adding noise etc.), and the result is passed back to each vehicle - anything that must be centrally computed for simulation purposes (such as ocean currents) is performed in the same way. It must be stressed, however, that behaviours described in this work are entirely distributed - the shoreside MOOS community is not used to control the behaviour of individual vehicles.

On each vehicle, the IvP helm solves for optimal control (heading, speed and depth) using:

$$x^* = \underset{x}{\operatorname{argmax}} \sum_{i=0}^{k-1} w_i f_i(x) \tag{2.1}$$

where each $f_i(x)$ is an objective function produced by each active behaviour on the vehicle, and each w_i is a relative priority weighting for the behaviour. As such, the IvP helm attempts to determine the parameters for which the sum of the objective functions for each active behaviour is maximized, and does so by evaluating the objective function of each behaviour over the entire decision space. How this is achieved is detailed in [3].

2.3 Swarm Robotics and Swarm Intelligence

Swarm robotics has emerged in the past decade as an area of research concerned with the application of swarm intelligence [4] concepts to multi-robot systems. Swarm intelligence is a form of distributed intelligence, in which the collective behavior of a group emerges out of the simple behaviors of its autonomous individuals through local peer interaction, and interactions with the environment. Although swarm robotics systems are not explicitly required to be distributed, research in swarm robotics is often inspired by biological systems, such as insect colonies (for example, cockroaches [5], ants [6] or bees [7]), flocks of birds [8], schools of fish [9], and bacteria colonies [10]. The absence of centralized control in such biological systems provides a number of advantages, and by seeking to replicate these advantages, many swarm robotics systems are typically distributed. This lack of centralization gives such systems implicit advantages of fault tolerance (failure of any single individual has little effect on the success of the group), flexibility (self organization and no reliance on global information), and scalability (behaviors are local so that the addition or removal of individuals has little impact on swarm performance). Swarm robotics has been a growing area of research over the past two decades, especially in the land and air domains. Unfortunately, until very recently little research has been performed in the underwater domain, and the limitations of the ocean environment provide a unique variation to the regular challenges of swarm autonomy. Here we provide a broad overview of different facets of swarm autonomy, and look especially at previous research directed at distributed swarm formation control. This section

is by no means an exhaustive review of swarm robotics, but the interested reader is directed to [11], [12], [13], [14] and [15] for good reviews of the recent state of the art in this research area.

Swarm robotics presents researchers with a range of potential advantages over single robot systems, and these advantages are the prime motivators behind swarm research. Swarm robotics systems are envisioned to be able to:

- Exploit the sensing capabilities of large groups, allowing the efficient discovery and exploration of areas of interest, as well as improve situational awareness.
- Provide superior robustness against mission failure, since the failure of a single agent in the group can be mitigated by other agents.
- Parallelize mission tasks amongst agents in the group, allowing a mission to be completed faster than if performed by a single agent. In addition, this distribution of tasks may enable swarms to achieve greater results, such as conducting missions over larger areas, manipulating objects or the environment more efficiently, or attacking with numbers.
- Be adaptable and scalable, allowing missions to continue via task reallocation with the addition and removal of agents in the group. Since interactions are localized, the addition or removal of agents does not require any change to control software.
- Be cost effective. By using many simple vehicles, rather than a single complex vehicle, swarms are able to have a greater cost effectiveness, by virtue of the fact that a loss of one simple vehicle in the swarm has less impact than the loss of a single more powerful vehicle.

However, these advantages come together with a couple of drawbacks. Firstly, operator command and control of large robotic swarms is difficult; centralized C&C schemes may not scale well with increasing number of agents, and decentralized C&C schemes may have trouble gathering and synthesizing data from all members of the swarm; deployment and retrieval of such swarms is also an open question. Secondly, because the design of behaviours of agents in the swarm is done with local interactions in mind, the global behaviour of the swarm can be difficult to predict, as it emerges from numerous locally interacting agents.

The design of robotic swarm systems include considerations that can be broadly defined into two categories - architecture and application. Within the architecture category, the designer must consider a number of facets that must be selected with respect to the intended application of the swarm. These include the selection of heterogeneous versus homogeneous robot swarms, centralized versus distributed control schemes, and communication structures.

Within the application category, the designer must develop specific or groups of swarm strategies that can most effectively address the desired task. These strategies include swarm behaviours such as aggregation [16], [17], dispersion [18], task allocation [19], coordinated collective motion [20], [21], object transportation [22], collective exploration and mapping [23], [24], and pattern formation. Within the scope of our work here, we can classify our swarm system as being homogeneous (within our application we wish to have single-type AUVs that either serve as acoustic receivers, or are able to sense a specific oceanographic phenomena), distributed (since the constraints of the underwater domain prohibit sufficient communication bandwidth for centralized control), having a local communications range (as acoustic communications becomes much less reliable at larger distances), and having a pattern formation behaviour (as this is what is required to achieve our desired goals). With this in mind, we limit our literature review to cover pattern formation approaches for robotic swarm systems that have a similar architecture. We do not provide an overview of differences in architecture or review approaches for applications other than formation control. As before, the interested reader is directed to [11], [12], [13], and [14] if they desire greater insight into these topics.

2.3.1 Distributed Swarm Formation Control

In swarm robotics, the term pattern formation control has been used in at least two different ways. Firstly, pattern formation has been used to broadly encompass multiple aspects of patterns of agents, including the establishment, maintenance and reconfiguration of patterns or lattices, rather than exclusively pattern establishment. Secondly, the term pattern formation has been used in literature interchangeably with the phenomenon of swarm flocking, which defines behaviours that are loosely geometric in nature. In term of this work, we refer to swarm formation control in the sense of the first definition - we mean pattern or lattice formation control as behaviours that produce and control well defined geometric patterns of agents in the swarm. In the context of this definition, past research can be broadly categorised into four groups, each of which we examine here. For further reading and other reviews of swarm formation control, we direct the reader to [25] and [26].

Physics-Based Approaches

One of the most common approaches to pattern formation in swarms is the physics-based approach. In this approach, formation generation is often inspired by the physics of atoms, molecules or crystals, or by the physics of springs. Physics-based formation design use virtual forces to coordinate the movement of agents in the swarm.

One of the earliest and most well-known physics-based approaches is the 'Physicomimetics' framework, introduced by Spears et al. [27], [28]. In their approach agents in a swarm

react to virtual forces inspired by natural physics laws. At an abstract level, agents are used to mimic physical structures of particles, with each particle having a position and a velocity. At each time step of the algorithm, particle positions are perturbed by an amount dependent on its current velocity; this velocity is itself modified at each time step in accordance to a force law applied to the particle by its neighbours; and this force law is essentially a repulsion/attraction force of certain radii centred at each particle. Using this approach, they are able to form hexagonal lattices which attempt to minimize an overall 'potential energy' of the swarm. They are also able to develop an analytical model of the potential energy wells that cause the formation of this hexagonal structure, providing an analysis of lattice quality. They also extend their work to produce square lattices, and provide strategies to prevent agents from falling into local minima. They demonstrate their Physicomimetics technique in both simulation and on a group of seven robots. In [29], Prabhu et al. extend this framework to produce stable hexagonal cells, and to remove lattice imperfections.

In [30], Pinciroli et al. present a similar approach, whereby local artificial potential fields centred around each agent are modelled after the interatomic Lennard-Jones potential model (used to approximate the interaction between neutral pairs of atoms) in order to produce a hexagonal lattice of simulated pico satellites. However, in order to induce aggregation in the agents, they introduce a global potential field centred at a desired coordinate, so as to draw all agents to a single point. For stabilization, they also include a third damping term to prevent oscillations commonly observed in physics-based approaches.

Similarly, Gazi and Passino [31] introduce different classes of attraction/repulsion artificial potential functions for distributed formation control. Their artificial potential functions include linear attraction/bounded repulsion, linear attraction/unbounded repulsion, and constant attraction/unbounded repulsion, each of which they analyse in the context of swarm cohesion, and provide simulation results.

An alternative approach to attraction/repulsion schemes is the artificial springs technique, first introduced by Fujibayashi et al. [32], where potentials are modelled as artificial springs with varying coefficients. In this work, each agent generates virtual springs between itself and neighbouring agents, based on the number of its neighbours within a specified radius, in order to produce a crystalline lattice structure. To produce a desired lattice, they modify the spring constant and natural length properties of each spring depending on the number of connections of the agents on either end of the spring; they also include a method of separating agents. By introducing certain combinations of springs and through tuning they are able to generate desired shapes.

Shucker and Bennett [33] present a similar approach, using a virtual spring mesh to control a formation of robots. In this work, the authors describe an algorithm that controls the creation and destruction of virtual springs between agents that operates in the following way - a robot R will create a spring with its neighbour S, if for every other neighbour T, the interior angle RTS is acute. They argue that such an approach will influence the swarm to create a hexagonal lattice, because such a lattice is the only optimal zero-energy state of the system. They provide simulation examples for swarm exploration and target tracking, and analyse the system in the case of catastrophic agent failures.

Finally, Stolkin and Nickerson [34] provide a review and comparison of different physics-based approaches, and propose a novel method that combines them - they use an attraction/repulsion model to form local clusters of agents, then use a spring technique to group clusters together. This allows local clusters to produce regular lattices, while attracting local clusters together quickly via spring-like attraction behaviours. Unfortunately this method is decentralized; it is not completely distributed.

Potential Field Approaches

Potential Field approaches to swarm formation are somewhat similar to physics-based approaches, but differ in one important way - the potential field is global, rather than local. In these approaches, this global potential field is used to actuate agents toward minima in the desired shape of the formation.

Perhaps one of the earliest use of this approach is presented by Bachmayer and Leonard in [35]. In this work, they use artificial local potentials to maintain group geometry, while at the same time using a gradient descent method to drive the group toward a minimum. They achieve this by having each agent sum an approximation of the world gradient with the local potentials of its neighbours; the world gradient is approximated using a single sensor which is assumed to be able to measure the gradient only in the direction of agent motion. Although usually such an approach would cause an agent to find a minimum only along a 'slice' of the world, the introduction of inter-vehicle potential functions enables the group to communicate enough information to determine the global world minimum in an emergent manner.

Chaimowicz et al. [36] present an approach to swarm formation whereby they create potential fields whose minimum lies along a 2D curve described by an implicit function. This implicit function is viewed as the zero isocontour of the 3D potential field, whose value is greater than zero outside the isocontour, and less than zero inside the isocontour. By making each agent perform a gradient descent on the potential field (and inverting the gradient when

the agent is within the isocontour), the authors are able to direct the agents to converge along the desired isocontour. They are able to produce a wide variety of desired formation shapes, including letters, using this implicit function approach. The authors provide an analysis of system performance, as well as results from simulations of tens of robots and experimental results with a team of six vehicles.

In her PhD dissertation, Barnes [37] demonstrates the results from a similar approach - artificial potential fields are generated using normal and sigmoid functions, as well as other limiting functions, to control overall swarm geometry and spacing. She presents the results from simulations of four to ten vehicles performing circle, wedge and ellipse formations. Using a fuzzy speed controller, she directly uses the gradient of the potential field to direct the movement of agents in the swarm.

Virtual Structure Approaches

The virtual structure approach was first introduced by Lewis and Tan [38]. In this approach a rigid formation (referred to as a structure) is defined, within which agents maintain a rigid geometrical relationship. In this way, the entire swarm is treated as a single rigid body, with agents acting as vertices of the body.

In the approach originally presented by Lewis and Tan [38], the authors present an algorithm which is composed of four steps - the first step involves aligning the virtual structure to the current positions of agents in the swarm, via an optimization problem to minimize the difference between actual and desired agent positions; the second step displaces the virtual structure toward a desired mission objective; the third involves computing agent trajectories so as to realign their positions with the structure within a specified time window; and the final step directs the agents to follow the calculated trajectories. Unfortunately, their original approach was centralized; however, Ren and Beard [39] present a decentralized extension to this work using local controllers.

Belta and Kumar present a control approach [40] that designs trajectories such that a rigid formation of agents will maintain their geometry. Specifically, the method outlined generates trajectories that minimizes the total energy associated with the translations and rotations of the robots, while maintaining their current formation. Their method involves three steps; the first generates optimal trajectories for the formation; the second projects these trajectories on a specified Euclidean group that represents the gross position and orientation of the swarm, the set of shape variables that describe the relative positions of robots in the swarm, and the control graph that describes each robot's control strategy; the third step performs a transla-

tion of the motion to position trajectory for each individual robot.

In [41], Egerstedt and Hu present a model-independent coordination strategy for swarm formations, where instead of designing control laws for the agents directly, they assume agents possess existing tracking controllers and instead design an algorithm that generates reference points. The desired formation structure is defined relative to a virtual formation leader, and this leader moves along a parametrized path. The trajectory of each agent is selected so as to propagate in the direction that minimizes a specified control function, where the magnitude of the motion depends on how well the agent tracks the structure. In turn, the velocity of the virtual leader along the path is dependent on the tracking errors of all the agents.

Leader-Follower Approaches

In leader-follower approaches, a hierarchy of agents is defined within the formation, meaning that such approaches inherently require the communication of unique agent IDs. Followers attempt to maintain formation with their respective leader(s), whose motions are either prescribed (such as following a specified path) or who themselves follow their own leader(s). In addition, leaders do not necessarily have to be physical agents in the swarm, as virtual leaders can also be used to effectively control group motion.

Desai et al. [42] presents a popular leader-follower control strategy whereby feedback linearisation is used, along with a formation plan defined by relative angles and distances between agents to produce two feedback controllers - the first is a separation/bearing controller between a follower and its leader, and the second is separation/separation controller between a follower and two leaders. Other authors expand upon this approach with other control strategies, such as dynamic feedback linearisation [43], model predictive control [44], [45], and first and second order sliding mode control [46].

In [47], Elkaim and Kelbley combine a leader-follower approach with a physics-based approach. They describe a formation whose geometric structure is maintained via inter-vehicle artificial forces, and which travels along a desired trajectory using artificial forces between each vehicle and a virtual leader which is guided along the trajectory. The general methodology of this approach is as follows - first, all of the forces that are acting upon the agents are calculated; secondly, the point at which the sum of these forces is zero is determined; finally this point is then used as the reference position for each individual vehicle, which moves towards the point using its own kinematic controller. This process is iteratively repeated.

Other Approaches

Besides the approaches categorized in the sections above, a considerable amount of research in swarm formation control does not fit comfortably in the physics-based, potential field, virtual structure or leader-follower groups. We examine some here.

In [48] Song and O'Kane present a novel decentralized algorithm for forming arbitrary multi-robot lattices, such as squares, hexagons, and octagon-squares. Their approach makes use of an author-defined 'lattice graph' (not to be confused with lattice graphs of graph theory). This lattice graph is a strongly connected directed multigraph whose edges describe a rigid-body transformation, and is used to define the desired lattice of the swarm. Their algorithm requires the broadcast of short messages between neighbouring agents, and executes the following series of four steps. First, each agent must decide whether to consider itself as a root robot (one that remains motionless), or a descendant robot (one that moves in accordance to a task assignment from a parent); this decision is made based on an author-defined 'authority', which depends on robot ID as well as information about robot's 'ancestors', and is done to ensure that agents form a stable forest of authority trees. Secondly, agents select a role; root agents always select the first lattice vertex as their role, and descendant agents accept their role by task assignment from their parent. Thirdly, after role selection, each agent computes a locally optimal task assignment for its neighbours, using the Hungarian algorithm; each agent broadcasts this assignment and their authority value to its neighbours. In the final step, each descendant agent moves toward the position assigned by its parent, while staying within communication range. The authors present the results of their algorithm using a simulation of between 50 and 250 robots.

Lee and Chong [49] demonstrate a distributed algorithm for lattice formation using simple geometric principles. In their approach, an agent selects two of its neighbours such that the triangular path travelled by the positions of three agents is minimized; then using these two neighbouring agents, the robot calculates the centroid of the triangle formed by itself and its neighbours, and measures the angle between the line connecting two neighbours and its horizontal axis; finally, in reference to the calculated centroid and using this angle and a desired inter-agent separation distance, the robot repositions itself to a target that is computed using simple trigonometric equations such that it forms an isosceles triangle with its two neighbours. The authors prove convergence of this algorithm using Lyapunov stability theory, and demonstrate formation self-repair in simulations of 100 robots.

Lee and Chong [50] also present another approach using similar geometric principles, but in which agents are referenced against neighbours and a local leader is dynamically selected.

Agents are given predefined unique IDs, and using these, they position themselves in relation to neighbours using rules specific to different formations. The leader is selected as the agent farthest from the centroid of all agents, and is used to control the movement of the entire group; as such, the authors assume that all agents are visible to one another.

Antonelli et al. [51] describe an approach to distributed lattice formation via a behaviour-based control architecture termed Null-Space-based behavioural control. This approach is rooted in graph theory, and uses an author-defined α-Lattice to define the desired lattice formation. The α-Lattice structure is described as a geometric configuration whereby all edges in a graph are of the same length. Utilizing this α-Lattice structure, the authors develop a control law such that the swarm of agents are driven to an α-Lattice formation within a specified tolerance, and implement this control law within a Null-Space-based control architecture, allowing them to demonstrate the lattice formation behaviour along with obstacle avoidance and move-to-target behaviours in simulation.

In [52], Raffard et al. employ dual decomposition techniques in order to generate optimal trajectories for a set of cooperative aircraft in a distributed fashion. Their algorithm solves the dual problem of an artificially decomposed primal problem, allowing them to replace one large computationally intractable problem with several smaller tractable problems. They present complexity analysis of convex and non-convex cases using examples in simulation.

Unique Considerations for the Underwater Environment

As mentioned previously, the majority of swarm robotics research has taken place in the context of land and air domains, and unfortunately, not much research has been undertaken with the intention of applying swarm intelligence concepts to robots in the underwater domain. Even less research exists in using swarm formation control strategies in this context. In [53], Hu et al. use an undirected graph to model information exchange between multiple AUVs, and the authors develop a distributed controller that satisfies an admissible impulse time sequence that represents the communications constraints of the underwater environment. The authors demonstrate the feasibility of their approach with simulations of four AUVs operating under their defined dynamics. In [54], Amory et al. present the development of a miniature AUV called the MONSUN II, which use their communication network to propagate their internal states and build formations. The authors outline a hierarchical communications structure and a set of action phases that are undertaken to produce a 'V' formation, and demonstrate results in a simulation of five vehicles. In [55], Kalantar and Zimmer study the formation of a large group of underwater vehicles for shape formation. Their approach works by partitioning the group of vehicles into two non-overlapping sets - the boundary and its interior. The

vehicles identified as part of the boundary are then driven into a desired shape using general theory of curve evolution, while interior vehicles maintain formation using a physics-based attraction/repulsion approach in order to form a uniform interior distribution of vehicles. In [56], Shao et al. control the formation of multiple biomimetic fish-like robots. Using a geometric leader-follower approach under the constraints of the fish robot dynamics, they present experimental results on a group of three robots in a line formation. In [57], Schmickl et al. outline the CoCoRo underwater swarm project, in which they detail their plans to build the hardware and software for a swarm of miniature underwater vehicles which behave according to bio-inspired motion principles and biologically-driven collective cognition mechanisms.

The underwater domain places unique constraints on swarm robotics and swarm intelligence, the primary limitation being that of communication. Underwater acoustic communication is generally limited by low bandwidth and intermittency due to the nature of the medium, in which multipath propagation, high ambient noise, and strong signal attenuation results in inter-symbol interference and low data rates [58]. A swarm system introduces an additional issue of message collisions, due to the number of agents needing to communicate. As such, the limitations in bandwidth and rate of acoustic communication will affect the amount of information that can be passed between agents in a swarm, and in turn affect possible approaches for swarm formation control and maintenance. The majority of approaches reviewed in this section have relied on being able to estimate the range and bearing of an agent's neighbours, at the very least; some approaches also require the ability for agents to communicate a unique identifier, and still others require even more information to be communicated. In the underwater domain, ranges between neighbours can be computed using time-of-flight with knowledge of the acoustic speed of propagation in the current ocean environment, along with accurately synchronised clocks (e.g. chip-scale atomic clocks) and acoustic pingers; bearings between neighbours can be estimated via an acoustic hydrophone array (e.g. a tetrahedral 3D array) and sufficient signal processing, or alternatively by using vector sensors; and unique IDs can be communicated between two vehicles either by using an acoustic modem and compressed encoding/decoding schemes [58], or via narrowband acoustic pingers with a unique frequency for each vehicle. In general, when investigating swarm formation control for the underwater domain, it is obvious that control strategies that minimize the amount of information passed between agents are highly advantageous. With this insight in mind, we are most interested in formation control approaches that use at most range, bearing and ID to generate and maintain formations. In addition to this, this limitation in communication means that external positioning infrastructure, such as GPS, cannot be used. Agents in an underwater swarm must position themselves relative to neighbours in a local frame of reference, rather than in a global reference frame.

In concluding the review of prior related work, I would like to underscore the fact that the amount of literature in the area of swarm robotics is immense, with the number of published papers on the subject of swarm formation control itself being large. In pruning this vast forest of possibilities, I have attempted to limit my examination to distributed algorithms that may be of use in our particular application, but there remain many more possible approaches that remain unsurveyed.

2.4 Assumptions and Scope

As described in section 1.1, the goal of this research was to develop and analyse approaches for the formation of patterns or lattices using a swarm of AUVs, for the purpose of sensing and measuring oceanographic phenomena. Four different algorithms for AUV formation control were implemented in MOOS-IvP, and a 'formation quality' metric was developed to compare the efficacy of each algorithm with respect to energy expenditure and robustness. In order to reasonably limit the amount of work undertaken during this investigation, for simulation purposes a number of assumptions were made and the scope of the work was limited as outlined in the following:

- We limit formation control to a 2D plane, with AUVs operating at a specified depth - depth control is handled by each AUV individually, without the use of neighbouring AUV depth information.
- Since we limit formation control to a 2D plane, the four behaviours described in this thesis produce an IvP objective function over the vehicle's heading and speed decision space. Depth control is handled by a separate, already existing behaviour called BHV_ConstantDepth.
- We assume that AUVs communicate via underwater acoustics (acoustic pings), with a uniform acoustic sound speed of $1500\,m/s$.
- We assume that each AUV is able to measure relative bearing and range to neighbouring AUVs within a radius of $550\,m$ via this acoustic communication.
- We assume that the bearing measurement has noise with a Gaussian distribution with a mean of $0\,^\circ$ and standard deviation of $5\,^\circ$.
- We assume that the range measurement has noise with a Gaussian distribution with a mean of $0\,m$ and standard deviation of $1.5\,m$.
- We assume that any additional information communicated between AUVs, such as unique ID, occurs during the measurement of bearing/range.

- Unless otherwise specified, we assume that communication and measurement of bearing/range occurs once every 30 s.
- Each AUV navigates using a local coordinate frame (with itself at the origin) using the relative bearing/range measurements of its neighbours - between subsequent acoustic pings, we assume the AUV navigation solution has no drift. Essentially, we assume the position of the AUV is accurate between each acoustic ping - this assumption is justifiable, since the AUV readjusts its relative position at every ping based on the relative location of its neighbours, and the occurrence of pings is fairly frequent.
- Due to this relative navigation approach, we assume that the global AUV position is accurate, as global positions do not impact the accuracy of relative formation control.
- We assume that AUVs have active buoyancy control, allowing them to drift freely at a desired depth.
- Unless otherwise specified, we limit the swarm size to 25 AUVs.
- We assume that AUVs can turn in place; although our behaviours do not explicitly require that the AUVs be able to do so.
- We limit the scope of this work to the development of the four formation control algorithms, the formation quality metric, and the comparison scenarios described in section 1.1.3.

2.5 Summary

This chapter has given the reader a brief background of topics related to the work of this thesis, as well as an abridged review of literature related to swarm robotics and distributed swarm formation control. In addition, I have outlined the assumptions and scope of the work undertaken in this thesis.

The remainder of this thesis is organized into four chapters. Chapter 3 provides a detailed explanation of the four lattice/pattern formation behaviours implemented for this thesis. Chapter 4 gives the reader an overview of the methodology, infrastructure, and comparison metrics used to evaluate each behaviour in simulation. Chapter 5 provides the results and an analysis of the comparison metrics. Finally, chapter 6 details a discussion of the results of the analysis, conclusions drawn from this research, and future work to be undertaken.

This page intentionally left blank.

Chapter 3

Distributed Formation Control Behaviours for AUVs

In this chapter we delve into the details of my four formation control behaviours, describing the implementation and algorithmic details of each. It is important to note, as previously mentioned, that all the behaviours described in this chapter are run on each vehicle in a distributed manner. As described in section 1.1.1, each of these behaviours has been developed for one specific purpose - to produce a desired pattern or lattice formation from a swarm of AUVs, and, once having done so, use minimal energy to maintain this formation in the presence of ocean currents, intrinsically utilizing these ocean currents so as to propel the formation. In this respect, these behaviours are aptly described as formation controllers; the desired formation is the set-point, and it is the job of these behaviours to maintain this set-point in the presence of external disturbances.

Each of the four behaviours was implemented as an IvP behaviour in the MOOS-IvP architecture briefly described in section 2.2. Each IvP behaviour produces a piecewise linearly defined objective function called an IvP function - IvP functions are defined over a specified decision space, which, in our case, is the speed/heading domain. The MOOS-IvP architecture allows an author to specify the properties of the IvP function using a number of tools. For example, figure 3.1 illustrates two functions defined over the speed and heading domains, and the IvP function outputted by their coupling - the coupled objective function is greatest at the red-colored peak, corresponding to the desired speed of 2 and heading of 135. The job of the IvP Helm MOOSApp is to arbitrate between the IvP functions outputted by multiple active IvP behaviours, ultimately outputting optimal desired values for decision variables in the vehicle's decision space. For greater insight, see [3] and [59].

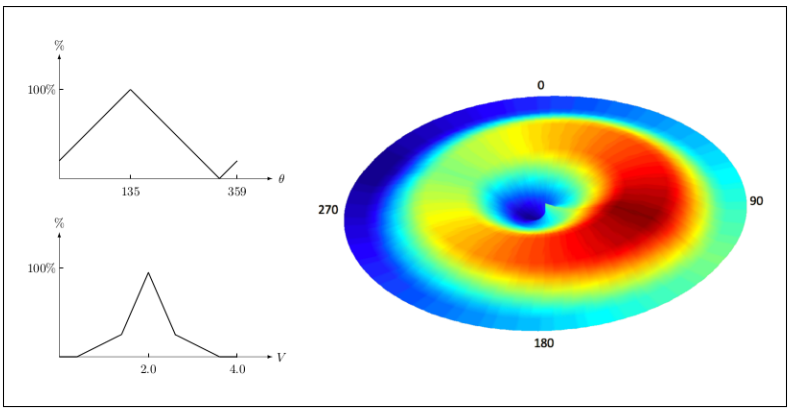

Figure 3.1: Top-left: an IvP function over the heading domain, with a peak at 135. Bottom-left: an IvP function over the speed domain, with a peak at 2. Right: an IvP function produced by the coupling of the two left functions, defined over the heading/speed domain [59].

3.1 Behaviour Class Hierarchy

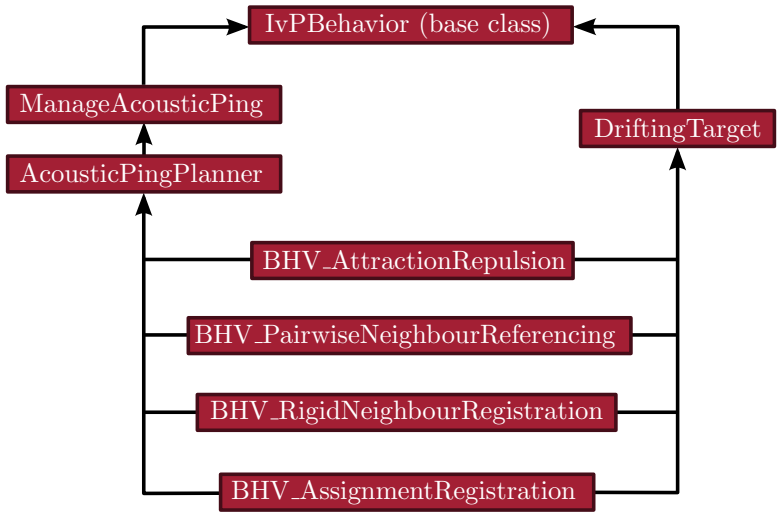

Figure 3.2: Behaviour class hierarchy for formation control behaviours.

Each of the four behaviours share some common functionality. Firstly, working under the expectation that the vehicle has existing low-level controllers for maintaining a desired speed, heading and depth, each behaviour must generate a target point for the vehicle such that it assumes a position in the formation so as to generate the desired pattern/lattice. This functionality is provided by the DriftingTarget behaviour. Secondly, each vehicle must be able to receive relative range/bearing measurements and possibly additional information (unique IDs) from neighbouring vehicles, interpret it, and provide it for use to determine the opti-

mal location for the target point. This functionality is provided by the ManageAcousticPing behaviour. Finally, some behaviours require a predetermined plan specifying the relative locations of each vehicle. This functionality is provided by the AcousticPingPlanner behaviour. Each of the four formation control behaviours inherit this common functionality from these behaviours, as illustrated in figure 3.2. Note that even though some behaviours do not require a predefined plan, for the sake of simplicity, all behaviours still inherit this functionality.

MOOS-IvP provides a base class called IvPBehavior upon which custom behaviours are built. IvPBehaviour has a number of overloadable functions related to the state of the behaviour, for example when the behaviour is idle, running, or complete, and the transitions between these states. Of these functions, the most important is the $OnRunState()$ function, which is the main loop of the behaviour and occurs when the behaviour is active and running. The objective function is generated within this loop in order to influence the trajectory of the vehicle. Unless otherwise stated, we describe the operation of each of my behaviours in terms of this $OnRunState()$ main loop.

3.2 AUV Navigation to Formation Target - DriftingTarget Behaviour

At the base of each of the formation control behaviours is the DriftingTarget behaviour. Ultimately, this behaviour is called upon to direct the vehicle towards a relative (x, y) target position by producing an IvP objective function over the heading/speed domain, and outputting a desired heading and speed in the target direction. The DriftingTarget behaviour is somewhat similar to two already existing MOOS-IvP behaviours - StationKeep and Waypoint, acting almost like a hybrid between the two. The objective of the DriftingTarget behaviour is to direct the vehicle to a relative (x, y) target position, and then once there, keep the vehicle within a certain $drifting\_radius$ around the target point. Once reaching the target position, the vehicle's speed is set to zero and it is left to drift freely. An illustration of this behaviour is shown in figure 3.3.

Table 3.1 lists behaviour-specific parameters which the user can specify, along with an explanatory description of each. Unless otherwise specified, the default values listed are those used during simulation and testing of my behaviours; note that although a moving average filter was implemented within the DriftingTarget behaviour, we do not use it in simulation (this is done by setting the length of the filter to 1).

Algorithm 1 shows pseudo-code for the main loop of the DriftingTarget behaviour. Note

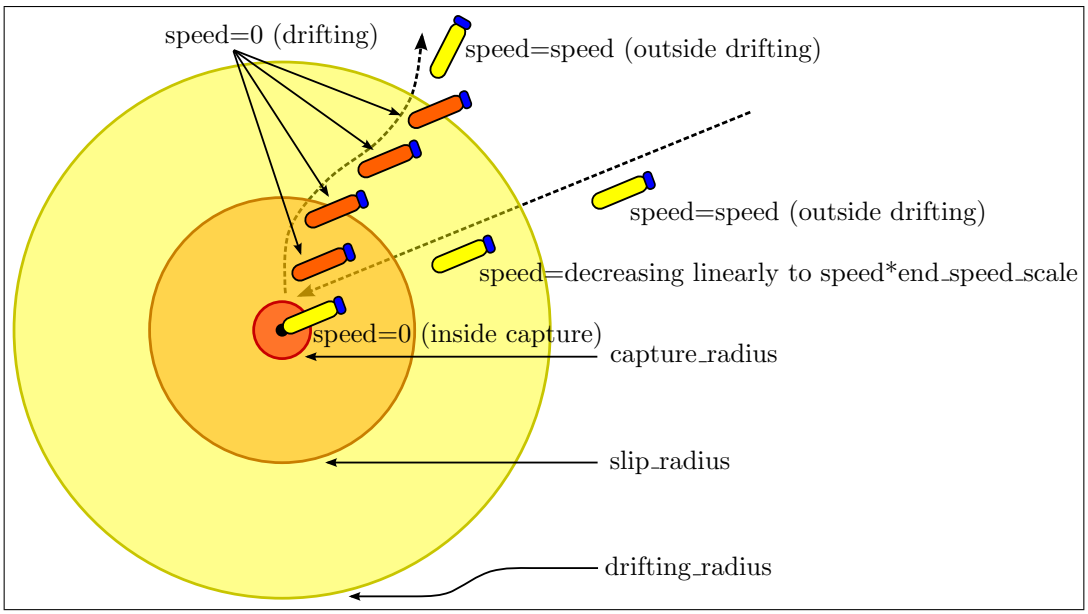

Figure 3.3: Illustration of the DriftingTarget behaviour.

that the target point is relative to the vehicle (i.e. the vehicle has a local frame of reference where it is always at the origin). Essentially, the behaviour works by directing the vehicle to a relative target point - when the vehicle is outside the *drifting_radius*, it outputs an objective function with a peak at a heading pointing towards the target and the user-specified speed; when the vehicle is between the *drifting_radius* and the *capture_radius*, the objective function reduces the speed linearly up to a multiplier of the user-specified speed; when the vehicle arrives at the target (either by entering the *capture_radius* or if the vehicle is within the *slip_radius* and it is determined that it is no longer moving toward the target), the objective function outputs a desired speed of zero, and no longer outputs a desired heading, allowing the vehicle to drift freely; once in this state, the behaviour does not influence the vehicle until it has moved outside the *drifting_radius*, whereby it once again directs the vehicle to the target point, and the cycle begins again.

One important thing to note is that if the position of the target is altered while the vehicle is in the drifting state, and the new position of the target is such that the vehicle is still within the *drifting_radius* of the new target, then the vehicle will stay in that state and continue drifting. As a result, this *drifting_radius* gives the vehicle a kind of 'slack' area, within which the formation control behaviour may alter the desired position of the vehicle, but the vehicle position is 'good enough' for the formation to be maintained. This functionality is illustrated in figure 3.4. In this way we are able to make a trade-off between the geometrical accuracy of the formation and power consumption - reducing the *drifting_radius* will make

the formation more accurate at the expense of motor use (and thus power consumption), since a smaller radius will make it more likely that the vehicle will move outside the $drifting\_radius$.

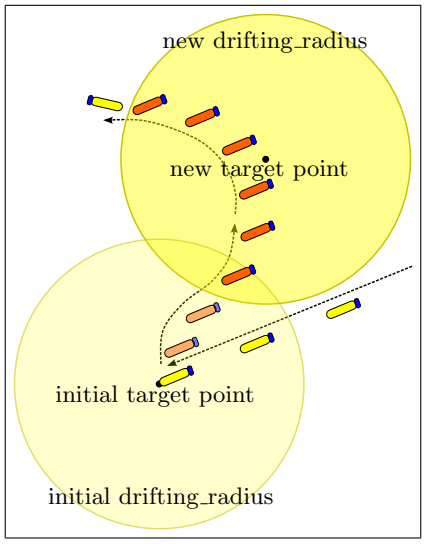

Figure 3.4: Illustration of the DriftingTarget behaviour - the AUV reaches the initial target and enters the drifting state; while in this state the target is altered such that the vehicle remains in the drifting radius of the new target; consequently, the AUV remains in the drifting state until it exits the drifting radius, whereby it turns to head toward the new target.

In order to specify the position of the target, a function called $addRelativeTargetTo-Filter(x, y)$ is available to behaviours that inherit from DriftingTarget. In essence, this function adds a given (x, y) position to the moving average filter, and sets the target position used by the main loop. This function is shown in algorithm 2.

Algorithm 1 DriftingTarget main loop.

procedure DRIFTINGTARGET::ONRUNSTATE()
  $dist\_to\_target \leftarrow \sqrt{target\_point.x^2 + target\_point.y^2}$
  $heading\_to\_target \leftarrow atan2(target\_point.y, target\_point.x)$
  if $dist\_to\_target > capture\_radius$ **then**
    if $dist\_to\_target < slip\_radius$ **then**
      if $dist\_to\_target \geq prev\_dist\_to\_target$ **then** ▷ vehicle is inside slip_radius and is no longer closing distance - vehicle has arrived
        $desired\_speed \leftarrow 0$
        $desired\_heading \leftarrow None$
      else
        $speed\_ratio \leftarrow (dist\_to\_target - capture\_radius) \div (drifting\_radius - capture\_radius)$
        $desired\_speed \leftarrow speed \times speed\_ratio + end\_speed\_scale \times (1 - speed\_ratio)$
        $desired\_heading \leftarrow heading\_to\_target$
      end if
    else if $dist\_to\_target < drifting\_radius$ **then**
      $speed\_ratio \leftarrow (dist\_to\_target - capture\_radius) \div (drifting\_radius - capture\_radius)$
      $desired\_speed \leftarrow speed \times speed\_ratio + end\_speed\_scale \times (1 - speed\_ratio)$
      $desired\_heading \leftarrow heading\_to\_target$
    else ▷ vehicle is outside drifting_radius
      $desired\_speed \leftarrow speed$
      $desired\_heading \leftarrow heading\_to\_target$
    end if
  else ▷ vehicle is inside capture_radius
    $desired\_speed \leftarrow 0$
    $desired\_heading \leftarrow None$
  end if
  $prev\_dist\_to\_target \leftarrow dist\_to\_target$
  $speed\_objective \leftarrow$ create IvPFunction for speed with peak at desired_speed
  $heading\_objective \leftarrow$ create IvPFunction for heading with peak at desired_heading
  $output\_objective \leftarrow$ create IvPFunction by coupling $speed\_objective$ and $heading\_objective$
end procedure

Algorithm 2 DriftingTarget function to set relative target.

```
1: procedure DriftingTarget::addRelativeTargetToFilter(x, y)
2:     target_list.append_last(point(x, y))
3:     if target_list.length() > targets_filter_size then
4:         target_list.delete_first()
5:     end if
6:     target_point ← sum(target_list.values()) ÷ target_list.length()
7: end procedure
```

Parameter	Description	Default
speed	The desired transit speed of the vehicle. (m/s)	1.0
capture_radius	The radius for satisfying the arrival at the target point. (m)	3.0
slip_radius	An 'outer' capture radius - if the vehicle is within this radius and is no longer closing the distance to the target point, it is declared to have arrived. (m)	10.0
drifting_radius	The radius within which the vehicle is allowed to drift freely once arriving at the target point. (m)	20.0
end_speed_scale	A speed multiplier - between the *drifting_radius* and the *capture_radius* the vehicle speed decreases linearly from *speed* to *speed* $\times$ *end_speed_scale*.	0.5
targets_filter_size	Length of the target point filter - the user can specify the length of a moving average filter for the target point.	1
display_filtered_target	Display a point in the $pMarineViewer$ visualizer for the filtered target point.	true
display_unfiltered_targets	Display points in the $pMarineViewer$ visualizer for all the points in the moving average filter.	false
display_radii	Display circles in the $pMarineViewer$ visualizer corresponding to the capture, slip and drifting radii.	true
display_statistics	Display statistics in the $pMarineViewer$ visualizer related to power consumption of the vehicle.	true
display_drift	Display a vector in the $pMarineViewer$ visualizer indicating the direction and magnitude of the ocean current acting on the vehicle.	true

Table 3.1: Parameters of the DriftingTarget behaviour.

3.3 Managing Acoustic Pings - ManageAcousticPing & AcousticPingPlanner Behaviours

In order for a formation control behaviour to function, it must be able to generate a target position for its vehicle such that the vehicle is directed to a desired point in the formation. To do so, acoustic pings containing information about the relative positions of neighbouring vehicles must be received and filtered (since bearing/range measurements have noise). This information is what allows a formation control behaviour to competently plan a target point for its vehicle, and the process of handling acoustic pings is performed by the following two behaviours - ManageAcousticPing and AcousticPingPlanner.

3.3.1 ManageAcousticPing Behaviour

The purpose of the ManageAcousticPing behaviour is simple - upon reception of a range and bearing measurement, this behaviour filters the measurement, converts it to Cartesian coordinates, and stores this point in memory as a neighbour position for use by inheriting behaviours. Table 3.2 lists behaviour-specific parameters which the user can specify, along with an explanatory description of each. Unless otherwise specified, the default values listed are those used during simulation and testing of my behaviours.

Algorithm 3 lists the pseudo-code for the two primary functions of the ManageAcousticPing behaviour which allow it to update neighbour positions from received pings. When a formation control behaviour of a vehicle receives a ping, it obtains a bearing measurement, a globally unique vehicle ID, and a time difference. It can then call the $updateNeighbourPosition(dt, \theta)$ function listed in algorithm 3 in order to localize the vehicle's neighbours. The basic process of this function is as follows - given the time difference (between transmission and reception of the ping) and the bearing of a ping, it first finds (in a storage dictionary) the corresponding vehicle which generated the ping (in our case this correspondence is performed explicitly by transmitting globally unique vehicle IDs, although for formation control behaviours that do not require IDs, this can be performed via a point-correspondence algorithm, thus reducing communication requirements); following this, it updates (or if a ping from that vehicle does not yet exist in the dictionary, adds) the position of the corresponding neighbour via a weighted moving average filter.

Filtering of the ping is performed using a weighted moving average filter, which operates as follows - when a new ping is received from a given neighbour, the $filterPing(ID, ping)$ function first retrieves a filter dictionary corresponding to that neighbour; it then adds a new entry to this dictionary with a key set to the current mission time, and the value set to the

received ping (time difference and bearing pair); following this, it iterates over all the key-value pairs in this dictionary, subtracting the key (which is the time at which that value was added) from the current time, and removing that pair from the dictionary if this time difference is greater than the user-specified filter time-out; the remaining values in the dictionary are weighted by a value inversely proportional to its age in the filter, summed, and averaged; finally, a polar to Cartesian conversion is applied to the range (where range is calculated as the time difference multiplied by the specified acoustic sound speed) and bearing, and this point is returned as the updated filtered position of the neighbour.

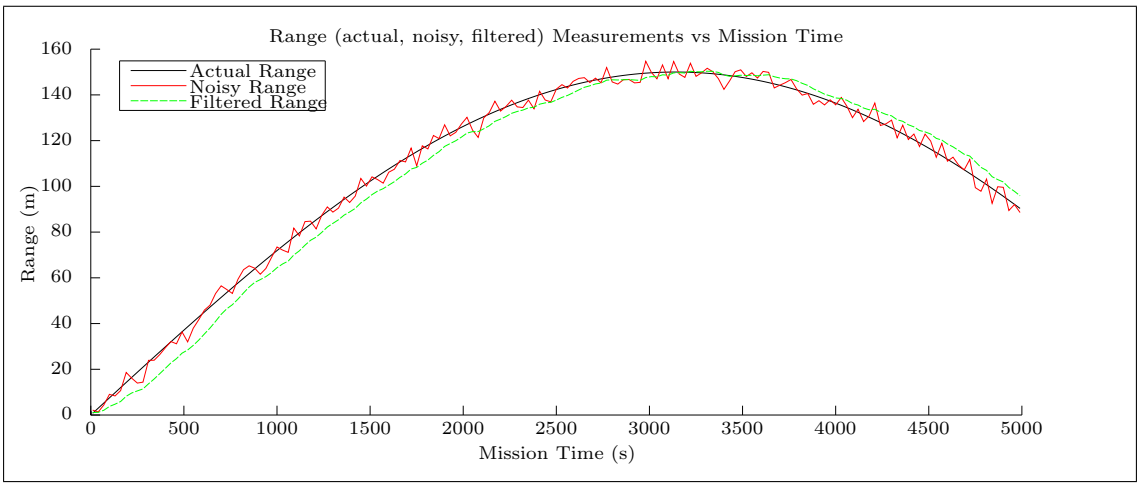

Figure 3.5: Plot comparing the actual, noisy (measured), and filtered range measurements; to highlight the filtering effect, the noise added to the actual range is Gaussian with a standard deviation of $3m$, and the filter time-out is set to $360s$; pings occur every $30s$.

This filtering approach does fairly well in removing the noise associated with bearing/range measurements, but, as expected, it adds a latency when comparing the actual position to the filtered position. This 'lag' can be seen in figure 3.5, along with the desired 'smoothing' effect. This associated latency results in inaccurate neighbour positioning primarily when either the neighbour or the vehicle is moving (the filtered position lags the actual position by a few meters). When both vehicles are still or drifting, then this filtering allows the vehicle to effectively remove noise from the measured positions of neighbouring vehicles.

Algorithm 3 ManageAcousticPing update function on ping reception.

procedure ManageAcousticPing::updateNeighbourPosition(dt, θ)
    $ping\_neigbour \leftarrow ping(dt, \theta)$
    $ping\_neigbour\_key \leftarrow$ find corresponding source ID of $ping\_neighbour$ ▷ in simulation this is done via explicit passing of vehicle IDs, but in reality can be done using some form of point-correspondence
    $neighbour\_pos\_dictionary[ping\_neighbour\_key] \leftarrow$ call $filterPing(ping\_neighbour\_key, ping\_neigbour)$
end procedure

procedure ManageAcousticPing::filterPing($ID, ping$)
    $filtered\_pos \leftarrow point(None, None)$
    $accum\_weights \leftarrow 0$
    $accum\_range \leftarrow 0$
    $accum\_sin\_bearing \leftarrow 0$
    $accum\_cos\_bearing \leftarrow 0$
    $curr\_time \leftarrow$ get current mission time
    $ping\_filter \leftarrow ping\_filter\_dictionary[ID]$
    $ping\_filter[curr\_time] \leftarrow ping$
    for key-value pair $(ping\_time, ping\_value)$ in $ping\_filter$ **do**
        if $(curr\_time - ping\_time) \geq ping\_filter\_timeout$ **then**
            $ping\_filter.delete(ping\_time)$
        else
            $range \leftarrow ping\_value.dt \times sound\_speed$
            $bearing \leftarrow ping\_value.\theta$
            $weight \leftarrow 1 - ((curr\_time - ping\_time) \div ping\_filter\_timeout)$
            $accum\_weights \leftarrow accum\_weights + weight$
            $accum\_range \leftarrow accum\_range + (weight \times range)$
            $accum\_sin\_bearing \leftarrow accum\_sin\_bearing + (weight \times \sin(bearing))$
            $accum\_cos\_bearing \leftarrow accum\_cos\_bearing + (weight \times \cos(bearing))$
        end if
    end for
    $range\_filtered \leftarrow accum\_range \div accum\_weights$
    $bearing\_filtered \leftarrow atan2(accum\_sin\_bearing, accum\_cos\_bearing)$
    $filtered\_pos \leftarrow point(x, y)$ from polar to Cartesian conversion of $(range\_filtered, bearing\_filtered)$
    return $(filtered\_pos, curr\_time)$
end procedure

procedure ManageAcousticPing::getNeighbourPos()
    return $neighbour\_pos\_dictionary$
end procedure

Parameter	Description	Default
ping_filter_timeout	Maximum ping age of the ping filter - the user can specify the maximum ping age of a weighted moving average filter; pings greater than this age are removed from the filter. (s)	120.0
contact_timeout	Maximum ping age of a neighbour - pings from neighbours remain in memory until this age (this value is not used by ManageAcousticPing, but is available for inheriting behaviours for neighbour management). (s)	120.0
sound_speed	The acoustic sound speed. (m/s)	1500.0
display_filtered_contact	Display a point in the $pMarineViewer$ visualizer of the filtered neighbour position.	false
display_unfiltered_contact	Display points in the $pMarineViewer$ visualizer for all the neighbour positions in the filter.	false

Table 3.2: Parameters of the ManageAcousticPing behaviour.

3.3.2 AcousticPingPlanner Behaviour

Since some of my formation control behaviours require a user-specified formation, some way of providing this information to behaviours had to be implemented. This is the job of the AcousticPingPlanner behaviour - essentially this behaviour provides the user with a method of specifying the desired geometry of the formation, and provides inheriting behaviours with a data structure holding this user-specified geometry. Table 3.3 lists behaviour-specific parameters which the user can specify, along with an explanatory description of each. As you can see, there is only one parameter, called $node\_offsets$. This parameter should be specified by the user multiple times, with one value for each vehicle in the formation, using the following string format - "$name = vehiclename, x = x\_position, y = y\_position$". $vehiclename$ specifies the globally unique vehicle ID (unused for some of my formation control behaviours), and $x\_position$ and $y\_position$ specify the global coordinates of the desired vehicle position in the formation. It is important that this collection of specified positions is consistent across all vehicles in the swarm (i.e. all vehicles share the same plan).

Parameter	Description	Default
node_offsets	A string allowing the user to specify the ID and position of a vehicle in a desired formation; this string must have the following format: "$name = vehiclename, x = x\_position, y = y\_position$"	None

Table 3.3: Parameters of the AcousticPingPlanner behaviour.

The AcousticPingPlanner behaviour does one thing - it reads in the specified $node\_offsets$ values, subtracts all other $node\_offsets$ values from the $node\_offsets$ value corresponding to the vehicle on which it is running, and stores this relative plan as a dictionary for use by inheriting behaviours. Pseudo-code for this is shown in algorithm 4.

The ManageAcousticPing and AcousticPingPlanner behaviours allow inheriting behaviours to access a dictionary of neighbour positions calculated from measured pings, and a dictionary of desired neighbour positions that are user-specified. A function is provided by both the ManageAcousticPing and AcousticPingPlanner behaviours in order to access these dictionaries - $getNeighbourPos()$ and $getNeighbourPlan()$, as shown at the ends of algorithm 3 and algorithm 4 respectively.

Algorithm 4 AcousticPingPlanner function to read user-specified formation plan.

```
1: procedure AcousticPingPlanner::setFormationPlan()
2:     read in all node_offsets and add to dictionary: neighbour_pos_plan[vehiclename] ←
   point(x_position, y_position)
3:     for all points in neighbour_pos_plan, subtract point corresponding to ownship position
4:     remove point corresponding to ownship position from neighbour_pos_plan
5: end procedure
6:
7: procedure AcousticPingPlanner::getNeighbourPlan()
8:     return neighbour_pos_plan
9: end procedure
```

3.4 Formation Control Algorithm 1 - BHV_AttractionRepulsion Behaviour

The first of my formation control behaviours was inspired by many of the physics-based approaches described in section 2.3.1, and in particular, utilizes ideas from Pinciroli et al. [30], Gazi and Passino [31], and Lee and Chong [49], but with a couple of novel differences. The idea behaind the BHV_AttractionRepulsion behaviour is rooted in the behaviour of atoms and inter-atomic forces. When thinking about the behaviour of atoms, it is reasonable to assume that atoms will organize themselves so as to minimize the total potential energy of the system - intuitively, this organization should produce a hexagonal lattice, as atoms pack themselves together much like balls in a box. Indeed, it is conjectured that the minimum energy state will produce a hexagonal lattice, with models such as the bubble raft [60] being used to describe the behaviour of crystalline materials. With this in mind, this first behaviour uses this inter-atomic artificial force approach to construct a hexagonal lattice from our swarm of AUVs, where each AUV acts like an 'atom' in a crystalline material.

Inheriting functionality from the ManageAcousticPing behaviour, BHV_AttractionRepulsion gains access to the positions of neighbouring vehicles. Table 3.4 reveals that this behaviour really has only one notable parameter - the *separation_distance*, which allows the user to specify the distance between vehicles in the hexagonal lattice (this can be thought of as the 'size' of the atoms). In theory, BHV_AttractionRepulsion does not require vehicles to exchange globally unique IDs, and does not require a user-specified plan; vehicles only need to know the positions of their neighbours, and the structure of the lattice formation is constructed in an emergent manner.

Parameter	**Description**	**Default**
contact_rangeout	Backup distance above which neighbour pings are ignored. (m)	650.0
separation_distance	Distance of lattice node separation. (m)	300.0
display_neighbour_hull	Display a polygon in the $pMarineViewer$ visualizer showing the convex hull of the 3 vehicles involved in the behaviour.	true

Table 3.4: Important parameters of the BHV_AttractionRepulsion behaviour.

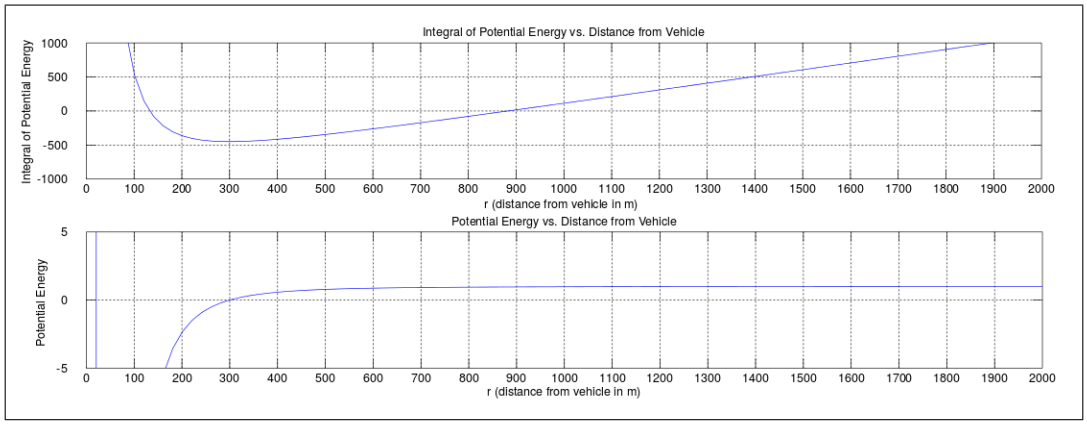

Figure 3.6: Top: Integral of attraction/repulsion potential function. Bottom: Potential function with repulsion as negative potential and attraction as positive potential; zero potential occurs at $300m$.

At the core of the BHV_AttractionRepulsion behaviour is an attraction/repulsion potential function which enables the creation of the desired lattice, similar to the potential functions described in the work of Pinciroli et al. [30], and Gazi and Passino [31]. These potential functions attract and repel neighbouring vehicles such that they stabilize at a desired distance from one another. Pinciroli et al. [30] use the well known Lennard-Jones potential function; however, a critical limitation of this function is that the attraction term decays as a power

law, meaning that the potential will fail to attract neighbours at large distances, or will only attract neighbours slowly. Gazi and Passino [31] present three different potential functions, which allow them to circumvent this issue. In a similar manner, I select a potential function with unbounded repulsion and constant attraction, listed in equation 3.2 as $\frac{df(r)}{dr}$, and shown in figure 3.6 as the lower plot. In this figure the desired separation distance is $300m$, the attractive portion of the function is constant at 1, and the repulsive portion has the form $-\frac{2a^2}{r^3}$ where a is a constant and r is the distance from the vehicle. It can be seen that the potential is zero at the desired distance of $300m$. My approach differs from those described in section 2.3.1 in one important respect - instead of directly using the potential as a means of controlling the vehicle to the zero potential distance, I instead use the integral of the potential, and perform direct optimization on this integral function. Equation 3.1 lists this function, $f(r)$, and it is illustrated at the top of figure 3.6. It is apparent that the minimum of $f(r)$ exists at the desired separation distance. $f(r)$ is given by:

$$f(r) = (\frac{a}{r})^2 + (r - b) \tag{3.1}$$

The derivative of $f(r)$ yields the potential function as described previously. Solving for the zero potential allows us to determine the value of a in terms of the desired separation distance, s, as follows:

$$\begin{aligned} \frac{df(r)}{dr} &= 1 - \frac{2a^2}{r^3} \\ \text{let } \frac{df(r)}{dr} &= 0 \\ 1 - \frac{2a^2}{r^3} &= 0 \\ a^2 &= \frac{r^3}{2} \\ \text{let} \qquad a &= \sqrt{\frac{s^3}{2}} \\ \text{let} \qquad b &= 3 \cdot s \end{aligned} \tag{3.2}$$

The value of b was selected to be $3 \cdot s$ for convenience. Substituting a and b back into equation 3.1 gives us $f(r)$ in terms of our desired separation distance, s, as follows:

$$
\begin{aligned}
f(r) &= (\frac{s^3}{2 \cdot r^2}) + (r - 3 \cdot s) \\
f(r := s) &= (\frac{s^3}{2 \cdot s^2}) + (s - 3 \cdot s) \\
&= -1.5 \cdot s
\end{aligned}
\tag{3.3}
$$

The minimum occurs when $r := s$, with a value of $-1.5 \cdot s$. Finally, to convert $f(r)$ to a 2D plane, we substitute r with $\sqrt{x^2 + y^2}$:

$$
f(x, y) = (\frac{s^3}{2 \cdot (\sqrt{x^2 + y^2})^2}) + ((\sqrt{x^2 + y^2}) - 3 \cdot s) \tag{3.4}
$$

$f(x, y)$ gives us a surface over which to minimize; when multiple instances of $f(x, y)$ are centred over the positions of a vehicle's neighbours, then performing a minimization over this sum of surfaces yields a locally optimal solution for the position of the vehicle, such that the vehicle attempts to maintain the desired separation distance. Thus, the surface over which we wish to minimize is the cost function given by:

$$
C(x, y) = \sum_{i=1}^{n} (\frac{s^3}{2 \cdot (\sqrt{(x - x_i)^2 + (y - y_i)^2})^2}) + ((\sqrt{(x - x_i)^2 + (y - y_i)^2}) - 3 \cdot s) \tag{3.5}
$$

Where (x_i, y_i) are the positions for n neighbours. The minimization that we perform is therefore given by:

$$
(x^*, y^*) = \underset{(x,y)}{\operatorname{argmin}} \sum_{i=1}^{n} (\frac{s^3}{2 \cdot (\sqrt{(x - x_i)^2 + (y - y_i)^2})^2}) + ((\sqrt{(x - x_i)^2 + (y - y_i)^2}) - 3 \cdot s) \tag{3.6}
$$

For the purposes of visualizing this surface, figure 3.7 illustrates an example where a vehicle has two neighbours; one at $(-280, -81)$, and the other at $(-25, -193)$. The output of minimization gives an optimal location of $(-45, 106)$; moving the vehicle to this position would result in it being $300m$ from both it's neighbours, as desired. Note that although the surface has two minima, reflected along the line defined by the positions of the vehicle's neighbours, the minimization is initialized at $(0, 0)$, resulting in the selection of the closer minimum via gradient descent.

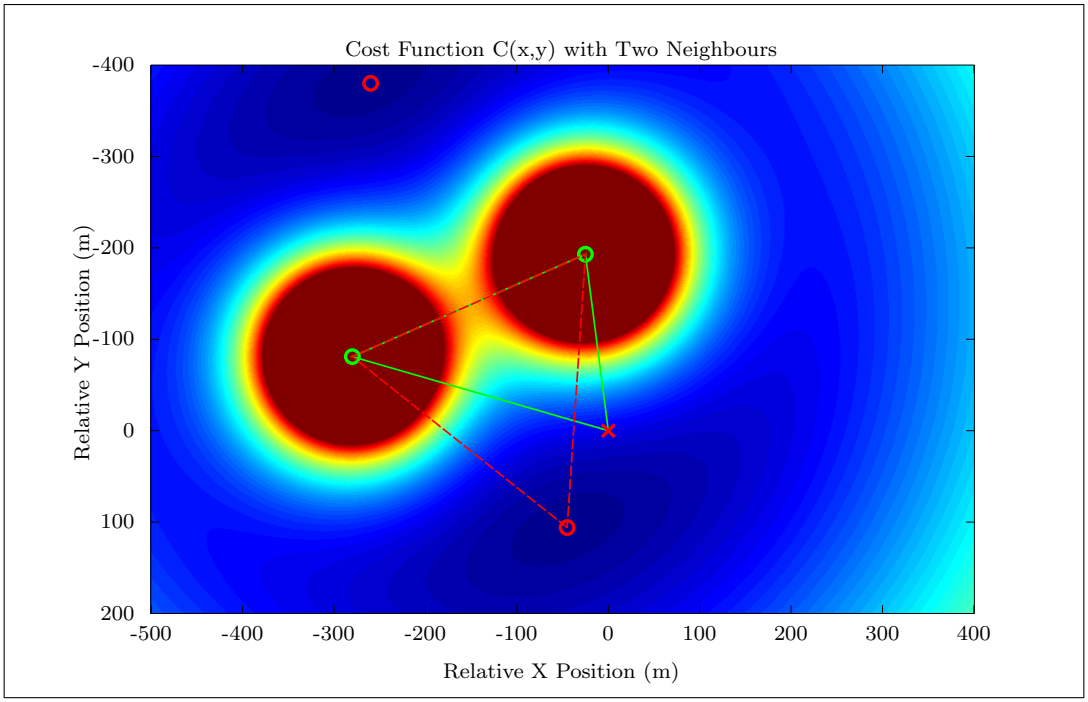

Figure 3.7: Cost function surface in vehicle's local reference frame, with two neighbour vehicles, one at $(-280, -81)$ and the other at $(-25, -193)$; Two minima are apparent, and the closer one (at $(-45, 106)$) is chosen as the target point of the vehicle; this target point results in the vehicle being at the desired separation distance of $300m$ from both its neighbours.

Unfortunately, simply minimizing over the summed cost functions of all a vehicle's neighbours (within the communications radius) does not result in the desired lattice formation. This can be explained by the fact that the summation of cost surfaces (analogous to attractive/repulsive forces) alters the minima depending on the number of neighbours and their positions with respect to the vehicle; vehicles at the outer edge of the formation tend to distribute themselves further apart, since less force is acting upon them, while vehicles surrounded by neighbours tend to be distributed in a more concentrated manner. This issue occurs in many physics-based approaches, and is usually solved by careful selection of which neighbours are used to influence the vehicle. In my approach, the vehicle selects only two neighbours, using the same approach as that of Lee and Chong [49] - the first neighbour is selected as the closest neighbour, and the second is selected such that the sum of the edge lengths of the triangle created by the vehicle and its two neighbours is minimal. However, selecting only two neighbours results in a new problem - the formation can 'fracture' and break apart. This occurs when two subsets of vehicles only select neighbours amongst their respective subsets; an illustration of this is shown in figure 3.8. In an attempt to prevent this, an additional term is added to the cost function which attracts the vehicle to the centroid point of all its neighbours. Thus, our final cost function is of the form:

$$C(x,y) = \sum_{(x_i,y_i)\in N_s} \left(\frac{s^3}{2\cdot(\sqrt{(x-x_i)^2+(y-y_i)^2})^2}\right) + \left((\sqrt{(x-x_i)^2+(y-y_i)^2}) - 3\cdot s\right) + 1e^{-5}\cdot\left(\sqrt{(x-\frac{\sum_{j=1}^{n}(x_j)}{n})^2+(y-\frac{\sum_{j=1}^{n}(y_j)}{n})^2}\right)^2 \quad (3.7)$$

Where N_s is the set of positions of the two selected neighbours, n is the total number of neighbours in the vehicle's communications radius, and (x_j, y_j) are the positions of these neighbours. This cost function is minimized in my implementation using the open source non-linear optimization library, NLopt [61].

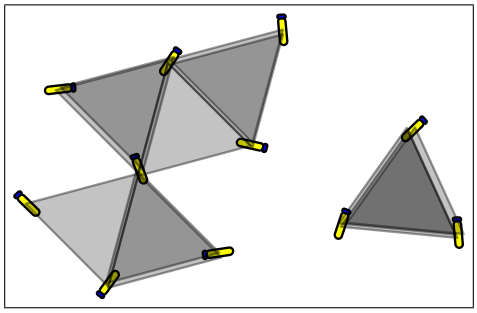

Figure 3.8: Illustration of formation 'fracturing' - grey triangles indicate a vehicle's neighbour selection triangle; here the 3 vehicles on the right have selected their two neighbours only amongst themselves, and similarly, the 8 left side vehicles have selected their two neighbours amongst themselves, resulting in the 3 vehicles drifting away from the majority.

Finally, given this insight of how BHV_AttractionRepulsion operates, we can summarize the approach in algorithm 5. This behaviour runs on each vehicle, in a distributed fashion. We have the following definitions - (x_{os}, y_{os}) denotes the ownship position, which is always set to $(0,0)$, as each vehicle operates in a local coordinate frame with itself at the origin; CR is the ownship sensing/communications radius, a circular area around the vehicle within which it can determine its neighbours positions (performed by the ManageAcousticPing behaviour); $N = \{(x_1,y_1),(x_2,y_2),...(x_n,y_n)\}$ is the set of neighbour positions within CR that the vehicle senses; finally $N_s = \{(x_i,y_i),(x_j,y_j)\}$ is the set of the two neighbour positions selected for the optimization procedure. Finally, the behaviour running on a swarm of 20 AUVs in the MOOS-IvP Simulation Test-Bed is shown in figure 3.9.

Algorithm 5 BHV_AttractionRepulsion main loop.

procedure BHV_AttractionRepulsion::onRunState()
  $N \leftarrow$ call $ManageAcousticPing :: getNeighbourPos()$
  $neighbour\_centroid \leftarrow (0, 0)$
  $closest\_neighbour \leftarrow None$
  $closest\_dist \leftarrow \infty$
  for $(x, y) \in N$ **do**
    if $\sqrt{x^2 + y^2} \leq contact\_rangeout$ **then**
      $neighbour\_centroid \leftarrow neighbour\_centroid + (x, y)$
      if $\sqrt{x^2 + y^2} < closest\_dist$ **then**
        $closest\_neighbour \leftarrow (x, y)$
      end if
    end if
  end for
  $neighbour\_centroid \leftarrow neighbour\_centroid \div N.size()$
  $N \leftarrow N \setminus closest\_neighbour$
  $second\_neighbour \leftarrow None$
  $shortest\_tri\_length \leftarrow \infty$
  for $(x, y) \in N$ **do**
    if $\sqrt{x^2 + y^2} \leq contact\_rangeout$ **then**
      if $\sqrt{(closest\_neighbour.x - x)^2 + (closest\_neighbour.y - y)^2} + \sqrt{x^2 + y^2} < shortest\_tri\_length$ **then**
        $second\_neighbour \leftarrow (x, y)$
      end if
    end if
  end for
  $N_s \leftarrow ((closest\_neighbour.x, closest\_neighbour.y),$
    $(second\_neighbour.x, second\_neighbour.y))$
  $cost \leftarrow \sum_{(x_i,y_i)\in N_s}(\frac{separation\_dist^3}{2 \cdot (\sqrt{(x - x_i)^2 + (y - y_i)^2})^2}) + ((\sqrt{(x - x_i)^2 + (y - y_i)^2}) - 3 \cdot separation\_dist) + 1e^{-5} \cdot (\sqrt{(x - neighbour\_centroid.x)^2 + (y - neighbour\_centroid.y)^2})^2$
  $(x^*, y^*) \leftarrow (0, 0)$
  $(x^*, y^*) \leftarrow \underset{(x,y)}{\operatorname{argmin}}(cost)$
  call $DriftingTarget :: addRelativeTargetToFilter(x^*, y^*)$
end procedure

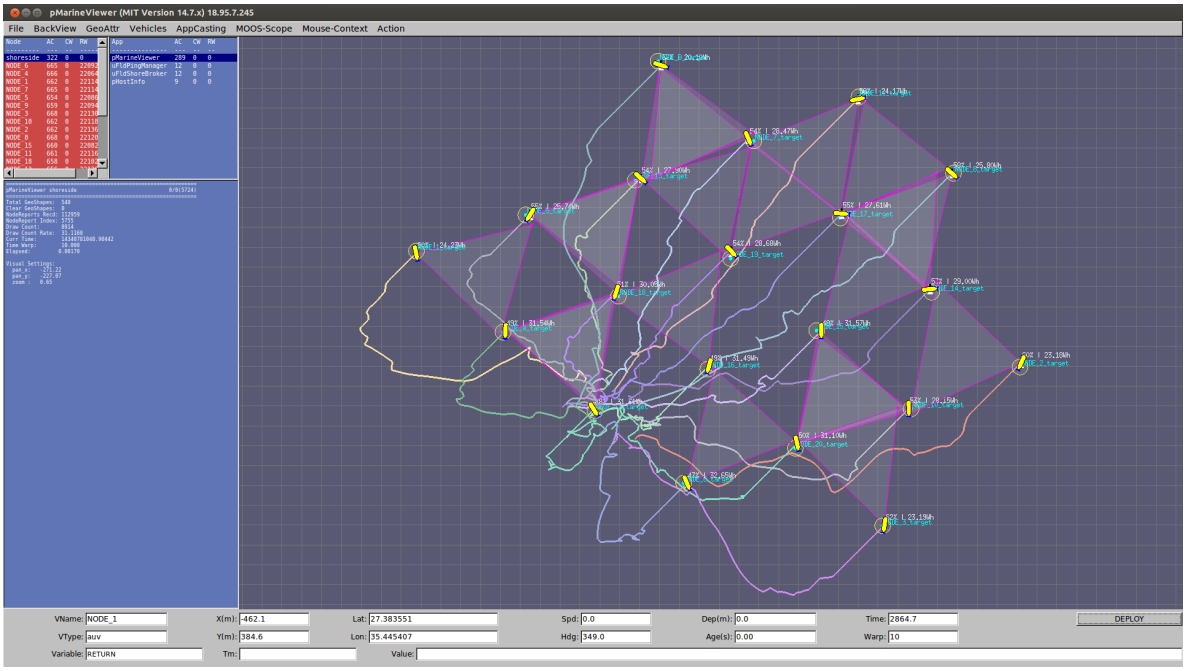

Figure 3.9: BHV_AttractionRepulsion running on a swarm of 20 AUVs in the MOOS-IvP Simulation Test-Bed.

3.5 Formation Control Algorithm 2 - BHV_PairwiseNeighbourReferencing Behaviour

The second of my formation control behaviours was motivated by a number of shortcomings observed in the BHV_AttractionRepulsion behaviour - BHV_AttractionRepulsion does not allow us to form lattice or pattern formations other than the hexagonal lattice; the movement of vehicles tend to be erratic and chaotic during formation establishment; and lattices have the tendency to form with 'defects'. As a result of these limitations, a new behaviour, called BHV_PairwiseNeighbourReferencing, was developed to observe how formation control could be improved with the use of additional information in the form of globally unique vehicle IDs and a user-specified formation plan. Considering the desired properties of the formation control behaviour - we wish it to be rotationally and transationally invariant (i.e. the formation should not operate within a predetermined frame of reference, and should be free to rotate and move under the influence of external forces), I came up with a simple formation control strategy based on trigonometric rules.

Inheriting functionality from the ManageAcousticPing and AcousticPingPlanner behaviours, BHV_PairwiseNeighbourReferencing gains access to the actual and desired relative positions of neighbouring vehicles. Table 3.5 reveals that this behaviour does not really have

any notable parameters. BHV_PairwiseNeighbourReferencing uses the parameters specified by AcousticPingPlanner to determine the globally unique IDs for vehicles, and to obtain the desired geometric plan for the formation (see section 3.3.2 for the AcousticPingPlanner parameters).

Parameter	**Description**	**Default**
contact_rangeout	Backup distance above which neighbour pings are ignored. (m)	650.0
display_unaveraged_targets	Display points in the $pMarineViewer$ visualizer corresponding to the target points from all neighbour pairs.	false
display_unaveraged_hull	Display a polygon in the $pMarineViewer$ visualizer showing the convex hull of the target points from all neighbour pairs.	false

Table 3.5: Important parameters of the BHV_PairwiseNeighbourReferencing behaviour.

The basic idea behind BHV_PairwiseNeighbourReferencing is this - given a pair of neighbours and a predefined plan that details how we wish the vehicle to be placed in reference to this pair, we can use simple trigonometric rules to calculate where the vehicle should actually be positioned in reference to this pair; essentially, the neighbour pair provides us with a reference frame in which we can perform our geometric calculations. When there are n neighbours within the vehicle's communications radius, then the number of possible pairs is $\frac{n(n-1)}{2}$. We consider each possible pair, look up this pair in our plan to determine how our vehicle should be placed in reference to it, then calculate the corresponding position given the actual positions of the neighbour pair. Obviously, this provides us with $\frac{n(n-1)}{2}$ target points for our vehicle - to collapse this group into a single point, the weighted centroid of the group is used as the target point of our vehicle, and the vehicle is driven to this position. The basic idea is illustrated in figure 3.10 for a single neighbour pair, and in figure 3.11 for three neighbour pairs.

We now provide a more detailed explanation of the steps of the BHV_PairwiseNeighbourReferencing algorithm. Given the set of n neighbour positions $N = \{(x_1, y_1), (x_2, y_2), ... (x_n, y_n)\}$ detected within the vehicle's communications radius CR, the behaviour loops through all possible pairs of neighbours and performs the following - for each neighbour pair, it first looks up in the formation plan the corresponding relative positions of the pair; it then calculates the distance from the vehicle to the midpoint of the pair in the plan, as well as the angle from this midpoint to the vehicle with respect to the line connecting the pair (as shown at leftmost in figure 3.11); it then uses this calculated distance and angle to project a target point from the midpoint of the actual positions of the pair (as shown in centre of figure 3.11); after performing these calculations for all pairs of neighbours, it then calcu-

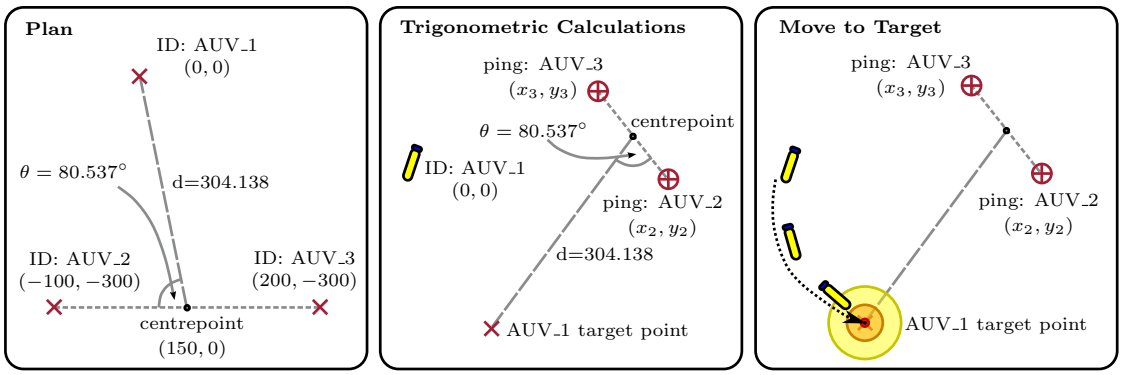

Figure 3.10: Illustration of the geometric principles behind BHV_PairwiseNeighbourReferencing running on AUV_1 for a single neighbour pair (AUV_2, AUV_3).

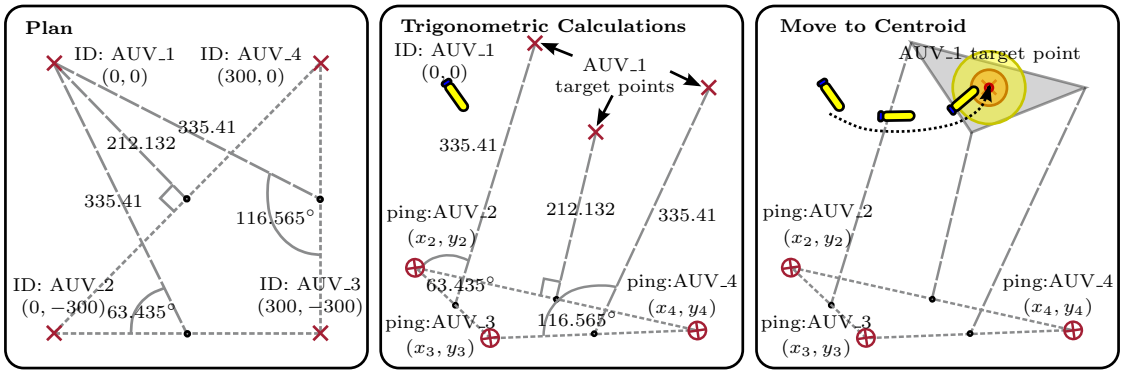

Figure 3.11: Illustration of the geometric principles behind BHV_PairwiseNeighbourReferencing running on AUV_1 for three neighbour pairs (AUV_2, AUV_3), (AUV_3, AUV_4) and (AUV_2, AUV_4).

lates the weighted centroid of all the target points generated by the neighbour pairs (where the weight is inversely proportional to the distance between the midpoint of the pair and the vehicle); finally, the vehicle is directed to this centroid point (as shown at rightmost in figure 3.11). This approach is summarized in algorithm 6, and as with all my behaviours, BHV_PairwiseNeighbourReferencing runs on each vehicle in a distributed fashion. We denote the set of planned vehicle positions as $N_p = \{(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ...(x_{pm}, y_{pm})\}$ for a swarm of m vehicles, where (x_{p1}, y_{p1}) corresponds to the planned position of the neighbour with position (x_1, y_1). The behaviour running on a swarm of 22 AUVs in the MOOS-IvP Simulation Test-Bed is shown in figure 3.12. Because the user can specify a desired formation plan and there are globally unique IDs, an advantage of this approach is that formations of arbitrary shapes can be constructed. In this case, the letters 'NR' are formed by the 22 AUVs.

Algorithm 6 BHV_PairwiseNeighbourReferencing main loop.

procedure BHV_PairwiseNeighbourReferencing::onRunState()
  $N \leftarrow$ call $ManageAcousticPing :: getNeighbourPos()$
  $N_p \leftarrow$ call $AcousticPingPlanner :: getNeighbourPlan()$
  $target \leftarrow (0, 0)$
  $weight \leftarrow 0$
  for $(x_i, y_i) \in N$ **do**
    for $(x_j, y_j) \in N \setminus (x_i, y_i)$ **do**
      $midpoint\_plan \leftarrow ((x_{pi}, y_{pi}) + (x_{pj}, y_{pj})) \div 2$
      $dist\_plan \leftarrow \sqrt{midpoint\_plan.x^2 + midpoint\_plan.y^2}$
      $angle\_plan \leftarrow atan2(midpoint\_plan.y, midpoint\_plan.x) - atan2(y_{pj} - y_{pi}, x_{pj} - x_{pi})$
      $midpoint\_actual \leftarrow ((x_i, y_i) + (x_j, y_j)) \div 2$
      if $\sqrt{midpoint\_actual.x^2 + midpoint\_actual.y^2} > contact\_rangeout$ **then**
        continue
      end if
      $subtarget \leftarrow midpoint\_actual + (dist\_plan \cdot \sin(angle\_plan), dist\_plan \cdot \cos(angle\_plan))$
      $subtarget \leftarrow (1 - \sqrt{midpoint\_actual.x^2 + midpoint\_actual.y^2} \div contact\_rangeout) \cdot subtarget$
      $target \leftarrow target + subtarget$
      $weight \leftarrow weight + (1 - \sqrt{midpoint\_actual.x^2 + midpoint\_actual.y^2} \div contact\_rangeout)$
    end for
  end for
  $target \leftarrow target \div weight$
  call $DriftingTarget :: addRelativeTargetToFilter(target.x, target.y)$
end procedure

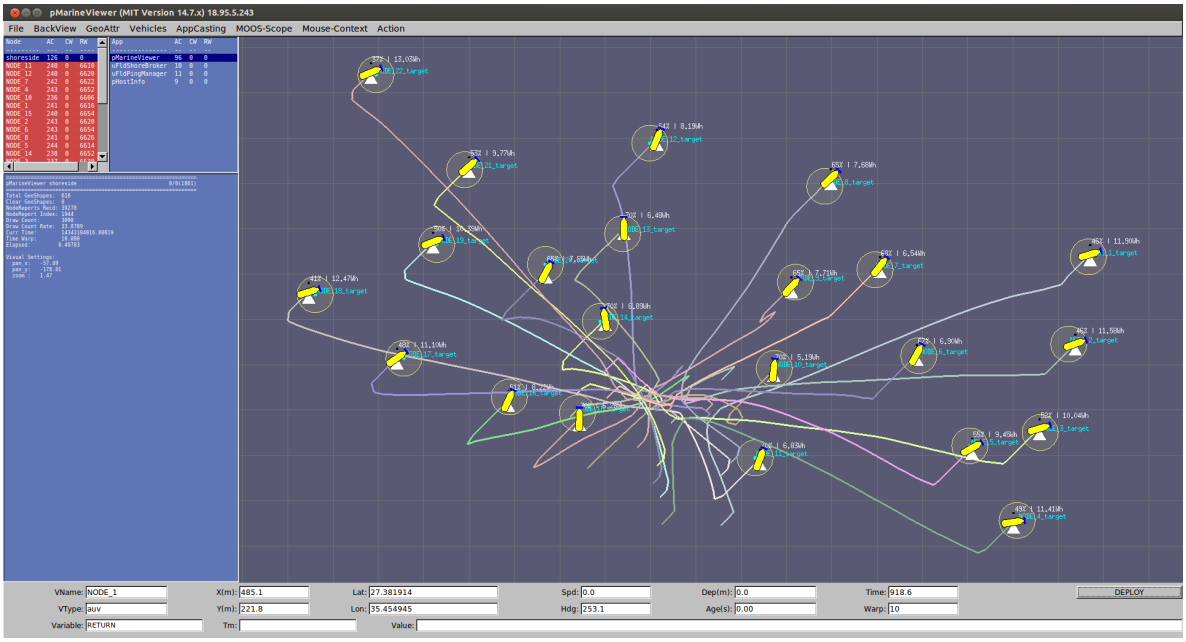

Figure 3.12: BHV_PairwiseNeighbourReferencing running on a swarm of 22 AUVs in the MOOS-IvP Simulation Test-Bed (the formed letters 'NR' are upside-down to the viewer).

3.6 Formation Control Algorithm 3 - BHV_RigidNeighbourRegistration Behaviour

Although the relatively simple approach of BHV_PairwiseNeighbourReferencing proved to be remarkably successful in constructing and maintaining a desired formation, the development of another behaviour was motivated by a desire to improve on efficiency. When running BHV_PairwiseNeighbourReferencing, it is noticeable that many of the vehicles tend to move away and then re-approach their final positions in the formation during construction, in a kind of 'arc-shaped' manoeuvre. This manoeuvre needlessly consumes energy; it would be more desirable if vehicles approached their positions in the formation directly along a straight-line path. With this motivation in mind, I developed the BHV_RigidNeighbourRegistration behaviour. Inheriting functionality from the ManageAcousticPing and AcousticPingPlanner behaviours, BHV_RigidNeighbourRegistration gains access to the actual and desired relative positions of neighbouring vehicles. BHV_RigidNeighbourRegistration uses the parameters specified by AcousticPingPlanner to determine the globally unique IDs for vehicles, and to obtain the desired geometric plan for the formation (see section 3.3.2 for the AcousticPing-Planner parameters).

BHV_RigidNeighbourRegistration was inspired, in part, by the iterative closest point (ICP) algorithm of Besl and McKay [62]. The ICP algorithm is widely used to align two

Parameter	Description	Default
contact_rangeout	Backup distance above which neighbour pings are ignored. (m)	650.0
ownship_weight	The weighting w_i associated with the ownship position in the plan (0,0) - a lower weight means that the plan will conform more closely to neighbours.	1.0
display_rigid_registration_points	Display points in the $pMarineViewer$ visualizer corresponding to the plan positions included in the rigid transformation calculation.	false
display_rigid_registration_hull	Display a polygon in the $pMarineViewer$ visualizer showing the convex hull of the plan positions included in the rigid transformation calculation.	false

Table 3.6: Important parameters of the BHV_RigidNeighbourRegistration behaviour.

point clouds without knowledge of the individual point correspondences between the two clouds - it can be described by three steps; firstly, correspondences are computed between the two point clouds, using some distance metric; secondly, the optimal rigid transformation is calculated between the corresponding points using a mean squared error cost function; finally, one point cloud is transformed to the other, and the process is repeated. Now, considering the problem we are trying to address, it is quite apparent that the second step of this algorithm is highly suited to our needs. In our case, given a user-specified formation plan and the use of globally unique IDs, the correspondences between the desired formation positions and the actual vehicle positions are explicitly known; this leaves us with simply calculating the optimal rigid transformation between the two, a problem known as the orthogonal Procrustes or the rigid point set registration problem.

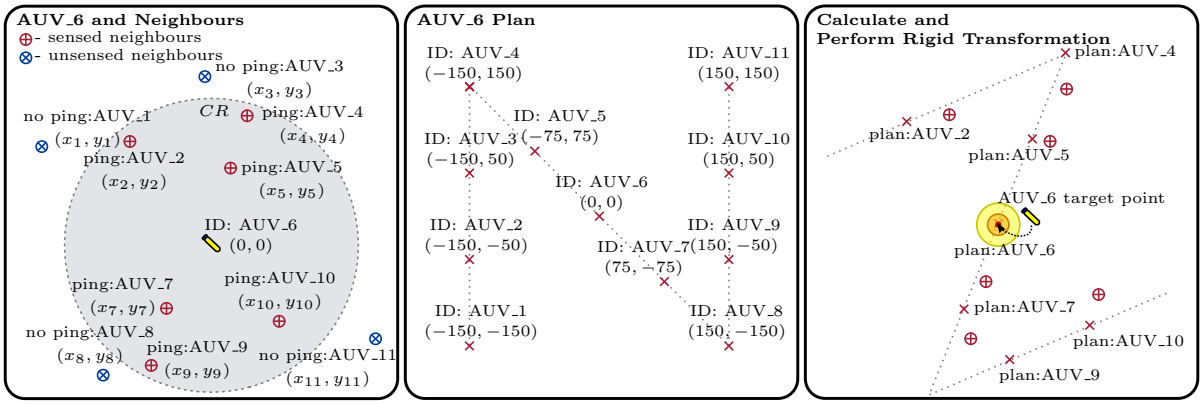

Figure 3.13: Illustration of the operational principles of BHV_RigidNeighbourRegistration; for the neighbours within the vehicles CR, the corresponding points from the plan are rotated and translated to best fit the actual neighbour positions (the CR is reduced for illustrative purposes).

The operational principles of BHV_RigidNeighbourRegistration are illustrated in figure 3.13, and the behaviour operates in a distributed manner in the following way - given the set of relative neighbour positions plus the ownship position $N = \{(x_{os}, y_{os}) = (0,0), (x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, with n detected neighbours within the vehicle's communications radius CR, the behaviour first looks up their corresponding relative positions in the user-defined formation plan, given by $N_p = \{(x_{p_{os}}, y_{p_{os}}) = (0,0), (x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ...(x_{pn}, y_{pn})\}$, where (x_{p1}, y_{p1}) corresponds to the planned position of the neighbour with position (x_1, y_1); given these two sets of points with known correspondences, it then calculates the rigid transformation that best aligns N_p with N in the least squares sense, as follows, where R is a rotation matrix and $\vec{t}$ is a translation vector:

$$(R, \vec{t}) = \underset{R,\vec{t}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i ||(R \begin{bmatrix} x_{pi} \\ y_{pi} \end{bmatrix} + \vec{t}) - \begin{bmatrix} x_i \\ y_i \end{bmatrix} ||^2 \tag{3.8}$$

Where w_i is a weighting associated with each point pair. This minimization problem has a closed-form solution using a singular value decomposition based method, the mathematical details of which can be found in Sorkine's notes [63]. The steps of the closed-form solution are:

1. Compute the weighted centroid of both point sets:

$$\begin{bmatrix} \widetilde{x_p} & \widetilde{y_p} \end{bmatrix} = \frac{\sum_{i=1}^{n} w_i \begin{bmatrix} x_{pi} & y_{pi} \end{bmatrix}}{\sum_{i=1}^{n} w_i} \quad , \qquad \begin{bmatrix} \widetilde{x} & \widetilde{y} \end{bmatrix} = \frac{\sum_{i=1}^{n} w_i \begin{bmatrix} x_i & y_i \end{bmatrix}}{\sum_{i=1}^{n} w_i}$$

2. Translate each point set by their centroids so that each is centred at their origins:

$$I = \begin{bmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \\ \vdots & \\ I_{n1} & I_{n2} \end{bmatrix} := \begin{bmatrix} x_{p1} & y_{p1} \\ x_{p2} & y_{p2} \\ \vdots & \\ x_{pn} & y_{pn} \end{bmatrix} - \begin{bmatrix} \widetilde{x_p} & \widetilde{y_p} \\ \widetilde{x_p} & \widetilde{y_p} \\ \vdots & \\ \widetilde{x_p} & \widetilde{y_p} \end{bmatrix} \quad ,$$

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \\ \vdots & \\ J_{n1} & J_{n2} \end{bmatrix} := \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \\ x_n & y_n \end{bmatrix} - \begin{bmatrix} \widetilde{x} & \widetilde{y} \\ \widetilde{x} & \widetilde{y} \\ \vdots & \\ \widetilde{x} & \widetilde{y} \end{bmatrix}$$

3. Compute the 2×2 covariance matrix:

$$S = I^T W J$$
$$\text{with} \quad W = diag(w_1, w_2, \ldots, w_n)$$

4. Compute the the singular value decomposition of S:

$$S = U \Sigma V^T$$

5. The rotation is then given by:

$$R = V \begin{bmatrix} 1 & 0 \\ 0 & det(VU^T) \end{bmatrix} U^T$$

6. And the translation is given by:

$$\vec{t} = \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} - R \begin{bmatrix} \widetilde{x_p} \\ \widetilde{y_p} \end{bmatrix}$$

Once this computation is performed, the calculated rotation and transformation is applied to the formation plan N_p (as shown at rightmost of figure 3.13). Finally, this transformation produces a target point for a vehicle corresponding to the vehicle's position in the transformed plan, and the vehicle is driven towards this point. The BHV_RigidNeighbourRegistration algorithm is summarized in algorithm 7. The open source linear algebra library, Armadillo, is used in my implementation [64].

Note that table 3.6 reveals that there is one notable parameter, the onwship_weight, that specifies the weighting w_i associated with the ownship position in the plan (which is always $(0, 0)$, since the plan for each vehicle has the vehicle at the origin). Other than this user-specified weight, the weights of neighbours are hard-set to 1.0. Varying this ownship_weight allows the user to determine how much the ownship is taken into account during the rigid transformation calculation - lowering it will cause the plan to conform more tightly to the vehicle's neighbours causing the vehicle's target to be further from it, while increasing it will cause the vehicle's target to remain closer to it; essentially this parameter can control how 'rigid' the formation is. Finally, we show in figure 3.14 the behaviour running on a swarm of 22 AUVs in the MOOS-IvP Simulation Test-Bed. As before, since the user can specify the formation plan and there are globally unique IDs, the letters 'NR' are formed by the 22 vehicles.

Algorithm 7 BHV_RigidNeighbourRegistration main loop.

procedure BHV_RigidNeighbourRegistration::onRunState()
  $N \leftarrow$ call $ManageAcousticPing :: getNeighbourPos()$
  $N_p \leftarrow$ call $AcousticPingPlanner :: getNeighbourPlan()$
  $target \leftarrow (0, 0)$
  $X_N \leftarrow$ empty $(n+1) \times 2$ matrix
  $X_N[0,:] \leftarrow [0 \quad 0]$
  $X_{Np} \leftarrow$ empty $(n+1) \times 2$ matrix
  $X_{Np}[0,:] \leftarrow [0 \quad 0]$
  $weights \leftarrow (n+1) \times 1$ matrix of ones
  $weights[0] \leftarrow ownship\_weight$
  for $(x_i, y_i) \in N$ **do**
    if $\sqrt{x_i^2 + y_i^2} \leq contact\_rangeout$ **then**
      $X_N[i,:] \leftarrow [x_i \quad y_i]$
      $X_{Np}[i,:] \leftarrow [x_{pi} \quad y_{pi}]$
    end if
  end for
  $[\widetilde{x} \quad \widetilde{y}] \leftarrow ((weights^T)(X_N))/sum(weights)$
  $[\widetilde{x_p} \quad \widetilde{y_p}] \leftarrow ((weights^T)(X_{Np}))/sum(weights)$
  $J \leftarrow$ matrix for all rows in X_N, subtract $[\widetilde{x} \quad \widetilde{y}]$
  $I \leftarrow$ matrix for all rows in X_{Np}, subtract $[\widetilde{x_p} \quad \widetilde{y_p}]$
  $S \leftarrow (I^T)(diag(weights))(J)$
  $[U, s, V] \leftarrow svd(S)$ ▷ perform singular value decomposition
  $R \leftarrow V \begin{bmatrix} 1 & 0 \\ 0 & det(VU^T) \end{bmatrix} U^T$
  $\vec{t} \leftarrow \left[\widetilde{x} \quad \widetilde{y}\right]^T - R \left[\widetilde{x_p} \quad \widetilde{y_p}\right]^T$
  $transformed\_X_{Np} \leftarrow (R)({X_{Np}}^T)$
  for all rows in $transformed\_X_{Np}$ add $\vec{t}$
  $target \leftarrow (transformed\_X_{Np}[0,0], transformed\_X_{Np}[0,1])$
  call $DriftingTarget :: addRelativeTargetToFilter(target.x, target.y)$
end procedure

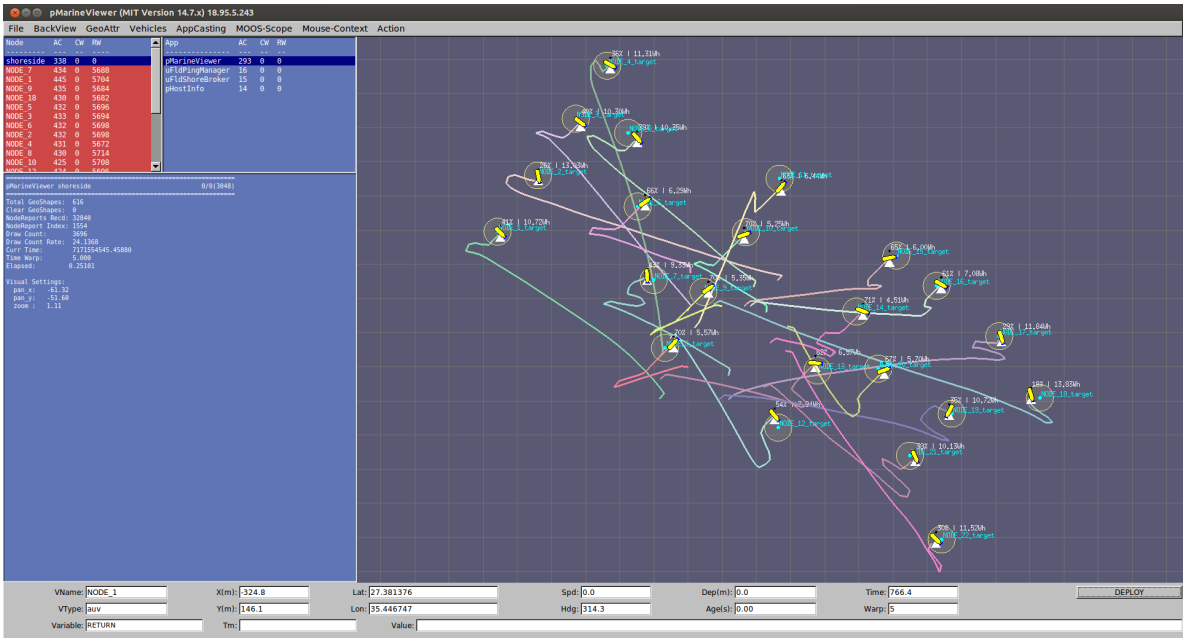

Figure 3.14: BHV_RigidNeighbourRegistration running on a swarm of 22 AUVs in the MOOS-IvP Simulation Test-Bed.

3.7 Formation Control Algorithm 4 - BHV_AssignmentRegistration Behaviour

Given a user-defined formation plan and the ability to exchange globally unique IDs between vehicles, BHV_RigidNeighbourRegistration appears to construct the desired formation in a fairly optimal way - the vehicles move to their respective positions in the formation using near-direct paths. A question then arises - is it possible to assign vehicles to positions in the formation dynamically? If we were able assign the correspondences dynamically in such a way that we minimize the length of time it takes to construct the formation, we would be able to avoid the numerous crossed paths that occur with BHV_RigidNeighbourRegistration during formation construction and in doing so, reduce overall energy expenditure. In addition, vehicles would no longer be required to exchange IDs, thus theoretically reducing communications requirements. With this question in mind, I developed the BHV_AssignmentRegistration behaviour. As with BHV_RigidNeighbourRegistration, BHV_AssignmentRegistration gains access to the actual and desired relative positions of neighbouring vehicles via ManageAcousticPing and AcousticPingPlanner. However, it uses the parameters specified by AcousticPingPlanner only to obtain the desired geometric plan for the formation (see section 3.3.2 for the AcousticPingPlanner parameters).

The main idea behind BHV_AssignmentRegistration is this - somehow compute an op-

timal assignment in the plan for the vehicle it is running on and its neighbours, based on the positions of the neighbours within its communications radius, CR; then calculate and perform the rigid transformation on the corresponding neighbour positions from the plan, in the same manner as was performed for BHV_RigidNeighbourRegistration; and finally, direct the vehicle to its transformed plan position, and iterate. The main issue that must be addressed is how to compute these optimal assignments, a problem that is extremely difficult to solve. An approach has to be developed where, given a set of neighbour positions that is a subset of the entire formation, we must somehow determine which portion of the formation best fits this subset - in essence, the vehicle must determine which point in the formation it is most suited to using only the positions of its neighbours. Now, if each vehicle is able to measure the positions of all other vehicles in the formation, then the problem is simplified dramatically - vehicles can simply use an assignment algorithm, such as the Hungarian or Kuhn–Munkres algorithm to determine an optimal assignment for itself and its neighbours, and since each vehicle has access to the positions of all others, running the same algorithm on all vehicles will naturally result in the same assignment solution being produced on all vehicles. Rather, the difficulty lies in the fact that each vehicle in the formation can only 'see' a subset of vehicles in the formation, and it is also likely that each vehicle 'sees' a different subset. Our main concern is how to find where in the formation each subset is best placed.

In an attempt to simplify the problem somewhat, we limit our formations to lattices - formations that have repetitive patterns with more than one line of symmetry. In this way the formation is made up of fundamental 'unit' shapes, which, intuitively, may make it easier to match vehicle positions to. In addition, we assume that at the beginning of formation construction all vehicles are within the CR of one another, providing a common assignment solution. My approach to the problem is essentially one of brute force, the steps of which are outlined here:

1. Given the set defined by the ownship position and positions of its n neighbours $N = \{(x_{os}, y_{os}) = (0,0), (x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, my algorithm first loops through all points in the user-defined formation plan.

2. For each point (x_{p0}, y_{p0}) in the plan it then selects that point plus the n nearest points to it in the plan (strictly speaking, for the n^{th} nearest point, if there exists any other points in the plan at the same distance, then those points are also selected), giving us the set $N_p = \{(x_{p0}, y_{p0}), (x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ...(x_{pm}, y_{pm})\}$, where $m \geq n$.

3. N and N_p are then aligned by subtraction of their respective centroids.

4. Following this, the algorithm then enters a second loop, during which the following occurs on each iteration:

(a) N is rotated by a user-specified $\delta\theta$, resulting in the set $N_\theta = \{(x_{\theta os}, y_{\theta os}), (x_{\theta 1}, y_{\theta 1}), (x_{\theta 2}, y_{\theta 2}), ...(x_{\theta n}, y_{\theta n})\}$; for clarity, we let $(x_{\theta os}, y_{\theta os}) := (x_{\theta 0}, y_{\theta 0})$.

(b) A cost matrix C of size $m \times n$ is built, in which $C(i,j) = \sqrt{(x_{\theta j} - x_{pi})^2 + (y_{\theta j} - y_{pi})^2}$; essentially this is a matrix of costs corresponding to the Euclidean distance between each rotated point in N_θ and all points in N_p.

(c) C is fed to an implementation of the Hungarian/Kuhn–Munkres [65] algorithm (derived from [66]) to determine the optimal assignment between the N_θ and N_p points sets; the total cost for this assignment is compared to the cost for the previous rotation, and kept if it is smaller.

(d) The loop terminates when a full rotation of N has been accomplished, resulting in a set N_θ with a corresponding minimum cost and rotation value.

5. After looping through all points in the formation, the algorithm has a cost, assignment, and plan point set N_p associated with each point; the lowest cost assignment and corresponding N_p are selected, and the points in N_p are rearranged in order such that point (x_i, y_i) in N is assigned to point (x_{pi}, y_{pi}) in N_p; if $m > n$ then the last $m - n$ elements are removed from N_p; the algorithm now has the point sets $N = \{(x_{os}, y_{os}) = (0,0), (x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$ and $N_p = \{(x_{p0}, y_{p0}), (x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ...(x_{pn}, y_{pn})\}$, where point i in N is assigned to point i in N_p.

6. Finally, given N and N_p, the algorithm calculates the optimal least-squares rigid transformation (equation 3.8) between the two in the same manner as is done by BHV_RigidNeighbourRegistration, and the vehicle is driven to the transformed ownship point.

Given the algorithmic complexity of BHV_AssignmentRegistration, we do not attempt to illustrate its operation. We note that this algorithm is computationally intensive, and does not scale well when the formation size is very large, since it loops through all points in the formation. However, because we restrict formations to lattices, it is possible to improve the computation time of the algorithm by noting that given the communication range of the vehicles, there are only a certain number of unique subsets of the formation - in this way the algorithm can be augmented by only examining the set of points forming this unique subset. In addition, the value of $\delta\theta$ significantly affects the processing time, as it determines how many rotations are performed - this value is set to a default of 45°, as shown in table 3.7. Finally, a summary of the approach is provided in algorithm 8, and figure 3.15 displays the behaviour running on a swarm of 20 AUVs in the MOOS-IvP Simulation Test-Bed. The 20 AUVs construct a square lattice of size 4×5, where each square is $300m \times 300m$.

Algorithm 8 BHV_AssignmentRegistration main loop.

procedure BHV_AssignmentRegistration::onRunState()
  $N \leftarrow$ call $ManageAcousticPing :: getNeighbourPos()$
  $N_p \leftarrow$ call $AcousticPingPlanner :: getNeighbourPlan()$
  $target \leftarrow (0, 0)$
  $X_N \leftarrow$ empty $(n+1) \times 2$ matrix
  $X_N[0, :] \leftarrow [0 \quad 0]$
  $weights \leftarrow (n+1) \times 1$ matrix of ones
  $weights[0] \leftarrow ownship\_weight$
  for $(x_i, y_i) \in N$ **do**
    if $\sqrt{x_i^2 + y_i^2} \leq contact\_rangeout$ **then**
      $X_N[i, :] \leftarrow [x_i \quad y_i]$
    end if
  end for
  $min\_cost\_list \leftarrow$ empty list
  $min\_assignments\_list \leftarrow$ empty list
  $min\_plan\_list \leftarrow$ empty list
  $num\_rotations \leftarrow 360.0 \div delta\_theta$
  for $(x_{pi}, y_{pi}) \in N_p$ **do**
    $X_{Np} \leftarrow$ empty $(m+1) \times 2$ matrix
    $X_{Np}[0, :] \leftarrow [x_{pi} \quad y_{pi}]$
    $X_{Np}[1 : end, :] \leftarrow$ fill with m closest points to (x_{pi}, y_{pi})
    $[\widetilde{x} \quad \widetilde{y}] \leftarrow ((weights^T)(X_N))/sum(weights)$
    $[\widetilde{x_p} \quad \widetilde{y_p}] \leftarrow ((weights^T)(X_{Np}))/sum(weights)$
    $X_N \leftarrow$ matrix for all rows in X_N, subtract $[\widetilde{x} \quad \widetilde{y}]$
    $X_{Np} \leftarrow$ matrix for all rows in X_{Np}, subtract $[\widetilde{x_p} \quad \widetilde{y_p}]$
    $min\_cost \leftarrow \infty$
    $min\_assignments \leftarrow None$
    $costs \leftarrow$ empty $m \times n$ matrix
    for $i = 1$ to $num\_rotations$ **do**
      $X_{N\theta} \leftarrow X_n$ rotated by $i \cdot delta\_theta$
      $costs[j, k] \leftarrow \sqrt{(x_{\theta k} - x_{pj})^2 + (y_{\theta k} - y_{pj})^2}$
      $[curr\_cost, curr\_assignments] \leftarrow$ call Kuhn–Munkres algorithm with *costs* matrix
      if $curr\_cost < min\_cost$ **then**
        $min\_cost \leftarrow curr\_cost$
        $min\_assignments \leftarrow curr\_assignments$
      end if
    end for
    $min\_cost\_list.append(min\_cost)$
    $min\_assignments\_list.append(min\_assignments)$
    $min\_plan\_list.append(X_{Np})$
  end for

42: $[\widetilde{x} \quad \widetilde{y}] \leftarrow ((weights^T)(X_N))/sum(weights)$
43: $J \leftarrow$ matrix for all rows in X_N, subtract $[\widetilde{x} \quad \widetilde{y}]$
44: $X_{Np} \leftarrow$ matrix element of min_plan_list corresponding to $min(min\_cost\_list)$
45: $X_{Np} \leftarrow X_{Np}$ with rows rearranged according to assignments specified by element of $min\_assignments\_list$ corresponding to $min(min\_cost\_list)$
46: $[\widetilde{x_p} \quad \widetilde{y_p}] \leftarrow ((weights^T)(X_{Np}))/sum(weights)$
47: $I \leftarrow$ matrix for all rows in X_{Np}, subtract $[\widetilde{x_p} \quad \widetilde{y_p}]$
48: $S \leftarrow (I^T)(diag(weights))(J)$
49: $[U, s, V] \leftarrow svd(S)$ ▷ perform singular value decomposition
50: $R \leftarrow V \begin{bmatrix} 1 & 0 \\ 0 & det(VU^T) \end{bmatrix} U^T$
51: $\vec{t} \leftarrow \left[\widetilde{x} \quad \widetilde{y}\right]^T - R\left[\widetilde{x_p} \quad \widetilde{y_p}\right]^T$
52: $transformed\_X_{Np} \leftarrow (R)({X_{Np}}^T)$
53: for all rows in $transformed\_X_{Np}$ add $\vec{t}$
54: $target \leftarrow (transformed\_X_{Np}[0,0], transformed\_X_{Np}[0,1])$
55: call $DriftingTarget :: addRelativeTargetToFilter(target.x, target.y)$
56: **end procedure**

Parameter	**Description**	**Default**
contact_rangeout	Backup distance above which neighbour pings are ignored. (m)	650.0
ownship_weight	The weighting w_i associated with the ownship position in the plan (0,0) - a lower weight means that the plan will conform more closely to neighbours.	0.1
delta_theta	The amount of rotation $\delta\theta$ applied to the set N on each iteration of the second loop. (degrees)	45.0
display_rigid_registration_points	Display points in the $pMarineViewer$ visualizer corresponding to the plan positions included in the rigid transformation calculation.	false
display_rigid_registration_hull	Display a polygon in the $pMarineViewer$ visualizer showing the convex hull of the plan positions included in the rigid transformation calculation.	false

Table 3.7: Important parameters of the BHV_AssignmentRegistration behaviour.

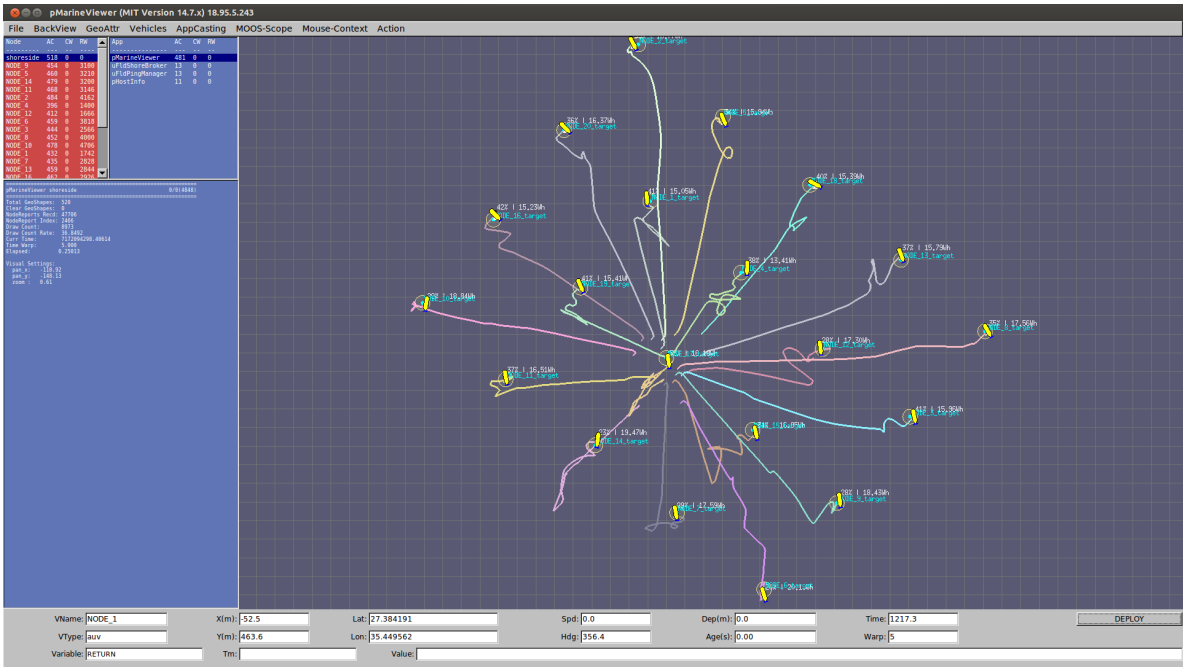

Figure 3.15: BHV_AssignmentRegistration running on a swarm of 20 AUVs in the MOOS-IvP Simulation Test-Bed.

3.8 Summary

This chapter has given the reader a comprehensive explanation of the algorithmic detail and structure of the behaviours developed for this thesis. In particular, the four formation control behaviours I have developed have been explained and discussed in depth, giving the reader an understanding of how each one operates.

The remainder of the thesis is organized into three chapters. Chapter 4 gives the reader an overview of the methodology, infrastructure, and comparison metrics used to evaluate each behaviour in simulation. Chapter 5 provides the results and an analysis of the comparison metrics. Finally, chapter 6 details a discussion of the results of the analysis, conclusions drawn from this research, and future work to be undertaken.

This page intentionally left blank.

Chapter 4

Methodology, Infrastructure, and Comparison Metrics

This chapter provides the reader with an abridged overview of the software environment and infrastructure used to simulate the swarm of AUVs, which we name the MOOS-IvP Ocean Simulation Test-Bed (screen-shots of which were provided throughout chapter 3). Additionally, the four comparison scenarios used during testing are examined, and a detailed explanation is provided of the formation quality metric developed in order to compare the ability of each formation control behaviour. Finally, a brief explanation of methodology used for testing and comparing each formation control behaviour is provided at the end of the chapter.

4.1 The MOOS-IvP Ocean Simulation Test-Bed

The MOOS-IvP Ocean Simulation Test-Bed is organized as illustrated in figure 4.1. As briefly mentioned in section 2.2 and as can be seen in the diagram, the architecture runs using a single centralized 'shoreside' MOOS community alongside multiple AUV MOOS communities, with each AUV in the swarm having their own community running identical MOOSApps. We briefly describe the applications and operation of the AUV and shoreside communities here, as well as how they interact with one another. We do not provide a detailed description of each application or the general operational mechanisms of MOOS, as many of the applications and their details can be found on the MOOS-IvP website [67], and the reader can obtain a better understanding of MOOS in [3]. We elaborate primarily on the MOOSApps developed especially for this thesis, indicated by the $\star$ symbol in the lists below.

Each simulated AUV runs the following MOOSApps within its community:

- MOOSDB: the central database in which variables from MOOSApps are published to and subscribed from.

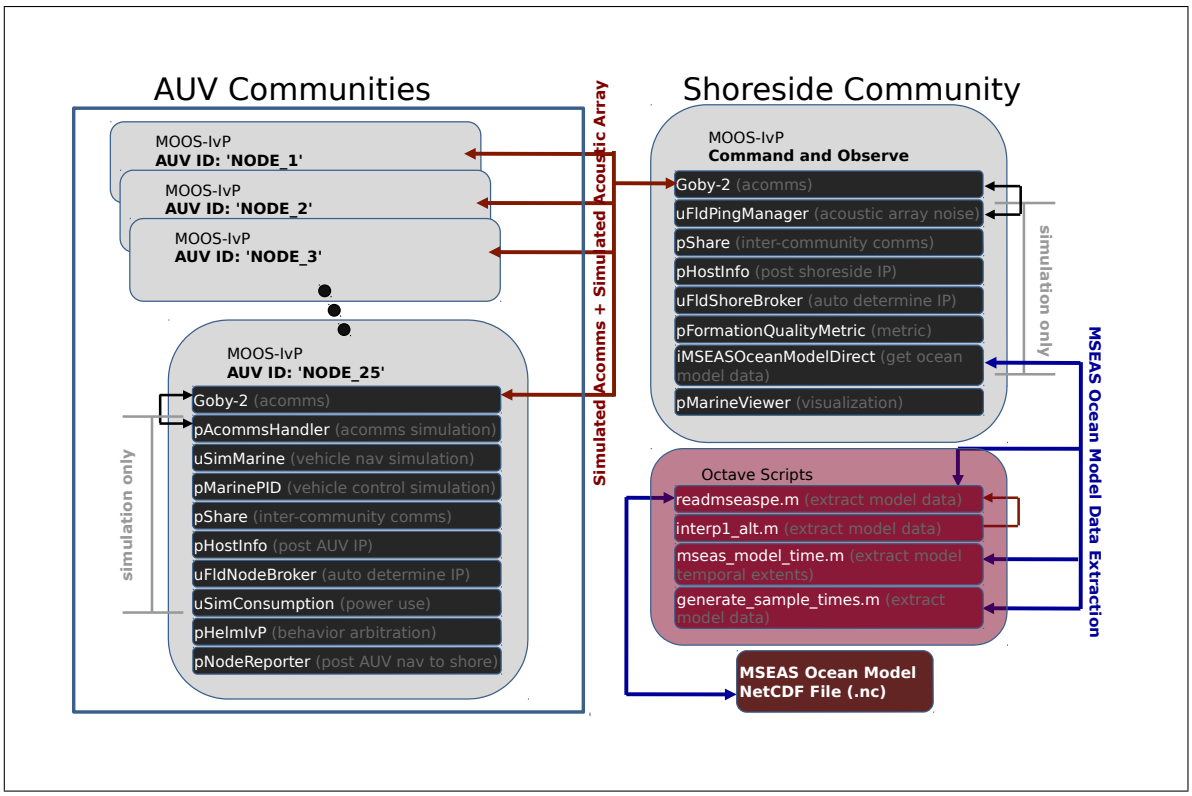

Figure 4.1: Diagram of the system architecture for the MOOS-IvP Ocean Simulation Test-Bed; a single MOOS 'Shoreside' community is run alongside multiple MOOS 'AUV' communities.

- pHelmIvP: the IvP Helm which arbitrates between multiple vehicle behaviours over a user-defined decision space, and ultimately publishes a steady stream of desired vehicle control values, typically heading, speed, and depth.

- ⋆ uSimMarineSwarm: a simple 3D vehicle simulator that continually updates the state of a simulated vehicle using current actuator values, the previous vehicle state, and the time elapsed since the previous update. This MOOSApp subscribes to MOOS variables related to the current navigation state as well as actuator values in order to determine how to update the navigation state of the vehicle. In addition, it subscribes to values related to external drift acting upon the vehicle, allowing it to simulate disturbances from wind or ocean currents. uSimMarineSwarm is a simple extension of the already available MOOSApp uSimMarine, with modifications that allow the AUV to maintain depth while stationary (i.e. active buoyancy control), and to allow the AUV to rotate in place. uSimMarineSwarm is used for all simulations in this work, but can be easily replaced by uSimMarine, or any other user-defined vehicle simulator if necessary.

- ⋆ pMarinePIDSwarm: a simple PID controller that is utilized in conjunction with uSimMarineSwarm, and which performs low-level PID control of vehicle heading, speed, and

depth. It subscribes to the high-level desired control values produced by the IvP Helm, and publishes low-level actuator values to realize the desired heading, speed, and depth. These low-level actuator values are then used by uSimMarineSwarm to update the simulated position of the vehicle. pMarinePIDSwarm is a simple extension of pMarinePID, with modifications that allow for active depth control and vehicle turn-in-place. As mentioned previously, this can be replaced by pMarinePID if a more traditional AUV needs to be simulated.

- pNodeReporter: this MOOSApp subscribes to vehicle state information, and publishes a single string summarizing this information, including vehicle ID, type, position, time of report, and length. In our system, this information is typically bridged via pShare to the shoreside community so as to provide it with information about all vehicles in the swarm. This allows the shoreside to display visual indicators of each vehicle, as well as to perform centralized functions necessary for simulation, such as inter-vehicle acoustic communications and ocean current effects.

- pShare: a foundational MOOSApp that allows user-specified variables to be shared between multiple MOOSDBs. In our system pShare bridges variables related to vehicle state, acoustic pings, and ocean currents between each AUV community and the shoreside community - no variables are bridged between AUV communities.

- pHostInfo: a MOOSApp that simply posts the IP address on which the community is running to the MOOSDB. This information is typically used by uFldNodeBroker.

- uFldNodeBroker: a tool used for brokering connections between each vehicle community and the shoreside community. This MOOSApp is used to automate the configuration of pShare (IP and port number) in order to allow the bridging of variables. When there are a large number of vehicles, configuring pShare can be cumbersome - uFldNodeBroker simplifies this process.

- pAcommsHandler: this MOOSApp implements a networking framework that allows for efficient acoustic communications via a dynamic encoding/decoding scheme known as the Dynamic Compact Control Language. This application was developed by GobySoft. More information is available at [68]. In addition, pAcommsHandler continuously publishes a ping variable on a timer indicating the maximum ping range, which is then bridged to the shoreside community for use by uFldPingManager. In the scenario that we wish to extend the work in this thesis with behaviours that require a larger amount of information to be communicated acoustically between vehicles, pAcommsHandler allows for the simulation of the state-of-the-art in acoustic communications, providing us with an indication of the limits of underwater communications and multi-vehicle operations.

- ⋆ pTrailViewer: a simple application that displays the vehicle trail - although the pMarineViewer visualizer already allows for in-built display of vehicle trails, this option can be computer intensive and limited in trail length. pTrailViewer allows the user to specify the distance between vertices in the trail, and randomizes the trail colour for a more user-friendly visualization of trails from multiple vehicles.
- ⋆ uSimConsumption: a MOOSApp that allows the user to specify the amount of energy the vehicle has at its disposal, and simulates the energy consumed by hotel, propulsion, and communication using a simple power consumption model. It then provides an estimate of the maximum length of the mission in days. The details of uSimConsumption are elaborated upon in section 4.1.1.
- ⋆ pNodeLogger: a simple MOOSApp that records information related to mission time, vehicle state, and vehicle energy consumption as a file of comma-separated values for post-processing.

The shoreside community runs the following MOOSApps:

- MOOSDB: the central database in which variables from MOOSApps are published to and subscribed from.
- pMarineViewer: a visualizer written in FLTK and OpenGL for rendering vehicles and their associated information to allow the user to visualize vehicles during simulation. The user is able to manipulate the viewer to monitor the swarm and access vehicle information. Information published by pNodeReporter and bridged by pShare is used to display the current state of the vehicles.
- pShare: a foundational MOOSApp that allows user-specified variables to be shared between multiple MOOSDBs. In our system pShare bridges variables related to vehicle state, acoustic pings, and ocean currents between each AUV community and the shoreside community.
- pHostInfo: a MOOSApp that simply posts the IP address on which the community is running to the MOOSDB. This information is typically used by uFldShoreBroker.
- uFldShoreBroker: a tool used for brokering connections between the shoreside community and multiple vehicle communities. This MOOSApp is used in coordination with uFldNodeBroker to automate the configuration of pShare (IP and port number) in order to allow the bridging of variables.
- ⋆ uFldPingManager: a MOOSApp to simulate the transmission and reception of acoustic pings. This MOOSApp uses the information published by pNodeReporter and pAcommsHandler and bridged by pShare to determine the positions of all vehicles and the

ping transmission time; then for each vehicle, it calculates whether or not a ping from every other vehicle will be received by it (i.e. the transmitting vehicle is within the receiving vehicle's ping range). If so, it then calculates the range, bearing, and elevation to this neighbouring vehicle, adds Gaussian noise to each parameter and a transmission delay based on the distance and acoustic speed, and publishes a ping string with the neighbouring vehicle ID and these noisy parameters, which is finally bridged back to the receiving vehicle via pShare.

- ⋆ iMSEASOceanModelDirect: this application provides a high-fidelity simulation of ocean currents using a netCDF datafile generated by the MSEAS MIT group. This netCDF file provides a 4 dimensional dynamical representation of ocean currents over specified temporal and spatial extents, and this model is accessed by iMSEASOceanModelDirect using a number of Octave scripts. As the mission progresses, this application requests model data from positions corresponding to the positions of vehicles, and this data is bridged back to each vehicle for use by uSimMarineSwarm as an external disturbance from ocean currents. The details of iMSEASOceanModelDirect are elaborated upon in section 4.1.2.

- ⋆ pFormationQualityMetric: this MOOSApp utilizes the positions of all vehicles in the swarm (obtained using the data published by pNodeReporter) to continually assess the formation quality metric, publishing this value to the MOOSDB. This metric provides a quantitative value of how well the swarm is maintaining the desired formation. The details of how the formation quality is assessed are provided in section 4.3.

- ⋆ pShoreLogger: a simple MOOSApp that records information related to mission time and formation quality as a file of comma-separated values for post-processing.

4.1.1 Energy Expenditure

The principal reason for using ocean currents in order to propel the swarm is to increase its endurance and maximize the mission duration. In order to get an idea of possible swarm endurance, a measure of the energy expenditure of each vehicle must be estimated. To this end, a simple power consumption model was used and implemented in the uSimConsumption MOOSApp. Essentially, uSimConsumption calculates the instantaneous power consumption due to propulsion, communications, and hotel load, and integrates this to obtain the total consumed electrical energy. The important configuration parameters of uSimConsumption are listed in table 4.1.

The power consumption from propulsion is calculated as follows - the total propulsion efficiency is given by:

Parameter	**Description**	**Default**
efficiency_prop	The efficiency of the propeller.	0.58
efficiency_shaft	The efficiency of the propeller shaft.	0.7
efficiency_gear	The efficiency of the propeller gears.	0.9
efficiency_motor	The efficiency of the propeller motor.	0.8
power_nominal	The power of the hotel load. (W)	1.5
consumption_per_ping	The power consumption of each acoustic ping. (Ws)	2.4
power_speed	The electrical power at a specified speed. ($newtons$ and m/s)	power=90.0,speed=1.0
energy_pack	The total energy available to the vehicle. (WH)	520.0
decaying_speed_ratio	If set to false, the propulsion power is hard-set to 90 newtons whenever the vehicle is thrusting; otherwise the propulsion power increases linearly through 0 and the point defined by *power_speed*.	true

Table 4.1: Important configuration parameters of the uSimConsumption MOOSApp.

$$\eta = efficiency\_prop \cdot efficiency\_shaft \cdot efficiency\_gear \cdot efficiency\_motor \tag{4.1}$$

The drag coefficient is calculated by (where $POWER$ and $SPEED$ are the power and speed parameters from *power_speed*):

$$\begin{aligned} drag &= \frac{POWER}{SPEED} \times \eta \\ c_d &= \frac{drag}{SPEED \times |SPEED|} \end{aligned} \tag{4.2}$$

The power consumption from propulsion is finally given by (where v is the speed of the vehicle in m/s, and Δ_t is the time difference since the last consumption calculation):

$$\begin{aligned} P_{prop} &= c_d \times v^2 \times |v| \times \frac{1}{\eta} \\ consum_{prop} &= P_{prop} \times \Delta_t \end{aligned} \tag{4.3}$$

This power consumption due to propulsion is continuously accumulated throughout the mission. Similarly, the power consumption due to hotel load is accumulated, and is calculated as follows:

$$consum_{hotel} = power\_nominal \times \Delta_t \tag{4.4}$$

Finally, the accumulated power consumption due to acoustic pings is simply incremented on each acoustic ping by the value of *consumption_per_ping*. These power consumption values for propulsion, hotel, and acoustics are given in Ws, and given the *energy_pack* in WH, we are able to calculate an estimate of the total mission duration. The total power consumed by the vehicle in WH is calculated and published by uSimConsumption on each iteration of the simulation, and is logged by pNodeLogger for use in analysing the energy efficiency of my formation control behaviours.

4.1.2 Simulation of Ocean Currents Using MSEAS Models

In order to simulate realistic ocean currents, the MOOS-IvP simulation test-bed was augmented to take advantage of the accurate ocean models produced by the MSEAS MIT research group [69]. The MIT Multidisciplinary Simulation, Estimation, and Assimilation Systems (MSEAS) group creates and utilizes physics-driven numerical models of dynamic oceanographic environments based on data from current and historical in-situ and remotely sensed measurements. The purpose of the MSEAS-MOOS interface is to allow the use of MSEAS ocean models within the LAMSS MOOS-IvP environment, allowing us to incorporate realistic ocean environments in our simulations. This MSEAS-MOOS interface is comprised of a set of Octave scripts and a single MOOSApp named iMSEASOceanModelDirect, that facilitates the use of MSEAS ocean models within the MOOS-IvP environment. We detail here these scripts and iMSEASOceanModelDirect, covering their roles and the method by which to configure the parameters of this interface.

Octave Scripts

The MSEAS group creates ocean models in a format called netCDF, and provides a Matlab file called readmseaspe.m that allows the user to extract and interpolate data from these netCDF files at specified positions and times. The foundation of the MSEAS-MOOS interface is an Octave script called readmseaspe_moos.m, which is in essence a direct translation of the readmseaspe.m file from Matlab to Octave. In addition to this script, three other scripts are used by the interface - interp1_alt.m, a script for 1-dimensional interpolation, mseaspe_model_time.m, a script which extracts the time extents of the ocean model, and generate_sample_times.m, a script which converts a user requested sample time from seconds to the format required by readmseaspe_moos.m. We briefly describe each of these scripts here. Note that these scripts are used internally by the MOOSApp iMSEASOceanModelDirect.

The Octave script readmseaspe_moos.m is the foundation of the MSEAS-MOOS interface, and is called within iMSEASOceanModelDirect in order to access desired ocean data from the netCDF file, at every vehicle position within the MOOS-IvP simulation. The input to this file is a string of requested variables (e.g. ocean velocities, temperature, salinity, etc.), a longitude, a latitude, a depth, and a time, and its output is the requested values interpolated by the dynamic model from the netCDF file. A significant advantage of this script is that it allows for multiple requests to be calculated and retrieved with a single function call. The script interp1_alt.m, is called by readmseaspe_moos.m for 1-D interpolation of ocean values. The next script, mseaspe_model_time.m performs a simple function that allows iMSEASOceanModelDirect to retrieve the temporal extents of the ocean model to be used. This is needed in order to correctly extract samples at the requested times. The final Octave script, generate_sample_times.m, is used to convert a sample request time from Unix time to the vector format (year, month, day, hours, minutes, seconds) required by readmseaspe_moos.m. In addition, it returns a cell array of these vectors, in order to perform multi-vehicle requests that are separate in space but equal in time. For a deeper understanding of how exactly the MSEAS netCDF files are generated and how readmseaspe.m accesses netCDF data, see [70].

Parameter	**Description**	**Default**
octave_path	The path to the Octave scripts used by iMSEASOceanModelDirect.	/path_to_scripts/
mseas_filepath	The path to the MSEAS netCDF .nc file to be read.	/path_to_netCDF/
mseas_varnames	The variables to be read/interpolated, usually u,v,w for zonal, meridional, and vertical ocean current velocities. Any combination of u,v,w can be used, in any order.	u,v
mseas_time_offset	The offset in Unix seconds from the the start of the MSEAS netCDF file. This allows the user to start the data access at a time offset.	0.0
node_communities	A comma-separated list of vehicle names that the user wishes to obtain velocities for (e.g. alpha,bravo,charlie,...).	NODE_1,...,NODE_25

Table 4.2: Important configuration parameters of the iMSEASOceanModelDirect MOOSApp.

iMSEASOceanModelDirect

iMSEASOceanModelDirect is a MOOSApp that runs on the shoreside community, and essentially utilizes the previously described Octave scripts to access desired ocean velocity data from a specified netCDF file. Given a list of vehicles whose positions are bridged to the shoreside, iMSEASOceanModelDirect performs a batch request of the ocean velocity data

via the readmseaspe_moos.m script, using the time elapsed since its launch, and publishes the ocean current values extracted from the netCDF file at the positions of each vehicle as variables. These variables are then bridged to each vehicle MOOS community for use as external disturbances. It performs this request in a 'as soon as possible' manner, sending the next request the moment the previous request has been satisfied. Important configuration parameters of iMSEASOceanModelDirect are listed in table 4.2. iMSEASOceanModelDirect is used to simulate realistic ocean currents in one comparison scenario for the evaluation of my formation control behaviours.

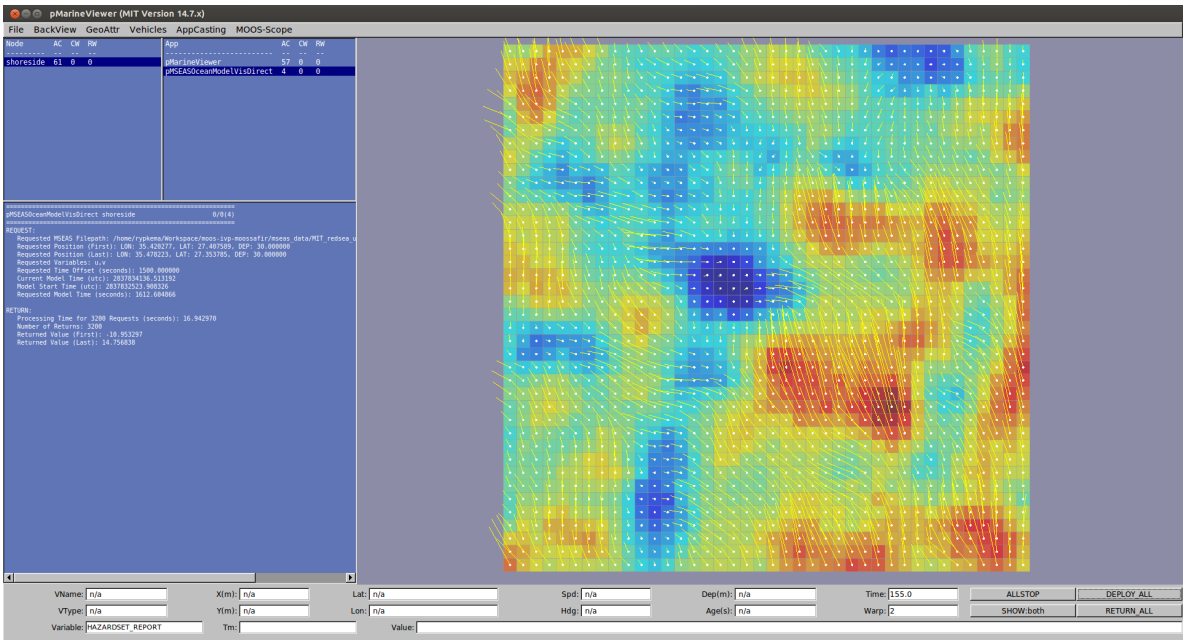

Figure 4.2: Visualization in the MOOS-IvP Simulation Test-Bed of zonal and meridional currents from an MSEAS ocean model netCDF file representing the Red Sea.

4.2 Comparison Scenarios

We evaluate the efficacy of each of my formation control behaviours using four comparison scenarios of different ocean currents. The first is with no currents at all, the second is with three current channels of linear velocity, the third is with an irrotational current vortex, and the last is with realistic ocean currents representative of the Red Sea.

In the first scenario, we purposely do not use any ocean currents, allowing us to compare the ability of each formation control behaviour to construct the desired formation in an idealized setting. This scenario is also used to test how well each formation control behaviour is able to deal with the loss of vehicles during formation construction, and to assess the effect

of changing communication rate on the formation construction process.

In the second scenario, we have three current channels of uniform linear flow, as shown on the left of figure 4.3. The central line of vehicles is placed such that they experience no external disturbance, while the two upper and lower lines of vehicles experience constant velocity ocean currents moving in opposite directions. As shown in the figure, the centre channel has zero velocity and is placed between $-150m \leq y \leq 150m$, the upper channel has a velocity of $0.1m/s$ along the positive x direction and exists at $y > 150m$, and the lower channel has a velocity of $0.1m/s$ along the negative x direction and exists at $y < -150m$. In the third scenario, we have an irrotational current vortex, where the velocity is inversely proportional to the distance from the axis of rotation, as illustrated on the right of figure 4.3. As shown in the figure, the vortex is centred at $(0, 0)$, and uses the following equations for the magnitude (in m/s) and direction (in $radians$) of current flow, giving us a counter-clockwise vortex that has a velocity inversely proportional to the distance from $(0, 0)$, with a maximum velocity of $0.5m/s$ at the center:

$$\begin{aligned} magnitude &= \frac{1}{\frac{\sqrt{x^2+y^2}}{200} + 2} \\ direction &= atan2(y, x) + \frac{\pi}{2} \end{aligned} \tag{4.5}$$

These two scenarios are used to test how efficiently each of my formation control behaviours can maintain a desired formation in the presence of external disturbances. In addition, we also use these scenarios to evaluate how well each behaviour is able to maintain formation when vehicles fail and are lost, and to see the effect of varying communication rate during formation maintenance.

Finally, the last scenario uses realistic ocean currents in the form of a netCDF file representing the Red Sea. This scenario is used as a general test to compare how well each formation control behaviour is able to construct the desired formation, to maintain it in the presence of ocean currents, and to utilize ocean currents for swarm propulsion.

4.3 The Formation Quality Metric

In order to effectively compare each of my formation control behaviours against one another, there must exist some method of quantifying how well the swarm adheres to the desired formation. To achieve this, I developed a metric to assess the 'formation quality', based on the approach used for the BHV_AssignmentRegistration formation control behaviour detailed in

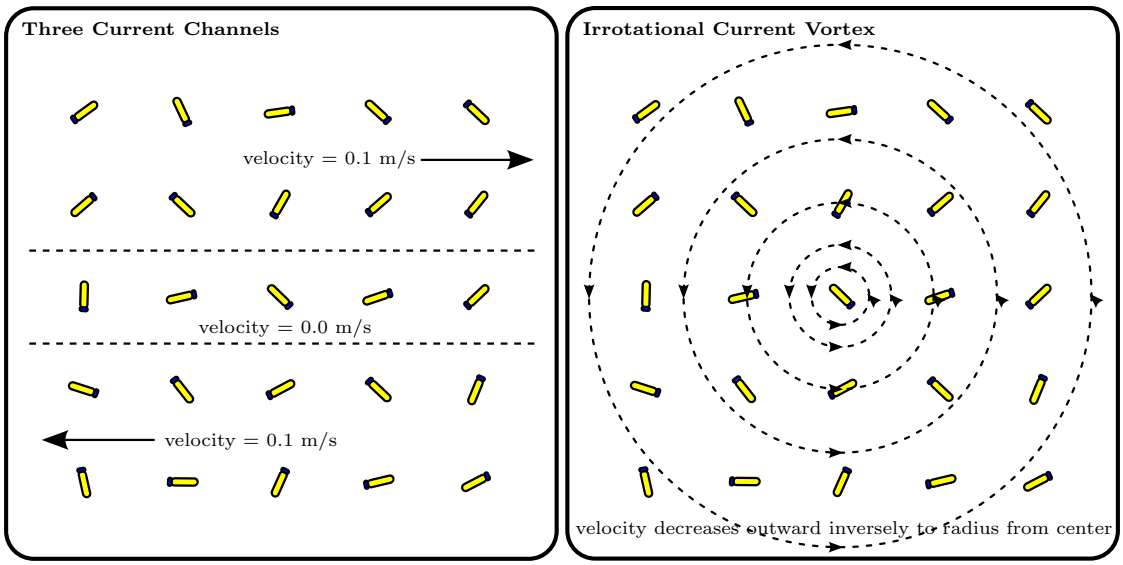

Figure 4.3: Illustration of the second and third comparison scenarios - three current channels of linear velocity, and an irrotational current vortex.

section 3.7. The formation quality metric is implemented in the pFormationQualityMetric MOOSApp which runs on the shoreside community, and operates in much the same way as BHV_AssignmentRegistration; the difference is that pFormationQualityMetric has direct access to the positions of all vehicles in the swarm. It uses these positions to assign vehicles to points in a formation plan using the Hungarian/Kuhn–Munkres algorithm, calculates an optimal rigid transformation between the two point sets, and finally calculates the average distance between the vehicle positions and their corresponding points in the formation plan. The value of this average distance between the actual vehicle position and a desired formation position is the formation quality metric.

In the case of my first formation control behaviour, BHV_AttractionRepulsion, a formation plan does not exist, and the formation is generated dynamically - so how can the formation quality metric be applied to it? We observe that the ideal formation for a given number of vehicles would have roughly the same height as its width, but because of the dynamic nature of this behaviour, this rarely occurs. To surmount this problem, for a given number of vehicles, we first generate a formation plan that is roughly equal in height and width, then add an additional 'layer' of vehicles around the perimeter of this formation plan, and supply this as the desired formation to pFormationQualityMetric. The additional layer provides a buffer when the formation constructed differs from the ideal, and in this way the formation quality metric is made more robust. To provide the reader with a better understanding of how the formation quality metric operates, each step of the algorithm is detailed below - figure 4.4

provides an illustration of the operational principles of the formation quality metric.

1. We begin with the set defined by the positions of all vehicles in the swarm, $N = \{(x_0, y_0), (x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, and the set defined by a user-specified formation metric plan, $N_p = \{(x_{p0}, y_{p0}), (x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ...(x_{pm}, y_{pm})\}$. As previously explained, in our case this formation metric plan is given by the ideal formation plus an additional outer 'layer' of positions.

2. N and N_p are then aligned by subtraction of their respective centroids.

3. We then enter a loop where:

   (a) N is rotated by a user-specified $\delta\theta$, resulting in the set $N_\theta = \{(x_{\theta 0}, y_{\theta 0}), (x_{\theta 1}, y_{\theta 1}), (x_{\theta 2}, y_{\theta 2}), ...(x_{\theta n}, y_{\theta n})\}$.

   (b) A cost matrix C of size $m \times n$ is built, where $C(i, j) = \sqrt{(x_{\theta j} - x_{pi})^2 + (y_{\theta j} - y_{pi})^2}$; essentially this is a matrix of costs corresponding to the Euclidean distance between each rotated point in N_θ and all points in N_p.

   (c) An implementation of the Hungarian/Kuhn–Munkres algorithm uses C to determine the optimal assignment between the N_θ and N_p points sets; the total cost for this assignment is compared to the cost for the previous rotation, and kept if it is smaller.

   (d) The loop terminates when a full rotation of N has been accomplished, resulting in a set N_θ with a corresponding minimum cost and rotation value.

4. After looping through a full rotation of N, the algorithm has the minimum cost and assignment corresponding to a certain rotation value, and the points in N_p are rearranged in order such that point (x_i, y_i) in N is assigned to point (x_{pi}, y_{pi}) in N_p; if $m > n$ then the last $m - n$ elements are removed from N_p; the algorithm now has the point sets $N = \{(x_0, y_0), (x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$ and $N_p = \{(x_{p0}, y_{p0}), (x_{p1}, y_{p1}), (x_{p2}, y_{p2}), ... (x_{pn}, y_{pn})\}$, where point i in N is assigned to point i in N_p.

5. Then, given N and N_p, the algorithm calculates and performs the optimal least-squares rigid transformation (equation 3.8) between the two in the same manner as explained in section 3.6.

6. Finally, after the rigid transformation is applied to N_p, the Euclidean distances between corresponding points of N and N_p are averaged, and this value is returned as the formation quality metric.

Examining the steps of the algorithm, we see that it is extremely similar to the BHV_AssignmentRegistration algorithm, the principal difference being that we do not have to loop

through point subsets of the formation plan (since all vehicles of the swarm are considered at once). It is also apparent that the formation quality metric is unbounded from above, and approaches zero as the 'quality' of the formation improves (i.e. the vehicle positions approach the desired formation). The important configuration parameters of pFormationQualityMetric are listed in table 4.3.

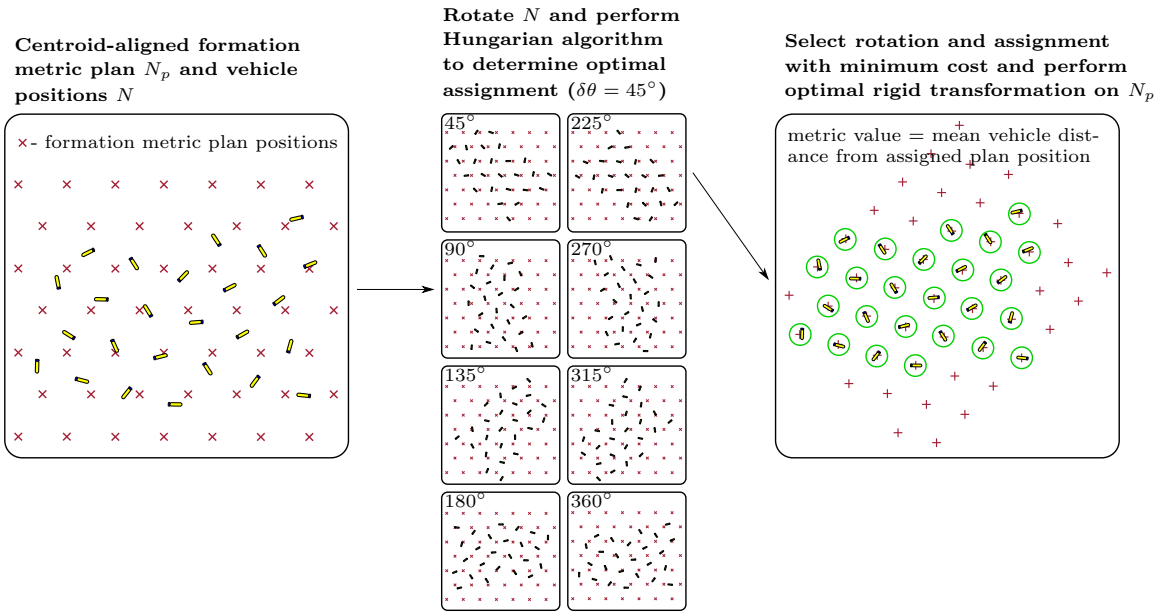

Figure 4.4: Illustration of the operational principles of the formation quality metric.

Parameter	Description	Default
display_rigid_registration	Display points in the pMarineViewer visualizer corresponding to the positions of the transformed formation metric plan.	true
delta_theta	The amount of rotation $\delta\theta$ applied to the set N on each iteration of the inner loop. (degrees)	10.0
node_offsets_metric	A string allowing the user to specify a position of a single point in the formation metric plan; this string must have the following format: "$x = x\_position, y = y\_position$".	none

Table 4.3: Important configuration parameters of the pFormationQualityMetric MOOSApp.

4.4 Putting it all Together - Testing Methodology

Now that we have described the simulation infrastructure, the comparison scenarios used for testing, and the formation quality metric, we are in a position to elaborate upon the

methodology used to compare the ability of my four formation control behaviours. We examine each behaviour during the construction of the formation, in maintaining the desired formation, and in using ocean currents for propulsion. How exactly we compare each behaviour in these three situations is briefly described in this section.

4.4.1 Formation Construction

To compare how efficiently each formation control behaviour is able to construct the desired formation, we use the scenario with no ocean currents and use the following methodology - first, 25 AUVs are randomly initialized in a $200m \times 200m$ box; we then instruct the AUVs to form a hexagonal lattice of five rows of five AUVs, with a separation distance of $300m$; during formation construction, the energy expenditure of all AUVs is recorded, as well as the formation quality metric; once the change in the metric value falls below a specified threshold, the simulation is stopped, and the mean energy expenditure over all AUVs is computed; finally, this process is repeated 5 times, and the mean energy expenditure and formation quality is averaged over all trials. We plot the trial-averaged mean energy expenditure and formation quality versus time, and use these plots to compare each behaviour against one another. Unfortunately, due to the computational intensity of BHV_AssignmentRegistration and the limitations of our computer, for BHV_AssignmentRegistration only 20 AUVs were simulated, resulting in a hexagonal lattice of five rows of four AUVs. This methodology was repeated with 20 AUVs with all behaviours except for BHV_AttractionRepulsion for a 4×5 square lattice with side length of $300m$.

To compare the effect of changing communication rate on formation construction, we use the same methodology with ping periods of $5s$, $15s$, $30s$, $60s$, and $120s$. In this case however, we use 20 AUVs, limit the desired formation to the hexagonal lattice, and only perform 2 trials for each ping period.

Finally, to compare the effect of AUV loss during formation construction, we use the same methodology, but remove a random AUV every $275s$ until 5 AUVs are removed. As with the communication rate test, we use 20 AUVs, limit the desired formation to the hexagonal lattice, and only perform 2 trials.

4.4.2 Formation Maintenance

To compare how efficiently each formation control behaviour is able to maintain the desired formation, we use the two scenarios illustrated in figure 4.3 and use the following methodology - first, the 25 AUVs are initialized in the ideal hexagonal lattice formation of five rows of five AUVs, with a separation distance of $300m$; we then instruct the swarm to maintain

this formation in the presence of ocean currents; the swarm is left to maintain this formation for approximately $7200s$ (about 2 hours), during which the energy expenditure of all AUVs is recorded, as well as the formation quality metric; The mean energy expenditure over all AUVs is then computed; finally, this process is repeated 2 times, and the mean energy expenditure and formation quality is averaged over all trials. We plot the trial-averaged mean energy expenditure and formation quality versus time, and use these plots to compare each behaviour against one another. This methodology was repeated with 20 AUVs with all behaviours except for BHV_AttractionRepulsion for a 4×5 square lattice with side length of $300m$. Note that for all formation maintenance trials, the centroid of the swarm is initially centred at the local coordinates of $(0, 0)$.

To compare the effect of changing communication rate on how well each behaviour can maintain the formation, we use the same methodology with ping periods of $5s$, $15s$, $30s$, $60s$, and $120s$. In this case however, we use 20 AUVs, limit the desired formation to the hexagonal lattice, and only perform 2 trials for each ping period.

Finally, to compare the effect of AUV loss during maintenance of the formation, we use the same methodology, but remove a random AUV every $1200s$ until 5 AUVs are removed. As with the communication rate test, we use 20 AUVs, limit the desired formation to the hexagonal lattice, and only perform 2 trials.

4.4.3 Formation Ocean Propulsion

Finally, as a general test of how well each behaviour is able to perform the ultimate objective of constructing a formation, maintaining it, and utilizing ocean currents for propulsion, we use the scenario of realistic Red Sea ocean currents, and use the following methodology - first, 20 AUVs are randomly initialized in a $200m \times 200m$ box; we then instruct the AUVs to form a hexagonal lattice of four rows of five AUVs, with a separation distance of $300m$; the swarm is then left to drift freely in the simulated ocean currents, until the mission reaches an elapsed time of approximately $18000s$ (about 5 hours) at which point the simulation is stopped; during the entire mission the energy expenditure of all AUVs is recorded, as well as the formation quality metric; The mean energy expenditure and the mean distance travelled over all AUVs is computed; finally, this process is repeated 2 times, and the mean energy expenditure, the formation quality, and the mean distance travelled is averaged over all trials. We plot the trial-averaged mean energy expenditure, formation quality, and the mean distance travelled versus time, and use these plots to compare each behaviour against one another. In addition, we repeat these 2 trials with all behaviours except for BHV_AttractionRepulsion for a square 4×5 lattice with side length of $300m$.

4.5 Summary

This chapter has given the reader an overview of the architecture used to simulate a swarm of AUVs running the four formation control behaviours, and in particular, provided the details of how vehicle energy expenditure is computed, how realistic ocean currents are simulated, and how the quality of the swarm formation is assessed. In addition, the comparison scenarios used to evaluate the efficacy of each formation control behaviour were described, and a brief explanation of the comparison testing methodology was provided.

The remainder of this thesis is organized into two chapters. Chapter 5 provides the results and an analysis of the comparison metrics, and chapter 6 details a discussion of the results of the analysis, conclusions drawn from this research, and future work to be undertaken.

Chapter 5

Results and Analysis of Metrics

Now that we have given the reader an explanation of the four formation control behaviours, as well as the simulation infrastructure developed to test these behaviours, and finally the testing methodology designed to compare the efficacy of each behaviour against one another, we present some results in this chapter that allow us to analyse how well each behaviour performs. We present graphs using the formation quality metric as well as swarm energy expenditure resulting from undertaking the testing methodology described in the previous chapter, and provide a brief analysis of these graphs as well as qualitative observations from watching each behaviour in action during simulations.

5.1 Scenario 1 - Formation Construction

As described in section 4.4.1, we test each behaviour's ability to construct a desired formation using the scenario of no ocean currents. For the first test, we instruct each behaviour to construct hexagonal and square lattices, and monitor energy consumption and the formation quality metric; for the second test, we modify the communications rate from $5s$ up to $120s$, and observe how this affects the construction process of a hexagonal lattice; and for the final test, we introduce node losses to the swarm during construction of a hexagonal lattice, and note any effects.

5.1.1 Construction

We begin with plots of the AUV trajectories during construction of the desired formation, as shown in the following figures. These plots illustrate the typical behaviour of each of the formation control algorithms during simulation. As mentioned previously, all AUVs begin at a random position within a $200m \times 200m$ box centred at the origin.

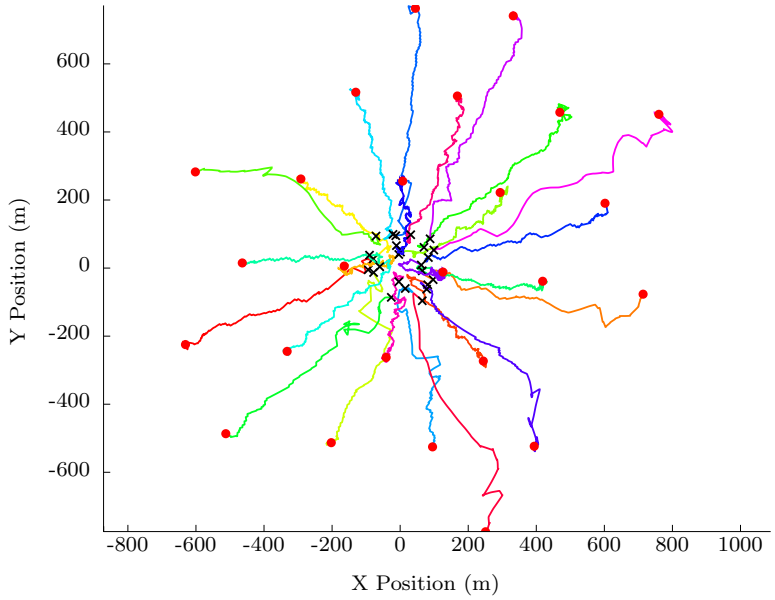

Figure 5.1: Trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_AttractionRepulsion (trial 2); black crosses indicate starting positions, red circles indicate final positions.

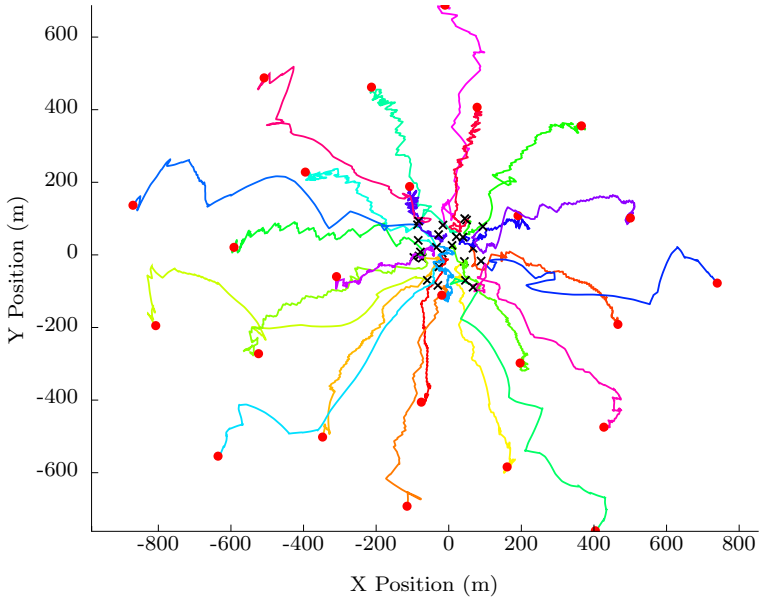

Figure 5.2: Trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_AttractionRepulsion (trial 4); black crosses indicate starting positions, red circles indicate final positions - defects are apparent in the final lattice.

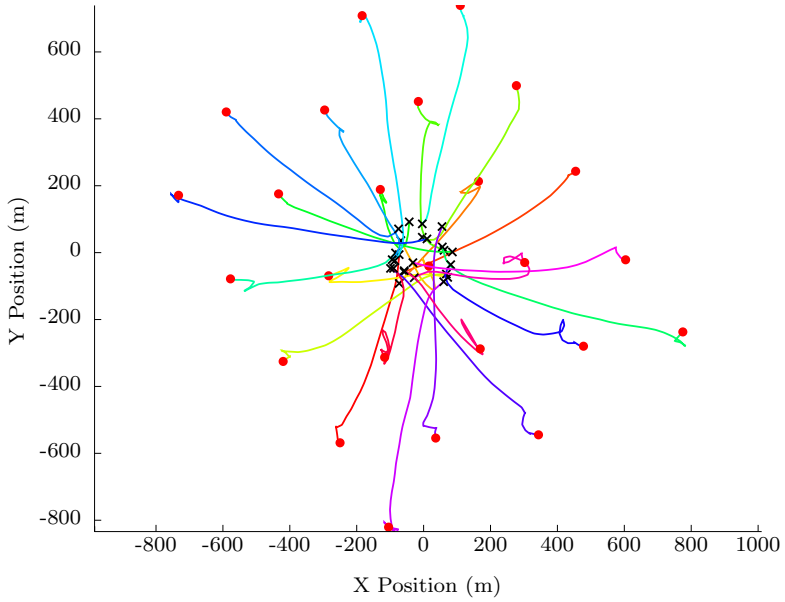

Figure 5.3: Trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_PairwiseNeighbourReferencing (trial 5); black crosses indicate starting positions, red circles indicate final positions.

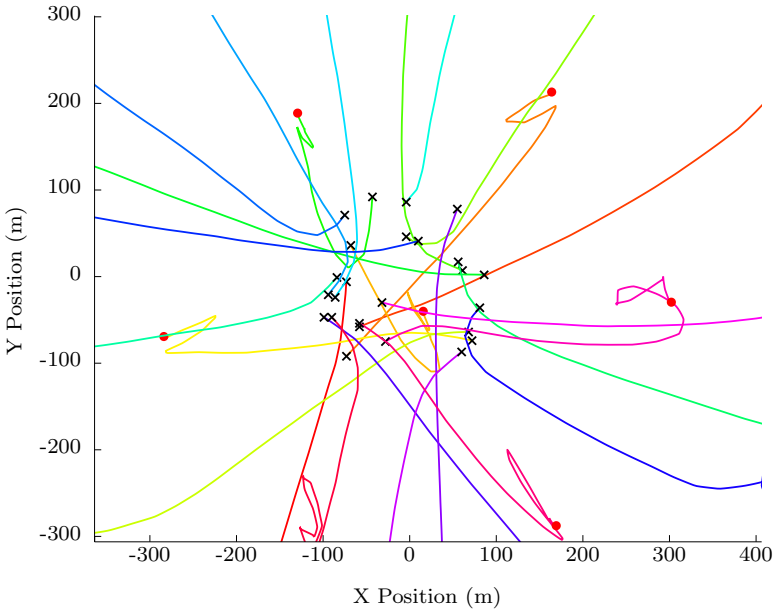

Figure 5.4: Zoomed-in view of the trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_PairwiseNeighbourReferencing (trial 5).

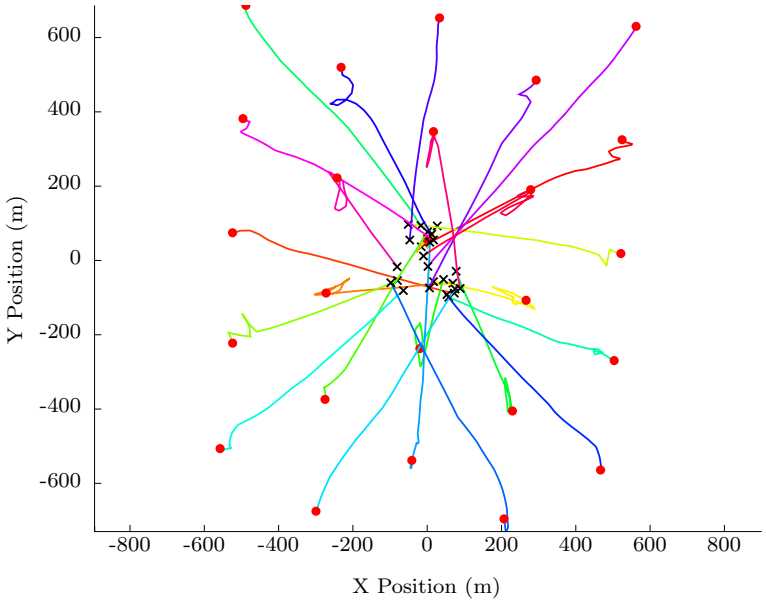

Figure 5.5: Trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_RigidNeighbourRegistration (trial 4); black crosses indicate starting positions, red circles indicate final positions.

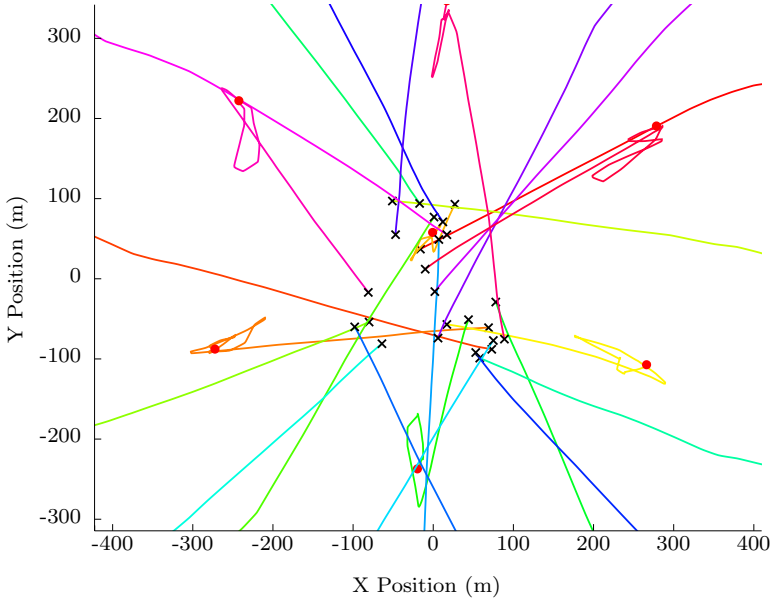

Figure 5.6: Zoomed-in view of the trajectories of 25 AUVs during hexagonal lattice formation construction with no ocean current - BHV_RigidNeighbourRegistration (trial 4).

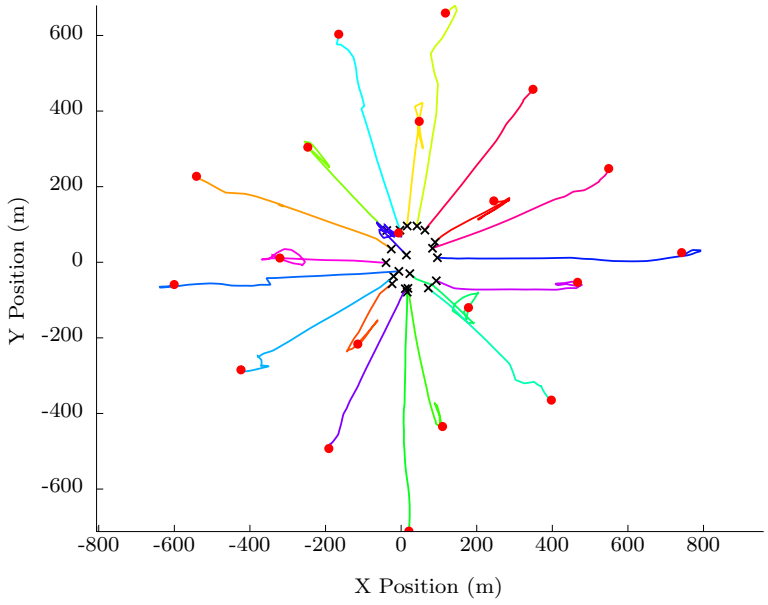

Figure 5.7: Trajectories of 20 AUVs during hexagonal lattice formation construction with no ocean current - BHV_AssignmentRegistration (trial 5); black crosses indicate starting positions, red circles indicate final positions.

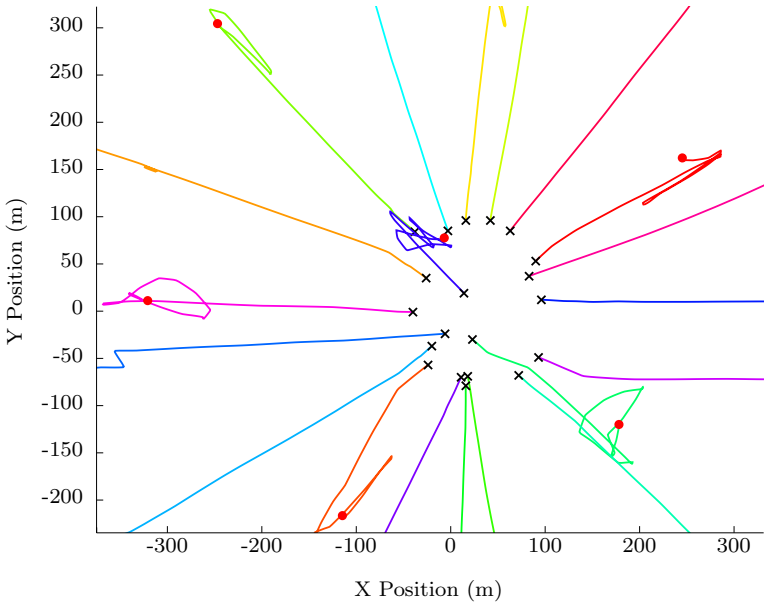

Figure 5.8: Zoomed-in view of the trajectories of 20 AUVs during hexagonal lattice formation construction with no ocean current - BHV_AssignmentRegistration (trial 5).

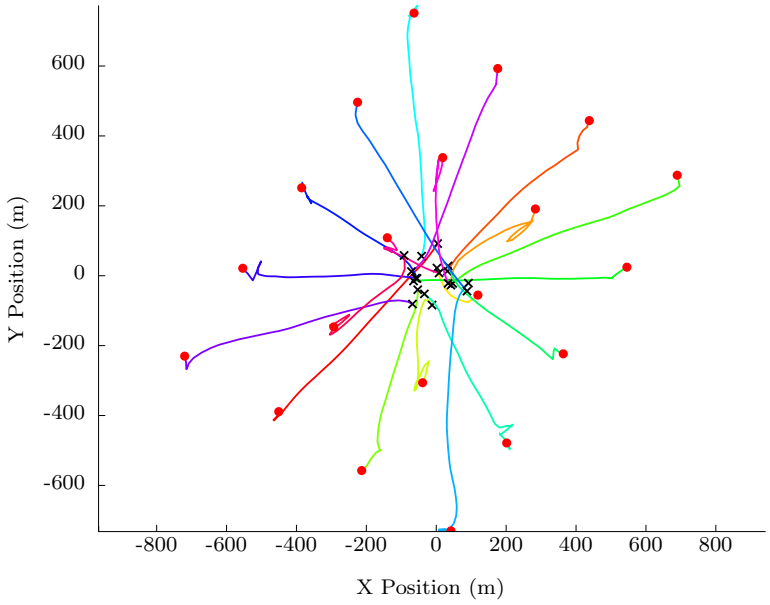

Figure 5.9: Trajectories of 20 AUVs during square lattice formation construction with no ocean current - BHV_PairwiseNeighbourReferencing (trial 1); black crosses indicate starting positions, red circles indicate final positions.

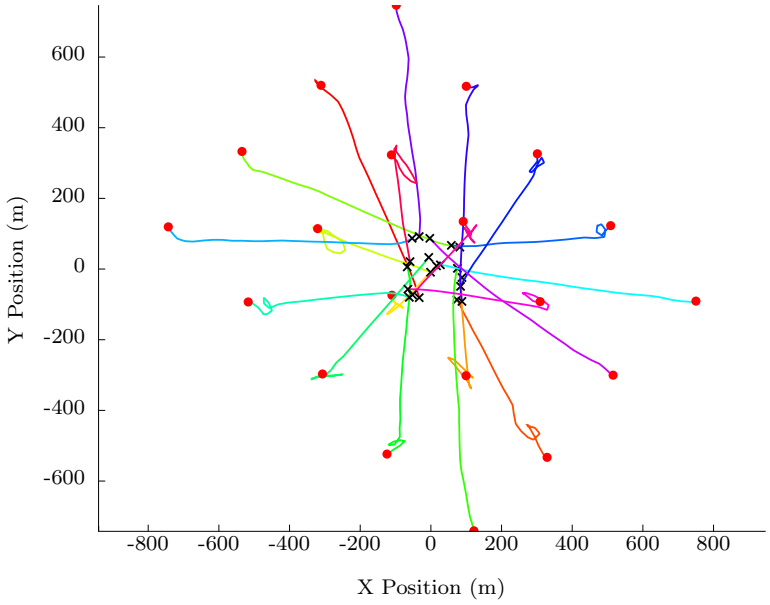

Figure 5.10: Trajectories of 20 AUVs during square lattice formation construction with no ocean current - BHV_RigidNeighbourRegistration (trial 1); black crosses indicate starting positions, red circles indicate final positions.

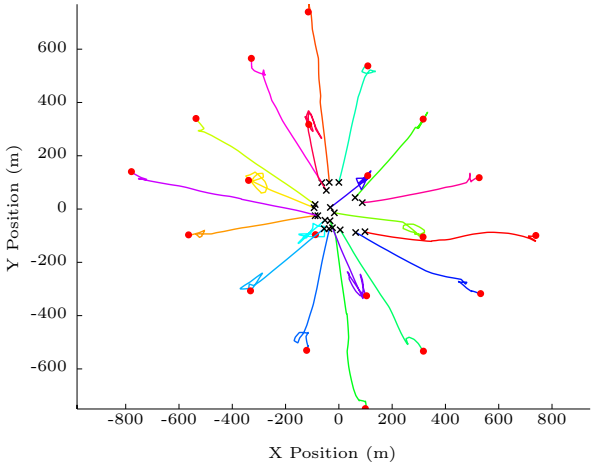

Figure 5.11: Trajectories of 20 AUVs during square lattice formation construction with no ocean current - BHV_AssignmentRegistration (trial 1); black crosses indicate starting positions, red circles indicate final positions.

BHV_AttractionRepulsion

Figures 5.1 and 5.2 illustrate typical trajectories of the AUVs during the construction of a hexagonal lattice formation using the BHV_AttractionRepulsion behaviour. Examining these figures, we can make a few observations. Firstly, the trajectories are extremely chaotic, with a lot of zigzagging movement occurring as the AUVs move toward their final positions; this movement can be explained by the fundamental fact that this behaviour is based upon an attraction/repulsion potential function - as the AUVs travel, they continuously select two neighbours according to the rules specified in section 3.4, and in doing so, are continuously influenced by different sets of neighbours, resulting in this back and forth zigzag movement. Secondly, since the user cannot specify a formation plan and the formation is constructed in a dynamic, emergent manner, the final constructed lattice does not have the ideal shape. Thirdly, the constructed lattice can end up with either a good 'quality', as seen in figure 5.1, or a bad 'quality' that contains defects, as seen in figure 5.2; looking at figure 5.1, we can see that the AUVs have positioned themselves such that they make the desired $300m$ equilateral triangles with any two neighbours; however, this is not the case with figure 5.2 - looking closely, there are actually two subsets of AUVs that have each produced a nice hexagonal lattice, and a 'hole' appears around the position $(-300, -300)$; what has occurred here is that the lattice has 'fractured' into two subsets, via the mechanism previously illustrated in figure 3.8 and elaborated upon in section 3.4. Unfortunately, this fracturing is a major drawback of the BHV_AttractionRepulsion behaviour, and can result in significant lattice defects during construction, and in lattice separation when drifting in ocean currents.

BHV_PairwiseNeighbourReferencing

Figures 5.3 and 5.9 illustrate typical trajectories of the AUVs during the construction of a hexagonal lattice and square lattice, respectively, using the BHV_PairwiseNeighbourReferencing behaviour. Examining these figures we can see that this behaviour constructs the desired lattices quite efficiently - each AUV has a pre-designated position in the user-specified formation plan, and they move toward that position somewhat directly. However, examining figure 5.4, which shows a zoomed-in view of the beginning of the trajectories, we see that the path of each AUV tends to 'arc' away from its final position, before each AUV moves back toward its final lattice position. This 'arcing' behaviour may result in unnecessary energy expenditure.

BHV_RigidNeighbourRegistration

Figures 5.5 and 5.10 illustrate typical trajectories of the AUVs during the construction of a hexagonal lattice and square lattice, respectively, using the BHV_RigidNeighbourRegistration behaviour. As with BHV_PairwiseNeighbourReferencing, this behaviour constructs the desired lattices very efficiently. Unlike BHV_PairwiseNeighbourReferencing however, each AUV approaches their pre-designated position in the formation plan directly - looking at figure 5.6, which shows a zoomed-in view of the beginning of the trajectories, we observe that the path of each AUV is quite straight, from their starting positions right up to their final positions.

BHV_AssignmentRegistration

Figures 5.7 and 5.11 illustrate typical trajectories of the AUVs during the construction of a hexagonal lattice and square lattice, respectively, using the BHV_AssignmentRegistration behaviour. These figures lead us to a couple of observations. Firstly, as with BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration, this behaviour appears to construct the desired lattices quite efficiently - each AUV moves toward a position in the desired formation plan fairly directly, along straight-line paths. Secondly, if we examine figure 5.8, which shows a zoomed-in view of the beginning of the trajectories, we can see that the trajectories essentially do not cross one another, unlike BHV_RigidNeighbourRegistration in figure 5.6. This is of course a direct result of the dynamic assignment portion of the BHV_AssignmentRegistration algorithm, explained in detail in section 3.7. Consequently, this algorithm has the potential to improve upon the energy efficiency of previous algorithms, by avoiding unnecessary AUV traversal across the paths of other vehicles.

Behaviour Comparison

We now present graphs for the mean energy expenditure over all AUVs, averaged over the five trials, versus mission time, as well as the formation quality versus mission time, for each

behaviour, and elaborate upon some observations gleaned from these datasets.

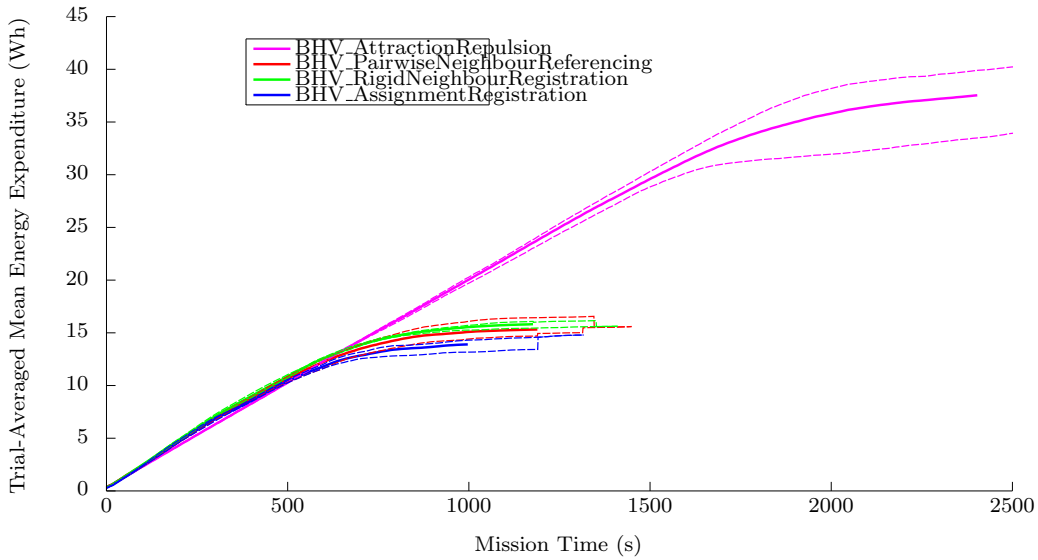

Figure 5.12: Trial-averaged mean energy expenditure over all AUVs during hexagonal lattice formation construction for each behaviour with no ocean current; solid lines indicate trial-averaged mean energy expenditure, dashed lines indicate minimum and maximum envelopes of mean energy expenditure from all trials.

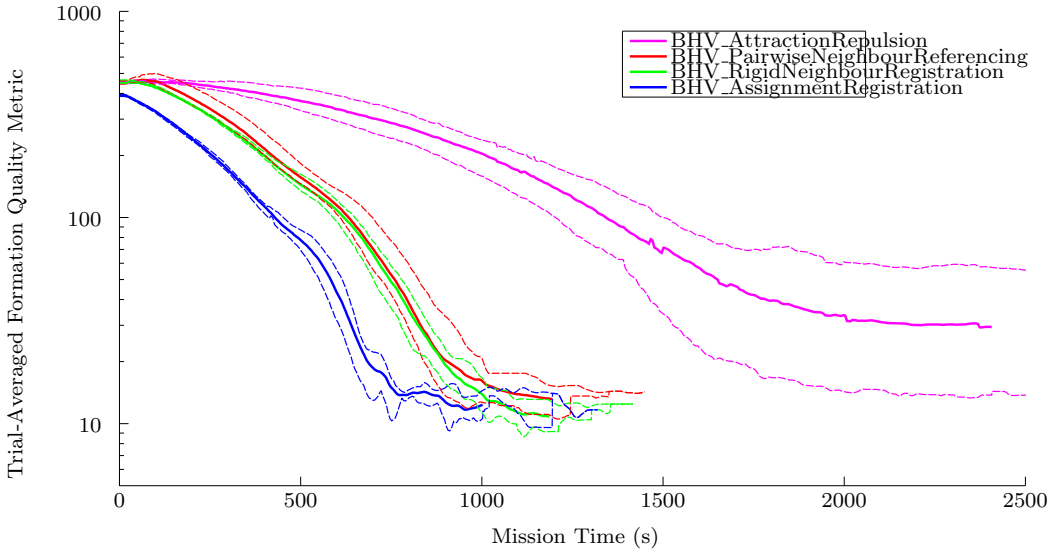

Figure 5.13: Trial-averaged formation quality metric during hexagonal lattice formation construction for each behaviour with no ocean current; solid lines indicate trial-averaged formation quality metric, dashed lines indicate minimum and maximum envelopes of formation quality metric from all trials.

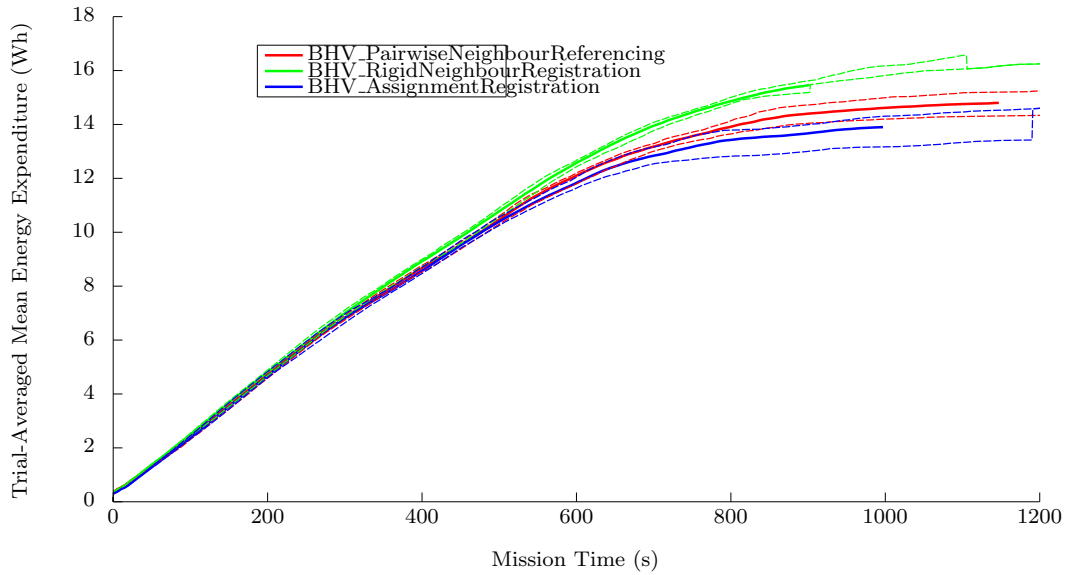

Figure 5.14: Trial-averaged mean energy expenditure over all AUVs during square lattice formation construction for each behaviour with no ocean current; solid lines indicate trial-averaged mean energy expenditure, dashed lines indicate minimum and maximum envelopes of mean energy expenditure from all trials.

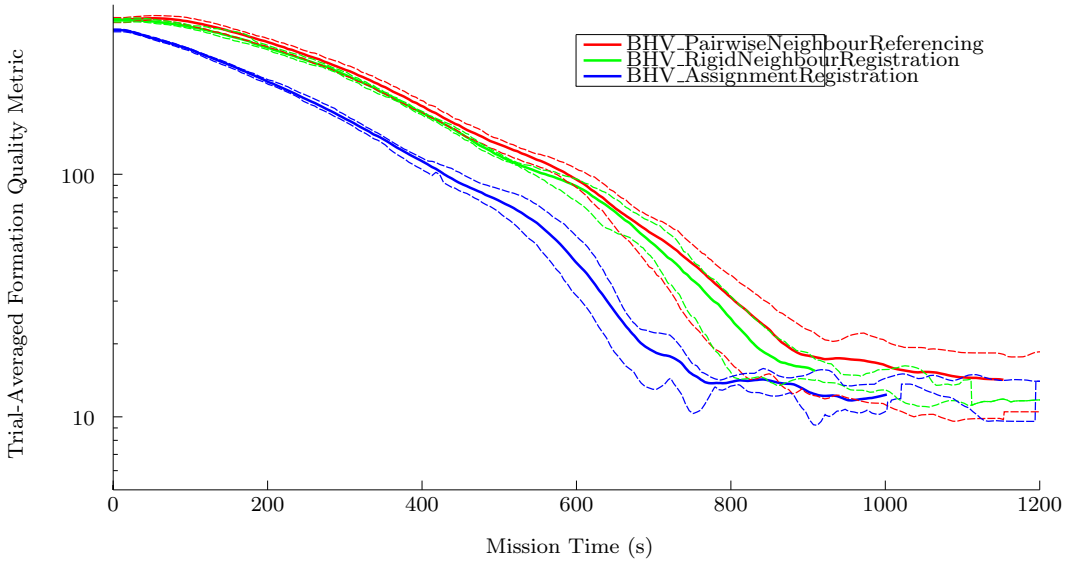

Figure 5.15: Trial-averaged formation quality metric during square lattice formation construction for each behaviour with no ocean current; solid lines indicate trial-averaged formation quality metric, dashed lines indicate minimum and maximum envelopes of formation quality metric from all trials.

Looking at figure 5.12, we can see that the chaotic movement of BHV_AttractionRepulsion during construction causes this behaviour to use a far greater amount of energy in comparison to all other behaviours. In addition, the plots in figure 5.13 indicate that BHV_AttractionRepulsion also constructs formations with a large variance in their quality metric, as illustrated by the maximum and minimum envelopes of the metric for this algorithm. This variance is due to the fact that this behaviour can create formations that have a tendency to fracture during construction, sometimes resulting in lattices which have defects (and thus a worse formation quality), as seen previously in figure 5.2.

We now compare the energy expenditure of the three other behaviours by examining figures 5.12 and 5.14. Interestingly, despite the fact that the AUV trajectories of BHV_RigidNeighbourRegistration are more direct than that of BHV_PairwiseNeighbourReferencing, the average energy expenditure of the former behaviour is greater in both tests; however, looking at figures 5.13 and 5.15, we can see that the quality of the lattice created by BHV_RigidNeighbourRegistration is generally better (remember, a quality metric that is lower corresponds to a better lattice 'quality'). Thus, we can deduce that the higher energy expenditure of this behaviour is used to maintain a higher quality formation. In addition, looking at the minimum and maximum envelopes of these two behaviours in figures 5.12 and 5.14, we observe that BHV_RigidNeighbourRegistration is significantly more consistent in its energy expenditure when compared to all other behaviours (it has a much lower variance), which may give it a slight advantage in usability.

Finally, we make a couple of observations about the BHV_AssignmentRegistration behaviour from an examination these four figures. Looking at figures 5.12 and 5.14, we can see that the energy expenditure of BHV_AssignmentRegistration is lower than all other behaviours. This is due to the fact that AUVs are dynamically assigned to positions in the formation based on a cost, causing the vehicles to generally have a shorter travel distance during formation construction in comparison to the other behaviours. The second consequence of this, as seen in figures 5.13 and 5.15, is that this behaviour reaches a specific formation quality level more quickly than any other behaviour. It is important to note a significant caveat of the BHV_AssignmentRegistration algorithm - its behaviour appears to be highly dependant on the user-specified $\delta\theta$ parameter. During these trials $\delta\theta = 45°$, meaning that the algorithm only has a choice between 8 different rotations; as a result, subsets of points tend to choose the same rotation as one another for their best point-set fit. However, if $\delta\theta$ is lowered, the algorithm has a larger number of rotation choices, and each point subset becomes more likely to select a rotation that is different to those chosen by other subsets. This results in more back and forth movement until the swarm converges on a formation, and in turn causes greater energy use. This behaviour can be seen later in this chapter, in sections 5.2, 5.3, and 5.4.

5.1.2 Construction with Varying Communications Rate

Here we present graphs of energy expenditure and formation quality during formation construction, using a number of different communications rates. We make some observations on the effect of these changes on the behaviour of each algorithm.

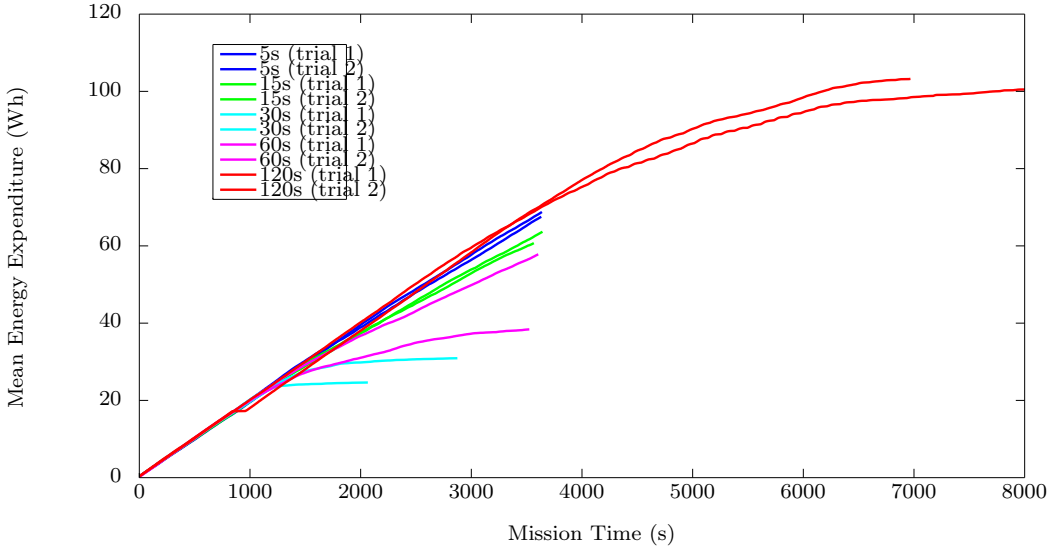

Figure 5.16: BHV_AttractionRepulsion - mean energy expenditure over all AUVs during hexagonal lattice formation construction with no ocean current and varying comms. rate.

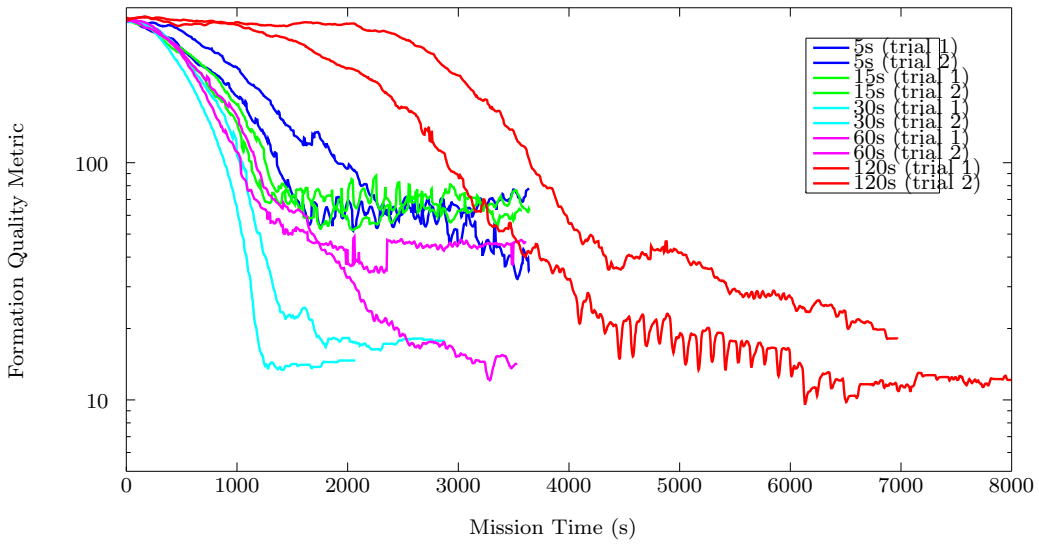

Figure 5.17: BHV_AttractionRepulsion - formation quality metric during hexagonal lattice formation construction with no ocean current and varying communications rate.

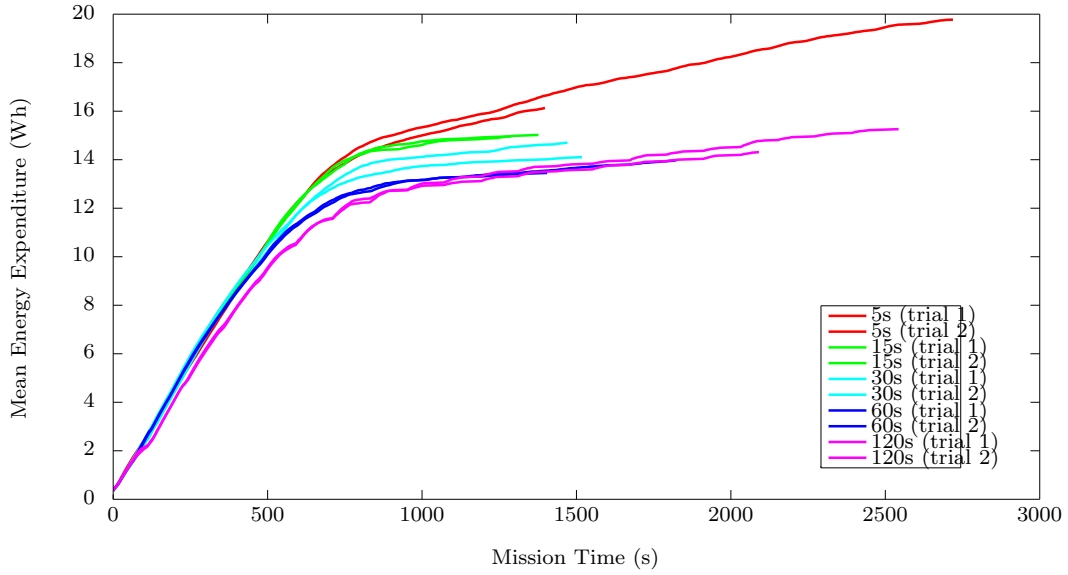

Figure 5.18: BHV_PairwiseNeighbourReferencing - mean energy expenditure over all AUVs during hexagonal lattice formation construction with no ocean current and varying communications rate.

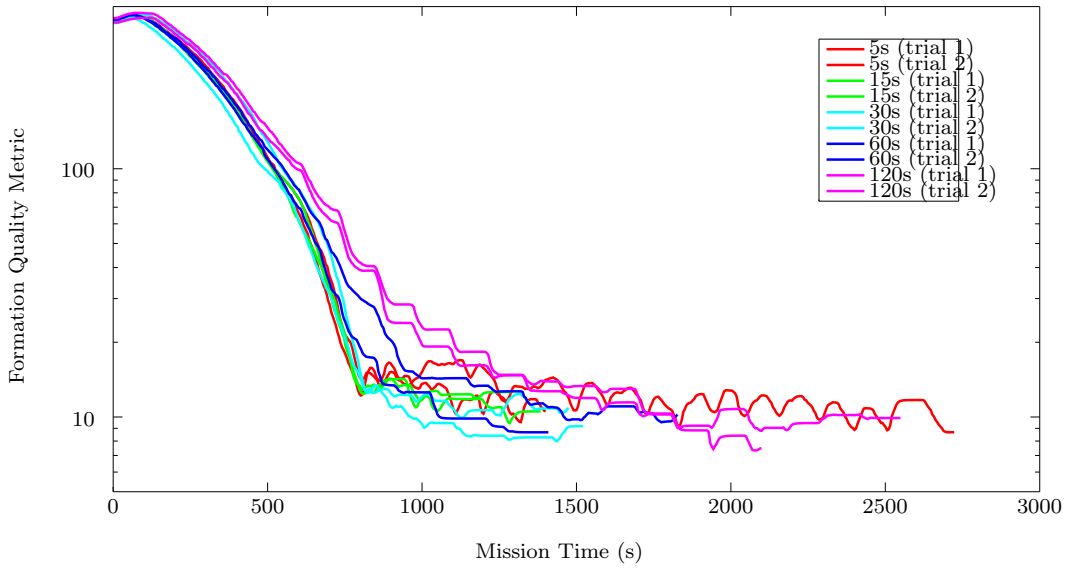

Figure 5.19: BHV_PairwiseNeighbourReferencing - formation quality metric during hexagonal lattice formation construction with no ocean current and varying communications rate.

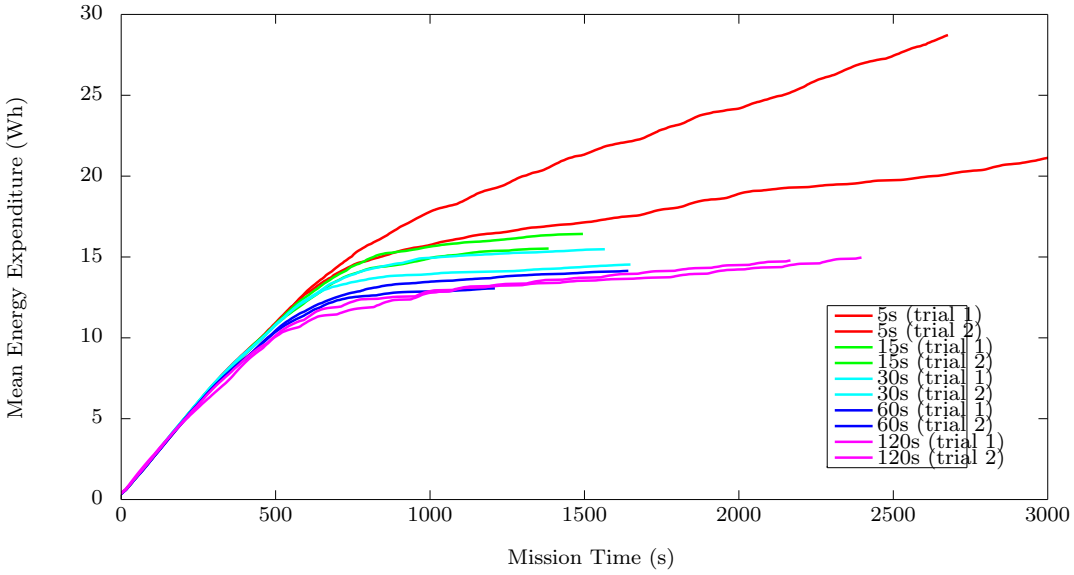

Figure 5.20: BHV_RigidNeighbourRegistration - mean energy expenditure over all AUVs during hexagonal lattice formation construction with no ocean current and varying communications rate.

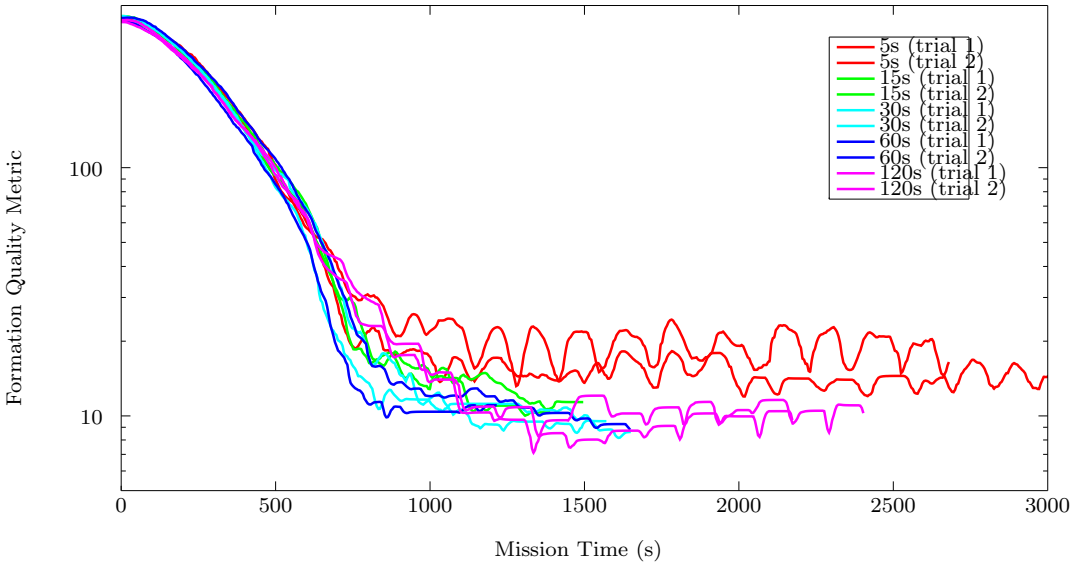

Figure 5.21: BHV_RigidNeighbourRegistration - formation quality metric during hexagonal lattice formation construction with no ocean current and varying communications rate.

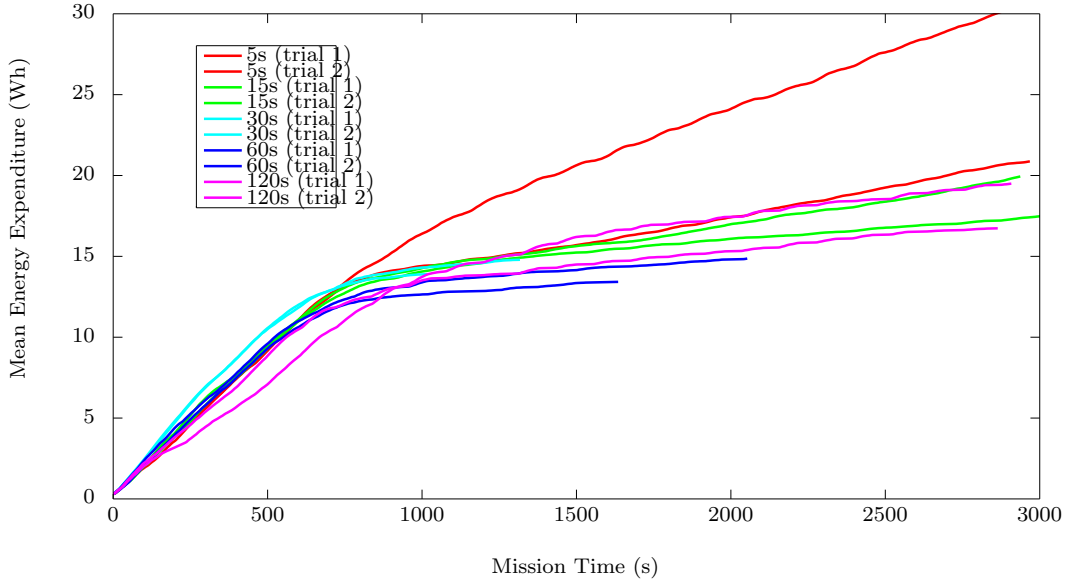

Figure 5.22: BHV_AssignmentRegistration - mean energy expenditure over all AUVs during hexagonal lattice formation construction with no ocean current and varying communications rate.

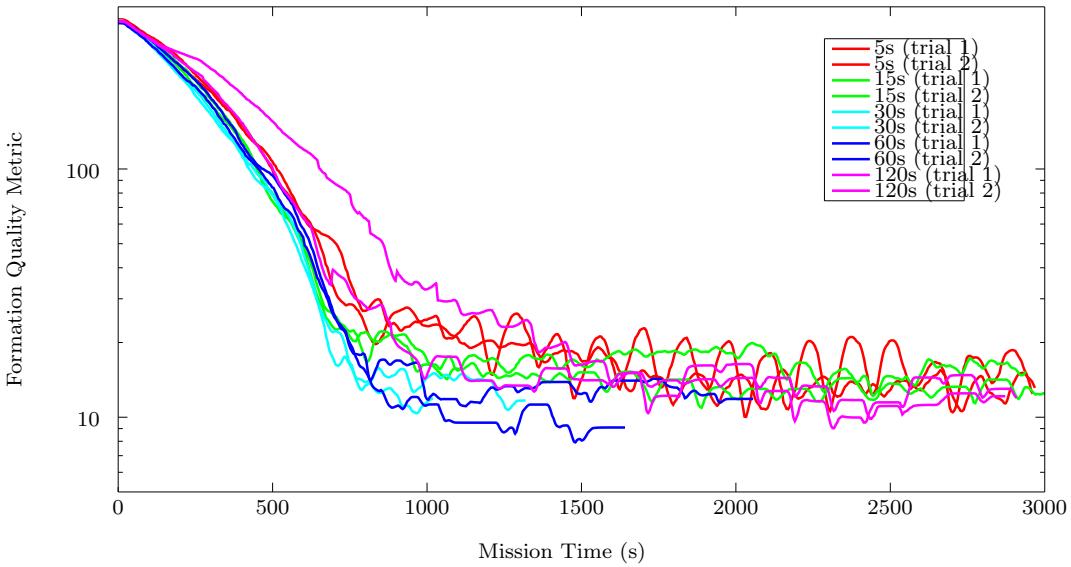

Figure 5.23: BHV_AssignmentRegistration - formation quality metric during hexagonal lattice formation construction with no ocean current and varying communications rate.

Figures 5.16 and 5.17 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, using the BHV_Attraction-Repulsion behaviour. Looking at these two graphs, we can make an interesting observation - the communications rate has a significant impact on the behaviour of the BHV_AttractionRepulsion algorithm. However, a higher communications rate (e.g. a $5s$ ping period) does not improve energy expenditure or improve the ability for the algorithm to more quickly achieve a specified formation quality. In fact, the opposite occurs for the following reason - when the ping period is low, vehicles update their neighbour positions more frequently; however, these range/bearing measurements are filtered to remove noise, thus introducing delay, and this delay is on top of the fact that neighbours have moved once the vehicle has estimated their relative position (in addition to the inherently noisy measurements); consequently, neighbour positions are inaccurate, and 'lag' behind the actual neighbour positions. This inaccuracy results in a strange behaviour - a vehicle detects a neighbour that is moving away from it, and thinks it is closer than it actually is, due to this 'lag'; as such, the vehicle moves further away from the neighbour to maintain the desired distance; its neighbour then sees that the vehicle has moved away, and so moves back towards the vehicle, but in the meantime, the updated neighbour position is further than the desired distance (again, due to 'lag'), and the vehicle also moves toward its neighbour; and in this way, the two vehicles 'oscillate' toward and away from one another continuously, a behaviour that was observed during simulations. Of course, this delay always occurs, but a higher ping period acts almost as a low-pass filter, helping to dampen this oscillatory effect, while lower ping periods appear to exacerbate these oscillations. However, looking at these two graphs, we can also note that higher ping periods ($60s$ and $120s$) also have a negative effect - this is likely because higher ping periods cause too large a delay when updating neighbour positions, causing the lattices to take a long time to form, resulting in the use of more energy. We can see that the default ping period of $30s$ appears to be near optimal for formation construction with BHV_AttractionRepulsion.

Figures 5.18 and 5.19 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, using the BHV_PairwiseNeighbourReferencing behaviour. These graphs appear to indicate that as the ping period increases, the mean energy expenditure tends to decrease, but at the cost of taking a larger amount of time to achieve a specified formation quality level. This is likely due to the fact that as the ping period increases, the vehicles tend to stop more frequently mid-construction, resulting in trajectories that tend to be straighter but which result in the desired formation taking longer to construct; at lower ping periods the 'arcing' behaviour of trajectories is generally more pronounced, a phenomena observed during simulations. However, we also notice that at a ping period of $5s$, the formation quality oscillates - this is due to the same oscillation phenomena described for low ping periods in the BHV_AttractionRepulsion algorithm.

Figures 5.20 and 5.21 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, using the BHV_RigidNeighbourRegistration behaviour. These graphs appear to indicate a similar effect of communications rate on formation construction as was observed in the BHV_PairwiseNeighbourReferencing behaviour. There is one significant difference however - the cause of lowering energy expenditure with increasing ping period is not due to a reduction in 'arcing' behaviour, as was the case with BHV_PairwiseNeighbourReferencing. The paths in this case are always fairly straight, but at lower ping periods this algorithm tends to have a larger amount of oscillatory or 'looping' movement at the end of each vehicle trajectory, causing a greater amount of energy expenditure, as well as a longer time to 'settle' to a specified formation quality level. This is likely due to phenomena similar to that seen in previous behaviours, where delay in measurements of neighbouring vehicle positions causes oscillations. As a result, we can see that the trajectories for the formation quality metric tends to improve as the ping period increases from $5s$ to $60s$; it then degrades at $120s$, simply because this ping period is too large to update neighbour positions in a timely manner. At $5s$ the oscillatory behaviour is very apparent in the wavelike path of the formation quality trajectory.

Figures 5.22 and 5.23 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, using the BHV_AssignmentRegistration behaviour. It is difficult to make any deductions from these graphs, as there do not appear to be any obvious trends. However, the oscillatory phenomena seen in previous behaviours is again quite apparent here, and can be seen in the formation quality trajectory for a $5s$ ping period.

To close, it is interesting to note that a higher communications rate does not necessarily corresponds to a better performance for any of the four formation control algorithms during formation construction - in fact, due to inherent latencies in updating neighbour positions leading to oscillatory behaviour, it is actually advantageous in most cases to use a ping period of $30s$ or more. Luckily for us, a higher ping period is better suited for the underwater environment, requiring less bandwidth and reducing the chance of message collisions.

5.1.3 Construction with Node Loss

Here we present graphs of energy expenditure and formation quality with loss of nodes during formation construction. A random AUV is selected and removed from the swarm every $275s$ until 5 AUVs are removed.

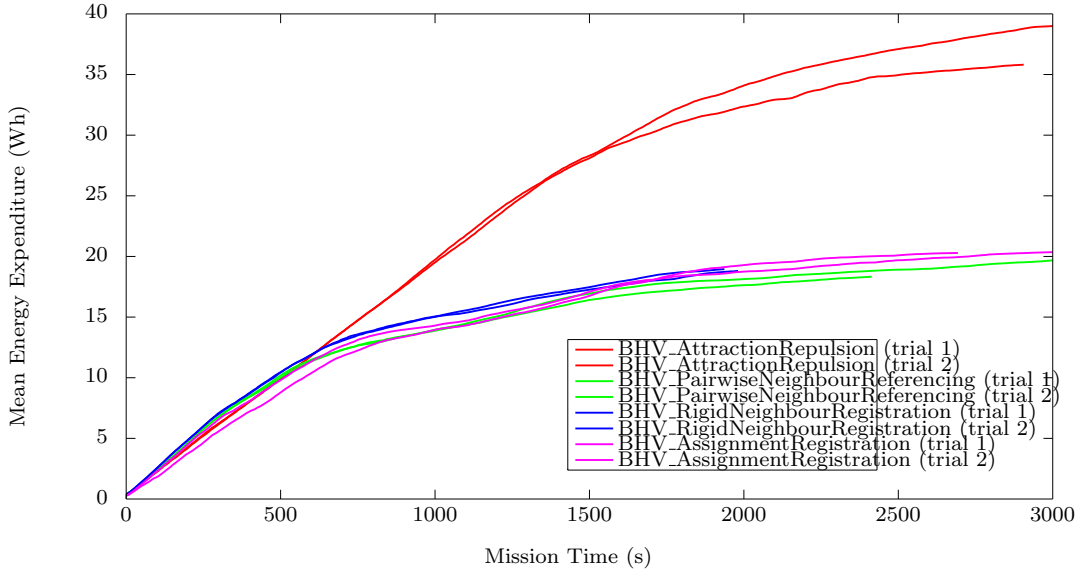

Figure 5.24: Mean energy expenditure over all AUVs during hexagonal lattice formation construction with no ocean current and with node loss.

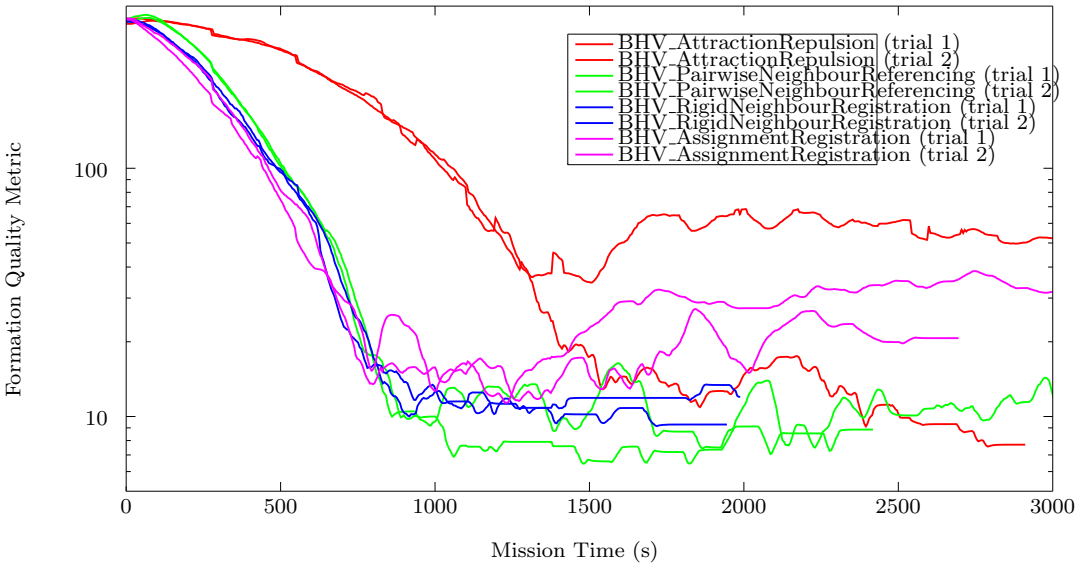

Figure 5.25: Formation quality metric during hexagonal lattice formation construction with no ocean current and with node loss.

Examining figures 5.24 and 5.25, node loss does not appear to have much effect on energy expenditure. However, looking at figure 5.25, we notice that when the fifth vehicle is lost (at around $1375s$), the formation quality noticeably worsens for both BHV_AttractionRepulsion (trial 1) and BHV_AssignmentRegistration (both trials). For BHV_AttractionRepulsion, this can be explained by the fact that a loss of a vehicle can increase the likelihood of defects or fractures occurring. For BHV_AssignmentRegistration, the loss of vehicles can significantly impact the ability for AUVs to find a good local fit for point subsets, resulting in the formation of a non-ideal lattice.

For BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration, node loss does not appear to have a great effect on either energy expenditure or formation quality. This is due to the inherent nature of these algorithms, which have pre-designated positions for AUVs in the user-specified formation plan. This means that when a node is lost, as long as there are still enough neighbours in the vicinity of a vehicle (at least 2 for BHV_PairwiseNeighbourReferincing, and at least 1 for BHV_RigidNeighbourRegistration), then the vehicle generally does well in maintaining its position, and thus the formation as a whole. Luckily, in each of these trials the situation never occurred where a vehicle was stranded (i.e. nodes were lost in such a way that an AUV no longer has contact with any neighbours); if this had occurred, it is very likely that the formation quality would have deteriorated significantly, as the lone vehicle was left to drift on its own. This is a downside of these two algorithms - they have no self-repairing capability. Generally speaking, BHV_RigidNeighbourRegistration is more robust to vehicle loss than BHV_PairwiseNeighbourReferencing, since in the former case a vehicle only requires a single neighbour in order to stay with the swarm, while in the latter case it requires at least two.

5.2 Scenario 2 - Formation Maintenance

We then tested each behaviour's ability to maintain a desired formation using the scenario of three current channels of linear velocity, as illustrated in figure 4.3 and described in section 4.4.2. As in the previous scenario, we first monitor the energy consumption and the formation quality metric as the swarm maintains a desired formation, then observe the effect of changing communications rate, and finally we observe how each algorithm copes with node loss.

5.2.1 Maintenance

We begin with plots of the AUV paths while the swarm maintains formation, as shown in the following figures. These plots illustrate the typical behaviour of each of the formation control algorithms during simulation. In these tests, the swarm begins in the ideal formation.

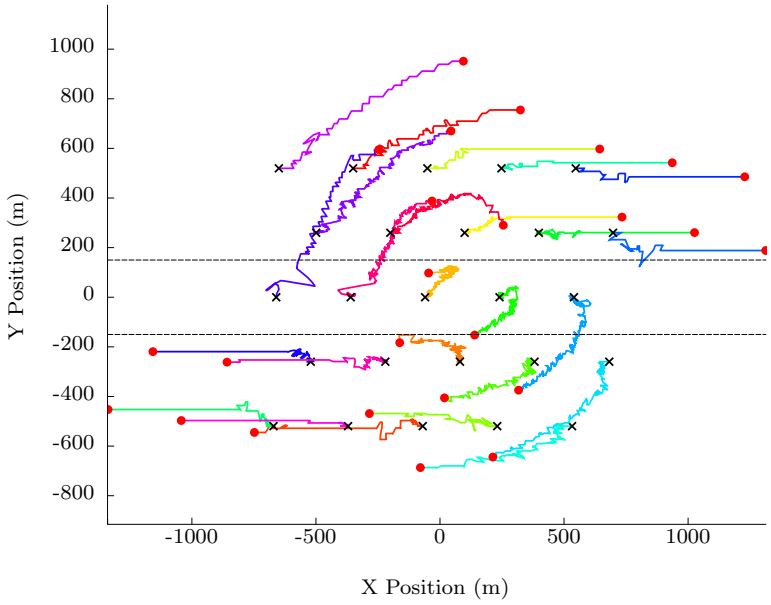

Figure 5.26: Trajectories of 25 AUVs during hexagonal lattice formation maintenance in three current channels - BHV_AttractionRepulsion (trial 2); black crosses indicate starting positions, red circles indicate final positions.

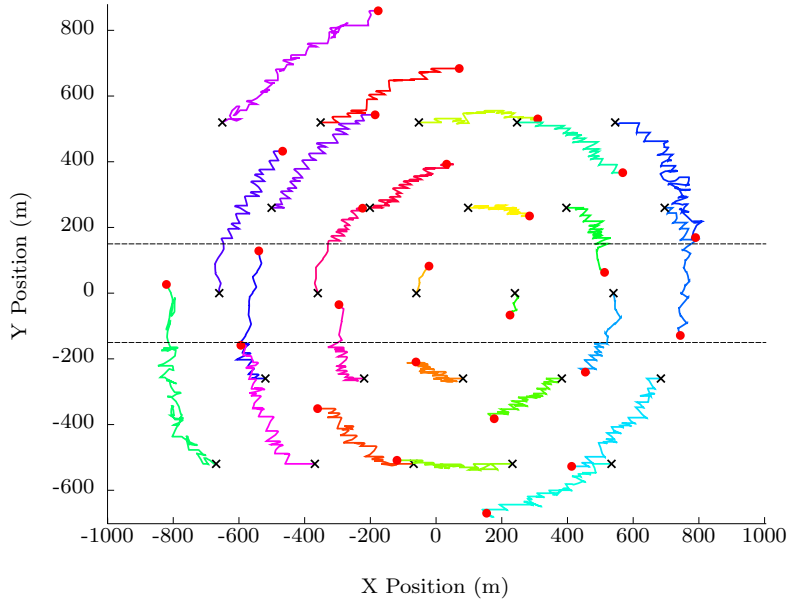

Figure 5.27: Trajectories of 25 AUVs during hexagonal lattice formation maintenance in three current channels - BHV_PairwiseNeighbourReferencing (trial 2); black crosses indicate starting positions, red circles indicate final positions.

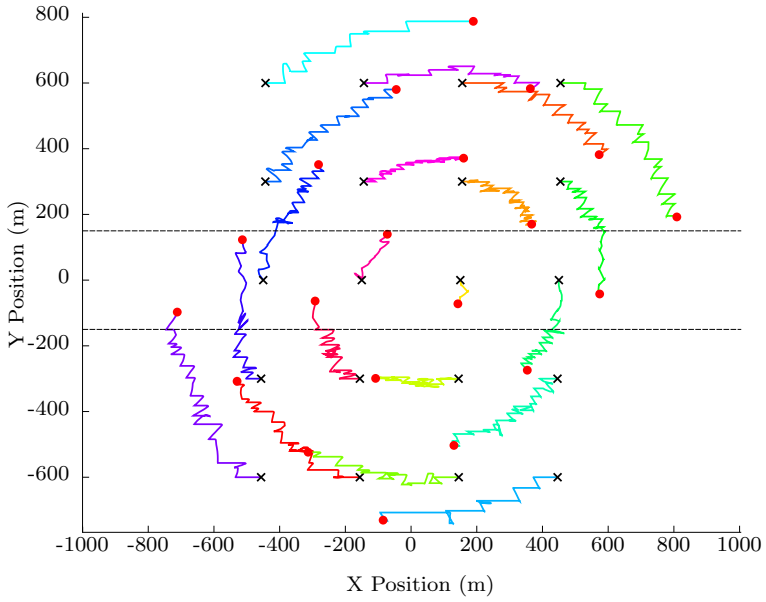

Figure 5.28: Trajectories of 20 AUVs during square lattice formation maintenance in three current channels - BHV_RigidNeighbourRegistration (trial 1); black crosses indicate starting positions, red circles indicate final positions.

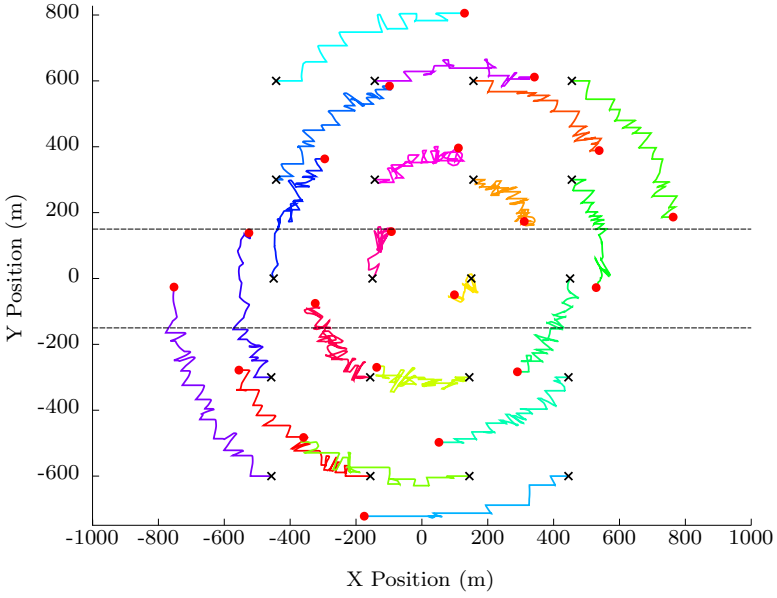

Figure 5.29: Trajectories of 20 AUVs during square lattice formation maintenance in three current channels - BHV_AssignmentRegistration (trial 2); black crosses indicate starting positions, red circles indicate final positions.

BHV_AttractionRepulsion

Figure 5.26 illustrates typical AUV trajectories of the BHV_AttractionRepulsion algorithm attempting to maintain a hexagonal lattice formation in this scenario, with the dashed lines indicating the boundaries between the three current channels. It is clearly apparent that, for the same reasons as described in previous sections, the lattice has fractured into at least 4 different groups of vehicles, as the upper and lower channels have pulled at the edges of the formation and 'torn' it apart.

BHV_PairwiseNeighbourReferencing

Figure 5.27 illustrates typical AUV trajectories of the BHV_PairwiseNeighbourReferencing algorithm attempting to maintain a hexagonal lattice formation in this scenario. It is interesting to note that the effect of the different current channels is clearly apparent in the trajectories of the AUVs - within the center channel (where there is no current), the AUV trajectories are much straighter, while the trajectories within the other channels are much more 'jagged', as the AUVs attempt to maintain formation by resisting vehicle drift caused by currents. Another interesting observation is that the entire formation rotates clockwise in the presence of these two opposite currents - this is exactly the type of behaviour you would expect of a rigid body if placed in a similar situation. Finally, we note that BHV_PairwiseNeighbourReferencing does a good job of maintaining the desired lattice formation in this three current channel scenario.

BHV_RigidNeighbourRegistration

Figure 5.28 illustrates typical AUV trajectories of the BHV_RigidNeighbourRegistration algorithm attempting to maintain a square lattice formation in this scenario (we show a square lattice for the sake of variety). As in BHV_PairwiseNeighbourReferencing, the effect of the different velocity current channels is again quite apparent in the trajectories of the AUVs, with straighter paths in the center channel. The rotation of the rigid body is even clearer in this plot, as the rectangular formation has rotated from 0° to approximately 30°. Finally, as with BHV_PairwiseNeighbourReferencing, BHV_RigidNeighbourRegistration is able to maintain the square lattice formation very well in this three current channel scenario.

BHV_AssignmentRegistration

Figure 5.29 illustrates typical AUV trajectories of the BHV_AssignmentRegistration algorithm attempting to maintain a square lattice formation in this scenario. Again we see the rotation of the formation, as well as the effect of the different current channels on the AUV trajectories. However, it appears that when compared to BHV_RigidNeighbourRegistration, this algorithm results in more AUV movement when attempting to maintain formation -

this can be seen when looking at some of the trajectories within the center channel, which appear to be more jagged when compared to those in the previous behaviour. Even so, BHV_AssignmentRegistration is able to maintain the desired formation quite well. We note an important caveat here - unlike during the tests of formation construction, the $\delta\theta$ parameter for this behaviour was set to 10° instead of the default 45°, to cope with the rotation induced in the formation by the currents of this scenario.

Behaviour Comparison

We now show some graphs of mean energy expenditure and formation quality for the two trials using each behaviour, and present some observations.

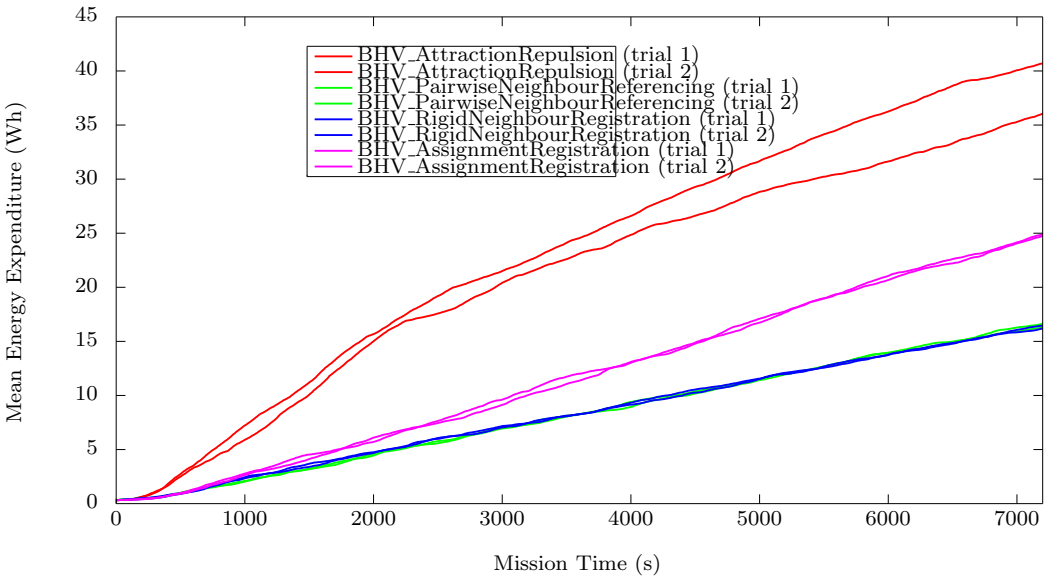

Figure 5.30: Mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels; two trials performed for each behaviour.

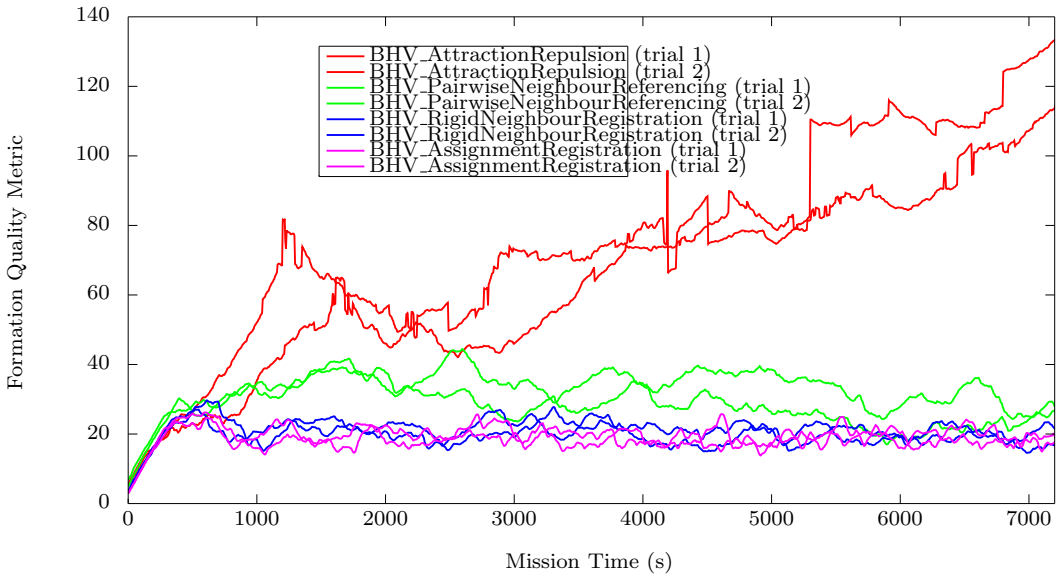

Figure 5.31: Formation quality metric while maintaining a hexagonal lattice formation in three current channels; two trials performed for each behaviour.

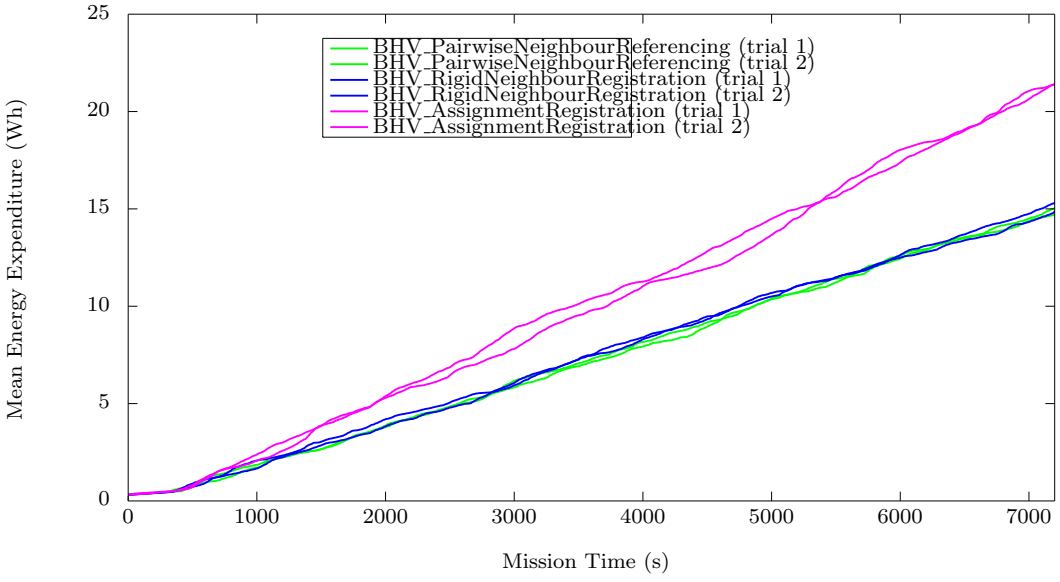

Figure 5.32: Mean energy expenditure over all AUVs while maintaining a square lattice formation in three current channels; two trials performed for three behaviours.

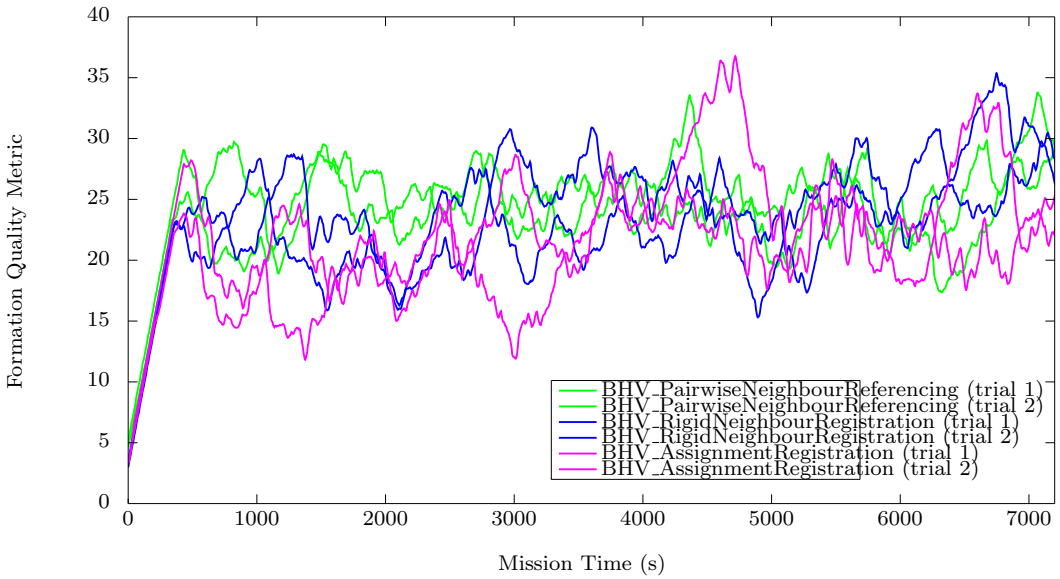

Figure 5.33: Formation quality metric while maintaining a square lattice formation in three current channels; two trials performed for three behaviours.

Figures 5.30 and 5.32 show the mean energy expenditure for two trials using each behaviour for formation maintenance in the three current channel scenario, for a hexagonal and square lattice respectively. Inspecting these two graphs, it is apparent that BHV_AttractionRepulsion expends energy very quickly while being unable to maintain the desired formation. BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration appear to use energy at an almost equal rate, while BHV_AssignmentRegistration uses energy at a faster rate than both these algorithms. Each of these three behaviours have a near-linear rate of energy expenditure, with BHV_AssignmentRegistration having the greatest slope of the three.

Figures 5.31 and 5.33 show the formation quality metric for two trials using each behaviour for formation maintenance in the three current channel scenario, for a hexagonal and square lattice respectively. Figure 5.31 demonstrates that BHV_AttractionRepulsion cannot maintain the desired formation, with the quality metric continuously increasing in both trials. This figure also appears to show that BHV_PairwiseNeighbourReferencing maintains a formation of a worse quality when compared to both BHV_RigidNeighbourRegistration and BHV_AssignmentRegistration, which maintain formations of very similar quality. However, figure 5.33 tells us a different story, where all three behaviours are able to maintain a formation of near equal quality, with BHV_AssignmentRegistration utilizing more energy to achieve this as seen in figure 5.32. Finally, we note that these three behaviours are remarkably consistent across both trials, in terms of both maintaining a specified quality and in rate of energy use.

5.2.2 Maintenance with Varying Communications Rate

In this section we present graphs of energy expenditure and formation quality with a variety of communications rates during formation maintenance in this scenario. We make some observations on the effect of this change on the behaviour of each algorithm.

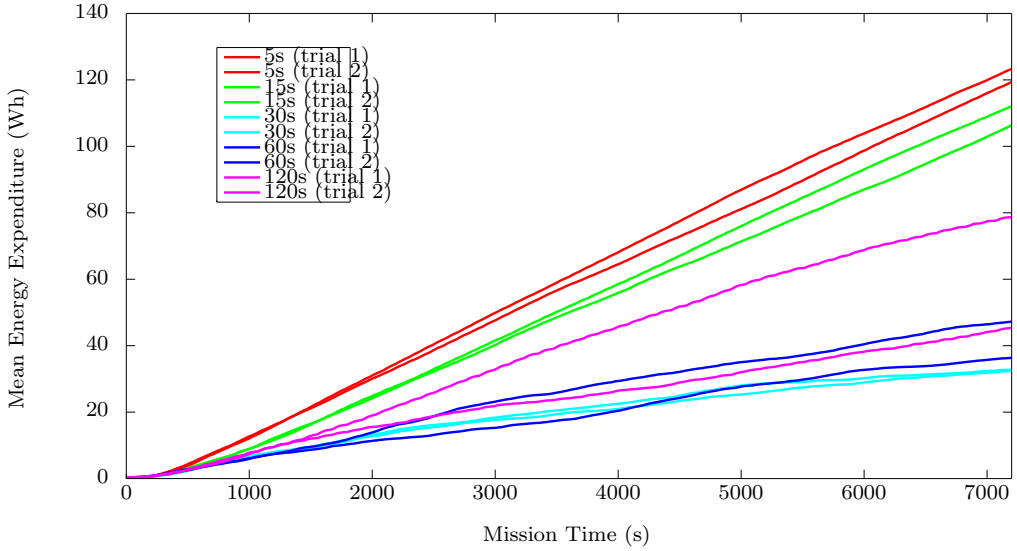

Figure 5.34: BHV_AttractionRepulsion - mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels and varying comms. rate.

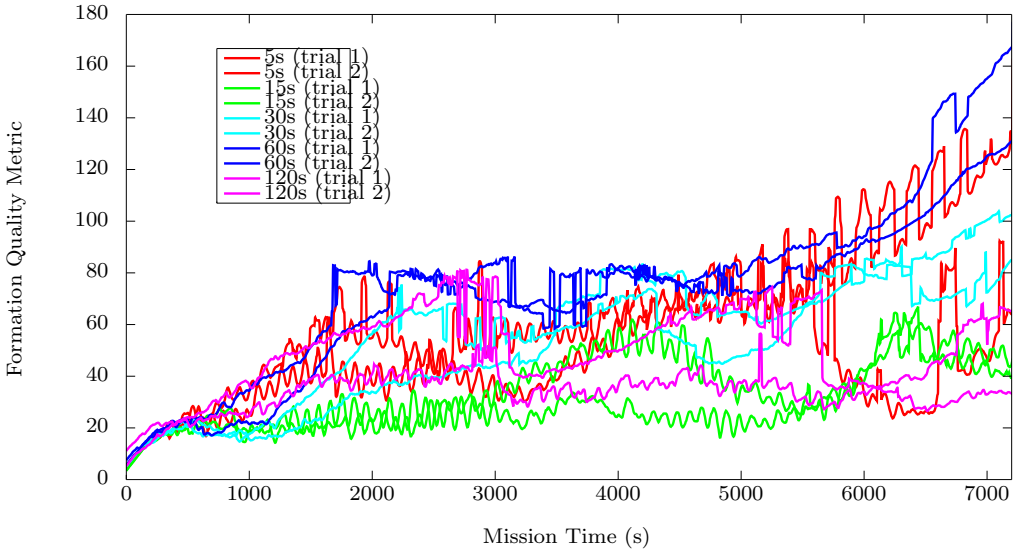

Figure 5.35: BHV_AttractionRepulsion - formation quality metric while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

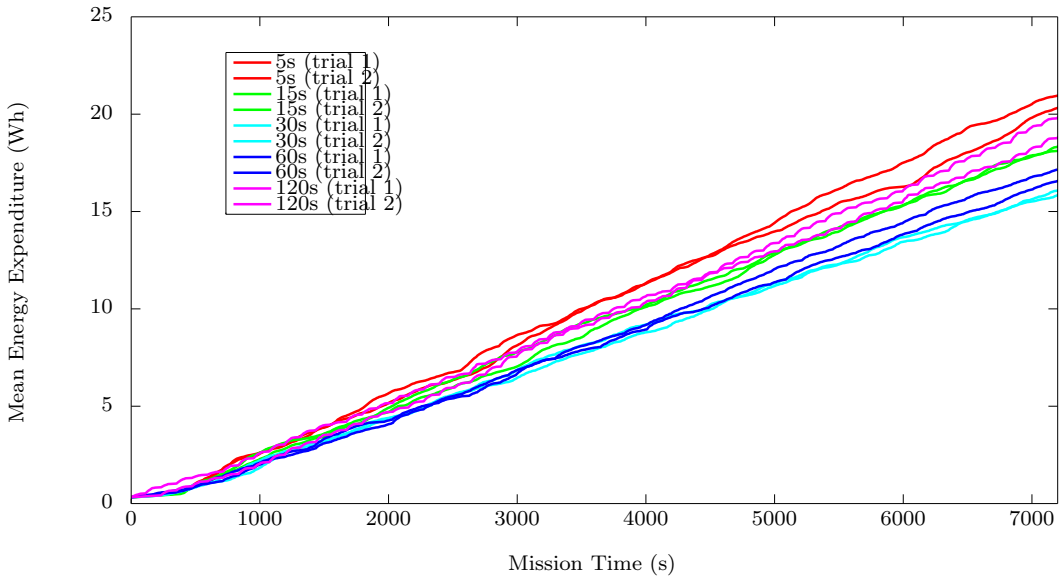

Figure 5.36: BHV_PairwiseNeighbourReferencing - mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

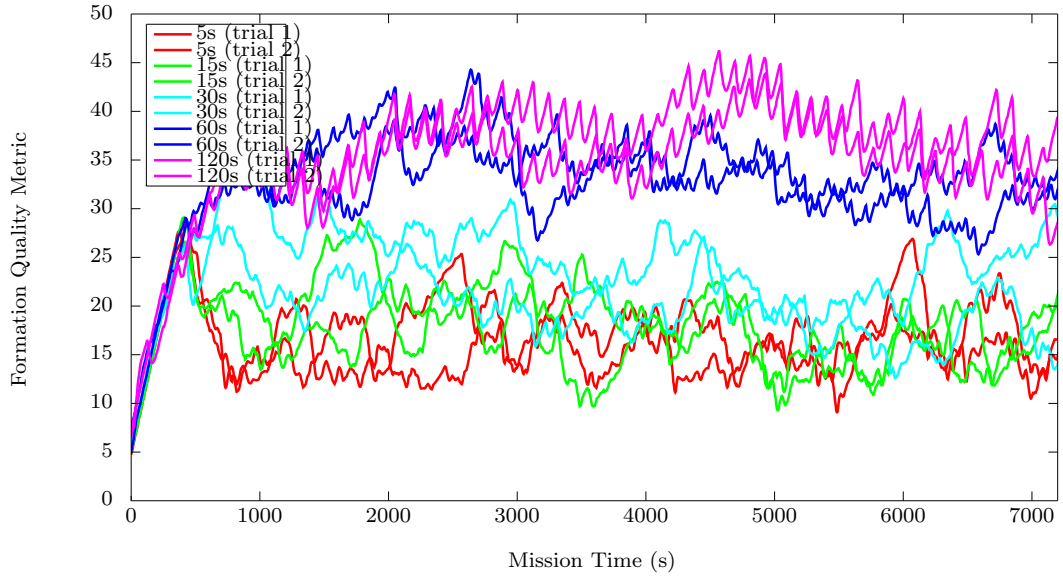

Figure 5.37: BHV_PairwiseNeighbourReferencing - formation quality metric while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

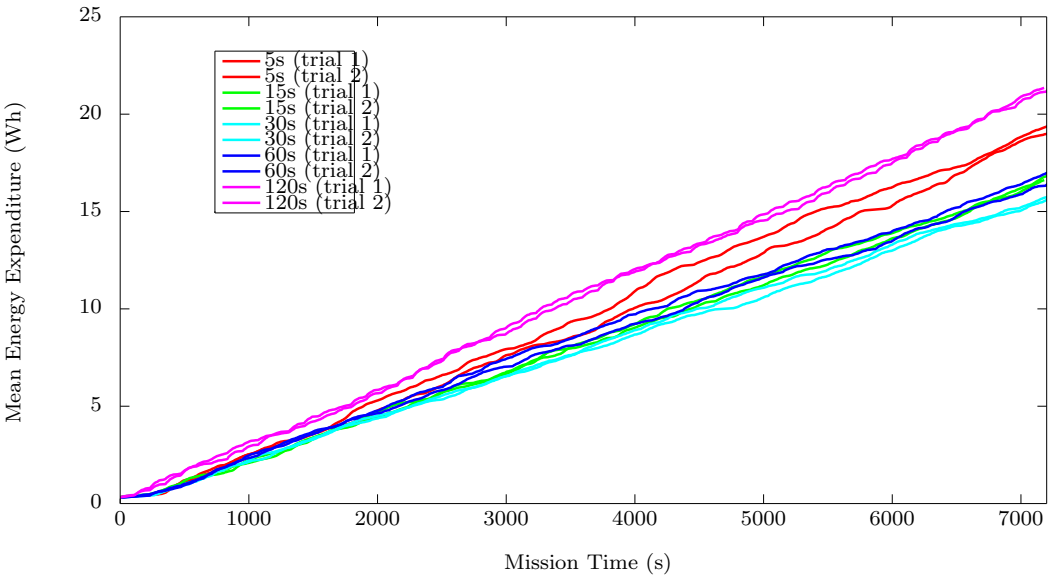

Figure 5.38: BHV_RigidNeighbourRegistration - mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

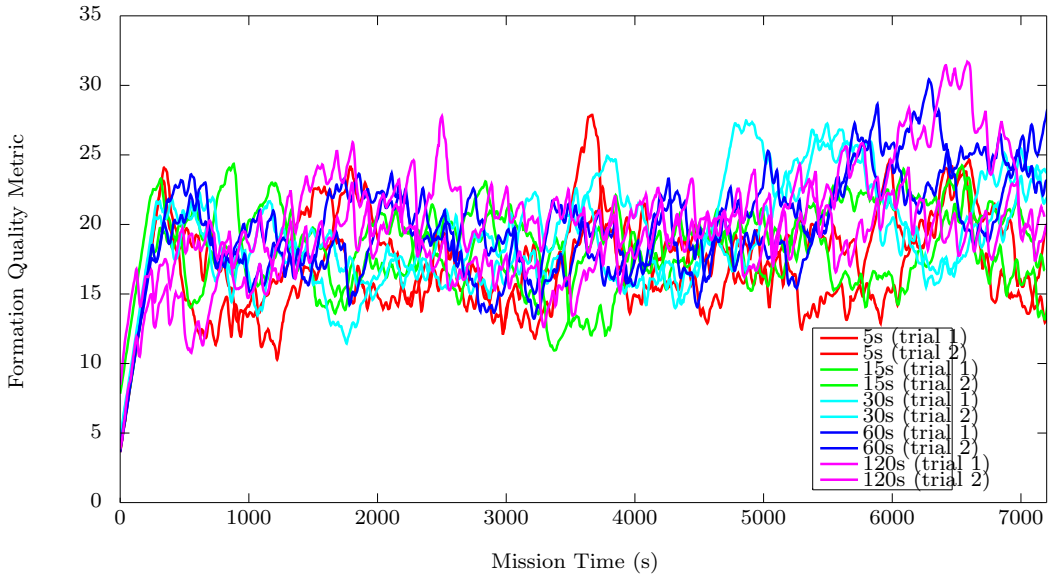

Figure 5.39: BHV_RigidNeighbourRegistration - formation quality metric while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

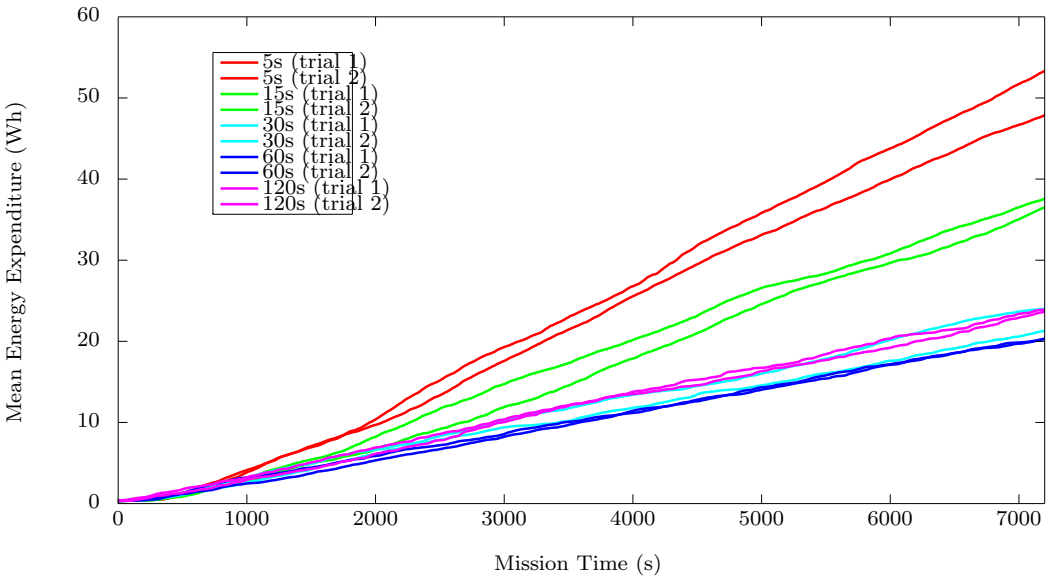

Figure 5.40: BHV_AssignmentRegistration - mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

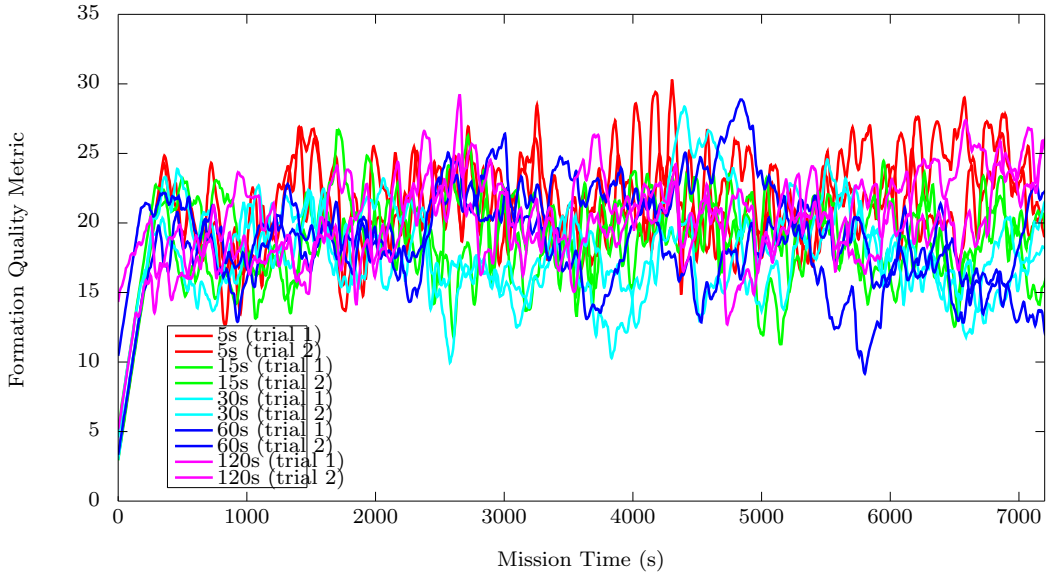

Figure 5.41: BHV_AssignmentRegistration - formation quality metric while maintaining a hexagonal lattice formation in three current channels and with varying communications rate.

Figures 5.34 and 5.35 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_Attraction-Repulsion behaviour attempts to maintain formation. Figure 5.34 in particular shows the same trend as was seen in the construction scenario for the same algorithm, where the rate of energy expenditure appears to decrease then increase with an increasing ping period. This occurs due to the same reasons as were explained in section 5.1.2 - that is, low ping periods induce oscillatory behaviour in the vehicles, while higher ping periods cause delays in updating neighbour positions, resulting in the vehicles 'straying' from their ideal formation positions, both of which increase energy expenditure. Since BHV_AttractionRepulsion cannot maintain the desired formation well, it is difficult to make any conclusions from figure 5.35 - no matter the rate of communications, the formation always fractures in this scenario, causing the formation quality to deteriorate rapidly. However, for ping periods of $5s$ and $15s$ we can clearly see the oscillatory behaviour in the plots of the formation quality metric, with the lines having wavelike qualities. Interestingly, this also occurs to a lesser extent for a ping period of $120s$, suggesting that both low and high ping periods induce oscillations.

Figures 5.36 and 5.37 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_PairwiseNeighbourReferencing behaviour attempts to maintain formation. Figure 5.36 appears to indicate that the rate of energy expenditure decreases then increases with increasing ping period, similar to BHV_AttractionRepulsion. It is difficult to explain why this occurs, but it is possibly due to similar reasons as for the previous behaviour. However, this trend is quite different to that seen for formation construction with BHV_PairwiseNeighbourReferencing, where the energy expenditure decreased with increasing ping period, by reducing the 'arcing' behaviour of vehicle paths during construction. In the case of maintaining formation, this 'arcing' behaviour is not a consideration. Examining figure 5.37, we can clearly see a trend where a better formation quality is maintained with a lower ping period - this is simply due to the fact that a higher communications rate allows vehicles to update their neighbour (and thus, target) positions more often. It is interesting to note that oscillatory behaviour is not very apparent in the formation quality metric graph, suggesting that it is not an issue for this algorithm in this scenario, possibly due to the constant movement caused by external currents.

Figures 5.38 and 5.39 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_RigidNeighbourRegistration behaviour attempts to maintain formation. Examining these graphs, we notice a similar trend occurring as those of the previous behaviours, where the rate of energy expenditure appears to decrease then increase with increasing ping period. This likely occurs for similar reasons as explained for the other behaviours, where a low ping period occasionally

induces some oscillatory behaviours in the vehicles, while large ping periods cause the vehicles to 'stray' from their optimal positions by delaying the updates of neighbouring vehicle positions. As with the other behaviours, a ping period of approximately $30s$ appears to provide the best 'damping' to these oscillations, while allowing for timely updating of neighbour positions. Looking at figure 5.39, its surprising to note that no matter the ping period, this algorithm is able to maintain a fairly consistent formation quality.

Figures 5.40 and 5.41 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_Assignment-Registration behaviour attempts to maintain formation. Figure 5.40 again shows a similar trend in energy expenditure with increasing ping period, and in this case, figure 5.41 clearly indicates oscillatory behaviour as seen in the plot of the formation quality metric for $5s$. In addition, the figure also illustrates that, in terms of the formation quality metric, this algorithm is reasonably unaffected by changes in the communications rate.

5.2.3 Maintenance with Node Loss

Here we present graphs of energy expenditure and formation quality with loss of nodes while the swarm attempts to maintain a hexagonal lattice formation in this scenario. A random AUV is selected and removed from the swarm every $1200s$ until 5 AUVs are removed.

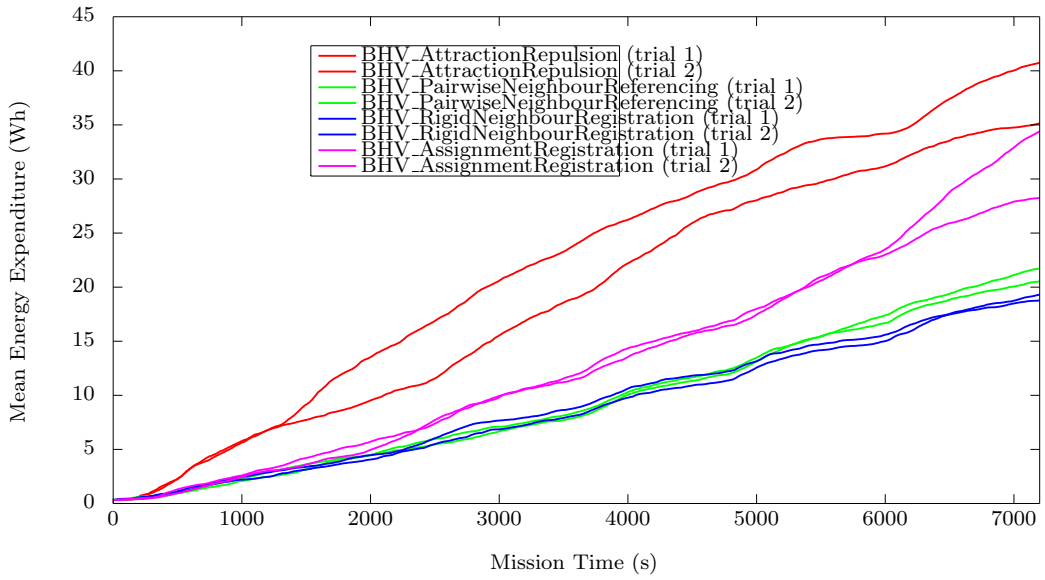

Figure 5.42: Mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in three current channels and with node loss.

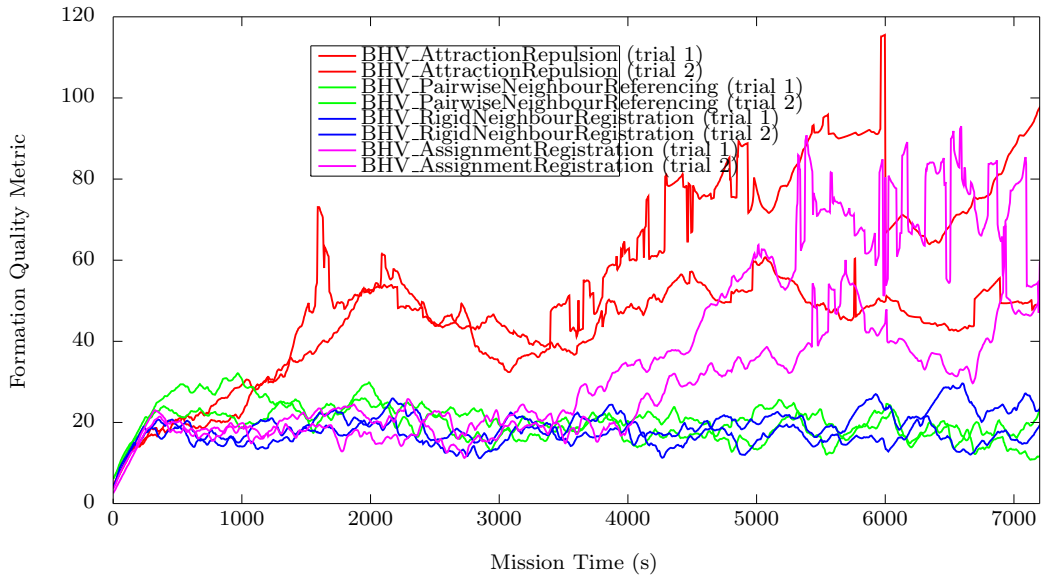

Figure 5.43: Formation quality metric while maintaining a hexagonal lattice formation in three current channels and with node loss.

Examining figure 5.42, we can observe a few interesting things. Firstly, for BHV_Attraction-Repulsion, for one trial, the loss of the first node (at around $1200s$) has caused a surge in energy expenditure as vehicles move and the formation attempts to self-repair; and in the second trial, we can clearly observe minor increases in energy expenditure during each of the five losses (at $1200s$, $2400s$, $3600s$, $4800s$ and $6000s$). Secondly, these same minor surges can be seen in the plots of each of the BHV_PairwiseNeighbourReferencing, BHV_RigidNeighbourRegistration and BHV_AssignmentRegistration behaviours at each of the five vehicle losses (although it is less noticeable for BHV_PairwiseNeighbourReferencing). In addition, for one of the trials of BHV_AssignmentRegistration, the loss of the fifth vehicle has caused a significant increase in energy expenditure, as vehicles move and rearrange themselves in an attempt to find their best local point-set fit.

Examining figure 5.43, we can make a couple of observations. Firstly, it is difficult to make any conclusions about the effect of node loss on the BHV_AttractionRepulsion algorithm, simply because even without node loss, this algorithm fails to maintain the desired formation. Looking at the BHV_AssignmentRegistration algorithm, we see that the third node loss (at around $3600s$) has caused a significant deterioration in formation quality for both of its trials, as it fails to either self-repair, or to maintain its current formation. Again, as was the case with the formation construction scenario, node loss does not appear to cause much effect to the formation quality of both the BHV_PairwiseNeighbourReferencing and the BHV_RigidNeighbourRegistration behaviours; as before, the vehicle loss did not cause any

vehicles to become 'stranded', and the swarms are able to maintain their current formation even with the absence of a few nodes.

5.3 Scenario 3 - Formation Maintenance

The next test intended to compare the ability of each algorithm to maintain a desired formation in the the scenario of an irrotational current vortex as illustrated in figure 4.3 and described in section 4.4.2. As in the previous scenarios, we measure the mean energy expenditure of the swarm as well as the formation quality metric as the swarm attempts to maintain the desired formation, then observe the effect of changing communications rate, and finally observe how each algorithm behaves when vehicles are lost.

5.3.1 Maintenance

We start this section with plots of AUV trajectories as the swarm attempts to maintain formation, as shown in the following figures. These plots illustrate the typical behaviour of each of the formation control algorithms in this scenario. As explained previously, the swarm begins in the ideal formation.

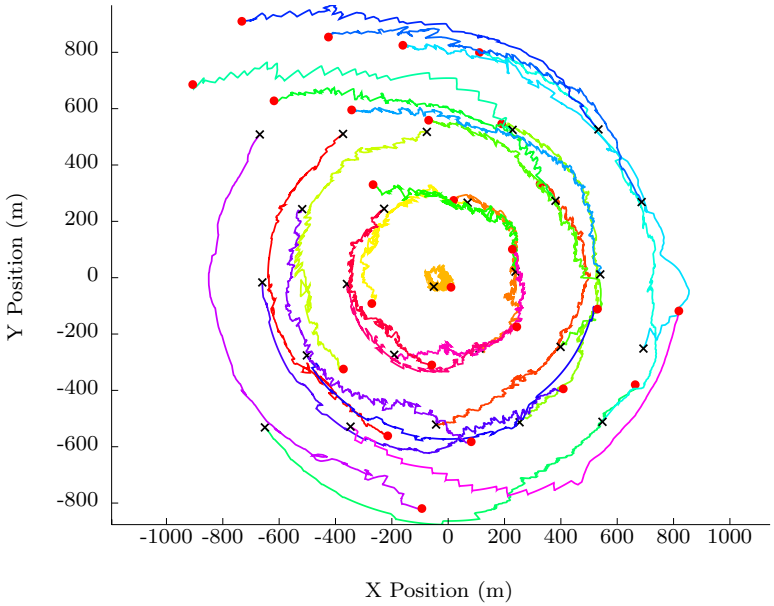

Figure 5.44: Trajectories of 25 AUVs during hexagonal lattice formation maintenance in an irrotational current vortex - BHV_AttractionRepulsion (trial 2); black crosses indicate starting positions, red circles indicate final positions.

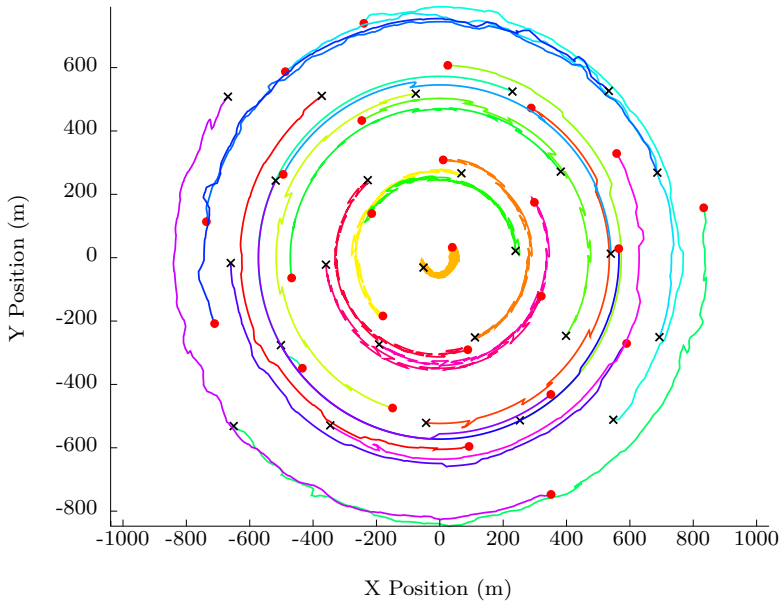

Figure 5.45: Trajectories of 25 AUVs during hexagonal lattice formation maintenance in an irrotational current vortex - BHV_PairwiseNeighbourReferencing (trial 1); black crosses indicate starting positions, red circles indicate final positions.

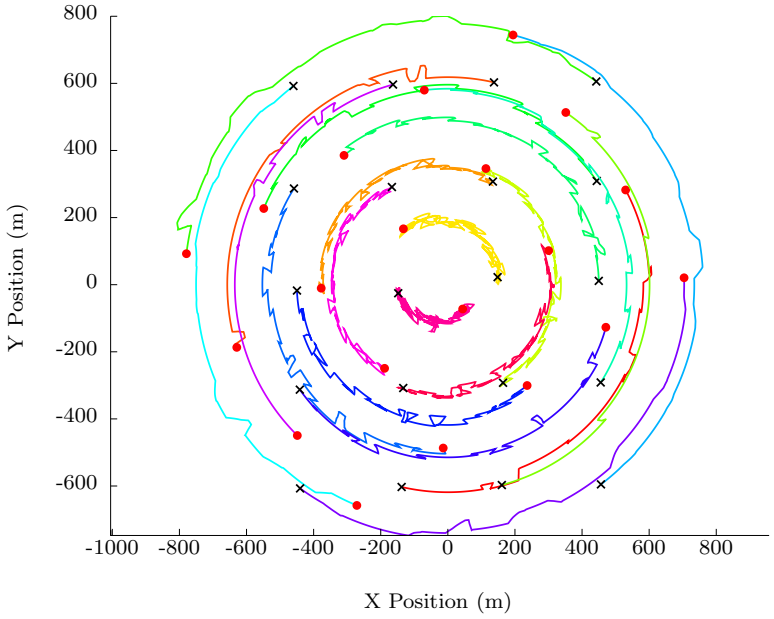

Figure 5.46: Trajectories of 20 AUVs during square lattice formation maintenance in an irrotational current vortex - BHV_RigidNeighbourRegistration (trial 2); black crosses indicate starting positions, red circles indicate final positions.

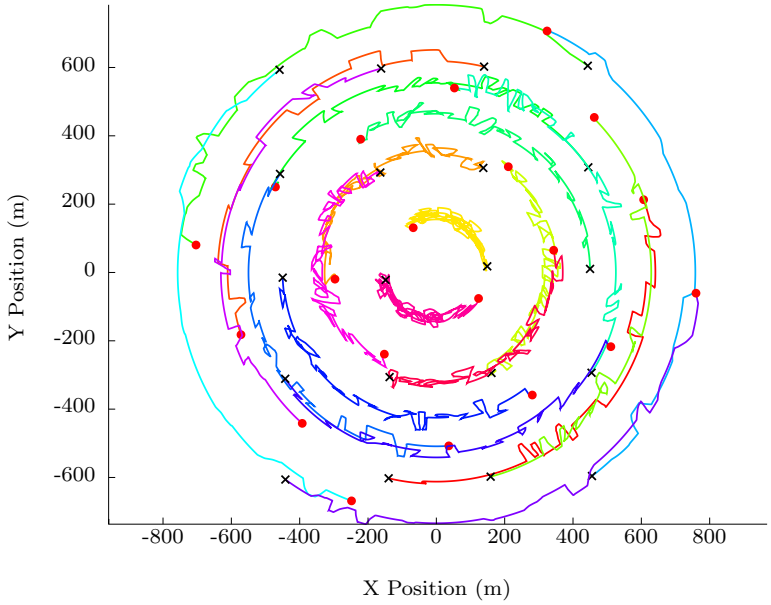

Figure 5.47: Trajectories of 20 AUVs during square lattice formation maintenance in an irrotational current vortex - BHV_AssignmentRegistration (trial 2); black crosses indicate starting positions, red circles indicate final positions.

BHV_AttractionRepulsion

Figure 5.44 illustrates typical AUV trajectories of the BHV_AttractionRepulsion algorithm attempting to maintain a hexagonal lattice formation in this scenario. Again, we can see that the lattice has fractured in 2 places, as the irrotational current vortex has 'pulled' the formation apart. However, the separation is not as bad as that caused by the previous scenario, suggesting that the irrotational current vortex does not put as much stress on the formation as the three current channel scenario. Again, this algorithm does a poor job of maintaining the desired formation, and the paths of the vehicles are very chaotic.

BHV_PairwiseNeighbourReferencing

Figure 5.45 illustrates typical AUV trajectories of the BHV_PairwiseNeighbourReferencing algorithm attempting to maintain a hexagonal lattice formation in this scenario. As expected, the formation has rotated anti-clockwise in the vortex, ending up at an offset of around 160° from its original orientation. Although easier to observe when in motion, it is interesting to note that it is not the outermost vehicles that thrust the least - AUVs that are positioned around two-thirds of the way between the center of the vortex and the edge of the formation are actually the vehicles that thrust the least and take most advantage of the water currents, drifting for the most amount of time without repositioning themselves (for example, the

AUV with the dark purple trajectory, which started one down and right from the top left of the formation). AUVs closer to the center of the vortex fight against the current to maintain formation, while the AUVs furthest from the center thrust to keep up with the rotational movement of the other vehicles. This is expected - the vehicles positioned nearer to the part of the vortex that rotates at the same velocity as the average rotational velocity of the formation can utilize the current without having to thrust. Finally, we note that the BHV_PairwiseNeighbourReferencing algorithm performs well at maintaining the desired formation.

BHV_RigidNeighbourRegistration

Figure 5.46 illustrates typical AUV trajectories of the BHV_RigidNeighbourRegistration algorithm attempting to maintain a square lattice formation in this scenario (we show a square lattice for the sake of variety). Again, as expected the rectangular formation has rotated anticlockwise from 0° to approximately 135°. Finally, as with BHV_PairwiseNeighbourReferencing, BHV_RigidNeighbourRegistration is able to maintain the square lattice formation very well in this scenario, and similarly, it is not the outermost vehicles that drift the most, but the AUVs that are second-outermost that thrust the least (e.g. the AUVs with the red trails).

BHV_AssignmentRegistration

Figure 5.47 illustrates typical AUV trajectories of the BHV_AssignmentRegistration algorithm attempting to maintain a square lattice formation in this scenario. Again we see the anticlockwise rotation of the formation, and similarly to the previous scenario, it appears that when compared to BHV_RigidNeighbourRegistration, this algorithm results in more AUV movement when attempting to maintain formation - this can be seen when comparing the trajectories between the two algorithms, with BHV_AssignmentRegistration appearing to have more jagged paths. This is likely due to the fact that this algorithm is continuously searching for a locally 'optimal' rigid transformation using the lowest cost output of the Hungarian algorithm; it is probable that the lowest cost 'jumps' between two solutions that are close in value, and this jumping occurs for many of the vehicles, resulting in a kind of 'vibration' of the lattice that is more clearly visible with the swarm in motion. Still, BHV_AssignmentRegistration is able to maintain the desired formation quite well. We note an important caveat here - as in the three current channel scenario, the $\delta\theta$ parameter for this behaviour was set to 10° instead of the default 45°, to cope with the rotation induced in the formation.

Behaviour Comparison

We now show some graphs of mean energy expenditure and formation quality for the two trials for each behaviour, and present some observations.

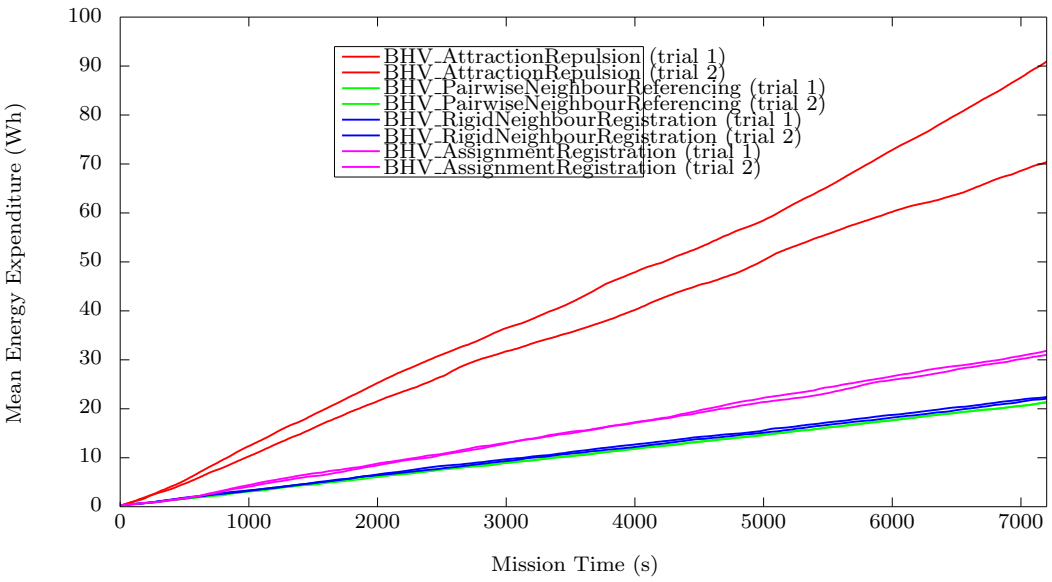

Figure 5.48: Mean energy expenditure over all AUVs while maintaining a hexagonal lattice formation in an irrotational current vortex; two trials performed for each behaviour.

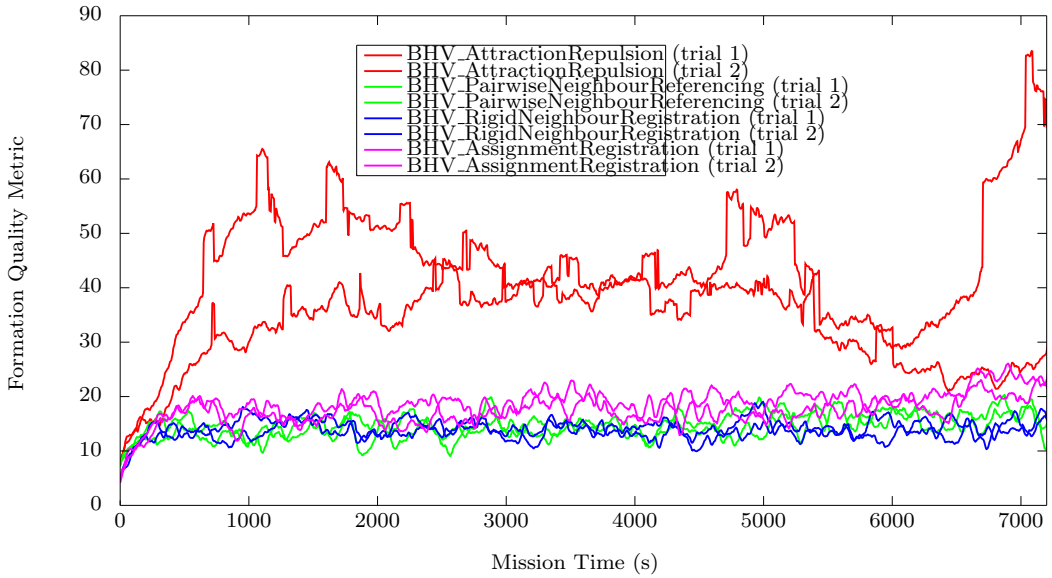

Figure 5.49: Formation quality metric while maintaining a hexagonal lattice formation in an irrotational current vortex; two trials performed for each behaviour.

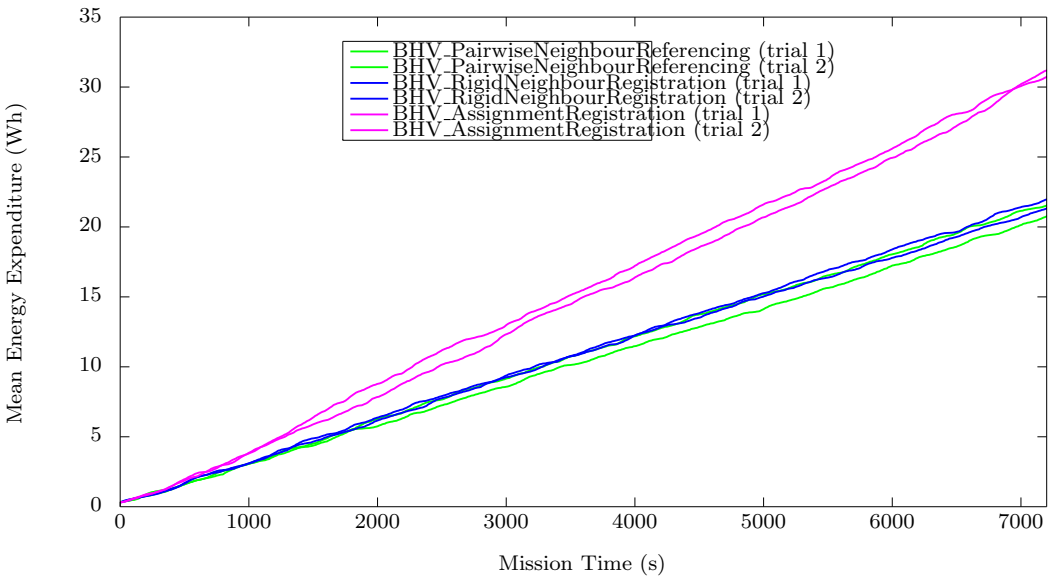

Figure 5.50: Mean energy expenditure over all AUVs while maintaining a square lattice formation in an irrotational current vortex; two trials performed for three behaviours.

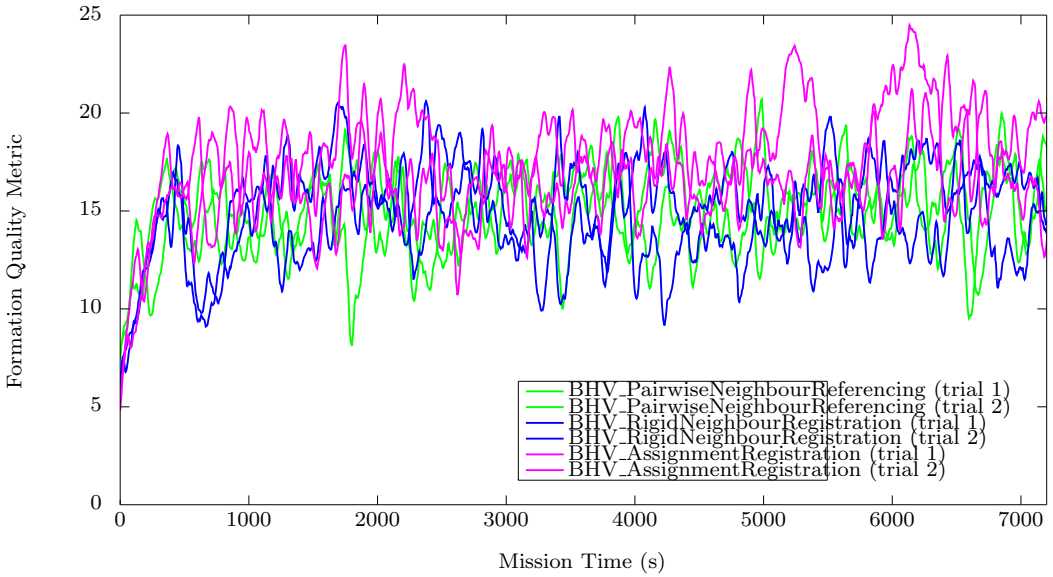

Figure 5.51: Formation quality metric while maintaining a square lattice formation in an irrotational current vortex; two trials performed for three behaviours.

Figures 5.48 and 5.50 show the mean energy expenditure for two trials using each behaviour for formation maintenance in the irrotational current vortex scenario, for a hexagonal and square lattice respectively. These two graphs lead us to observations similar to those that were described in the three current channel scenario.

Figures 5.49 and 5.51 show the formation quality metric for same two trials with each behaviour for a hexagonal and square lattice respectively. Figure 5.49 demonstrates that BHV_AttractionRepulsion cannot maintain the desired formation, with the quality metric becoming quite large in both its trials. This figure also appears to show that BHV_Assignment-Registration maintains a slightly worse quality formation when compared to both BHV_Rigid-NeighbourRegistration and BHV_PairwiseNeighbourReferencing, which maintain formations of nearly the same quality; figure 5.51 supports this observation, but to a lesser extent.

5.3.2 Maintenance with Varying Communications Rate

We now show some graphs of energy expenditure and formation quality with a variety of communications rates during formation maintenance in this scenario. We make some observations on the effect of these changes on the performance of each algorithm.

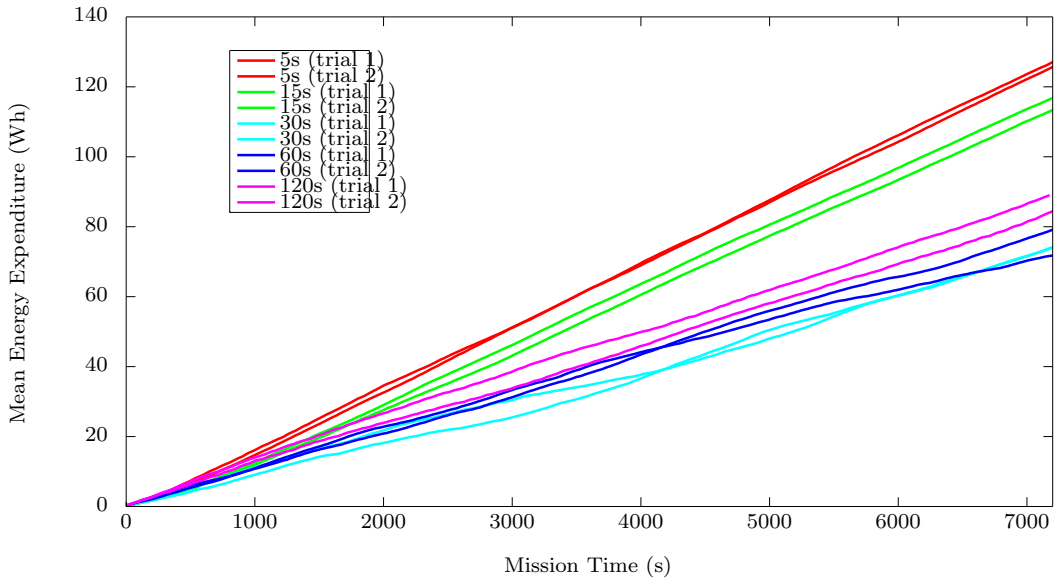

Figure 5.52: BHV_AttractionRepulsion - mean energy expenditure over all AUVs while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

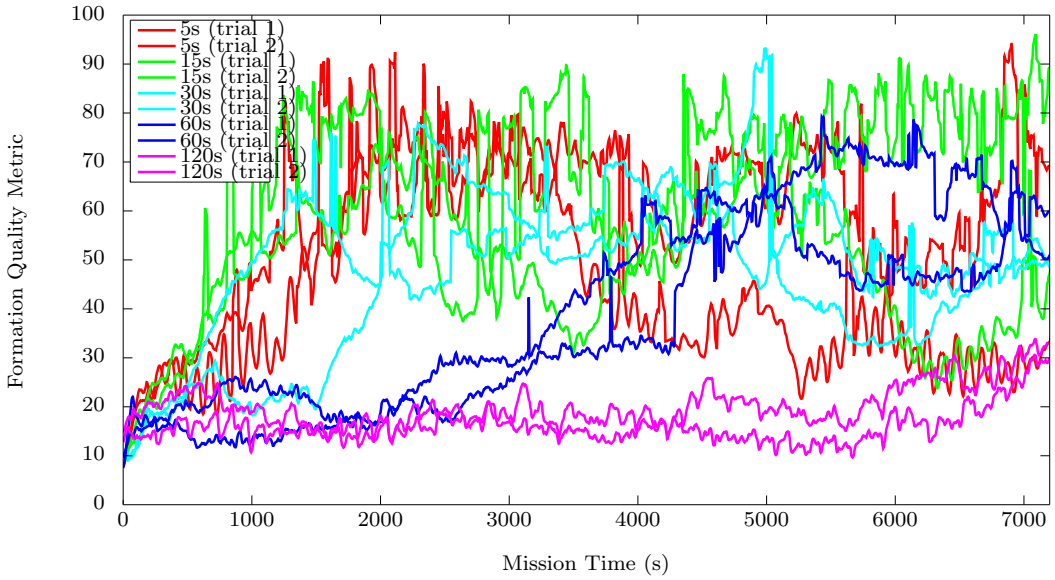

Figure 5.53: BHV_AttractionRepulsion - formation quality metric while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

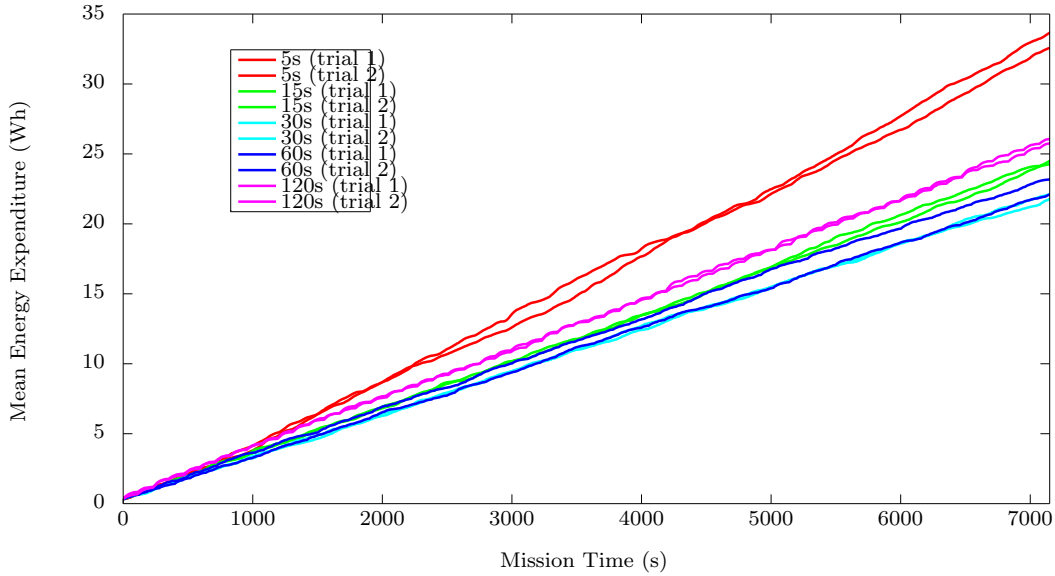

Figure 5.54: BHV_PairwiseNeighbourReferencing - mean energy expenditure over all AUVs while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

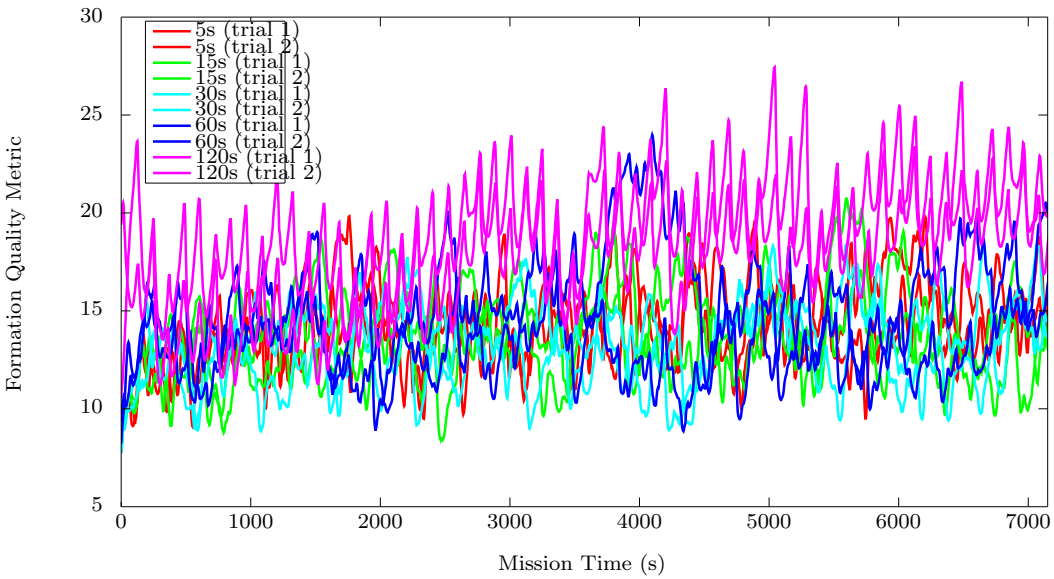

Figure 5.55: BHV_PairwiseNeighbourReferencing - formation quality metric while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

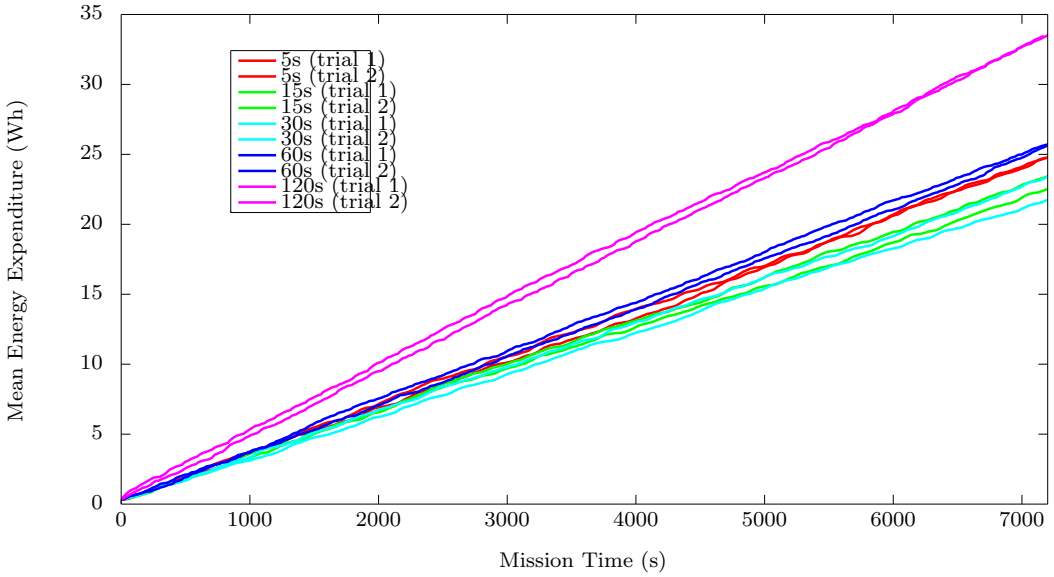

Figure 5.56: BHV_RigidNeighbourRegistration - mean energy expenditure over all AUVs while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

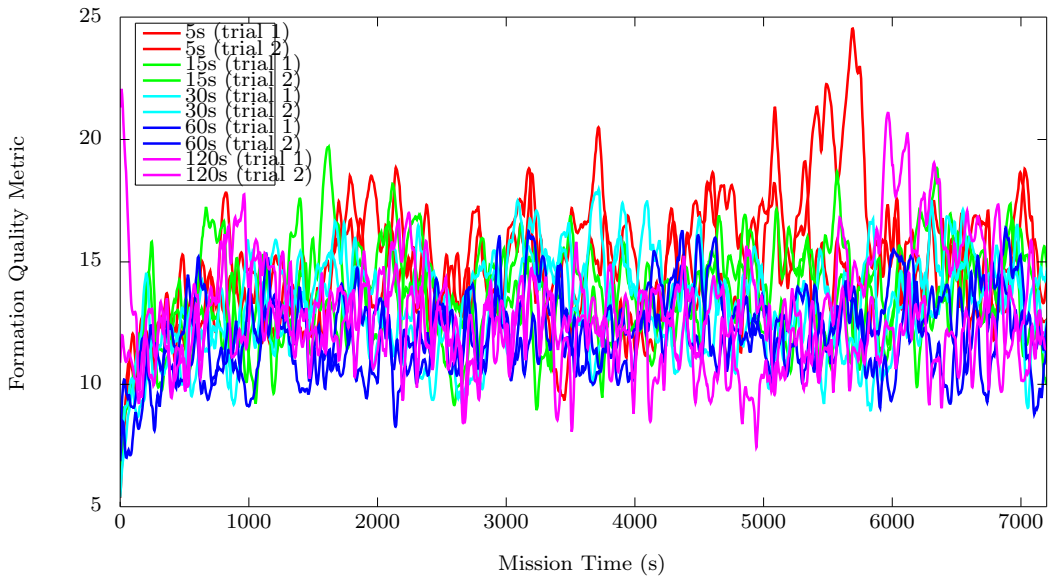

Figure 5.57: BHV_RigidNeighbourRegistration - formation quality metric while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

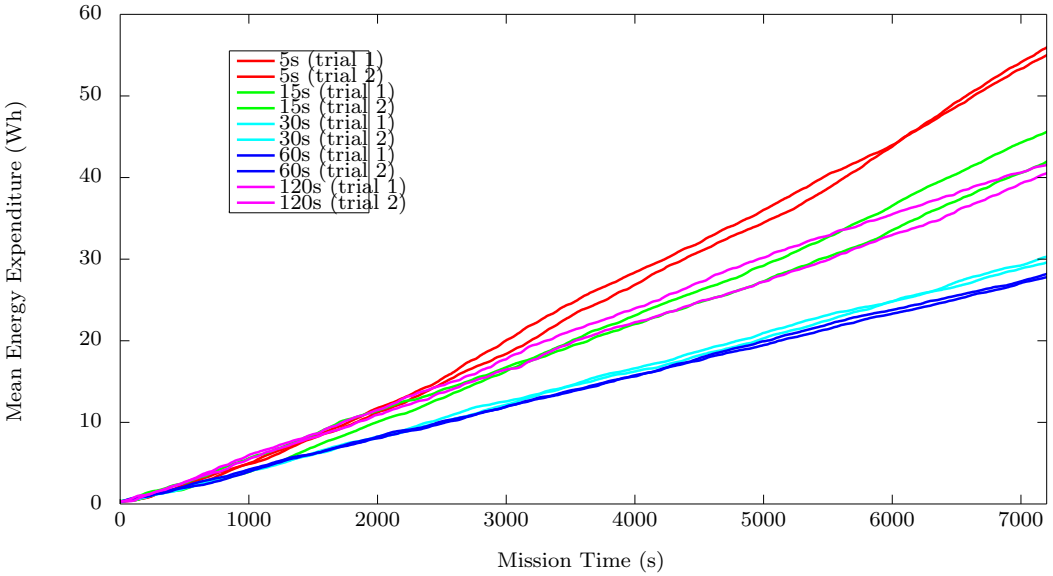

Figure 5.58: BHV_AssignmentRegistration - mean energy expenditure over all AUVs while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

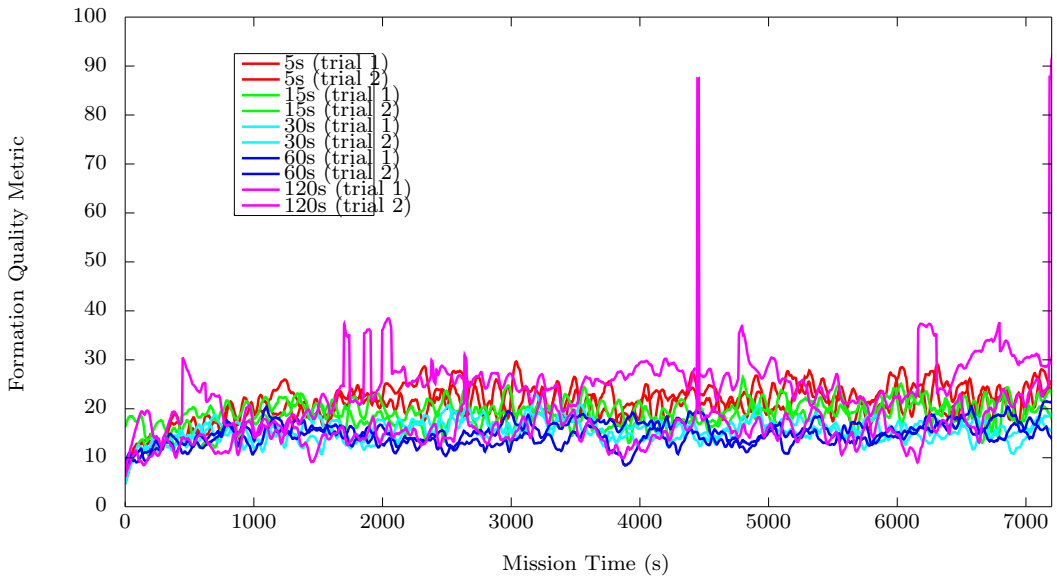

Figure 5.59: BHV_AssignmentRegistration - formation quality metric while maintaining hexagonal lattice formation in an irrotational current vortex and with varying communications rate.

Figures 5.52 and 5.53 show plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_Attraction-Repulsion behaviour attempts to maintain formation. There are not many additional observations to make with these results that were not mentioned in previous tests, other than the fact that it appears that an increasing ping period results in this algorithm being able to better maintain formation. This is likely due to the fact that this scenario does not stress the formation as much as the three current channel scenario, inducing rotation rather than 'pulling' at the formation; reducing the communications rate thus allows the formation to freely rotate in the vortex for a longer period of time, while increasing the communications rate induces oscillatory behaviour.

Figures 5.54 and 5.55 show plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_PairwiseNeighbourReferencing behaviour attempts to maintain formation. These graphs reinforce the observations described in the three current channel scenario. From figure 5.55, it appears that BHV_PairwiseNeighbourReferencing is able to maintain the formation to a high standard regardless of the communications rate, although the formation quality appears to be slightly worse for ping periods of $60s$ and $120s$.

Figures 5.56 and 5.57 illustrate plots of the mean energy expenditure and the formation

quality metric, respectively, for a variety of communications rates, while the BHV_RigidNeighbourRegistration behaviour attempts to maintain formation. Again, the observations made from the results of the three current channel scenario can be applied to these plots, and as before, it appears that BHV_RigidNeighbourRegistration is able to maintain the formation to a high standard no matter the ping period.

Figures 5.58 and 5.59 illustrate plots of the mean energy expenditure and the formation quality metric, respectively, for a variety of communications rates, while the BHV_AssignmentRegistration behaviour attempts to maintain formation. As with the other algorithms, no new observations can be made from these plots that were not made from the results of the previous scenario - except for the fact that there are two spikes that occur in the formation quality metric for one of the $120s$ ping period trials. Reviewing the logs with the swarm in motion, these spikes were caused by a single vehicle at the edge of the lattice deciding to reposition itself to what it decided was a more optimal position relative to the swarm; this is a result of the nature of the algorithm. In effect, the vehicle moved from one point in the lattice to a neighbouring point, and consequently, the formation quality metric increase then decreased again while this vehicle was in transit.

5.3.3 Maintenance with Node Loss

We now present graphs of energy expenditure and formation quality with loss of nodes while the swarm attempts to maintain a hexagonal lattice formation in the irrotational current vortex. A random AUV is removed from the swarm every $1200s$ until 5 AUVs are removed.

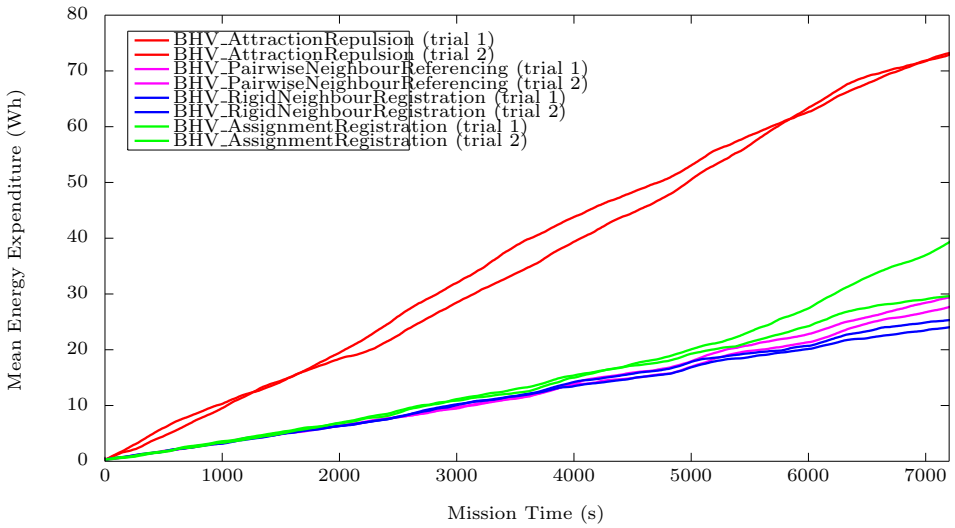

Figure 5.60: Mean energy expenditure over all AUVs while maintaining hexagonal lattice formation in an irrotational current vortex and with node loss.

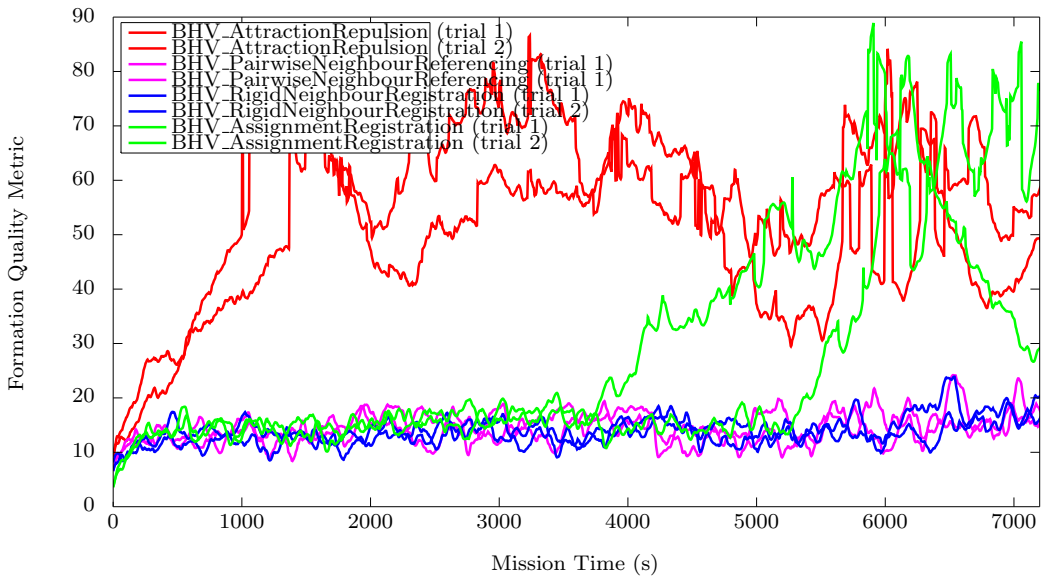

Figure 5.61: Formation quality metric while maintaining hexagonal lattice formation in an irrotational current vortex and with node loss.

Looking at figure 5.60 we see that the plots for each behaviour follow similar trajectories as in the previous scenario with vehicle loss. As in the previous scenario, we can see the slight changes in energy expenditure whenever a vehicle is lost (at $1200s$, $2400s$, $3600s$, $4800s$, and $6000s$). We make an additional observation that is also apparent in the plot of the previous scenario - unlike most of the tests, where the rate of energy expenditure of BHV_PairwiseNeighbourReferencing is usually lower than that of BHV_RigidNeighbourRegistration, in the tests with node loss the opposite occurs; this seems to suggest that BHV_RigidNeighbourRegistration is more robust to vehicle loss, which we expect due to the fact that each vehicle with BHV_PairwiseNeighbourReferencing needs at least 2 neighbours to position itself, while only 1 neighbour is needed with BHV_RigidNeighbourRegistration.

Figure 5.61 reinforces many of the observations described in the three current channel scenario with node loss. However, we make a couple of observations about the BHV_AssignmentRegistration algorithm - for the first trial, we see that the combination of the third and fourth vehicle losses at $3600s$ and $4800s$ has caused the formation to deteriorate in quality quite dramatically, with the formation being unable to self-repair; for the second trial, the fourth node loss at $4800s$ caused AUVs in the swarm to rearrange themselves, but eventually the formation was able to self-repair, resulting in the formation quality metric increasing then falling back to its original level. As before, BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration maintain a specified formation quality metric even in the face of vehicle loss, because these algorithms have no self-repair capability, and vehicles were

lost in such a way that no AUVs were left stranded.

5.4 Scenario 4 - Formation Propulsion

As described in section 4.4.3, for the final few tests we observe each behaviour's ability to construct and maintain a desired formation, while utilizing ocean currents for propulsion, using the scenario of simulated Red Sea ocean currents provided by the MSEAS group. We instruct each behaviour to construct hexagonal and square lattices, monitor energy consumption and the formation quality metric, and use these results to analyse the ability of each algorithm to address the objectives of the swarm. We begin with plots of typical AUV trajectories for each behaviour in this scenario.

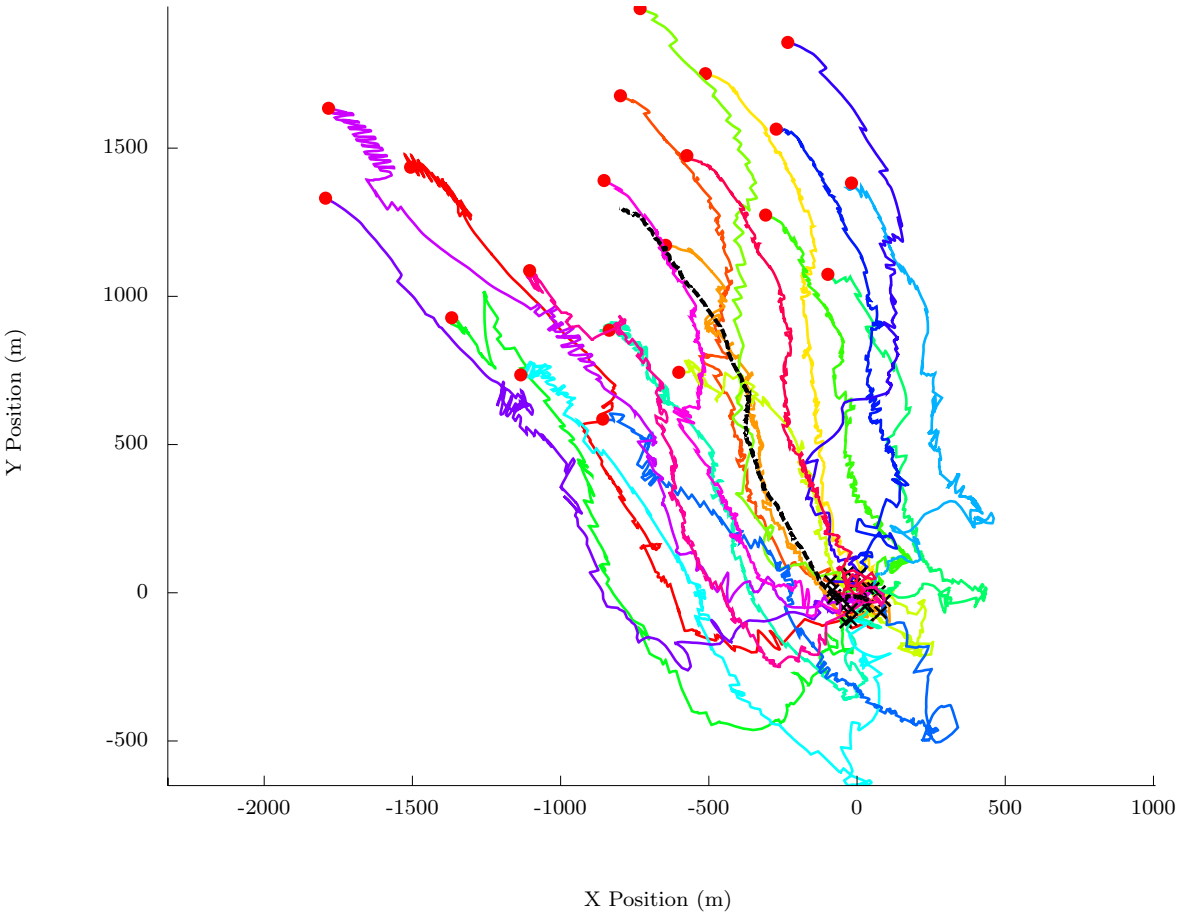

Figure 5.62: Trajectories of 20 AUVs during hexagonal lattice formation construction and maintenance in the simulated Red Sea - BHV_AttractionRepulsion (trial 2); black crosses indicate starting positions, red circles indicate final positions; dashed black line indicates the trajectory of the formation centroid.

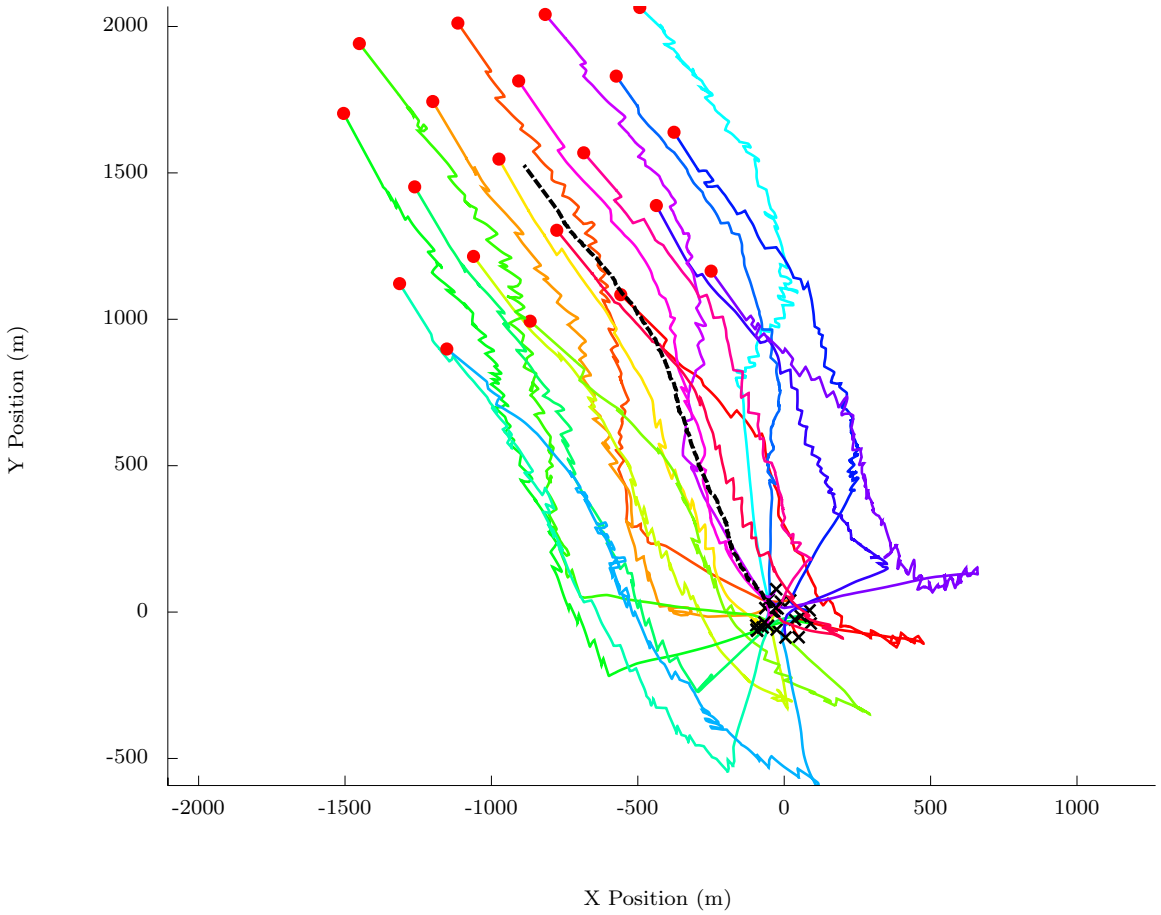

Figure 5.63: Trajectories of 20 AUVs during hexagonal lattice formation construction and maintenance in the simulated Red Sea - BHV_PairwiseNeighbourReferencing (trial 1); black crosses indicate starting positions, red circles indicate final positions; dashed black line indicates the trajectory of the formation centroid.

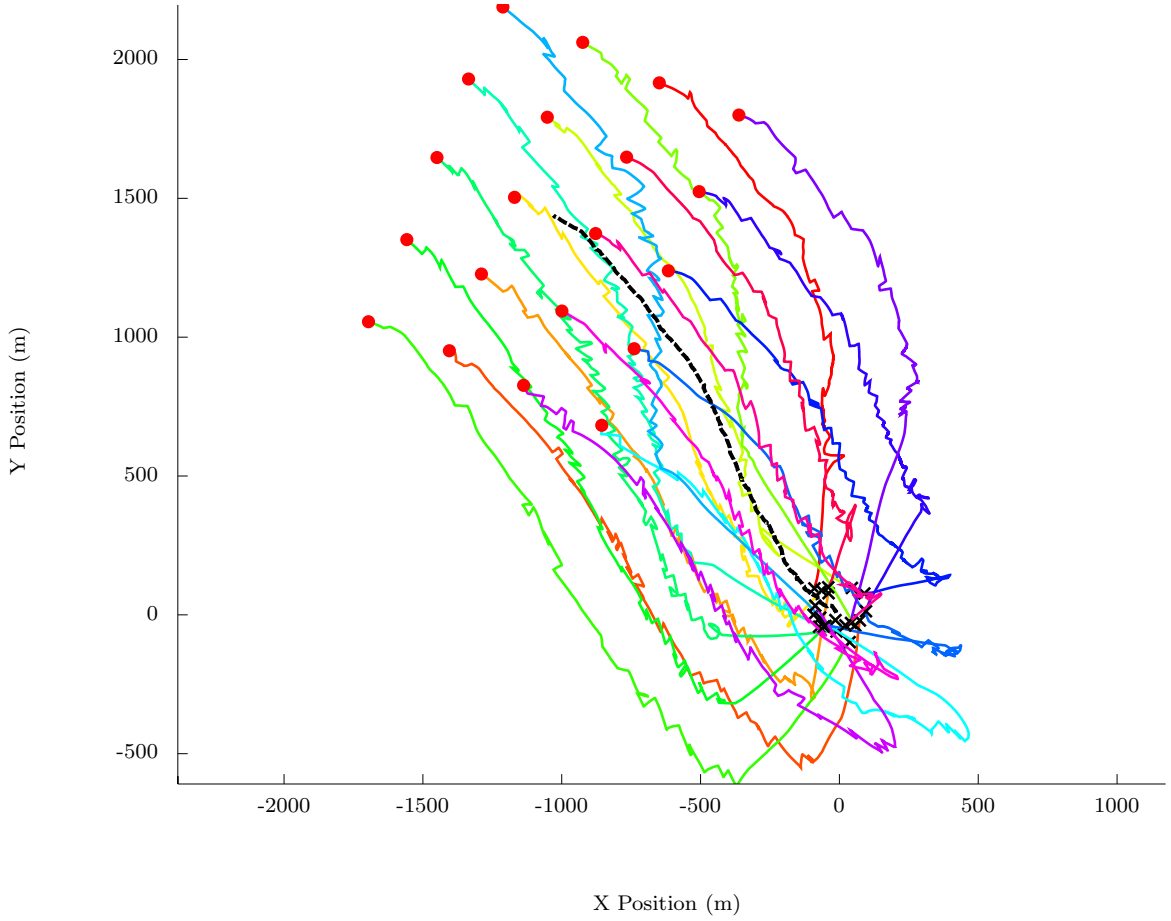

Figure 5.64: Trajectories of 20 AUVs during square lattice formation construction and maintenance in the simulated Red Sea - BHV_RigidNeighbourRegistration (trial 2); black crosses indicate starting positions, red circles indicate final positions; dashed black line indicates the trajectory of the formation centroid.

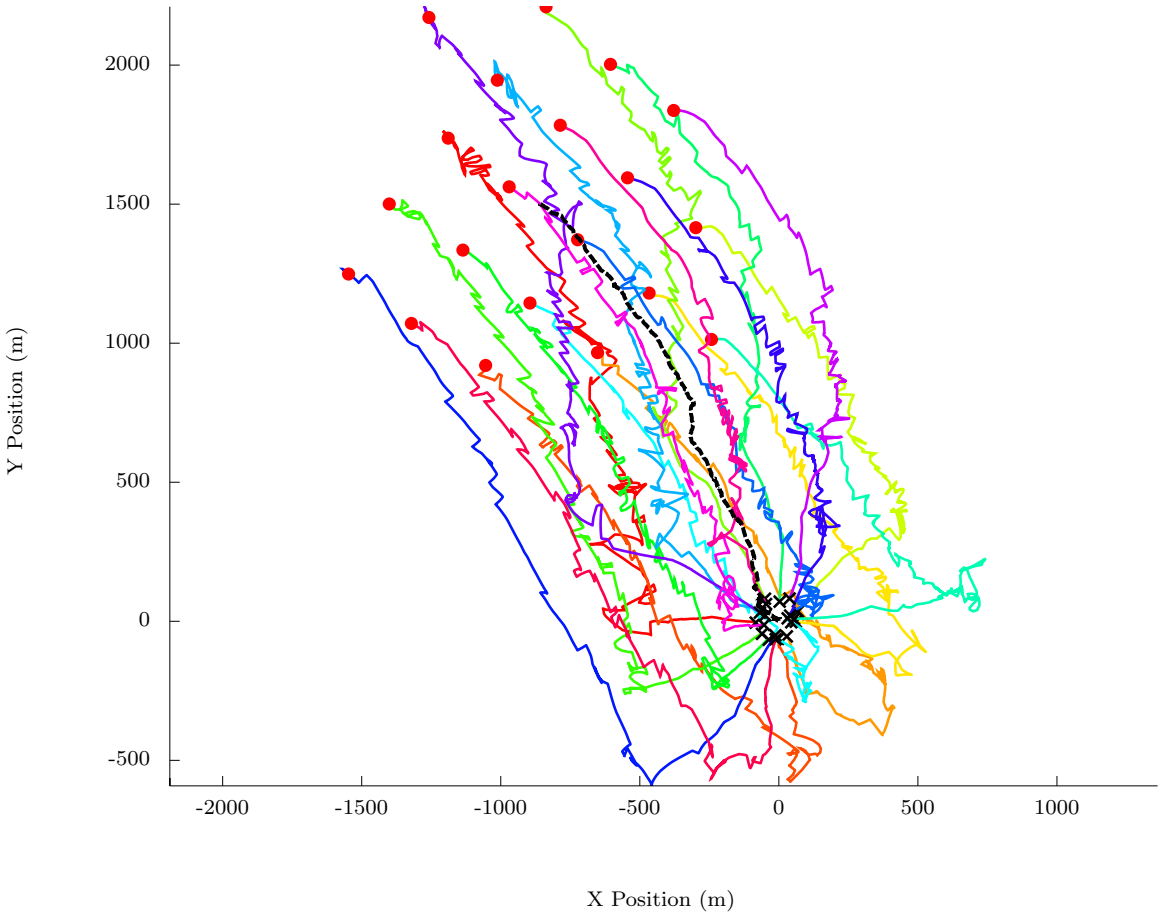

Figure 5.65: Trajectories of 20 AUVs during square lattice formation construction and maintenance in the simulated Red Sea - BHV_AssignmentRegistration (trial 2); black crosses indicate starting positions, red circles indicate final positions; dashed black line indicates the trajectory of the formation centroid.

BHV_AttractionRepulsion

Figure 5.62 illustrates typical AUV trajectories of the BHV_AttractionRepulsion algorithm when constructing and maintaining a hexagonal lattice formation in the scenario of simulated Red Sea currents. Many of the observations expressed about BHV_AttractionRepulsion from its behaviour in the other scenarios are applicable here - the chaotic movement while maintaining the formation and especially during formation construction, and the fracturing and separation of the formation while it drifts in ocean currents are both visible here. In addition, notice the trajectories of the three AUVs in the top left corner - here we can clearly see what was often described previously as oscillatory behaviour; the vehicles with the purple and red paths have fallen into oscillations right at the end of the mission, where these two vehicles are constantly moving toward and away from one another, creating a very noticeable zigzag pattern. This is the type of oscillatory behaviour that is exacerbated by higher communications rates. We note that, during this trial, BHV_AttractionRepulsion was able to create a high quality formation free of defects, but was unable to maintain it as the formation drifted; during the second trial, this algorithm was unable to create a defect-free formation, and the formation also fractured during drifting. All in all, it is apparent that BHV_AttractionRepulsion is not able to achieve our desired objectives very well.

BHV_PairwiseNeighbourReferencing

Figure 5.63 illustrates typical AUV trajectories of the BHV_PairwiseNeighbourReferencing algorithm when constructing and maintaining a hexagonal lattice formation in the scenario of simulated Red Sea currents. This plot leads us to many of the observations stated in previous scenarios for this algorithm - it is able to construct the desired formation quite efficiently, with fairly direct paths (the 'arcing' trajectory is visible here again at the start of these paths), and it is able to maintain the desired formation very well for the entire duration of the mission. We note the jagged trajectories of the vehicles, as they drift for long stretches of time until exiting their respective drifting radii, whereupon they thrust back to their targets to remain in formation.

BHV_RigidNeighbourRegistration

Figure 5.64 illustrates typical AUV trajectories of the BHV_RigidNeighbourRegistration algorithm when constructing and maintaining a square lattice formation (for the sake of variety) in the scenario of simulated Red Sea currents. Again, we note many of the same observations as described in previous scenarios - this algorithm is able to construct the desired formation quickly and efficiently using straight line paths, and is able to maintain the formation very well for the entire mission duration. As with BHV_PairwiseNeighbourReferencing, we note

the characteristic jagged trajectories of vehicles that are drifting and taking advantage of ocean currents while also attempting to maintain the desired formation.

BHV_AssignmentRegistration

Figure 5.65 illustrates typical AUV trajectories of the BHV_AssignmentRegistration algorithm when constructing and maintaining a square lattice formation in the scenario of simulated Red Sea currents. Firstly, we stress that the $\delta\theta$ parameter of this algorithm was again set to 10° for these tests. We make a couple of observations that were not seen in previous scenarios - firstly, during construction the vehicle trajectories are somewhat more chaotic than in the construction scenario in section 5.1; this is precisely due to the change in the $\delta\theta$ parameter from 45° to 10°. As touched upon previously, this change means that the algorithm now has a greater choice of rotations to select from as its best local point-set fit, likely resulting in each vehicle selecting a different rotation; this results in more chaotic movement until the entire swarm converges on the same rotation and settles into formation, causing more energy use. This rotation selection process is also the cause of the greater amount of movement in the trajectories as the swarm attempts to maintain formation (which was also visible in the previous two formation maintenance scenarios). We also note a second observation - the swarm has created a lattice that is not ideal, with two vehicles (in the top left and bottom right) being out of their 'ideal' formation positions. This is somewhat expected, since this algorithm dynamically assigns vehicle positions in the formation using a local point-set fit, and vehicles do not have a set formation position. Even so, each vehicle has correctly placed itself on a point in the square lattice. As in previous scenarios, this algorithm is able to construct and maintain the desired formation quite well.

Behaviour Comparison

We now present some graphs of the mean energy expenditure and the formation quality metric resulting from each behaviour constructing and maintaining a hexagonal and a square lattice formation in the simulated Red Sea ocean currents. Two trials were performed using each behaviour for both lattices (except for BHV_AttractionRepulsion, which can only construct hexagonal lattices).

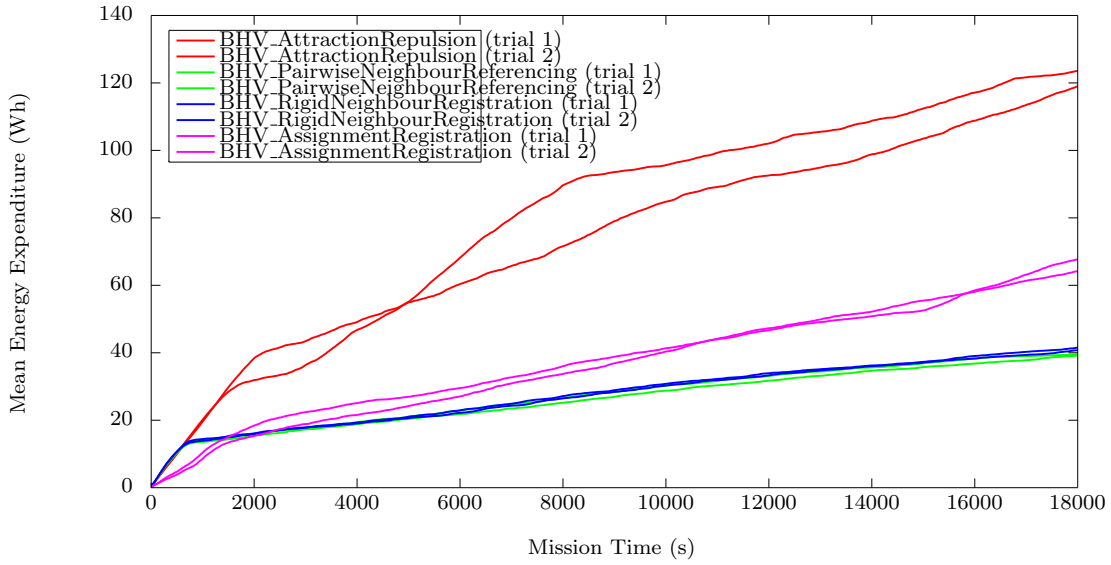

Figure 5.66: Mean energy expenditure over all AUVs while constructing and maintaining a hexagonal lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

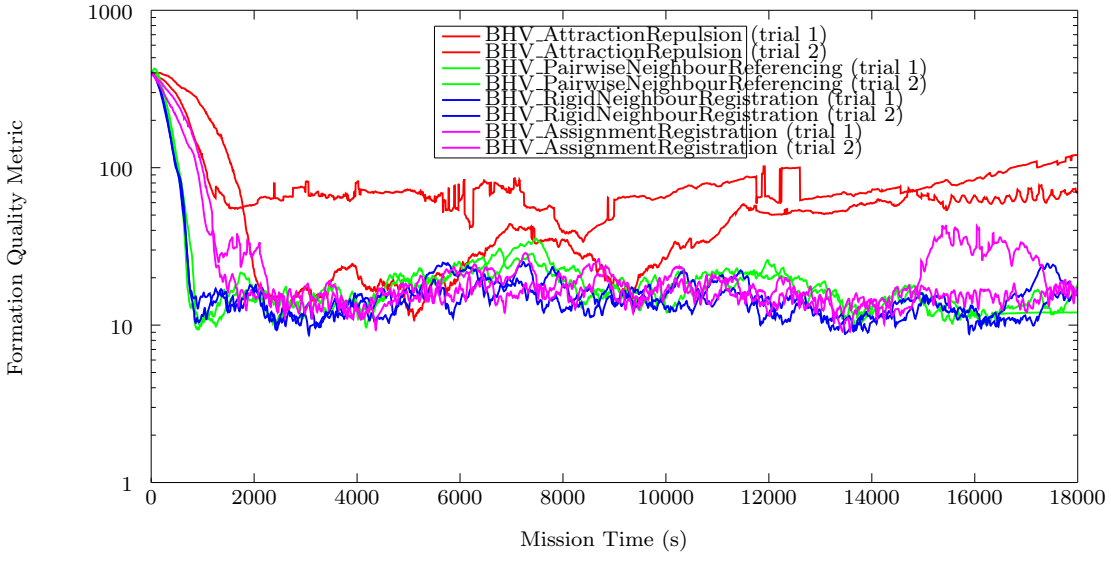

Figure 5.67: Formation quality metric while constructing and maintaining a hexagonal lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

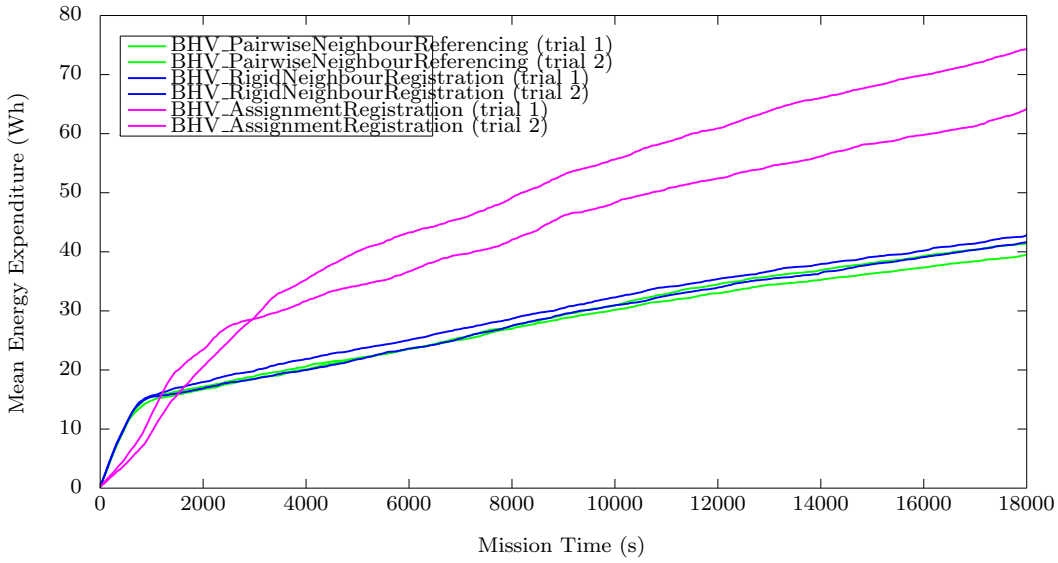

Figure 5.68: Mean energy expenditure over all AUVs while constructing and maintaining a square lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

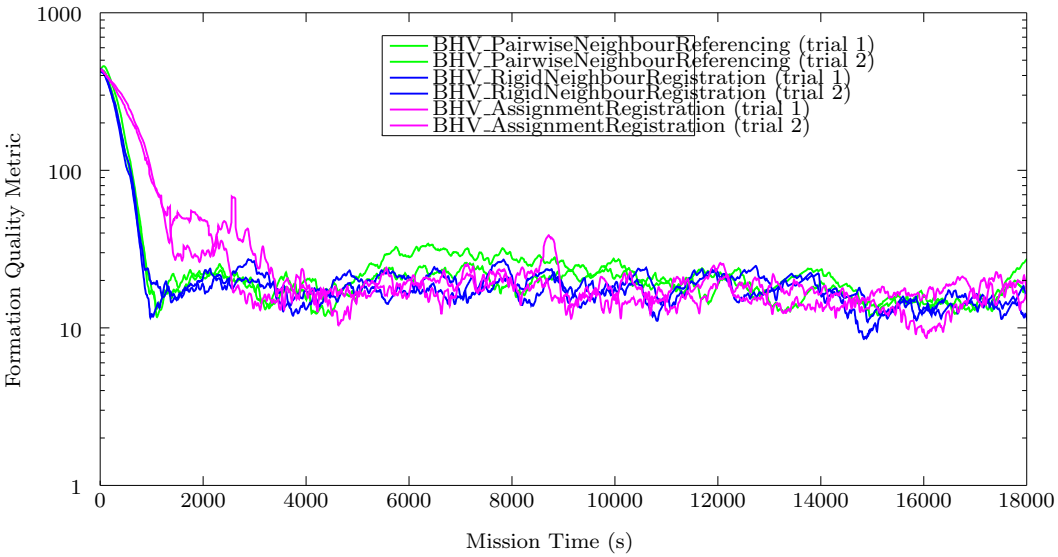

Figure 5.69: Formation quality metric while constructing and maintaining a square lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

Figures 5.66 and 5.68 illustrate the mean energy expenditure over all AUVs for formation construction and maintenance in simulated Red Sea currents, for a hexagonal and square lattice formation respectively; figures 5.67 and 5.69 show the corresponding formation quality plots; two trials were performed for each algorithm. These figures lead us to many of the same conclusions as in the previous scenarios, but there is one major difference - in comparison to the previous formation construction scenario in section 5.1, the construction portion for the BHV_AssignmentRegistration algorithm behaves in a markedly different way. In the previous scenario, BHV_AssignmentRegistration was able to construct the formation with the minimum energy expenditure and minimum time when compared to all other behaviours, but in this scenario it performs worse than both BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration. It can be seen that BHV_AssignmentRegistration takes longer to reach the same formation quality metric when compared to the two other algorithms, and does so by expending more energy. It is precisely due to the change in the $\delta\theta$ parameter of BHV_AssignmentRegistration that this occurs, and for the same reasons as previously explained at the start of section 5.4. Thus, we can see just how sensitive this algorithm is to changes in the $\delta\theta$ parameter - determining an optimal value for both efficient formation construction as well as maintenance can potentially allow for BHV_AssignmentRegistration to outperform the other behaviours.

In terms of formation quality, again we see that BHV_AttractionRepulsion is unable to maintain the desired formation - in one trial the formation separates around half way through the mission, causing the formation quality to deteriorate significantly, while in the second trial it is never able to construct the formation to the same quality as the other behaviours. All other behaviours are able to construct and maintain the formation to a similar quality level - although it can be argued that BHV_RigidNeighbourRegistration has a slightly higher quality than both BHV_PairwiseNeighbourReferencing and BHV_AssignmentRegistration, but at the cost of an energy expenditure that is slightly higher than BHV_PairwiseNeighbourReferencing, mostly used during the construction phase.

Energy Expenditure in Detail

In figures 5.66 and 5.68, we see that the energy expenditure of BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration have very similar trajectories. To try and determine which behaviour is superior in terms of energy expenditure, we take a closer look at their energy expenditure here. In the following graphs we plot the mean energy expenditure over all AUVs for both trials and for both the hexagonal and square lattice formations, but we also plot the minimum and maximum envelopes of energy expenditure for all vehicles in the swarm.

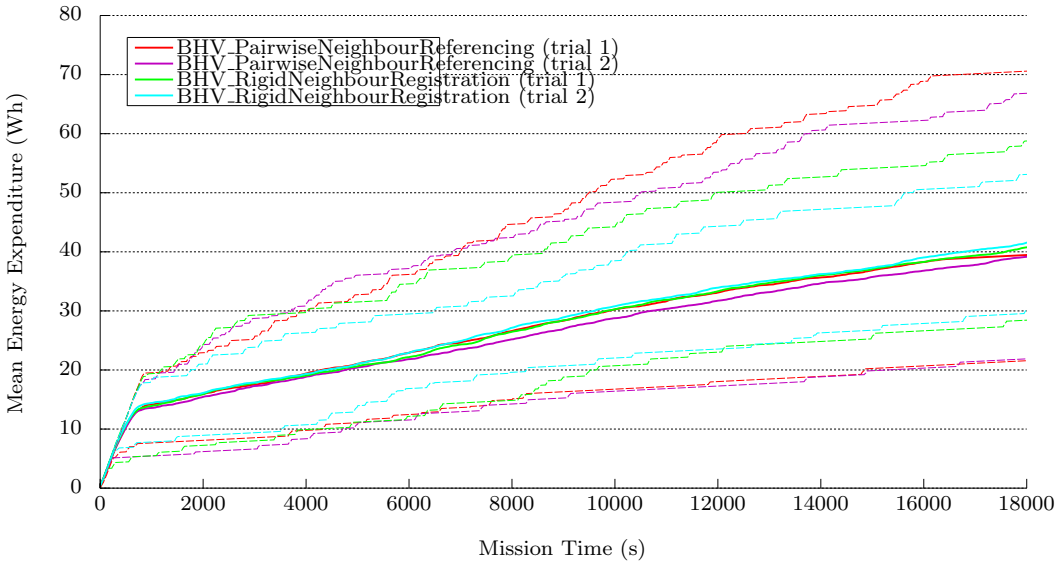

Figure 5.70: Mean energy expenditure over all AUVs while constructing and maintaining a hexagonal lattice formation in simulated Red Sea ocean currents; energy expenditure detail, where dashed lines indicate the minimum and maximum envelopes of energy expenditure over all AUVs.

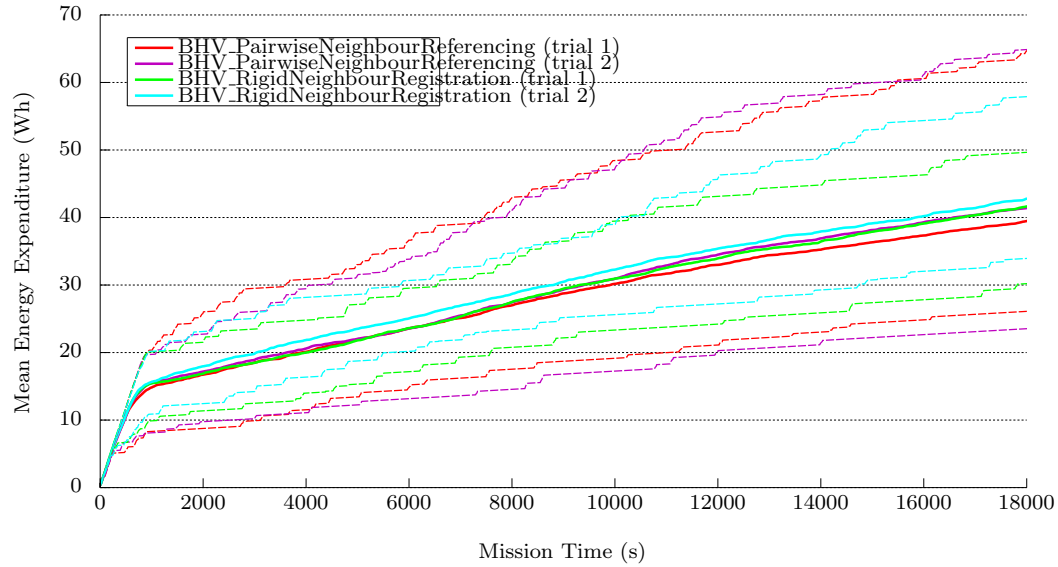

Figure 5.71: Mean energy expenditure over all AUVs while constructing and maintaining a square lattice formation in simulated Red Sea ocean currents; energy expenditure detail, where dashed lines indicate the minimum and maximum envelopes of energy expenditure over all AUVs.

Examining figures 5.70 and 5.71, we notice something interesting - in all cases, BHV_RigidNeighbourRegistration has a much lower variance in vehicle energy expenditure than BHV_PairwiseNeighbourReferencing. By the end of the mission at $18000s$, the difference in energy expenditure between the vehicle that has used the most energy and the vehicle that has use the least energy is around $20Wh$ to $30Wh$ for BHV_RigidNeighbourRegistration, while for BHV_PairwiseNeighbourReferencing, it is approximately $40Wh$ to $50Wh$. Therefore, energy usage for vehicles using the BHV_RigidNeighbourRegistration algorithm is significantly more consistent, which may give this behaviour a distinct advantage - if an application requires the entire swarm to be operational in order to effectively address its goal, then the formation becomes less effective the moment the first vehicle has run out of energy; and it appears that using the BHV_PairwiseNeighbourReferencing algorithm will result in the swarm losing its first AUV to energy depletion more quickly than when using the BHV_RigidNeighbourRegistration algorithm. The cause of this difference in variance between the two behaviours is difficult to ascertain. However, it is possibly due to the following reason - when using BHV_PairwiseNeighbourReferencing, vehicles at the outer edge of the swarm (and especially the corners) have fewer neighbours from which to perform their geometric calculations; as a result, the centroid by which a vehicle's final target is calculated is influenced by fewer vehicles (and thus undergoes less averaging), causing the target to be more susceptible to the noise from neighbour range/bearing measurements, which in turn causes the target to move more frequently, resulting in greater energy expenditure. The opposite is true in particular for the AUV at the center of the swarm. For BHV_RigidNeighbourRegistration this is not an issue - neighbouring vehicles are used to determine the optimal rigid transformation of the formation, effectively inferring the location of neighbours that are out of range of the vehicle.

Energy Expenditure and Distance Travelled

In this particular scenario we are able to produce another set of graphs with an interesting metric - the distance travelled by the centroid of the swarm divided by the mean energy expenditure, over time. This gives us some idea of the travel efficiency of the swarm in m/Wh, a metric that may be useful for certain applications - for example, if we wish to map or sample a given area in the ocean, based on predicted ocean currents. Graphs of swarm travel distance divided by mean energy expenditure are presented here for each trial and for each behaviour. We also provide graphs of mean energy expenditure versus the distance travelled by the centroid of the swarm.

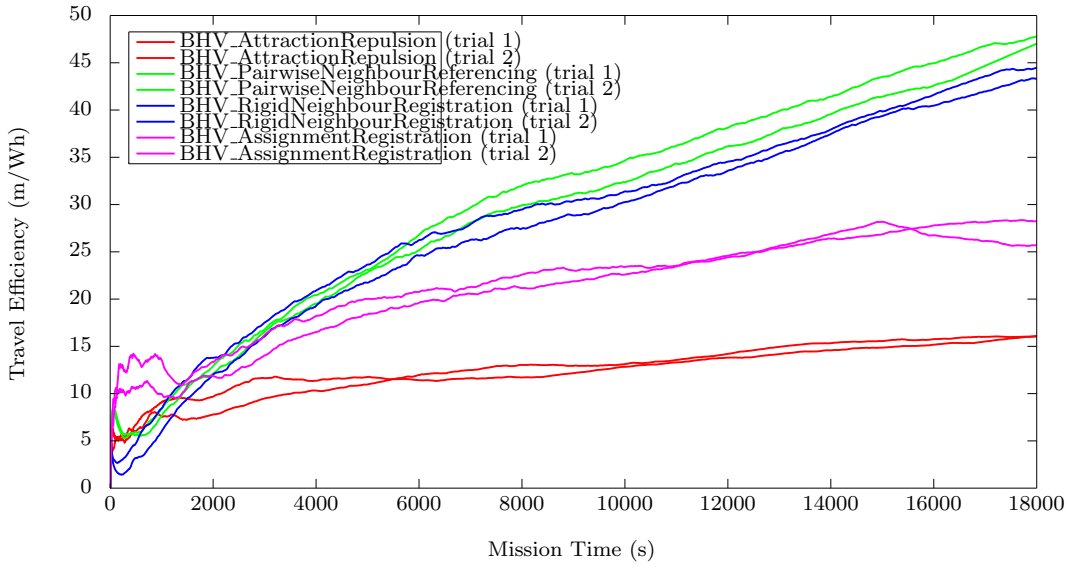

Figure 5.72: Swarm centroid travel distance divided by mean energy expenditure over all AUVs while constructing and maintaining a hexagonal lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

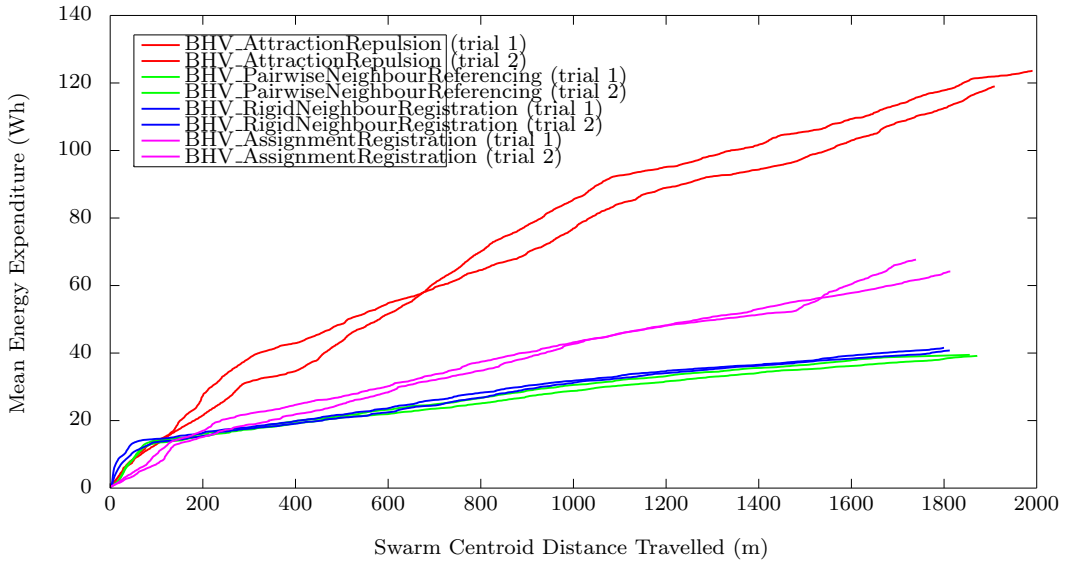

Figure 5.73: Mean energy expenditure over all AUVs versus swarm centroid travel distance while constructing and maintaining a hexagonal lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

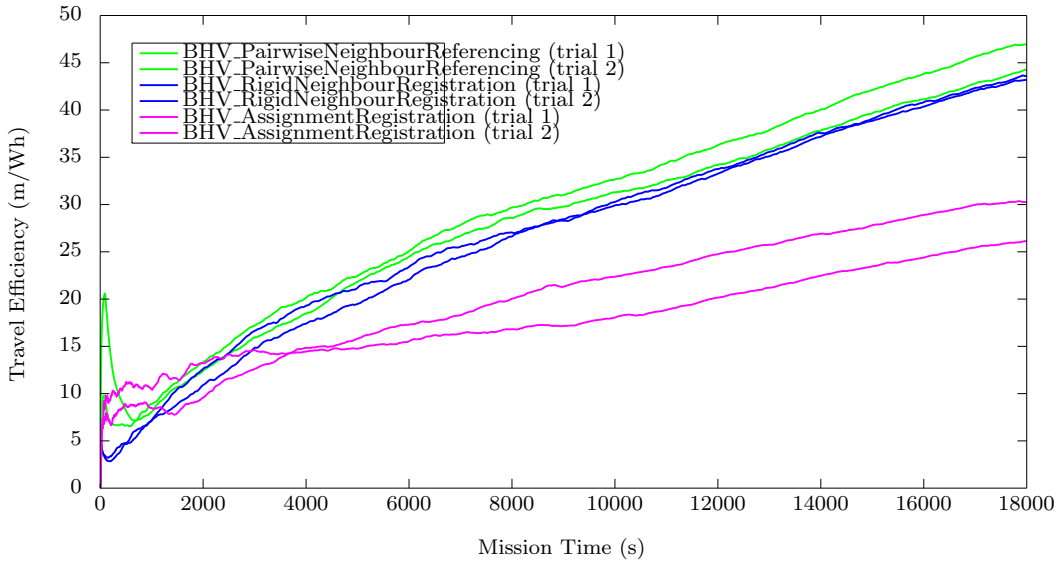

Figure 5.74: Swarm centroid travel distance divided by mean energy expenditure over all AUVs while constructing and maintaining a square lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

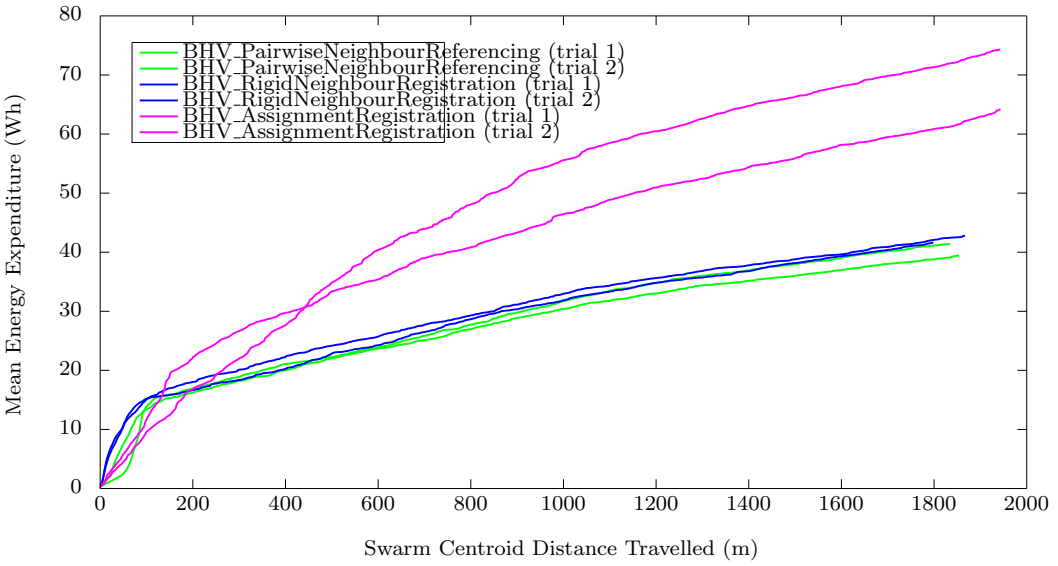

Figure 5.75: Mean energy expenditure over all AUVs versus swarm centroid travel distance while constructing and maintaining a square lattice formation in simulated Red Sea ocean currents; two trials performed for each behaviour.

Figures 5.72 and 5.74 show the ratio of distance travelled by the swarm centroid to the mean energy expenditure of the swarm, for a hexagonal and square lattice formation respectively; figures 5.73 and 5.75 show the corresponding mean energy expenditure versus the distance travelled by the swarm centroid. We can disregard the first $1000s$ to $2000s$ of these figures, since this is the construction phase for the majority of the algorithms (for a closer look at the time point at which the formation has been constructed, we can examine the formation quality metric plots in figures 5.67 and 5.69), and during this period the distance travelled is disproportionate to the amount of energy expended (the rate of energy expenditure during the construction phase is much larger than during the maintenance phase, and during this time the swarm centroid does not move very much). Looking at figures 5.72 and 5.74, we can clearly see that BHV_PairwiseNeighbourReferencing outperforms the other behaviours in terms of distance travelled by the swarm per Wh of energy expended, and by the end of the mission, this algorithm ends up with a ratio of around $45m$ travelled per Wh; this is followed by BHV_RigidNeighbourRegistration, with a ratio of just under $45m$ travelled per Wh; and then by BHV_AssignmentRegistration, with a ratio of around $25m$ to $30m$ travelled per Wh. These observations are backed up by figures 5.73 and 5.75, which show that BHV_PairwiseNeighbourReferencing utilizes the least energy per meter travelled for the majority of the mission, at least in the sense where energy expenditure is averaged over all AUVs. Whether or not this performance is at the expense of formation quality is debatable, but can be argued for by examination of figures 5.67 and 5.69.

Ratio of Mean Time Thrusting to Total Time & Ratio of Mean Distance Thrusting to Mean Total Distance

Finally, there are two additional metrics of interest extracted from the datasets of this scenario - here we present the ratio of the amount of time spent thrusting averaged over all AUVs divided by the total mission time, as well as the ratio of the amount of ground covered while thrusting averaged over all AUVs divided by the total distance covered averaged over all AUVs. Graphs are provided for all algorithms (excluding the BHV_AttractionRepulsion behaviour) for both trials and for both the hexagonal and square lattice formations. Since each vehicle is either thrusting or drifting, the ratios for the time spent drifting and the ground covered while drifting can be inferred as the inverse of these plots. In addition, in each plot we include the minimum and maximum envelopes of these ratios over all AUVs, as an indication of the variance in the values of both ratios. As one of the major goals of this work is to develop an algorithm that takes most advantage of ocean currents for propulsion, the behaviour that minimizes the amount of time and distance spent thrusting can be seen as preferable.

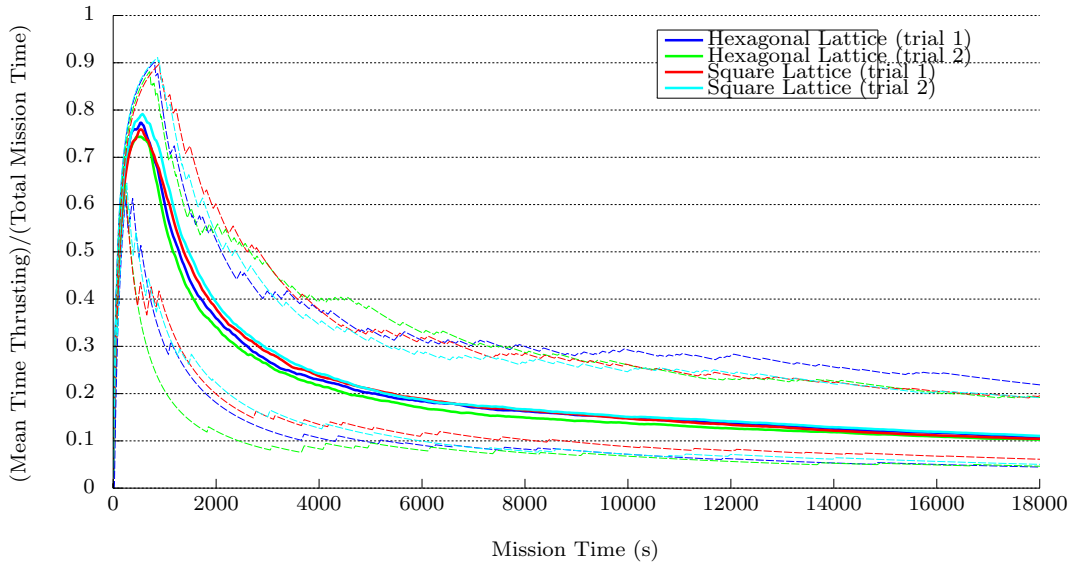

Figure 5.76: BHV_PairwiseNeighbourReferencing - ratio of time spent thrusting averaged over all AUVs to total mission time in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

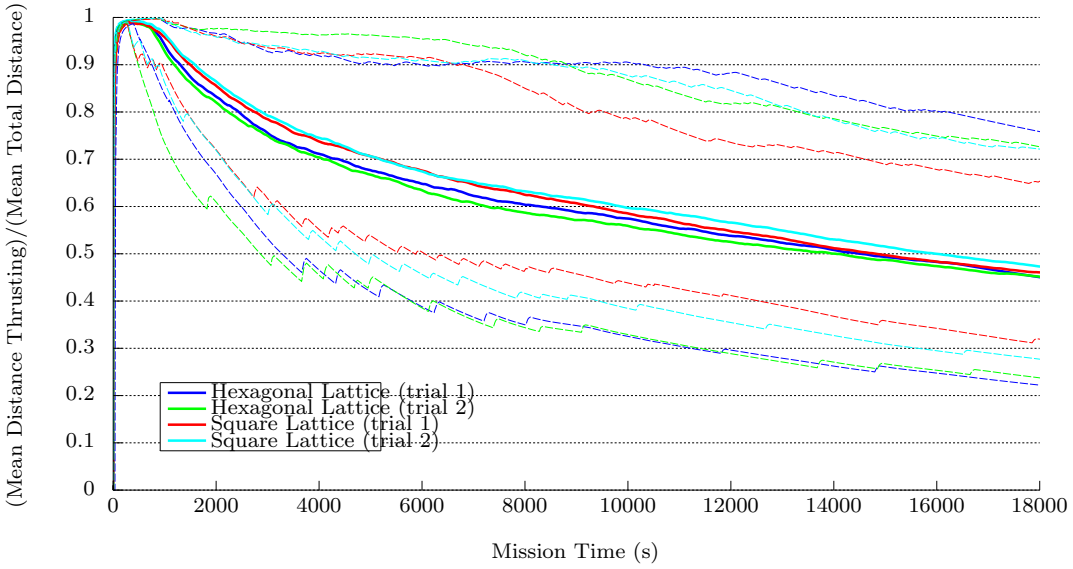

Figure 5.77: BHV_PairwiseNeighbourReferencing - ratio of distance travelled while thrusting averaged over all AUVs to total distance travelled averaged over all AUVs in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

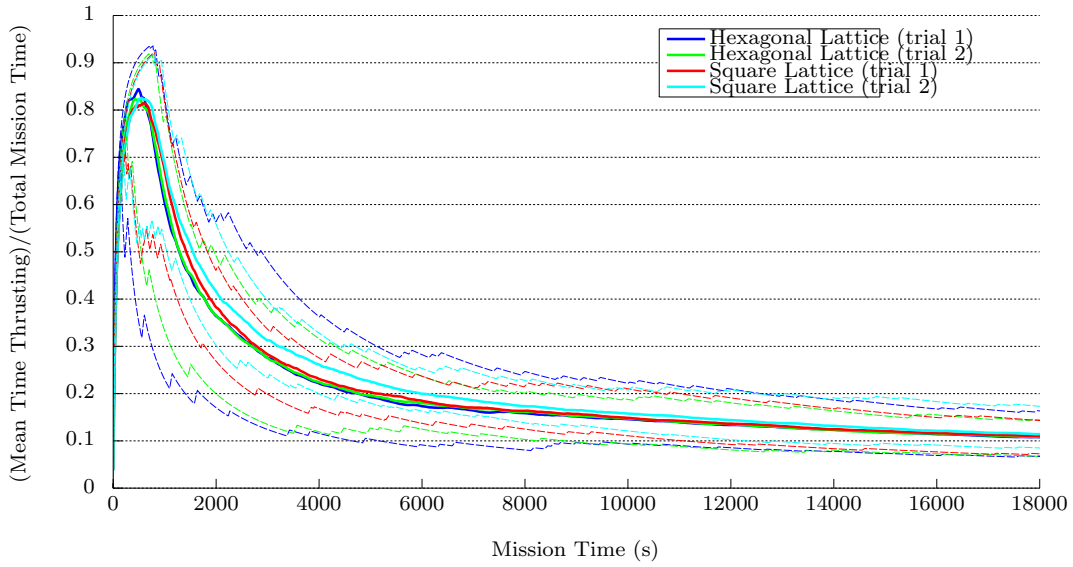

Figure 5.78: BHV_RigidNeighbourRegistration - ratio of time spent thrusting averaged over all AUVs to total mission time in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

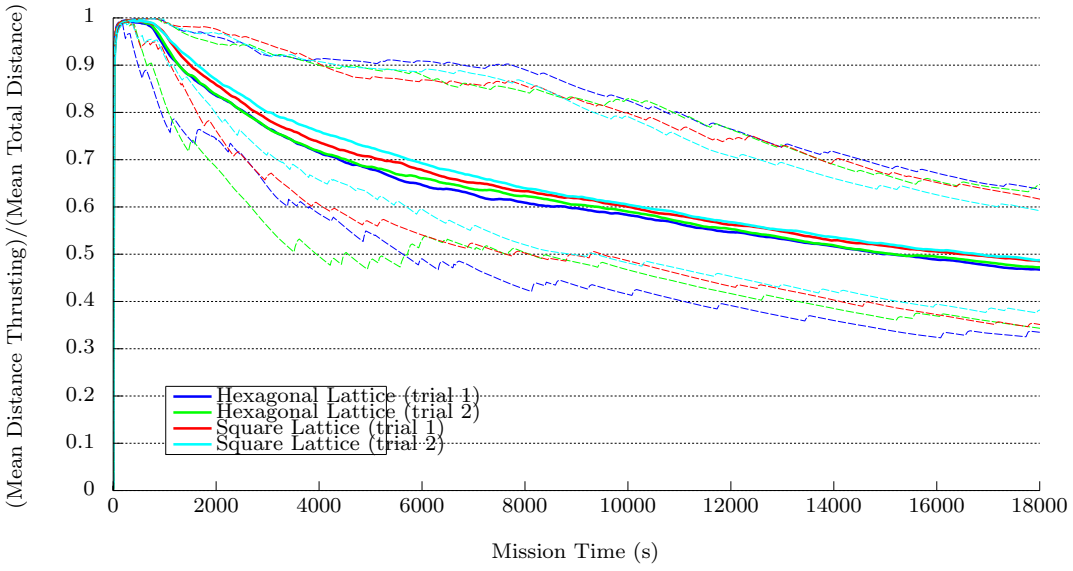

Figure 5.79: BHV_RigidNeighbourRegistration - ratio of distance travelled while thrusting averaged over all AUVs to total distance travelled averaged over all AUVs in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

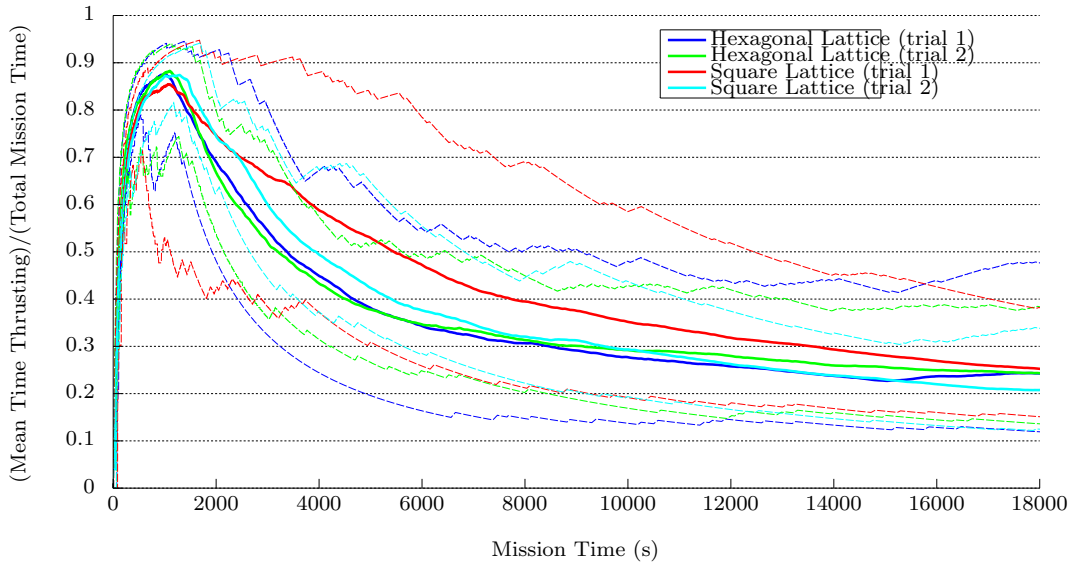

Figure 5.80: BHV_AssignmentRegistration - ratio of time spent thrusting averaged over all AUVs to total mission time in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

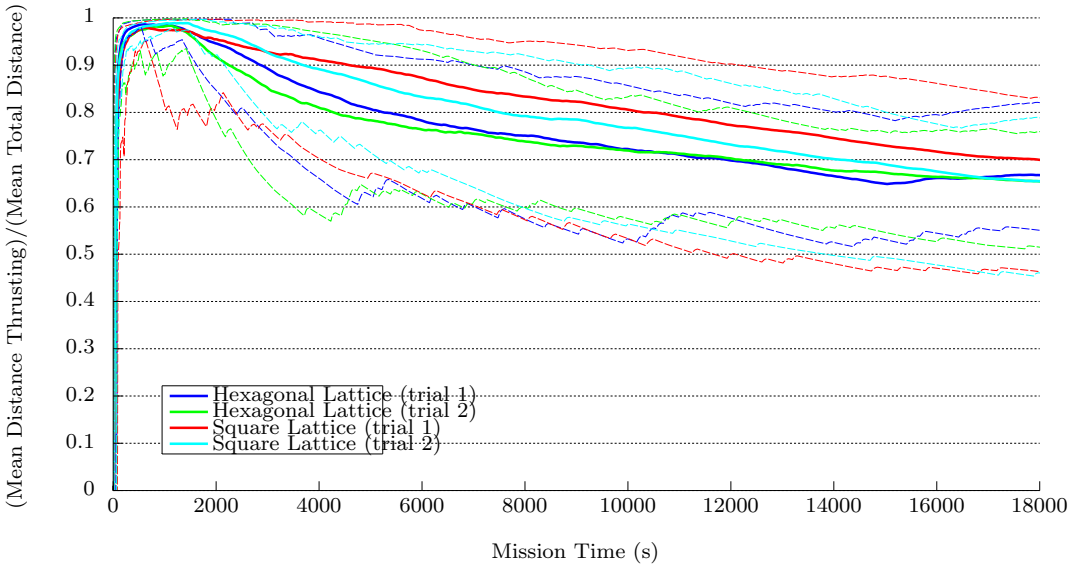

Figure 5.81: BHV_AssignmentRegistration - ratio of distance travelled while thrusting averaged over all AUVs to total distance travelled averaged over all AUVs in the Red Sea scenario; dashed lines indicate minimum and maximum envelopes of this ratio over all AUVs.

Examining figures 5.76 to 5.81, we can make a few of observations. Firstly, as expected, during the construction phase both the ratio of time and distance spent thrusting quickly approaches 1, as all vehicles move in order to form the desired lattice. Secondly, the ratio of time spent thrusting falls much more quickly than the distance travelled while thrusting - this is expected, since the simulated ocean currents only propel the vehicles at $\frac{1}{10}$ to $\frac{1}{5}$ the speed as it is capable of when thrusting; as such, although the time spent thrusting is a small fraction of the total time (approaching 0.1 for both BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration), the distance travelled is disproportionately larger (reaching just under 0.5 for these two behaviours). These figures indicate again the superiority of the BHV_PairwiseNeighbourReferencing algorithm, which is able to just edge out the BHV_RigidNeighbourRegistration in terms of minimizing time and distance spent thrusting. However, examining the minimum and maximum envelopes, we again see, as was the case with the energy expenditure metric, that BHV_RigidNeighbourRegistration is much more consistent, with a smaller variance in ratios. This consistency may endow BHV_RigidNeighbourRegistration with an advantage over BHV_PairwiseNeighbourReferencing, as previously explained. As was the case with many of the previous metrics, BHV_AssignmentRegistration is the worst performer in this metric out of these three behaviours.

5.5 Some Closing Observations

The previous sections have presented the reader with a raft of observations drawn from an examination of a variety of metrics and scenarios. We close this chapter with a few final comments about the behaviour of each algorithm when viewed in motion, providing some qualitative observations.

We begin with the BHV_AttractionRepulsion behaviour. It is fairly obvious from both the figures in this chapter, as well as from observing the algorithm in action, that this behaviour is not able to address the goals of this project set out at the beginning of this thesis. As visible in the majority of the metrics, and in the AUV trajectories in each of the scenarios, this algorithm results in chaotic vehicle movement, low formation quality (due to defects and formation fracturing), and a high rate of energy expenditure, making it unsuitable for our purposes. Its behaviour varies quite significantly with changes in the communications rate, which is an additional downside. In terms of node loss, the algorithm endows the formation with some ability to self-repair (which was observed a number of times during the course of testing), although it is debatable whether this is an advantage for our particular application. It is interesting to note that the majority of the existing literature that describe algorithms of physics-based swarm formation control do not introduce external disturbances as was the case in this work; in addition, many of the previous approaches limit themselves to observing

the formation only during the construction phase. It would be interesting to test these other physics-based approaches with the same stresses applied to the formation as were introduced in these scenarios, to observe how well they are able to cope. However, I find it unlikely that a physics-based approach can suitably handle the objective of this work.

The BHV_PairwiseNeighbourReferencing behaviour provided a few interesting results. This relatively simple approach using trigonometric principles resulted in formations that were efficiently constructed, and were well maintained in ocean currents. In fact, in terms of energy expenditure, this algorithm quite consistently outperformed all other behaviours in each scenario. In terms of formation quality, results were mixed, but in most cases the algorithm was able to maintain a formation quality level comparable to BHV_RigidNeighbourRegistration, the behaviour that most consistently performed best in this metric. However, this behaviour does have some disadvantages. Firstly, in terms of the formation quality metric, its value changes (sometimes significantly) depending on the communications rate. Secondly, in terms of energy expenditure, the variance in this value can be quite large, and certainly is larger than the variance seen when using the BHV_RigidNeighbourRegistration algorithm, over all AUVs. In the presence of vehicle loss, this behaviour is able to maintain the desired formation quite robustly; however, we again make the caveat that during these trials (with a loss of $\frac{1}{4}$ of AUVs in the swarm), vehicles were not lost in such a way that any one remaining vehicle could not sense at least two neighbours. Again, a downside of this approach is that each vehicle requires at least two neighbours to position itself, versus only one when using BHV_RigidNeighbourRegistration. Finally, we note that this behaviour is able to address the problem targeted by this work very well; and is the best performer in terms of energy and travel efficiency, at least in the sense where these metrics are averaged over all vehicles in the swarm.

BHV_RigidNeighbourRegistration shares many of the same qualities as the BHV_PairwiseNeighbourReferencing behaviour. This algorithm was also able to efficiently construct the desired formation, and maintain it well in ocean currents. In terms of formation quality, it most often outperformed all other behaviours, and it maintains a formation quality level very well and very consistently in the face of changing communications rate, providing robustness in this sense. In terms of energy expenditure, it almost achieves a similar efficiency as that provided by the BHV_PairwiseNeighbourReferencing algorithm, which is the best performer; however, as noted previously, the swarm using the BHV_RigidNeighbourRegistration algorithm always has a much lower variance in energy expenditure over all vehicles, giving vehicles a much more consistent rate of energy use no matter their position in the formation. In our application this may be a significant advantage, if we consider our mission to be over the moment the first vehicle in the swarm has run out of energy. In the presence of vehicle loss this algorithm is also able to maintain the formation without loss in formation quality; how-

ever, as with BHV_PairwiseNeighbourReferencing, during these trials vehicles were not lost in such a way that any one of the remaining vehicles became stranded. However, we note an additional advantage of this algorithm over BHV_PairwiseNeighbourReferencing in terms of vehicle loss - this algorithm is more robust, simply because any one vehicle in the swarm only requires a single neighbour to position itself. As with BHV_PairwiseNeighbourReferencing, BHV_RigidNeighbourRegistration does not provide the swarm with any capacity for self-repair, given that vehicles have a pre-designated position in the formation; whether or not this is a disadvantage in our application is debatable. Finally, we state that this algorithm is also able to address our desired application very well.

Finally, some notes about the BHV_AssignmentRegistration behaviour - although it was unable to fulfil our conjecture of improved energy expenditure through the dynamic assignment of vehicles to formation positions, this novel algorithm behaved in interesting ways; and certainly, in comparison to BHV_AttractionRepulsion (the only other behaviour that does not require the communication of globally unique vehicle IDs) it performed quite well in terms of energy expenditure, and especially in terms of formation quality. For algorithms that only use range/bearing measurements to neighbouring vehicles, this algorithm likely provides a much better approach to formation control than any physics-based approach, allowing us to construct lattice formations of different shapes using minimal information. We again stress the fact that the performance of this algorithm appears to be highly sensitive to its user specified $\delta\theta$ parameter, which controls the number of its rotation choices for local point-set fitting. The algorithm essentially works by continuously rotating the point-set comprising of the vehicle and its neighbours by a specified $\delta\theta$ offset, comparing this point-set to different parts of the formation plan, and selecting the rotation that results in the lowest cost output of the Hungarian algorithm. With a larger $\delta\theta$ (as in the first scenario, where it was set to 45°), the algorithm has fewer rotation choices, and each vehicle is more likely to select a rotation that is consistent across the swarm, enabling the efficient construction of the desired formation; however, this also causes issues during formation maintenance, especially when the formation rotates in ocean currents. If the formation rotates, there occur specific angles where the minimum cost is very similar between two different rotations, and vehicles end up becoming indecisive, alternately selecting these two rotations; and since $\delta\theta$ is large, jumping between the two angles can cause the formation to fracture, a behaviour that was observed during testing in the Red Sea scenario (which prompted us to select a lower $\delta\theta$ value). Selecting a lower $\delta\theta$ value (as in the last three scenarios, where it was set to 10°) avoids this issue, but at the expense of lower stability and efficiency during the formation construction phase (since vehicles then tend to select different rotation values from each other for their best local point-set fit). This is the reason why the BHV_AssignmentRegistration was able to outperform all other behaviours in terms of both energy efficiency and quickness in achieving a high

formation quality during the first scenario, but was unable to do so in the final Red Sea scenario. Thus, if we were somehow able to balance this trade-off, and reduce its sensitivity to the $\delta\theta$ parameter, this algorithm has the potential to perform very well in our desired application.

We make one final note about the loss of vehicles in the swarm, and whether or not a swarm self-reparation capability is an advantage or not for our particular scenario. Firstly, we note that both BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration do not provide the swarm with the ability to self-repair, a consequence of pre-designating vehicles to formation positions; however, this inability to self-repair resulted in better performance in terms of formation quality during tests of node loss. In contrast, both BHV_AttractionRepulsion and BHV_AssignmentRegistration both provide the swarm with some ability to self-repair. BHV_AttractionRepulsion provides this because of its physics-based nature, and BHV_AssignmentRegistration provides this because it dynamically assigns vehicles to formation positions. However, this capability resulted in a worse formation quality performance for these two behaviours, since by its very nature, self-repair results in a degradation of formation quality as a neighbouring vehicle transits to fill the 'hole' generated by the loss of a vehicle, before improving again. Secondly, we question whether or not self-reparation is advantageous in our application - given the unreliable nature of acoustic communications in the underwater environment, it is entirely possible that vehicles may not be able to sense some neighbours from time to time. When this occurs, how do we decide if this neighbour absence is due to the actual loss of the neighbour or due simply to loss of communications? If this check is not robustly implemented, we could end up with a situation where a vehicle believes that it has lost its neighbour, and if self-repair was an option, it may move to place itself at the position where its neighbour was last sensed; such a situation can cause confusion in the formation and possibly vehicle collision, as two vehicles place themselves in the same location. As such, I argue that self-reparation is in fact a possible hindrance for swarms in the underwater domain.

5.6 Summary

This chapter has provided the reader with graphical results of the testing methodology described in chapter 4, and detailed a large number of observations that were inferred from these results.

In the next and final chapter of this thesis, we provide a discussion of the results of the analysis, conclusions drawn from this research, and future work to be undertaken.

Chapter 6

Discussion and Conclusion

The operational understanding and underlying technology of autonomous underwater vehicles has now matured to a point whereby smaller and less expensive AUVs are beginning to become a reality. The low cost of these vehicles, and the consequently higher tolerance for risk, means that large multi-robot operations will soon become a very real possibility. The application of swarm intelligence concepts to these multi-robot systems has the potential to open up new areas of application for AUVs. This thesis has presented a study into one particular swarm intelligence strategy for a group of AUVs, namely formation control; the successful implementation of this strategy on such a system in the field presents an opportunity to more effectively characterise complex oceanographic phenomena and the ocean environment.

This thesis has presented three main contributions towards the goal of the successful control of a swarm of AUVs in formation. The first contribution is the augmentation of the existing MOOS-IvP architecture in order to produce a MOOS-IvP Simulation Test-Bed that is capable of effectively simulating a large number of AUVs in realistic ocean currents with the associated vehicle models and sensors. The second contribution is the development and implementation of four different formation control behaviours for a swarm of AUVs - BHV_AttractionRepulsion, which was inspired by the physics of atoms; BHV_PairwiseNeighbourReferencing, which uses simple trigonometric principles; BHV_RigidNeighbourRegistration, which was based on the rigid point-set registration problem; and BHV_AssignmentRegistration, a novel algorithm which performs dynamic point correspondence and uses local rigid optimal transformations on subsets of points. Each of these behaviours was evaluated against a number of metrics, and their relative performance analysed in great detail. The third and final contribution was the introduction of a metric to quantify how well a swarm of vehicles adheres to a desired formation. The main findings of this work are:

- The MOOS-IvP architecture is well suited for the effective simulation of a large number of AUVs using fairly realistic vehicle models, sensors, and models of the ocean environ-

ment. It provides a reliable test-bed for the study of swarm intelligence concepts with unmanned surface vehicles and autonomous underwater vehicles.

- The formation quality metric developed in this work provides a robust quantitative assessment of how well a swarm of vehicles is able to conform to a desired formation.
- Each of the four formation control strategies is able to construct and maintain lattice formations in a distributed manner, to differing levels of effectiveness.
- Two of the four behaviours, BHV_PairwiseNeighbourReferencing and BHV_RigidNeighbourRegistration, were the best performers across all metrics (including energy expenditure, formation quality, and average time and distance spent thrusting). These two behaviours are also able to construct formations of any arbitrary shape, at the cost of requiring the communication of globally unique vehicle identifiers.
- Based on the analysis of the metrics from each behaviour detailed in chapter 5, as well as an understanding of the advantages and disadvantages of each behaviour, the author recommends the use of the BHV_RigidNeighbourRegistration behaviour for implementation on physical vehicles in future work. BHV_RigidNeighbourRegistration performs almost as well as BHV_PairwiseNeighbourReferencing in terms of energy expenditure and travel efficiency, outperforms all other behaviours in terms of formation quality, is robust to changing communications rate and vehicle loss, and allows us to produce formations of arbitrary shape.

It is hoped that the contributions and findings of this thesis have provided a firm foundation for future study into formation control of a group of AUVs, especially for the purposes of field-testing such a system.

6.1 Future Work

The work undertaken in this thesis leaves us with many avenues of future work to pursue. First and foremost, the practical implementation and testing of these behaviours on actual, real-world vehicles is a priority. To simplify this endeavour, we propose two stages of practical field-testing. In the first stage, we recommend the use of our fleet of Kingfisher autonomous surface craft (ASC), along with simulated acoustic communications, to evaluate the fundamental ability of our algorithms. ASCs are still subjected to ocean currents, and the use of simulated acoustic communications avoids the initial complexities of developing the required communications hardware. In the second stage, we propose the use of low-cost, miniature AUVs (such as the Bluefin SandShark [71]) equipped with Chip Scale Atomic Clocks (CSACs) and acoustic pingers to determine range between vehicles using time of flight; a 3D hydrophone

array (for example, in a triangular pyramid configuration) to determine bearing information to neighbouring vehicles; and either acoustic modems or the use of unique pinger frequencies to communicate unique vehicle IDs.

The second avenue for future research is the study of the theoretical properties of our formation control algorithms. The focus of this thesis was on the practical implementation and understanding of these algorithms, but a theoretical understanding may prove useful. This may begin with an analysis of the computational complexity of each of our behaviours, followed by an understanding of their properties in terms of scalability, and finally, a study of their convergence properties and whether or not any guarantees can be made in terms of convergence to a desired formation.

A third path of future work is further research into, and improvement of, the BHV_AssignmentRegistration behaviour. Consider the following question - imagine gathering twenty strangers on a football field; you tell each of them to arrange themselves in a grid of 4×5 people spaced $20m$ apart, with the restriction that they are not allowed to communicate to each other in any manner whatsoever; how well would this group of strangers be able to construct the desired grid? In the author's opinion, it is likely that the group would be able to perform this task, but not without a great deal of indecisive back-and-forth movement, and certainly not in an efficient manner. Now imagine that each person was subjected to an external force that slowly shifted them about the field. This anecdote is analogous to what we are trying to perform using only range/bearing measurements with a swarm of AUVs; and finding an efficient solution to this problem is not an easy task. The BHV_AssignmentRegistration algorithm is a novel approach that appears to be surprisingly competent at performing this exercise, and it is the author's belief that its performance can be improved with further work, especially with regards to the selection of point subsets from the formation plan for comparison to the vehicle and its neighbours, as well as in gaining a better understanding of the effect of varying values of the $\delta\theta$ parameter, and finally in investigating its capabilities for swarm self-reparation. One possibility for further research into the improvement of this algorithm is the use of shape contexts [72]. Shape contexts are feature descriptors used widely in computer vision research to perform shape matching and object recognition; the use of a descriptor similar to the shape context descriptor with this algorithm opens up the possibility of using BHV_AssignmentRegistration to construct arbitrary formations by attempting to match subsets of vehicles to different parts of the desired formation based on its shape. All in all, the BHV_AssignmentRegistration algorithm shows promise as an approach to formation control using minimal neighbour information, and warrants further investigation.

The final avenue for further research is the combination of our formation control behaviours

with other existing MOOS-IvP behaviours for obstacle and area avoidance. The IvP Helm is useful precisely because it arbitrates between different behaviours based on priority, and as such, it is ideally suited for the inclusion of additional behaviours. In this thesis only two behaviours were active at any one time, and they presided over control of domains that did not intersect - the formation control behaviour influenced vehicle heading and speed, and a depth control behaviour maintained vehicle depth. As such, we did not consider vehicle collisions - testing should be performed with obstacle avoidance behaviours to prevent such collisions. In addition, we would eventually like the swarm to operate in environments of complex geometry, or environments that include keep-out areas (such as ocean infrastructure) - future work should include simulations with behaviours that maintain the formation while restricting the swarm to valid operational areas, and the evaluation of swarm performance in this context.

Bibliography

[1] B.W. Hobson, et al., *Tethys-class long range AUVs - extending the endurance of propeller-driven cruising AUVs from days to weeks*. Autonomous Underwater Vehicles (AUV), 1-8, 2012.

[2] A. Caffaz, et al., *The Hybrid Glider/AUV Folaga*. Robotics & Automation Magazine, Volume 17, Issue 1, 31-44, 2010.

[3] M. Benjamin, H. Schmidt, P. Newman, J. Leonard, *Nested Autonomy for Unmanned Marine Vehicles with MOOS-IvP*. Journal of Field Robotics, Volume 27, Issue 6, 834-875, 2010.

[4] G. Beni, *From Swarm Intelligence to Swarm Robotics*. Proceedings of the 2004 International Conference on Swarm Robotics, 1-9, 2005.

[5] C. Pinciroli, R. O'Grady, A.L. Christensen, M. Dorigo, *Self-organised recruitment in a heteregeneous swarm*. International Conference on Advanced Robotics, 1-8, 2009.

[6] D. Payton, et al., *Pheromone Robotics*. Autonomous Robots, Volume 11, Issue 3, 319-324, 2001.

[7] A. Jevtic, A. Gutierrez, D. Andina, M. Jamshidi, *Distributed Bees Algorithm for Task Allocation in Swarm of Robots*. IEEE Systems Journal, Volume 6, Issue 2, 296-304, 2012.

[8] C.W. Reynolds, *Flocks, Herds and Schools: A Distributed Behavioral Model*. Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, 25-34, 1987.

[9] Y. Yang, Y. Tian, *Swarm robots aggregation formation control inspired by fish school*. IEEE International Conference on Robotics and Biomimetics, 805-809, 2007.

[10] A. Shklarsh, G. Ariel, E. Schneidman, E. Ben-Jacob, *Smart Swarms of Bacteria-Inspired Agents with Performance Adaptable Interactions*. PLoS Computational Biology, Volume 7, Issue 9, 1-11, 2011.

[11] J.C. Barca, Y.A. Sekercioglu, *Swarm Robotics Reviewed*. Robotica, Volume 31, Issue 3, 345-359, 2013.

[12] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, *Swarm robotics: a review from the swarm engineering perspective*. Swarm Intelligence, Volume 7, Issue 1, 1-41, 2013.

[13] I. Navarro, F. Matía, *An Introduction to Swarm Robotics*. ISRN Robotics, Volume 2013, 2013.

[14] Y. Tan, Z. Zheng, *Research Advance in Swarm Robotics*. Defence Technology, Volume 9, Issue 1, 18-39, 2013.

[15] L. Bayindir, E. Sahin, *A Review of Studies in Swarm Robotics*. Turkish Journal of Electrical Engineering & Computer Sciences, Volume 15, Issue 2, 115, 2007.

[16] D.V. Dimarogonas, K.J. Kyriakopoulos, *Connectedness Preserving Distributed Swarm Aggregation for Multiple Kinematic Robots*. IEEE Transaction on Robotics, Volume 24, Issue 5, 1213-1223, 2008.

[17] M. Gauci, et al., *Self-Organised Aggregation without Computation*. The International Journal of Robotics Research, Volume 33, Issue 8, 1145-1161, 2014.

[18] J. Mclurkin, J. Smith, *Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots*. 7th International Symposium on Distributed Autonomous Robotic Systems, 2004.

[19] F. Ducatelle, A. Foerster, G.A. DiCaro, L.M. Gambardella, *New task allocation methods for robotic swarms*. Proceedings of the 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions, 2009.

[20] A.E. Turgut, H. Celikkanat, F. Gokce, E. Sahin, *Self-organized flocking in mobile robot swarms*. Swarm Intelligence, Volume 2, Issue 2-4, 97-120, 2008.

[21] E. Ferrante, et al., *Self-organized flocking with a mobile robot swarm: a novel motion control method*. Adaptive Behavior, 2012.

[22] J. Chen, et al., *Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots*. IEEE Transactions on Robotics, Volume 31, Issue 2, 307-321, 2015.

[23] F. Ducatelle, C.P.G.A. DiCaro, L.M. Gambardella, *Self-organized cooperation between robotic swarms*. Swarm Intelligence, Volume 5, Issue 2, 73-96, 2011.

[24] A. Howard, L.E. Parker, G.S. Sukhatme, *Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection*. International Journal of Robotics Research, Volume 25, Issue 5-6, 431-447, 2006.

[25] E. Bahceci, O. Soysal, E. Sahin, *A Review: Pattern Formation and Adaptation in Multi-Robot Systems*. Tech. Report, Robotics Institute, Carnegie Mellon University, 2003.

[26] Y.Q. Chen, Z. Wang, *Formation Control: A Review and A New Consideration*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 3181-3186, 2005.

[27] W. Spears, et al., *Distributed, physics-based control of swarms of vehicles*. Autonomous Robots, Volume 17, Issue 2-3, 137-162, 2004.

[28] W. Spears, et al., *An Overview of Physicomimetics*. Swarm Robotics, Lecture Notes in Computer Science, Volume 3342, 84-97, 2005.

[29] S. Prabhu, W. Li, J. McLurkin, *Hexagonal Lattice Formation in Multi-Robot Systems*. 11th International Conference on Autonomous Agents and Multiagent Systems, 2012.

[30] C. Pinciroli, et al., *Self-organizing and scalable shape formation for a swarm of pico satellites*. NASA/ESA Conference on Adaptive Hardware and Systems, 2008.

[31] V. Gazi, K.M. Passino, *A class of attraction/repulsion functions for stable swarm aggregations*. 41st IEEE Conference on Decision and Control, Volume 3, 2842-2847, 2002.

[32] K. Fujibayashi, et al., *Self-organizing formation algorithm for active elements*. 21st IEEE Symposium on Reliable Distributed Systems, 2002.

[33] B. Shucker, J.K. Bennett, *Scalable Control of Distributed Robotic Macrosensors*. Distributed Autonomous Robotic Systems 6, Part 9, 379-388, 2007.

[34] R. Stolkin, J.V. Nickerson, *Combining multiple autonomous mobile sensor behaviors using local clustering*. IEEE Military Communications Conference, 2005.

[35] R. Bachmayer, N.E. Leonard, *Vehicle networks for gradient descent in a sampled environment*. 41st IEEE Conference on Decision and Control, Volume 1, 112-117, 2002.

[36] L. Chaimowicz, N. Michael, V. Kumar, *Controlling Swarms of Robots Using Interpolated Implicit Functions*. IEEE International Conference on Robotics and Automation, 2487-2492, 2005.

[37] L. Barnes, *A potential field based formation control methodology for robot swarms*. Ph.D. Dissertation, University of South Florida, 2008.

[38] M.A. Lewis, K.H. Tan, *High Precision Formation Control of Mobile Robots Using Virtual Structures*. Journal of Autonomous Robots, Volume 4, Issue 4, 387-403, 1997.

[39] W. Ren, R. Beard, *Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach*. Journal of Guidance, Control, and Dynamics, Volume 27, Issue 1, 73-82, 2004.

[40] C. Belta, V. Kumar, *Motion generation for formations of robots: A geometric approach*. IEEE International Conference on Robotics and Automation, Volume 2, 1245-1250, 2001.

[41] M.B. Egerstedt, X. Hu, *Formation Constrained Multi-Agent Control*. IEEE Transactions on Robotics and Automation, Volume 17, Issue 6, 947-951, 2001.

[42] J.P. Desai, J.P. Ostrowski, V. Kumar, *Modeling and control of formations of nonholonomic mobile robots*. IEEE Transactions on Robotics and Automation, Volume 17, Issue 6, 905–908, 2001.

[43] G.J. Mariottini, et al., *Leader-follower formations: uncalibrated vision-based localization and control*. IEEE International Conference on Robotics and Automation, 2403–2408, 2007.

[44] K. Wesselowski, R. Fierro, *A dual-mode model predictive controller for robot formations*. IEEE Conference on Decision and Control, 3615–3620, 2003.

[45] D. Gu, H. Hu, *A model predictive controller for robots to follow a virtual leader*. Robotica, Volume 27, 905-913, 2009.

[46] J. Sanchez, R. Fierro, *Sliding mode control for robot formations*. IEEE International Symposium on Intelligent Control, 438-443, 2003.

[47] G.H. Elkaim, R.J. Kelbley, *A lightweight formation control methodology for a swarm of non-holonomic vehicles*. IEEE Aerospace Conference, 2006.

[48] Y. Song, J.M. O'Kane, *Decentralized Formation of Arbitrary Multi-Robot Lattices*. IEEE International Conference on Robotics and Automation, 2014.

[49] G. Lee, N.Y. Chong, *Self-configurable mobile robot swarms with hole repair capability*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 1403-1408, 2008.

[50] G. Lee, N.Y. Chong, *Decentralized formation control for small-scale robot teams with anonymity*. Mechatronics, Volume 19, Issue 1, 85-105, 2009.

[51] G. Antonelli, F. Arrichiello, S. Chiaverini, *Flocking for multi-robot systems via the Null-Space-based Behavioral control*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 1409-1414, 2008.

[52] R.L. Raffard, C.J. Tomlin, S.P. Boyd, *Distributed Optimization for Cooperative Agents: Application to Formation Flight*. IEEE Conference on Decision and Control, 2004.

[53] Z. Hu, C. Ma, L. Zhang, A. Halme, *Distributed formation control of autonomous underwater vehicles with impulsive information exchanges and disturbances under fixed and switching topologies*. IEEE International Symposium on Industrial Electronics, 99-104, 2014.

[54] A. Amory, et al., *Towards Fault-Tolerant and Energy-Efficient Swarms of Underwater Robots*. IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 1550-1553, 2013.

[55] S. Kalantar, U.R. Zimmer, *Distributed shape control of homogeneous swarms of autonomous underwater vehicles*. Autonomous Robots, Volume 22, Issue 1, 37-53, 2007.

[56] J. Shao, J. Yu, L. Wang, *Formation Control of Multiple Biomimetic Robotic Fish*. IEEE International Conference on Intelligent Robots and Systems, 2503-2508, 2006.

[57] T. Schmickl, et al., *CoCoRo–The Self-Aware Underwater Swarm*. IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, 120-126, 2011.

[58] T. Schneider, *Advances in Integrating Autonomy with Acoustic Communications for Intelligent Networks of Marine Robots*. Ph.D. dissertation, Joint Program in Applied Ocean Science and Engineering at the Massachusetts Institute of Technology and the Woods Hole Oceanographic Institution, 2013.

[59] M. Benjamin, P. Newman, H. Schmidt, J. Leonard, *Extending a MOOS-IvP Autonomy System and Users Guide to the IvPBuild Toolbox*. MIT Computer Science and Artificial Intelligence Laboratory Technical Report, 2009.

[60] L. Bragg, J.F. Nye, *A Dynamical Model of a Crystal Structure*. Proceedings of the Royal Society of London, Volume 190, Issue 1023, 474-481, 1947.

[61] S.G. Johnson, *The NLopt nonlinear-optimization package*. http://ab-initio.mit.edu/nlopt, 2014, accessed 11 June 2015.

[62] P.J. Besl, N.D. McKay, *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 14, Issue 2, 239-256, 1992.

[63] O. Sorkine, *Least-Squares Rigid Motion Using SVD*. https://igl.ethz.ch/projects/ARAP/svd_rot.pdf, 2007, accessed 13 June 2015.

[64] C. Sanderson, *Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments*. http://arma.sourceforge.net/, 2015, accessed 14 June 2015.

[65] H.W. Kuhn, *The Hungarian method for the assignment problem*. Naval Research Logistics Quarterly, Volume 2, Issue 1-2, 83–97, 1955.

[66] L.S. Zehnder, *Munkres' Assignment Algorithm with RcppArmadillo*. http://gallery.rcpp.org/articles/minimal-assignment/, 2013, accessed 15 June 2015.

[67] M. Benjamin, *Welcome to the MOOS-IvP Home Page*. http://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php, 2015, accessed 17 June 2015.

[68] T. Schneider, *Goby Underwater Autonomy Project*. http://gobysoft.org/index.wt/software/goby, 2015, accessed 1 July 2015.

[69] *MSEAS - multidisciplinary simulation, estimation, and assimilation systems*. http://mseas.mit.edu/, 2015, accessed 9 July 2015.

[70] P.J. Haley, P.F.J. Lermusiaux, *Multiscale two-way embedding schemes for free-surface primitive equations in the "Multidisciplinary Simulation, Estimation and Assimilation System"*. Ocean Dynamics, Volume 60, Issue 6, 1497-1537, 2010.

[71] *Bluefin SandShark*. http://www.bluefinrobotics.com/products/bluefin-sandshark/, 2015, accessed 12 August 2015.

[72] S. Belongie, J. Malik, J. Puzicha, *Shape Context: A New Descriptor for Shape Matching and Object Recognition*. Conference on Neural Information Processing Systems, 831-837, 2000.