DESIGN OF A LOW-COST UNMANNED SURFACE VEHICLE FOR SWARM ROBOTICS RESEARCH IN LABORATORY ENVIRONMENTS

by © Calvin Gregory

A thesis submitted to the School of Graduate Studies

in partial fulfilment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October 2020

St. John's Newfoundland

Abstract

Swarm robotics is the study of groups of simple, typically inexpensive agents working collaboratively toward a common goal. Such systems offer several benefits over single-robot solutions: they are flexible, scalable, and robust to the failure of individual agents. The majority of existing work in this field has focused on robots operating in terrestrial environments but the benefits of swarm systems extend to applications in the marine domain as well. The current scarcity of marine robotics platforms suitable for swarm research is detrimental to progress in this field. Of the few that exist, no publicly available unmanned surface vehicles can operate in a laboratory environment; an indoor tank of water where the vessels, temperature, lighting, etc. can be observed and controlled at all times. Laboratory testing is a common intermediate step in the hardware validation of algorithms. This thesis details the design of the microUSV: a small, inexpensive, laboratory-based platform developed to fill this gap.

The microUSV system was validated by performing laboratory testing of two algorithms: a waypoint-following controller and orbital retrieval. The waypoint-following controller was a simple PI controller implementation which corrects a vessel's speed and heading to seek predetermined goal positions. The orbital retrieval algorithm is a novel method for a swarm of unmanned surface vehicles to gather floating marine contaminants such as plastics. The vessels follow a circular path, orbiting around a central collection location and veer outwards to retrieve contaminants they detect outside the designated area. This method can potentially be used to cluster floating plastics together from a large region to facilitate cleanup.

Acknowledgements

I wish to extend my deepest thanks to Dr. Andrew Vardy and Dr. Ralf Bachmayer for their guidance and supervision throughout this degree program: I could never have succeeded without you. Special thanks to Paul Brett and Joe Singleton for their support and providing access to facilities at the Marine Institute. You continue to open both physical and metaphorical doors for me. I am grateful to Michael Dawson, Anthony Randell, Whymarrh Whitby, and Zach Brown who provided invaluable suggestions and insight during the design process. Thanks also to Hilary Sinclair, Jerry Smith, and Mitchell Burke who helped me with fabrication and the lab technicians, Craig Mitchell, Trevor Clark, and Matt Curtis for helping me set up my experiments. Finally, I would like to thank my parents for their constant encouragement and confidence in my accomplishments; especially when I was in doubt.

The work discussed in this thesis was partially supported by the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Tables

List of Figures

List of Code Listings

List of Acronyms

ASV	Autonomous Surface Vehicle. 11
AUV	Autonomous Underwater Vehicle. 10
BOM	Bill of Materials. 40
BOTS	Bio-Inspired Robotics Lab. 2, 10
CVSS	CVSensorSimulator. 5, 45, 73, 91, 114
FDM	Fused Deposition Modeling. 28, 113
GM	Metacentric Height. 25
GPS	Global Positioning System. 38, 61, 88
GZ	Righting Lever Length. 25
HSV	Hue, Saturation, Value. 52
IMU	Inertial Measurement Unit. 36, 48
INS	Inertial Navigation System. 61

IP Internet Protocol. 73

JSON JavaScript Object Notation. 74

LiPo Lithium Polymer. 117

MRS Multi-Robot Systems. 1

OEM Original Equipment Manufacturer. 18

PCB Printed Circuit Board. 117

PI Proportional-Integral. 74

PLA Polylactic Acid. 22

protobuf Protocol Buffer. 51

RC Remote Control. 13

TCP Transmission Control Protocol. 51

UAV Unmanned Aerial Vehicle. 9

UDP User Datagram Protocol. 51

USB Universal Serial Bus. 43, 71

USV Unmanned Surface Vehicle. 3, 10, 20, 84, 113

XML Extensible Markup Language. 51

Chapter 1

Introduction

Swarm robotics is a subset of Multi-Robot Systems (MRS): any collection of two or more robots working together [38]. A swarm robotics system is a MRS that has taken inspiration from the collaborative and self-organizing behaviors of social insects such as bees and ants [22]. It emphasizes the decentralized control of large numbers of small, inexpensive robots performing simple actions and interactions to accomplish tasks as well as, or often better than, a single large, expensive, and complex robot could achieve. Şahin [31] defines swarm robotics as follows:

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

The characteristics required of the physical agents Şahin describes are further elaborated on by Brambilla et al. [22]. They offer the following list of attributes that are required to be considered a swarm robotics system:

- The robots are autonomous.

- The robots are situated in the environment and can act to modify it.
- The robots' sensing and communication capabilities are local.
- The robots do not have access to centralized control and/or to global knowledge.
- The robots cooperate to tackle a given task.

Swarm robotics systems offer many benefits over traditional single or multi-robot systems in select applications. They are flexible, scalable, and robust [31, 95]. The workers in an ant colony are flexible and will reallocate themselves depending on the current available resources and needs of the colony without any central governing authority to coordinate their actions. Similarly, a swarm robotics system can implement and switch between modular behaviors to handle a wide variety of scenarios without requiring any changes to the physical robots themselves. Swarm solutions offer easy scalability: by increasing the number of robots a swarm's work scope and/or speed can be increased almost indefinitely. They are also able to handle the loss or failure of an arbitrary number of individuals with ease as the rest of the swarm will continue its mission uninterrupted. This fault-tolerant nature makes swarm systems very robust.

Such characteristics are potentially beneficial to missions conducted in any domain; terrestrial, aerial, or marine. Although the majority of existing research in this field is focused on terrestrial robots, there are examples of aerial and marine swarm robotics platforms in development as well [27, 32, 10]. The members of the Bio-Inspired Robotics Lab (BOTS) at Memorial University are interested in contributing to the research and development of swarm robotics in the marine domain, among other topics. Current interests include researching collective bathymetric mapping/-

surveying tasks and clustering of floating contaminants such as oil and plastic. These interests prompted the lab to begin an internal research initiative focused on marine swarm robotics.

A key aspect of the planned BOTS marine swarm robotics project was the acquisition of a hardware platform for algorithm testing and validation. The project had access to indoor laboratory testing facilities and a limited budget so it was decided to search for a small Unmanned Surface Vehicle (USV) that could be outfitted for indoor swarm testing operations at low cost. Such a testbed could be used as a stepping-stone to evaluate algorithms before investing in larger, more costly hardware for open water experiments on lakes or oceans. Unfortunately we were unable to find such an existing platform. The commercial and open-source USVs evaluated were too large, too expensive, and/or customized too heavily towards a specific application to serve as a general purpose research platform for the BOTS project. This prompted the researchers to develop their own.

This thesis will discuss the design, development, and testing of a novel, general purpose, low-cost marine swarm robotics research platform for use in laboratory environments called the microUSV. An article describing this platform titled *microUSV: A low-cost platform for indoor marine swarm robotics research* was published in April of 2020 [41]. It will also include a discussion of a preliminary algorithm for clustering floating marine contaminants called Orbital Retrieval which was implemented and tested on the microUSV platform. A publication discussing this work is expected to be published within the next year.

1.1 Thesis Objectives and Outline

This thesis aims to address a gap in the available marine robotics platforms. Its objectives include:

- Evaluate existing marine robotics platforms and identify their shortcomings specific to swarm robotics research applications.
- Design the hardware and software for a new marine swarm robotics research platform which addresses these shortcomings.
- Demonstrate the suitability of this new platform through experimental validation of marine swarm robotics research applications.

Chapter 2

The second chapter of this thesis reviews existing swarm robotics platforms and existing USV platforms. It also provides a formal problem definition for the development of the microUSV and the platform's design requirements and constraints.

Chapter 3

The third chapter describes the hardware design of the microUSV. It discusses the design decisions and justifications for the design of the microUSV's mechanical and electrical subsystems.

Chapter 4

The fourth chapter describes the microUSV system's software design. It discusses the general design of the control software running onboard the vessels as well as the

AprilTag-based pose tracking system called CVSensorSimulator (CVSS).

Chapter 5

The fifth chapter details the experiment used to validate the functionality of the microUSV system: a waypoint following test using a simple PI controller. The validation test scenarios included a linear path following test, an elliptical path following test, and a multi-vehicle test.

Chapter 6

The sixth chapter is a discussion of the Orbital Retrieval algorithm: a novel approach for collecting floating marine contaminants using a swarm of USVs. It includes a description of the algorithm, description of the procedures used to test it and analysis of the experimental results.

Chapter 7

The final chapter features concluding remarks and a discussion of future work. This includes a discussion of flaws in the microUSV design and potential solutions to be implemented in a future design revision.

Chapter 2

Problem Definition

This chapter provides a discussion of existing swarm robotics research platforms. It demonstrates the need for a new marine swarm robotics platform to fill the gap in existing research hardware. Such platforms are crucial to supplement simulation results when validating new algorithms experimentally. The chapter concludes by defining the design criteria and constraints for the development of such a platform including a formal problem statement.

Simulation is the most popular tool for developing and testing algorithms among swarm robotics researchers due to the high cost and complexity of acquiring and maintaining a swarm of physical robots [22]. Modern simulation environments are robust and fairly easy to use so many experiments in swarm robotics are conducted solely through simulation or models, some of the most popular environments being ARGoS [70], Gazebo [51], CoppeliaSim [76], and Stage [91]. Although useful, these predominantly terrestrial swarm simulators, like any simulated environment, are imperfect. They must sacrifice some level of accuracy in modeling the physical world to quickly compute results. For controller development this tradeoff in accuracy for speed is acceptable; beneficial even. It allows researchers to iterate on their ideas more frequently and more easily. They are not, however, an accurate reflection of the real world and cannot be treated as such.

Simulation of marine robots is particularly difficult. The marine environment is dynamic, complex, and hostile to robots. Selection of marine robotics simulators is sparse [29]: few among those available are capable of handling multiple robots and those that can are imperfect [83, 69]. Computational fluid dynamics simulations remain an unsolved problem, mostly relying on numerical methods which are very computationally expensive [86]. These tools have proven to be invaluable for ship analysis and design but are not substitutes for hardware testing [61]. This is doubly true for marine robotics where inaccuracies in the environment and vehicle models may compound to generate even larger errors. All simulator results must eventually be validated on a hardware platform to be trusted.

2.1 Overview of Swarm Robotics Platforms

2.1.1 Terrestrial Swarm Platforms

Ground robots are by far the most popular domain among swarm robotics researchers. The planar dynamics of terrestrial robots makes them easier to control than other kinds of robots such as aerial or marine vehicles and are thus easier to study. As a result, there exists a modest selection of general purpose ground swarm robotics research platforms built by different laboratories around the world [64, 48, 39, 60].

Such platforms are almost universally small robots designed to operate in a controlled laboratory environment. Their size makes them relatively inexpensive to build which is a virtual necessity when assembling a swarm with dozens of individuals.

Terrestrial swarm robotics platforms can be subcategorized into two groups: those that use wheels for locomotion [17, 16] and those that use vibrations [5]. Wheeled robots typically operate using a differential drive configuration which allows them to pivot in place and easily achieve any desired pose in a 3D planar state space. Robots using vibration for locomotion are holonomic meaning they can also maneuver anywhere in a planar environment. Robots using vibrations to move require smaller actuators than those of a wheeled robot. They are typically much cheaper and much slower than their wheeled counterparts.

The low cost of vibration-based locomotive platforms is a large contributing factor in the success of the Kilobot [77]: the single most popular hardware platform among swarm robotics researchers. It has been used in many experiments and ongoing projects as the platform of choice due to its simple design and cheap materials costing an order of magnitude less than similar existing platforms at the time of its creation [74]. Hundreds of these $14 robots can be assembled without incurring egregious cost and they are general purpose enough to be useful in a wide variety of swarm experiments such as formation control and swarm synchronization.

For some applications a general purpose solution like the Kilobot is not suitable. Problems that require environmental manipulation, construction and stigmergy for instance, demand larger, more expensive robots equipped with grippers and more powerful actuators [94, 15]. Such platforms are typically custom built to meet the demands of the application they target. Although more expensive than the small general purpose alternatives, the larger swarm robots are still often used to test and validate base-level swarm robotics concepts [26].

2.1.2 Aerial Swarm Platforms

An Unmanned Aerial Vehicle (UAV) has very complex and fast system dynamics. This makes them difficult to control as even the simplest action for a terrestrial robot, staying in place, requires active control for a UAV to achieve [27]. The topic of swarm UAV research has received a steady growth in interest in spite of, or perhaps due to, its inherent challenges. This is largely because of the improving quality and decreasing cost of UAV hardware components allowing construction of many cheaper UAVs, and growing interest in using aerial robots for surveillance and military applications [28].

There are two types of aerial swarm robotics platforms: multirotor and fixed-wing aircraft [27]. Quadrotors are the most popular among these due to their minimal complexity, maneuverability, and ability to achieve stable flight and hover [44]. Swarms of quadrotors have been demonstrated to function both in laboratory environments with access to motion capture systems [72] and outdoors using systems such as GPS inertial navigation, Ultra-wideband positioning, and Visual-Inertial Odometry for localization [92, 93]. UAVs can also operate in heterogeneous swarms, in tandem with terrestrial or marine robots [59, 90, 71].

2.1.3 Marine Swarm Platforms

The marine category of swarm robotics platforms includes both underwater [24, 10] and surface marine robots [81, 30]. An underwater robot is commonly known as an

Autonomous Underwater Vehicle (AUV) while a surface robot is called an Unmanned Surface Vehicle (USV). Similar to the dichotomy in control complexity between terrestrial and aerial robots, USV control is two dimensional and thus much simpler than the three dimensional control of AUVs. This makes USVs a more appealing platform for most general purpose research on swarm robotics applications in marine environments. The discussion of marine swarm robotics platforms in this thesis will be limited to USV platforms.

Dedicated marine swarm robotics platforms are very rare. Those that exist are typically custom built and outfitted specifically for their target application such as environmental monitoring [57]. It is far more common for an existing, general purpose USV platform with onboard autonomy and communication capabilities to be repurposed for swarm applications [32, 71]. Laboratory marine robotics platforms are even rarer. The only known laboratory-based USV platform for swarm robotics research is the mCoSTe system developed by Larkin et al. [53]. It has been tailored for use in the flow tracking experiments it was initially conceived to test but is general enough to be repurposed for testing other concepts such as the swarm heterogeneity experiment described in [73]. This platform is not available for purchase commercially, nor is it open source. Although the individual vessels are inexpensive, the full system relies on an expensive external pose tracking system for localization making it a costly platform overall. These factors make it an unsuitable platform for the Bio-Inspired Robotics Lab (BOTS) marine swarm robotics project. The discussion of marine swarm robotics platforms must, therefore, be expanded to include a discussion of existing general USV platforms as potential candidates.

2.1.3.1 Overview of Unmanned Surface Vehicles

A USV, sometimes known as an Autonomous Surface Vehicle (ASV), is an autonomous boat capable of augmenting or even replacing the use of manned vessels in dirty, dangerous, and/or monotonous tasks. These platforms are commonly used for oceanographic research and military applications such as surveillance and environmental monitoring [56, 58]. Designed to survive in harsh open-water environments, these vehicles are typically large and expensive.

The majority of industrial USVs are custom designed and built for their target application and are not generally available for purchase commercially. Those that are available on the market can range in size from 1.3 to 8 meters in length making them unsuitable for indoor operations. These commercial vessels can cost between $30,000 and several million USD per vessel depending on their sensor payload [6, 2, 3]. This makes the cost of purchasing a fleet of commercially available USVs prohibitive for many laboratories and researchers. The capabilities and engineering support that accompany commercial USV platforms make them an appealing choice for deployment as a final solution or industrial product but their cost makes them unsuitable for swarm robotics research and development purposes.

Open source USV platforms offer an alternative to purchasing commercial options. Open source solutions are generally much less expensive and can be built and modified by an individual user or researcher to suit their specific needs. They, like commercial USVs, are generally developed for operation on open bodies of water and so must be large enough to achieve stability under the influence of wind and waves. Most are too large to operate indoors and none are equipped with the sensing equipment needed to do so. Such platforms are slowly being introduced to the research community such as the Jetyak [49], ARCAB [25], and SMARTBoat 3 [46]. These three platforms showcase the three predominant approaches to USV development for research.

The Jetyak is a fully functional USV capable of carrying the same instrumentation and navigation equipment as large commercial USVs. It is built by outfitting a traditional marine vessel, in this case a kayak, with the equipment needed for onboard automation and locomotion. Its $15,000 USD fabrication cost, while cheaper than commercial USV systems, make it too expensive to produce in large quantities for multi-robot systems research.

The Arctic Research Centre Autonomous Boat (ARCAB) demonstrates the second common approach: developing a traditional marine vessel from the ground up. This type of platform often reflects the design approach seen in industry but with a smaller size and cheaper components. In the case of the ARCAB system, the 900×700 mm catamaran design is a stable mobile sensor platform large enough to accommodate the needs of the researchers who built it but not much more. It does, however, have the same cost issue as the Jetyak: at 3000 euro per vessel the cost of acquiring multiple vessels is prohibitive.

The SMARTBoat 3 is a recently published system with a much lower reported cost of $200 per vessel. It achieves this by utilizing the third design approach: building a custom vessel targeted at a narrow range of applications. The SMARTBoat 3 is intended as a cheap autonomous environmental monitoring platform and only features relevant to that task need be included. This allowed the team developing this vessel to eschew conventional design practices like utilizing traditional hull shapes and materials as the vessel's speed and efficiency are non-essential. The SMART-

Boat 3 design uses repurposed off-the-shelf components for floatation which heavily constrains the vessel's shape. As such, the vessel's toroidal hull-form does not reflect the hydrodynamics of larger USV's, the eventual target platform for any algorithm researchers would develop while using it. This makes it an unsuitable testing platform for general marine swarm robotics research.

The final category of USV under consideration are those developed for, and by, amateur roboticists and Remote Control (RC) boat hobbyists [14, 1, 19]. These platforms are normally constructed on a budget in the hundreds, not thousands, of dollars. As such, hobbyist USVs are typically small, in the 300 mm to 1 m range, and inexpensive. RC hobbyists typically have different objectives than most researchers; they often seek to optimize a particular aspect of their vessels such as speed and maneuverability or else they are seeking to produce scale models of larger vessels as accurately as possible. The shortcoming of such platforms for research purposes is their incomplete set of features: very few hobbyist products are equipped for localization or onboard autonomy. Nonetheless these platforms can provide an excellent source of inspiration for the development of small scale USV research platforms and the market for RC boat components represents an inexpensive means to acquire small marine-grade parts that might otherwise need to be custom made.

The advantages and disadvantages of the available USV platforms cited above are summarized in Table 2.1.

Name	**Category**	**Pros**	**Cons**
Heron USV [6]	Commercial	Large, modular, stable	Expensive, outdoor use only
C-Worker 8 [2]	Commercial	Large, long range	Expensive, outdoor use only
CAT-Surveyor [3]	Commercial	Large, stable	Expensive, outdoor use only
Jetyak [49]	Open-Source	Uses commercial hardware, long range, publicly available	Expensive, outdoor use only
ARCAB [25]	Open-Source	Stable, publicly available	Expensive, outdoor use only, difficult fabrication
SMARTBoat 3 [46]	Open-Source	Inexpensive, publicly available	Outdoor use only, abnormal dynamics
n3m0 [14]	Hobbyist	Inexpensive, simple, publicly available	Slow fabrication, unstable
Unnamed Boat Autopilot Project [1]	Hobbyist	Inexpensive, simple	Difficult to customize for research applications
Lake Maps NL USV [19]	Hobbyist	Inexpensive, stable	Difficult fabrication, not publicly available, outdoor use only
aPad [57]	Research Platform	Demonstrated success in research	Application specific, outdoor use only, abnormal dynamics
mCoSTe [53]	Research Platform	Indoor use, inexpensive	Not publicly available, expensive localization system

Table 2.1: Pros and Cons of Available Marine Swarm Robotics Research Platform Candidates

2.2 Problem Statement

Based on the review of existing USV platforms, none found were suitable for use in the BOTS marine swarm robotics project. They are predominantly large, expensive, and designed for outdoor operation. Of the platforms discussed the most appropriate would be the mCoSTe system [53] which, as discussed, is not available for purchase, not open source, and relies on an expensive external pose tracking system. There is a gap in the capabilities of available USV platforms. A marine equivalent to the Kilobots system is needed: an inexpensive platform which operates indoors that can be utilized to validate concepts and simulation results through hardware experiments. The design, fabrication, and testing of such a platform was the primary goal of the work discussed in this thesis.

Problem statement: *Develop a low-cost, general purpose, open source USV platform for marine swarm robotics research which can operate indoors in a laboratory environment.*

2.2.1 Design Requirements and Constraints

The proposed platform must be a functional USV. As such, certain properties are expected. It must:

1. *Float and be hydrostatically stable in water.*

2. *Endure continous partial immersion in water.*

3. *Be capable of self-powered, untethered locomotion and maneuvering.*

4. *Be capable of autonomous operation.*

Self-powered locomotion and autonomous operation demand the inclusion of an onboard power system and actuators as well as a means of decision-making, typically in the form of an onboard computer or microcontroller(s). It must also include a means of sensing its local environment to inform the decision process.

Additionally, the aim of the proposed platform was to serve as a testbed for, and eventual stepping stone towards, full-scale USV swarms. It should, therefore:

5. *Reflect the dynamics of equivalent larger existing USVs.*

This means that the vessel design is limited to using traditional hull forms and control interfaces seen in commercial and research USV platforms. These include mono-hull, catamaran, or trimaran vessel designs with a propeller-and-rudder configuration or differential drive propellers.

The proposed platform is intended as a general purpose platform for swarm robotics research. As such, it should include features commonly utilized in swarm robotics experiments. These include:

6. *Inter-vessel communication capabilities.*

7. *Customizable or modular tooling options.*

The ability for a swarm of of robots to communicate amongst themselves is a very common requirement among modern swarm robotics algorithms. A platform meant to validate such algorithms must include this capability. It should also be capable of supporting tools or equipment customized for a given task such as object aggregation or dispersion, collective transport of objects, flocking behaviors, and collective mapping tasks. As a general purpose platform, no single set of tools would be acceptable for all applications so the proposed platform must be modifiable to suit a researcher's current needs. A modular system for tool mounting would be a suitable alternative to a fixed tooling payload. This requirement also ties into the vessel's stability requirements: The platform must be sufficiently stable to accommodate any such custom tool or device within reason.

The proposed platform is intended for indoor operations in a controlled laboratory environment. This expectation simplifies or eliminates many of the most challenging aspects of USV design such as seakeeping requirements, communication range, battery life, corrosion resistance, and waterproofing. The members of BOTS have access to Memorial University's Deep Tank facility: a 3.65×3.65 m indoor tank of water with an overhead gantry for testing purposes. The proposed platform must be capable of maneuvering in an environment as confined as this tank or more. To provide sufficient space to maneuver in an environment of fixed size, the vessel's size must be constrained. It must:

8. *Have a length overall of less than* $\frac{1}{10}$ *the shortest side of its test tank.*

The platform must be low-cost. This means the total cost to build and run the system should be minimized. This constraint influences the design of the whole system from the selection of components and materials, the techniques used to fabricate it, and the choice of peripheral equipment surrounding the vessels themselves during operation. The vessel's cost also benefits from the size constraint, requiring less materials and less powerful actuators.

Finally, the platform must be open source. This comes with the expectation that other researchers may seek to replicate and use the platform for their own work. It should, therefore, be as simple to replicate as possible. The vessel design should:

9. *Use readily available off-the-shelf components wherever possible.*

10. *Use only simple fabrication techniques.*

The use of off-the-shelf, or Original Equipment Manufacturer (OEM), components is beneficial to both the platform's cost and replicability. Such parts are generally cheaper and easier to acquire than custom parts. The restriction to use of only simple fabrication methods is an acknowledgement that swarm robotics researchers come from diverse backgrounds and with distinct sets of knowledge. Not all are experts in machining and fabrication. The platform should eschew techniques that are excessively complex or require the use of expensive machinery. This should simplify the construction and maintenance process for those researchers who are less comfortable dealing with custom hardware, making the platform more accessible to the research community as a whole.

The platform discussed in this thesis was designed to meet each of these requirements, summarized in Table 2.2. They informed all design decisions discussed in subsequent chapters.

Item No.	**Importance**	**Requirement**
1	Must	Float and be hydrostatically stable in water.
2	Must	Endure continuous partial immersion in water.
3	Must	Be capable of self-powered untethered locomotion and maneuvering.
4	Must	Be capable of autonomous operation.
5	Should	Reflect the dynamics of equivalent larger existing USVs.
6	Must	Include inter-vessel communication capabilities.
7	Must	Include customizable or modular tooling options.
8	Must	Have length overall of less than $\frac{1}{10}$ the shortest side of its test tank.
9	Should	Make use of readily available off-the-shelf components.
10	Should	Require only simple fabrication techniques.

Table 2.2: Marine Swarm Robotics Research Platform Design Requirements

Chapter 3

Hardware Design

The microUSV, shown in Figure 3.1, is a proposed open source small Unmanned Surface Vehicle (USV) platform designed for swarm robotics experiments in laboratory environments [41]. It was designed to be inexpensive to produce, capable of onboard autonomous control, and easily reprogrammable. The design leverages 3D printing and off-the-shelf hobbyist electronics to produce a marine robotics research platform that costs just over $500 CAD for a single unit and is much cheaper when produced in multiples. It measures 23 cm in length and so is capable of maneuvering in the 3.65×3.65 m enclosed indoor water tank available for testing. Model files and source code for the microUSV are available through an open access license on the project's public online repository [40] which also includes detailed vessel fabrication and assembly instructions.

This chapter will discuss the design of the microUSV's hardware subsystems.

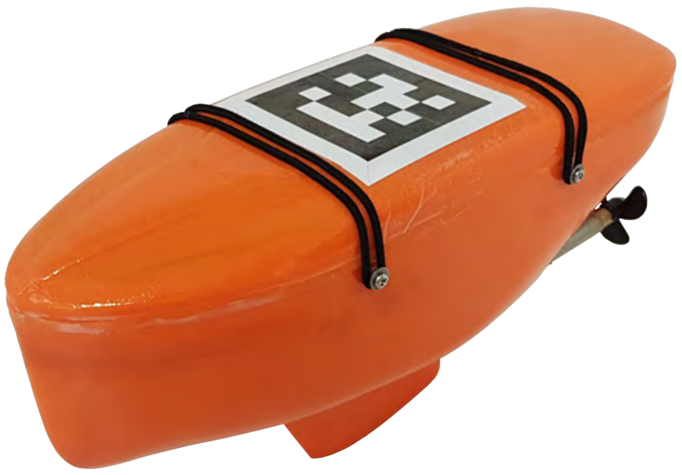


Figure 3.1: microUSV

3.1 Mechanical Subsystems

The mechanical design process behind the microUSV followed a simplified version of the Ship Design Spiral visualized by Evans [34]. This iterative design process is very popular in ship design due to the intrinsically interrelated nature of a vessel's design parameters. The microUSV design is far simpler than a traditional manned vessel but alternately reworking and improving the vessel's hull design, propulsion systems, and general arrangement yielded a better vessel design with each iteration. Figure 3.2 shows some of the hull prototypes produced during the design process and illustrates the progression in hull design over the course of the project. The design discussed in this chapter is the final result of this iterative process.

3.1.1 Hull

The microUSV's indoor operating environment made a small vessel design essential. The vessel has a length overall of 230 mm, a beam of 89.2 mm, and a depth of

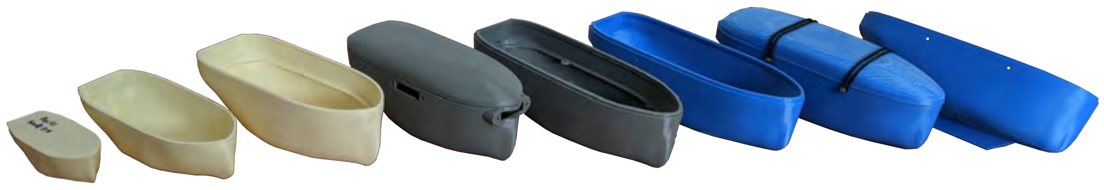


Figure 3.2: Sequential Hull Design Iteration Prototypes

121.5 mm as seen in Figure 3.3. A laboratory tank is a tightly confined operating environment when compared to a lake or ocean, with a surface area measured in m^2 rather than km^2. A small vessel is better suited to operating in such a small body of water than a full sized USV designed for the open water. A full sized USV with a length overall of 1.3m, such as the vessel described in [6], or greater would be nearly incapable of maneuvering in a 3.65×3.65 m test tank. One or more scaled-down vessels are fully capable of maneuvering in such a confined environment and, by extension, are better at demonstrating the behaviors being investigated by the operator(s).

A small vessel design is also beneficial for its reduced material costs: A smaller vessel requires smaller parts which influences not only the volume of materials required but also their selection. Using smaller parts reduces the vessel's weight which results in lower loads on the hull so structural components can be made smaller and lighter. The microUSV's hull design eschews more traditional structural materials like steel and aluminum for parts made of Polylactic Acid (PLA) plastic produced using an Ultimaker 2+ 3D printer. The printer's print volume constrained the size of custom components that could be included in the design to $223 \times 223 \times 205$ mm maximum. Multiple plastic hull designs were prototyped, and the material was deemed strong

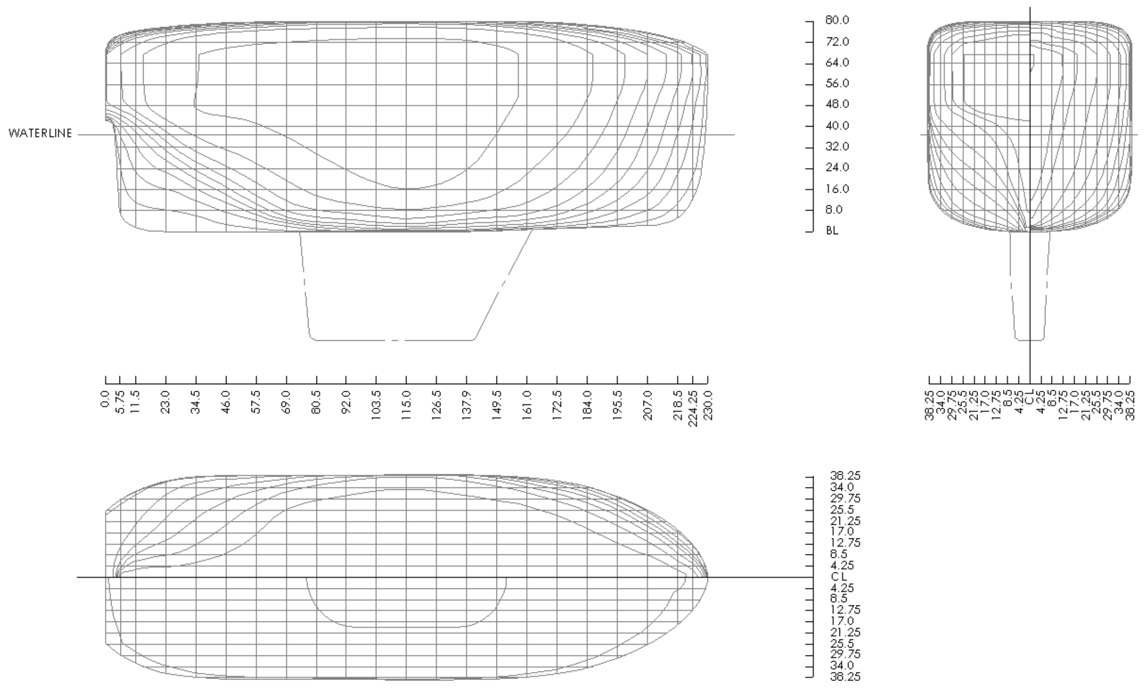


Figure 3.3: microUSV Hull Lines Plan (Dimensions in mm)

enough to support the vessel's small operating loads.

The vessel's hull form was designed using the lines plan of a tugboat [7] for reference. A tugboat hull was chosen for its stability and compact design. Good stability is of particular importance to the microUSV due to the use of an external pose detector system which will be discussed in Chapter 4. The compactness of this hull type is beneficial due to the increased usable internal cavity volume relative to the vessel's size. The vessel's low length-to-beam ratio is important to allow sufficient width of the internal volume for onboard electronics to be mounted without greatly increasing the vessel's length overall. The microUSV's hull was further compressed lengthwise relative to the reference lines plan to decrease this length-to-beam ratio and allow the hull to be fabricated as a single component on the 3D printer's limited build platform space.

3.1.1.1 Stability

The importance of mounting onboard electronics also influenced the decision to use a monohulled design over the catamaran design often seen in other small USVs [6, 3]: Mounting heavy components like batteries at or below the waterline improves the vessel's stability without increasing the beam length which results in a vessel with a smaller footprint, better able to navigate confined spaces. Due to the microUSV's small size, a catamaran design would require the twin hulls to be scaled up to provide enough internal space for component mounting. They would quickly exceed the volume necessary to keep the vessel afloat and further increase the vessel's footprint. Since the microUSV operates exclusively indoors it does not need to survive wind and large waves, so the catamaran's tradeoff in size for stability was deemed unnecessary. The microUSV's monohulled design utilizes other means to achieve the necessary stability.

A fin keel was added to improve vessel stability without resorting to widening the hull. A keel's surface area catches water as the vessel rolls, inducing a drag force to resist the motion. The keel form chosen was a NACA series 0018 hydrofoil tapered along its depth [45]. The symmetrical members of the NACA series of 4-digit foils are popular for use as ship rudders, keels, daggerboards, and the like [67]. Foils used for these purposes typically fall within the 0010 to 0035 range with keels being at the lower end of that range, 0010 to 0015, to minimize drag as the additional lift generated by a wider foil is not as critical as it would be for a rudder [79]. The microUSV keel's NACA profile at 0018 falls just outside the typical range. A wider profile was used to accommodate the inclusion of a steel ballast plate inside of it.

The steel ballast plate in the keel lowers the vessel's center of gravity, thus increasing the Metacentric Height (GM). Molland [62] defines GM as "The vertical separation of the metacenter and the center of gravity as projected on to a transverse plane" where the metacenter is "The intersection of successive vertical lines through the center of buoyancy as a ship is heeled progressively", heel angle referring to the degree of roll a ship is experiencing. The metacenter is the effective pivot point for a ship experiencing roll and its height dictates the stability of a vessel.

Increasing a vessel's GM increases its stability due to the resultant increase in the vessel's Righting Lever Length (GZ). GZ is the horizontal distance between a vessel's center of gravity and the line of action of its buoyancy force. When offset due to roll, the gravity and buoyancy forces generate a couple moment which rights the vessel. A longer righting lever increases the righting moment experienced by a vessel as it rolls, returning it to an upright position faster. In the case of the microUSV, the vessel has an estimated natural roll period of 1.17 seconds calculated using the Weiss formula shown in Equation 3.1 where k_{xx} refers to a vessel's radius of gyration and g is the acceleration due to gravity [63]. The microUSV's GZ curve, calculated using an iterative righting lever simulation with respect to vessel heel angle [78] and shown in Figure 3.4, shows no angle of vanishing stability, defined as the x-intercept on a vessel's GZ curve, meaning it can theoretically right itself from any degree of roll.

$$T_\phi = \frac{2\pi k_{xx}}{\sqrt{gGM}} \tag{3.1}$$

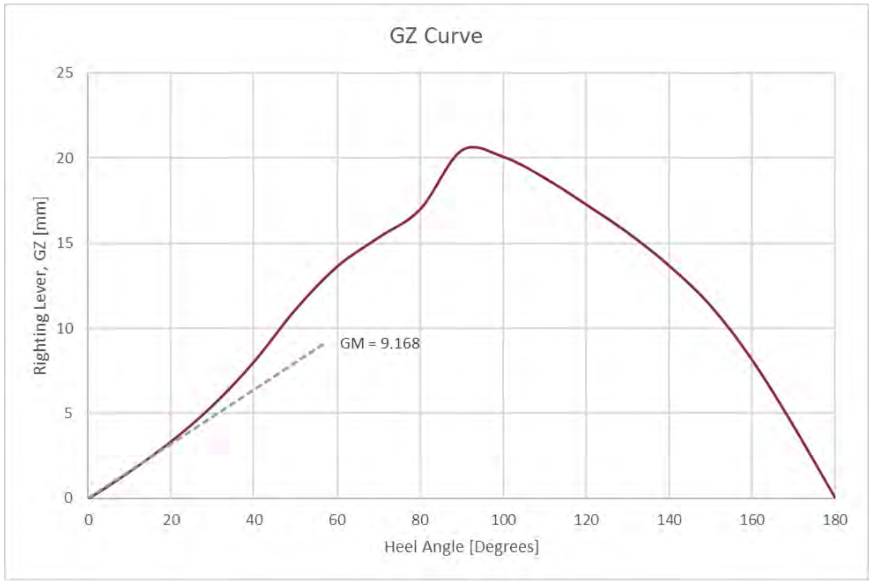


Figure 3.4: Righting Lever Length vs Heel (Roll) Angle Plot

3.1.1.2 Waterproofing

The hull design includes a lid to keep water out of the internal cavity in the case of excessive roll or splashes. This lid is mounted to the hull using a double lap joint around the hull's entire upper edge, shown in Detail B of Figure 3.5, and is kept in place by a pair of hair elastics. The design does not include a gasket or O-ring and so is splash proof but not completely watertight. In order to achieve a fully waterproof design a gasket flange or O-ring groove would need to be added with many tightly spaced bolts along the joint between hull and lid to squeeze the sealing elastomer component.

Assuming a number 2 machine screw size, the maximum bolt spacing suggested in Shigley's [23] would demand a minimum of 13 bolts, with accompanying nuts and washers, to seal the lid of the microUSV. These fasteners would contribute an extra 20 to 30 grams of metal to the vessel's weight with an extra estimated 10 to 12 grams

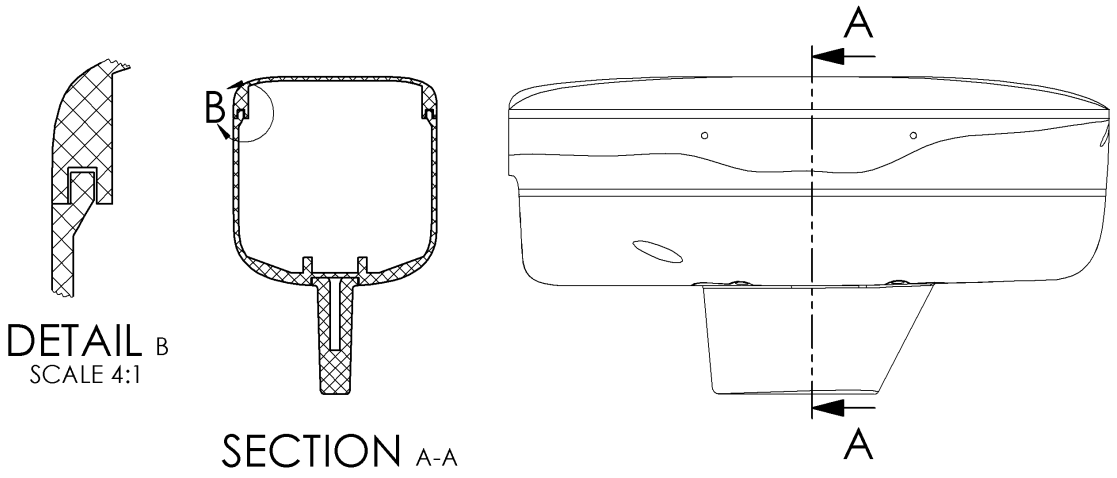


Figure 3.5: Lid Seal - Double Lap Joint

worth of 3D printed plastic to form a flange on which to mount the bolts and sealing elastomer. These components would add significant weight to the vessel, almost 10% of the vessel's total weight, and require increased buoyancy, and therefore hull size, to support them. These components would all be situated near the top of the vessel and would shift the center of gravity upward, reducing its GM and stability. The microUSV was designed to operate on the water's surface and should never need to endure being submerged under water, so the costs of a fully waterproof design outweighed its benefits.

The vessel has, however, been successfully tested for very brief periods of immersion: it can survive for one second fully submerged in water without any water reaching the internal cavity and can likely endure longer immersions, but this has not been tested. It is easily capable of surviving the splash from a wave or an accidental immersion during retrieval.

The PLA plastic used for the hull is naturally waterproof, but it is also a biodegradable material: Its integrity may gradually deteriorate due to exposure to UV light or temperature changes over time. Additionally, the layered building process used by Fused Deposition Modeling (FDM) 3D printers is imperfect. There is a chance of introducing small, often unnoticeable, defects and gaps in the walls of a part during the printing process. Although a 3D printed hull made with an accurate printer can easily survive a handful of immersions in water, it is unlikely to endure months or years of repeated use without leaking which would likely destroy the electronics housed inside. Therefore, an additional waterproof coating of epoxy was added to the hull's external surfaces.

3.1.2 Propulsion System

The microUSV's propulsion system consists of a pair of propellers driven directly by DC motors, the components for which can be seen in Figure 3.6. Each motor connects to its drive shaft via a universal joint shaft coupler to mitigate small shaft misalignments. The drive shafts are arranged in parallel with a spacing of 48 mm and independently controlled motors to allow differential drive. A differential drive system was chosen over a more traditional propeller and rudder arrangement to improve the vessel's maneuverability: making the microUSV better able to navigate its confined operating environment.

The vessel's drive shafts are kept watertight with a pair of custom stuffing tubes. A stuffing tube is a simple assembly packed with grease which forms a seal around a rotating shaft. The microUSV's stuffing tubes are made of stainless steel with a bushing at each end to support the shaft as it rotates and the void space between

ITEM NO.	DESCRIPTION	QTY.
1	Drive Shaft	1
2	Stuffing Tube	1
3	#5 Washer UHMW	2
4	Drive Dog	1
5	#5-40 Locknut	1
6	U-Joint Shaft Coupler	1
7	Propeller	1
8	Pololu Gearmotor	1

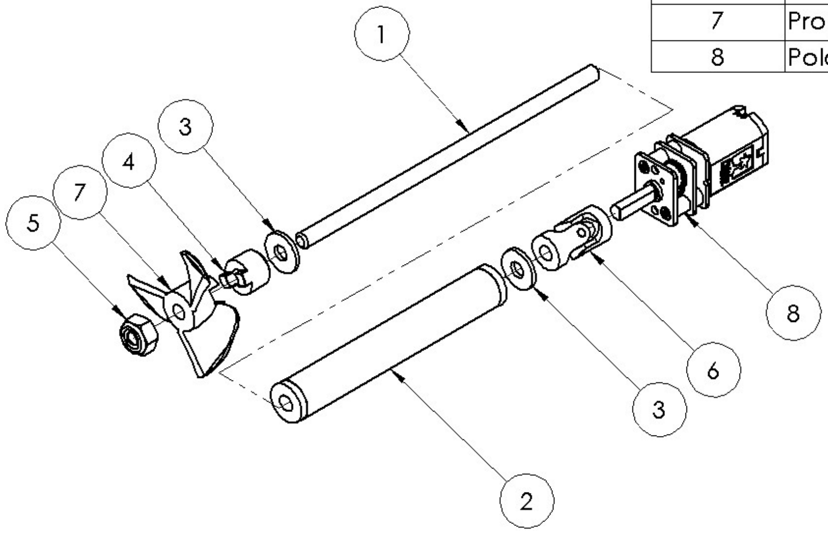


Figure 3.6: Drive Train Assembly Exploded View Drawing

shaft and tube is filled with petroleum jelly. These tubes are mounted to the hull and themselves sealed in place using a marine-grade silicone sealant. The stuffing tubes needed to be custom made as all existing off-the-shelf stuffing tube and drive shaft systems were too long to fit the vessel's hull.

The propulsion system consists of mostly off-the-shelf components to reduce costs: the propellers, drive dogs, and shaft couplers were all sourced from RC boat part suppliers. The drive shaft, motors, and components for the stuffing tubes were also purchased off-the-shelf but required further modification.

3.1.3 General Arrangement

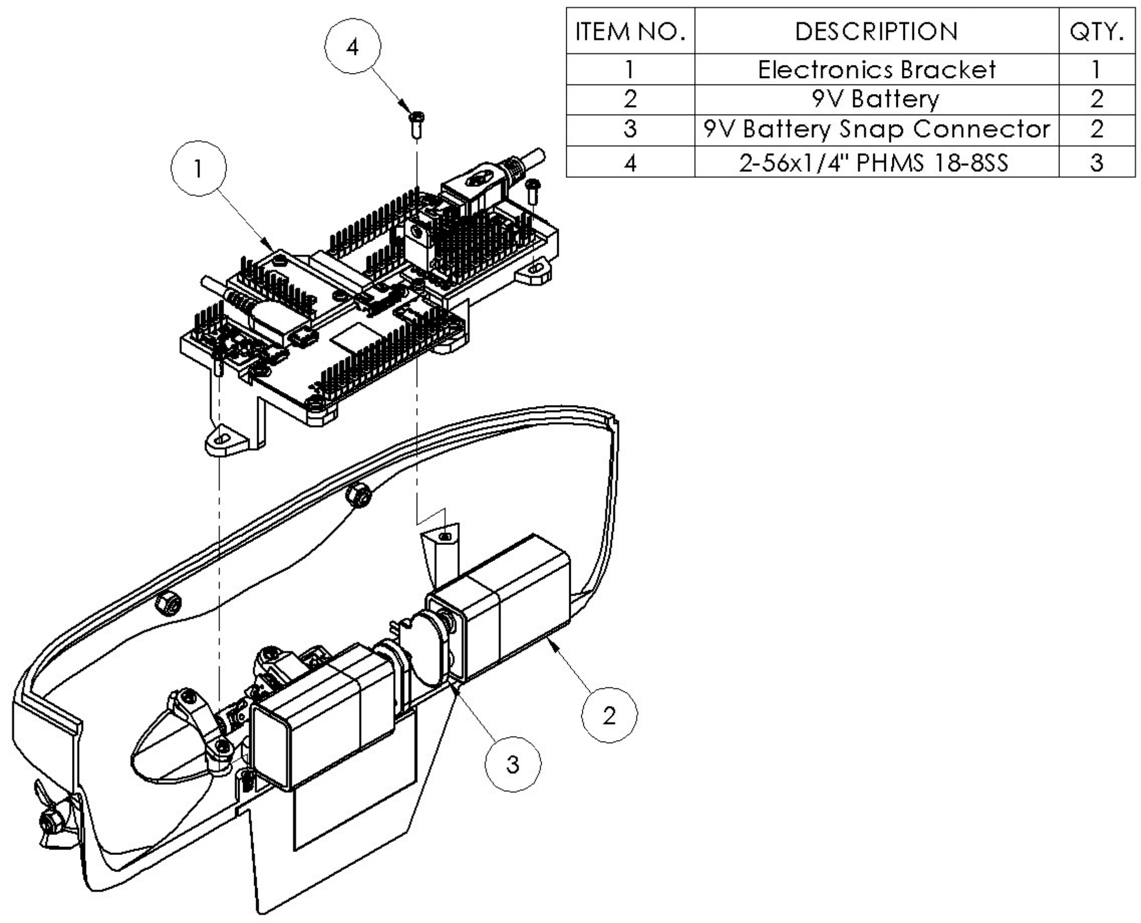
The location of the microUSV's center of gravity is an important factor contributing to the vessel's stability. It is a key variable used in calculating a vessel's GM [62]. To maximize GM, the center of gravity should be positioned as low as possible on the vessel. The vessel's center of gravity must also be aligned with its center of buoyancy in order to avoid statically pitching or rolling to one side which would increase the ship's wetted area, negatively impacting performance. If the centers of gravity and buoyancy were misaligned, with the center of gravity too far forward for instance, the vessel would pitch forward, submerging more of the forward section of its hull. This shifts the vessel's center of buoyancy forward until the buoyancy and gravity force vectors align and their couple moment is removed. The new equilibrium point is statically pitched forward.

A vessel's center of buoyancy is a function of its external geometry only which is both difficult to change and influences countless other factors from vessel footprint to hydrodynamic characteristics. For the sake of aligning the centers of gravity and buoyancy it is easier to consider the center of buoyancy fixed and to move the internal components in order to shift the vessel's center of gravity into alignment instead. This process is called creating a general arrangement.

The general arrangement of a vessel is the layout of internal volumes and components. In the case of the microUSV there is only one internal volume so this task is limited to the arrangement of internal components. The objectives of this process were to align the vessel's centers of gravity and buoyancy while maintaining easy access to all components critical for operation and maintenance such as the power switch and batteries. The general arrangement must also accommodate any special requirements unique to certain components such as the motors being mounted in line with the drive shafts or the wireless antenna being mounted as high as possible inside the vessel. These factors governed the design of the microUSV's onboard electronics bracket which holds the majority of its internal components and so has the largest impact on the location of the vessel's center of gravity.

3.1.3.1 Onboard Electronics Bracket

The electrical components onboard the microUSV are all mounted to a 3D printed bracket except for the batteries and motors which are mounted directly to the hull. The motors are mounted independently to ensure alignment with the drive shafts and the batteries are mounted below the electronics bracket in order to keep the vessel's center of gravity as low as possible to improve stability. This electronics bracket is mounted on three threaded posts inside the hull as far forward as possible. These screws are easily accessible and can be easily removed to change batteries or bench test and debug the electrical system. The electronics bracket mounting process is illustrated in Figure 3.7.



ITEM NO.	DESCRIPTION	QTY.
1	Electronics Bracket	1
2	9V Battery	2
3	9V Battery Snap Connector	2
4	2-56x1/4" PHMS 18-8SS	3

Figure 3.7: Electronics Bracket Assembly Drawing

The void space near the aft end of the vessel created by this bracket design is intentional. The center of buoyancy is nearly centered along the vessel's length but the drive shafts and stuffing tubes amount to a significant weight at the aft end of the vessel which cannot be moved. The forward position of the electronics bracket and batteries is to counteract this imbalance, shifting the center of gravity forward into alignment.

3.1.3.2 Component Mounting

All internal components onboard the microUSV are mounted using screws and 3D printed brackets. All screws onboard the vessel are the same size, phillips head number 2-56 machine screws, to minimize the number of tools required for assembly. With the exception of four through-holes above the waterline used to connect the lid to the hull, none of the mounting points on the vessel can use a traditional stack of screws, nuts, and washers. The design instead utilizes heat-set threaded inserts. These are bonded into plastic components using heat from a soldering iron and allow screws to be attached inside the hull without introducing holes through it which must be sealed and would represent additional points of failure.

The placement of these mounting brackets, screws, and heat-set inserts is influenced not only by the locations of the components they are used to mount but also the assembly procedure used to mount them. The heat-set inserts must be installed using a hot soldering iron which would damage any part of the 3D printed hull it contacted. The axes of the heat-set inserts must, therefore, provide sufficient clearance to avoid interference between the soldering iron and the hull during the installation process. Two methods were used to achieve this. The first is used for the three vertical connection points used to mount the electronics bracket: simply placing the mounting holes far enough away from the hull walls to provide the necessary clearance. The second is used for mounting the stuffing tubes and motors: the insert holes are angled away from the hull walls, 20 degrees offset from vertical, to allow the soldering iron to access a point that is directly below an overhang without interference.

3.1.4 Modular Tool Mounting

The lid of the microUSV is held in place by a pair of hair elastics. These elastics attach to four screws on the outside of the hull near the joint between hull and lid which serve as tie-down posts. These screws serve a second purpose as well: they offer a modular mounting point for experiment-specific tools. 3D printed brackets can be attached to these screws and serve as a base for any custom tool or sensor without needing to redesign and rebuild the hull. Figure 3.8 shows the microUSV outfitted with a pair of side-mounted nets. This configuration was used in the contaminant clustering experiment discussed in Chapter 6. Examples of other potential tool configurations include side-mounted hooks for capturing other vessels or tethering to them, drag or trawling nets to simulate the behaviors of fishing fleets, or an onboard camera to enable a multitude of detection and recognition tasks.

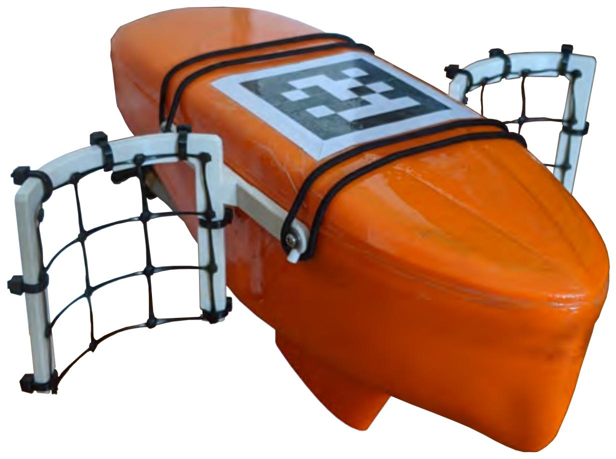


Figure 3.8: microUSV equipped with nets for contaminant clustering experiments

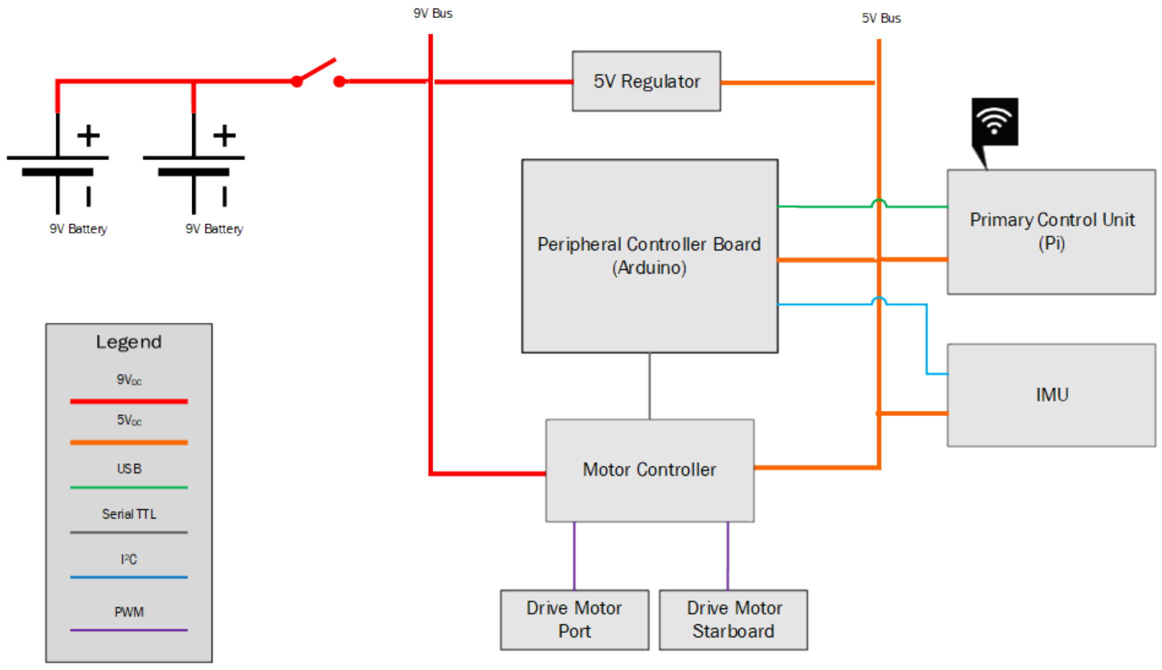


Figure 3.9: System Integration Diagram

3.2 Electrical Subsystems

The microUSV's onboard electronics system is designed to utilize readily available hobbyist components. This minimizes cost and allows damaged or defective components to be replaced quickly. It also grants access to the large user communities supporting each of these products, potentially simplifying the debugging process for a new user. A system integration diagram is provided in Figure 3.9.

3.2.1 Electronic Devices

To achieve onboard autonomous control and reprogrammability, a single-board computer was selected as the primary control unit: A Raspberry Pi Zero W. The Pi Zero is the smallest of the popular family of pocket-sized computers and is well suited to small robotics projects. The Pi Zero W features a built-in wireless module used for communication between each vessel and the computer running the CVSensorSimulator server software, the specifics of which are discussed in Chapter 4. It runs the Raspbian operating system which grants it the same functionality as most desktop Linux systems. This allows the microUSV's onboard control software to be written in any number of popular high-level programming languages and easily changed during testing.

The Raspberry Pi communicates with an Arduino Nano, which serves as a peripheral control device. The Arduino is included to simplify the interface between the vessel's primary controller and peripheral devices; a motor controller and Inertial Measurement Unit (IMU). It also offers room for expansion with eight unused digital pins and six unused analog pins available for additional devices or sensors.

The peripheral devices, a Qik 2s9v1 dual serial motor controller and MinIMU-9 v5, were both sourced from Pololu Robotics and Electronics. These are each very small devices with easily accessible drivers and support communities which fulfill their designed roles in the microUSV system: The IMU is included to augment any custom odometry system a user may deploy while the motor controller drives the vessel's motors. The current localization system implementation does not utilize the IMU data as the computer vision based sensor simulation software discussed in Chapter 4 proved sufficient for real-time localization without it. The sensor is included in anticipation of the need of future users to develop and test more complex localization systems without needing to modify the vessel's hardware. All onboard electronic devices are shown in Figure 3.10.

The motors selected for the microUSV were also sourced from Pololu. They are

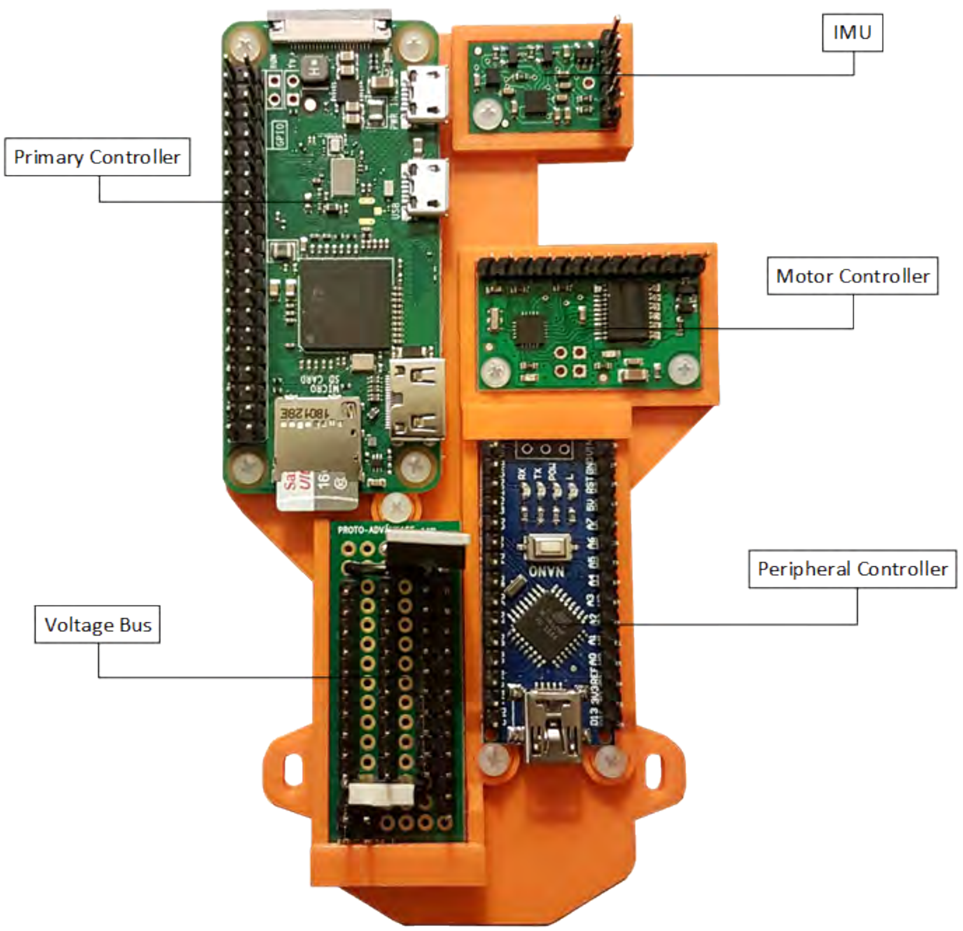


Figure 3.10: Electronics Bracket with Labelled Components

12 V DC motors with a 5:1 gearbox. Since the torque requirements for the chosen 28mm diameter propeller were so small, the priorities when selecting a motor were its small size and power consumption. The motor's voltage rating was also a primary consideration to interface with the microUSV's power system.

3.2.1.1 Onboard Sensors

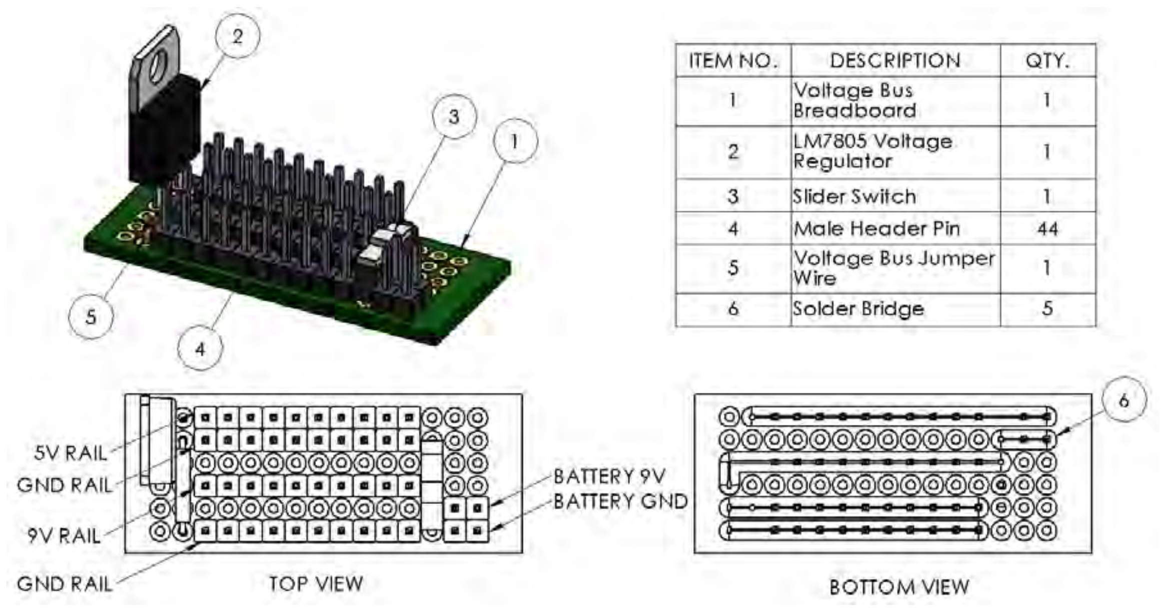
The sensors usable on the microUSV are limited due to the vessel's size and indoor operating environment. Traditional navigation sensors for full scale USVs such as a Global Positioning System (GPS) and magnetometer cannot function in indoor environments and powerful rangefinder systems such as LIDAR can weigh as much as the rest of the microUSV's components combined and would need to be mounted high on the vessel to function properly. Mounting one of these systems onboard would quickly sink or flip the vessel. Those sensors that are small and light enough to fit the microUSV cannot produce data of sufficient quality to be reliable. Unable to use traditional odometry sensors, the design instead uses an external pose detector system discussed in Chapter 4 based on computer vision. The only physical sensor included in the microUSV design is an IMU which are available in small sizes and low-costs with good performance. Similar to the Augmented Reality for Kilobots system [74], the addition of virtual sensors using external hardware allows for more complex behaviors without greatly increasing the hardware cost or complexity of the vessels.

Each microUSV is marked with a unique AprilTag attached to its lid. These tags allow the position of each vessel to be tracked when in view of an overhead camera using the AprilTag detection algorithm [68]. These tags must be kept in view of the camera in order to receive up-to-date pose estimates and calculate a vessel's simulated sensor values. The AprilTag detection algorithm can detect tags with a large degree of skew but to ensure the most reliable performance the microUSV design attempts to minimize roll motion and so must be very stable, a dominant factor in the vessel's hull design.

3.2.2 Power System

The vessel's power system was designed to be as simple as possible. Power is provided by a pair of standard 9V alkaline batteries. Using standard batteries greatly simplifies vessel maintenance and they are much cheaper than lithium-polymer batteries which are popular in mobile robotics applications. Since the microUSV operates exclusively indoors, ease of maintenance was deemed more important than extended battery life as the vessels can be easily pulled from the water and have fresh batteries installed between experiments if necessary.

The Raspberry Pi, IMU, and motor controller cannot accept the 9V battery power with a maximum input voltage for each device ranging from 5V to 5.5V. A simple voltage regulator circuit, shown in Figure 3.11, was added to the design to accommodate these requirements providing regulated 5V DC and unregulated 9V DC. The unregulated 9V DC power is used to drive the motors. This circuit doubles as a voltage bus allowing devices to be quickly connected to power without disturbing the rest of the system using standard 2.54mm header pin connectors. The 5V section of the voltage bus has its ground and live rail running adjacent to each other while the 9V section has a 2.54mm (one pin row) separation between its live and ground



ITEM NO.	DESCRIPTION	QTY.
1	Voltage Bus Breadboard	1
2	LM7805 Voltage Regulator	1
3	Slider Switch	1
4	Male Header Pin	44
5	Voltage Bus Jumper Wire	1
6	Solder Bridge	5

Figure 3.11: Voltage Bus Drawing

rails. By using the appropriately sized connector housings on their jumper wires, a two-pin housing for 5V devices and a 3-pin housing with a gap in the middle slot for 9V devices, the risk of connecting a device to the wrong input voltage is greatly reduced.

3.3 Fabrication and Assembly

A detailed Bill of Materials (BOM) for all components required to build a microUSV can be found in Appendix A.1. A simplified BOM is included in Table 3.1, which summarizes the components required to produce a single microUSV with cost estimates in Canadian dollars.

The instructions for fabrication, assembly, and testing of the microUSV are provided in detail on the project repository Wiki page [40]. The major steps are summa-

Component Category	**Description**	**Cost [$CAD]**	**Sources of Materials**
3D Printed Components	Custom 3D printed components including the Hull, Lid, and brackets.	90.44	3D Printer (Custom)
Electronic Devices and Wiring	Electronic devices such as the Raspberry Pi, Arduino, and motor controller as well as the components to connect them	200.18	BuyaPi, Digikey, Pololu
Mechanical Components and Fasteners	Screws, Nuts, Propellers, Drive Shafts, etc.	146.39	Amazon, Hobby King, McMaster-Carr
Adhesives and Sealants	Epoxy, silicone sealant, Loctite	85.00	Amazon, Hardware Stores
	Total:	**522.00**	

Table 3.1: Generalized Bill of Materials

rized below and require an estimated two days of fabrication effort for a single user to complete with an additional three days of indirect production time also required for steps such as 3D printing and adhesive curing.

- 3D print all necessary components
- Insert a ballast plate into the keel
- Mount the keel to the bottom of the hull
- Apply epoxy coating to hull and lid
- Cut and assemble two stuffing tubes
- Thread the tips of two drive shafts
- Assemble and solder the voltage bus
- Solder header pins onto all other electronic devices
- Mount all electronic devices to the electronics bracket
- Cut and terminate eight jumper cables
- Use jumper cables to connect electronic devices
- Bore propellers and one end of each shaft coupler to accommodate a 1/8" drive shaft
- Mount motors, drive shafts, and stuffing tubes inside the hull
- Mount batteries and electronics bracket inside the hull

- Install desktop and onboard software

- Configure lab environment

3.4 Peripheral Devices

In addition to the vessels themselves, several other devices are required for the microUSV system to function: A Universal Serial Bus (USB) camera, a computer, and a wireless router. Figure 3.12 illustrates the arrangement of these devices to form the external pose detector system discussed in Chapter 4.

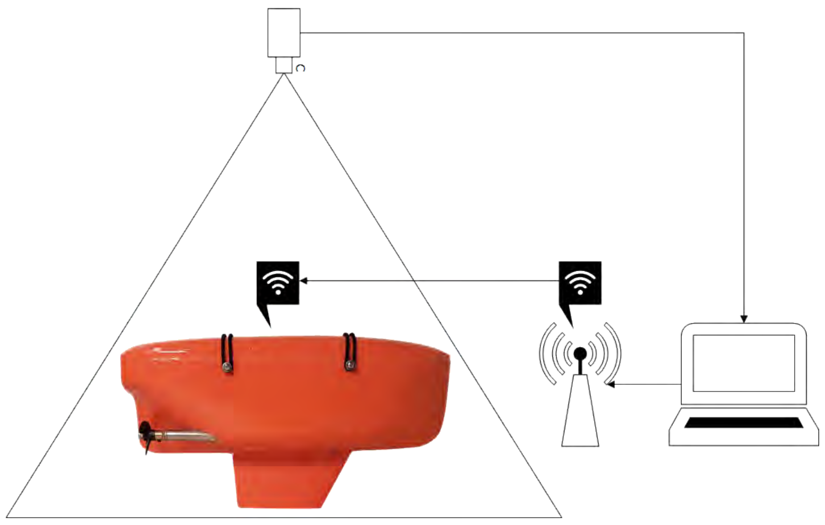


Figure 3.12: External Pose Detector System Data Flow Diagram

A custom single-camera system was chosen over a commercial pose tracking system due to the significantly lower cost. The camera must be mounted above the operating area in order to capture video of the vessels in operation or more specifically, their AprilTags. A Logitech C920 camera was used for this purpose during preliminary development but was later replaced with an Intel RealSense D435. The high video feed resolution of these cameras allows for detections at a longer range, thus increasing the size of the operating area however the increased video bandwidth demands more processing time and so can introduce latency. The best results were achieved during testing using either camera set to output a 720p video feed. The D435 was ultimately selected because it offered lower latency than the C920, allowing the system to achieve a higher update frequency, and also features a global shutter which eliminates the occasional spatial distortion introduced by the C920's rolling shutter. The camera is connected via USB to a computer, referred to henceforth as the server computer.

The server computer is responsible for running the AprilTag detection algorithm to continuously update a 2D pose estimate for each microUSV visible in the video feed. The computer used during testing was running an Intel i7-8750H CPU. It also receives update requests from the microUSVs and responds with simulated sensor data messages. The server computer sends these messages over a wireless network via a router to which all the microUSVs are connected. The wireless router used in testing was a NETGEAR Nighthawk AC1900 with a maximum bandwidth of 1300 Mbps however a router of this quality is not necessary. Each vessel requires approximately 200 Bytes/second of bandwidth so a small fleet of vessels can easily be managed by a low-end wireless router.

Chapter 4

Software Design

This chapter discusses the design of the microUSV's control software. All software used for this project was either available through open source libraries or custom written for this application. The complete source code is publicly available via the project repository [40] or GitHub (`https://github.com/CalvinGregory/microUSV`).

4.1 System Architecture

Three software applications running concurrently are required for microUSV operation: the main controller application running onboard each vessel called MUSVController, a small motor controller application called PeripheralController, and a server application called CVSensorSimulator (CVSS). These applications are summarized in Table 4.1. As discussed in Chapter 3, the selection of sensors suitable for use on the microUSV was limited due to its small size and indoor operating environment. Instead of physical sensors, the system design calls for the use of an external pose tracking system to simulate the sensor values observed by each vessel.

Application Name	**Language**	**Hardware Device**
MUSVController	Python	Onboard Raspberry Pi
PeripheralController	C	Onboard Arduino Nano
CVSensorSimulator	C++	External Server Computer

Table 4.1: microUSV Control Software Applications

CVSS is responsible for continuously calculating and updating these values based on the data acquired from an overhead camera positioned above the operating area. During operation one or more vessels will query the server computer for their sensor data. The sensor values are sent to the Raspberry Pi onboard each microUSV which is running the MUSVController application. This application is responsible for interpreting the simulated sensor data to determine the vessel's next action, i.e. the speed at which to spin its port and starboard propellers. These two speed values are then sent to the vessel's onboard Arduino which is connected to the motor controller and running the PeripheralController application. This application parses motor speed messages from the Raspberry Pi and forwards the values to the motor controller which applies voltage to each motor accordingly. This process, illustrated in Figure 4.1, repeats continuously to form a feedback loop.

4.1.1 Server Software

The Linux server computer runs the CVSS application which uses computer vision to continuously estimate the values observed by each microUSV's virtual sensors. Its primary purpose is providing a global pose estimate to each vessel using the

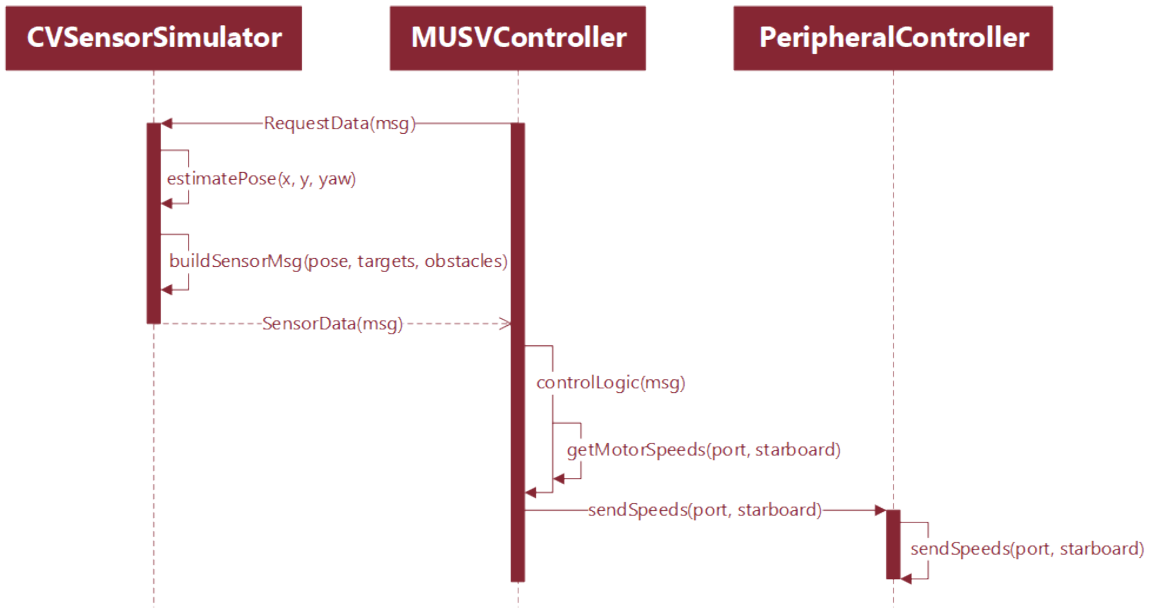

CVSensorSimulator
MUSVController
PeripheralController
RequestData(msg)
estimatePose(x, y, yaw)
buildSensorMsg(pose, targets, obstacles)
SensorData(msg)
controlLogic(msg)
getMotorSpeeds(port, starboard)
sendSpeeds(port, starboard)
sendSpeeds(port, starboard)

Figure 4.1: Control Software Inter-Application Message Sequence Diagram

AprilTag attached to its lid. Since USVs operate in a planar environment they are only concerned with 2D pose information. The server software's global pose estimates consist of a vessel's position in the x and y axes and its heading angle (x, y, θ). To simplify the interface to the AprilTag library [68] written in C, CVSS was written in C++. As a global manager agent, it has access to all pose estimates simultaneously and so can also be used to simulate multiple sensors for behaviors such as collision avoidance and target tracking.

The microUSV platform does not meet the full definition of a swarm robotics platform due to the inclusion of this centralized agent. The server's inclusion does not, however, prevent it from fulfilling its primary goal of providing a suitable swarm research platform: the server only simulates sensor values and all control decisions occur onboard the vessels themselves. The individual members of the swarm do not have full access to the server's global knowledge and so can still effectively replicate the behaviors of decentralized agents.

The global information available to the application via the server offers some benefits to the system as well: it allows the simulation of different sensor configurations and inter-vessel communication schemes by artificially constraining access to data. Different sensor types, arrangements, ranges, and resolutions can all be tested and tuned without altering the underlying hardware. Researchers can also easily test algorithms using multiple communication schemes. As an example, a system with no inter-vessel communication can be simulated by restricting a vessel's sensor data access to its own data. A system with limited inter-vessel communication range can be simulated by restricting a vessel's sensor data access to its own data in addition to the data of any other vessels within an arbitrary distance.

4.1.2 Vessel Control Software

There are two devices onboard each vessel running custom software: A Raspberry Pi (the primary controller) and an Arduino (the peripheral controller). The Arduino application, PeripheralController, simply acts as a pass-through device for motor speed messages. It receives integer motor speed commands from the primary controller and forwards them to the Qik motor controller using Pololu's Arduino library for the device. This application can also be modified to include an interface to the onboard IMU, also a Pololu device with an accompanying library, which connects to the Arduino. The Inertial Measurement Unit (IMU) data can be used to augment a state estimation system implemented on the Raspberry Pi.

The microUSV's primary controller runs the MUSVController application. This application was written in Python due to the Raspbian operating system's native support for the language and the ability to prototype quickly. It contains the logic for autonomous control of the vessel which includes requesting and parsing sensor data from CVSS.

4.2 Inter-Application Communication

The microUSV system has two inter-application communication links: the CVSS to MUSVController link between each vessel and the central server and the MUSVController to PeripheralController link onboard a vessel between its primary controller and peripheral controller boards as shown in Figure 4.1. The link between a vessel and the server is handled wirelessly over WiFi while the link between two devices onboard each vessel is a wired connection over USB.

The wired link between a microUSV's onboard Raspberry Pi and Arduino is used to send motor speed commands from the main controller application to the motor controller interface. This interface can also be used to communicate with other peripheral devices onboard the vessel such as an IMU but at the time of writing, only the motor controller interface was required. These are simple messages consisting of a pair of integers; one for the starboard motor speed and one for the port motor speed. These speed values must fall within the range of -127 and +127 to be considered valid commands so the PeripheralController application truncates any values it receives outside that range to the maximum positive or negative speed as appropriate. Each message is prepended by two consecutive * characters as seen in Code Listings 4.1 and 4.2. This short header is included to avoid passing any noise values picked up by the serial line to the motor controller. The messages are sent over a serial port at a baud rate of 115200 for minimal latency. The MUSVController application uses the pyserial library to encode its serial messages while the Arduino can handle parsing serial messages natively.

```
def send_speeds(arduino, portSpeed, starboardSpeed):
  arduino.write(struct.pack('<cchh', '*', '*', \
                starboardSpeed, portSpeed))
  // struct.pack encodes byte information as a string for
  // serial transmission. The argument '<cchh' indicates the
  // string will use little-endian encoding and contain 2
  // chars followed by 2 short ints.
  return
```

Code Listing 4.1: Motor Speed Message Send Function

```
if(Serial.available()) {
  if(Serial.read() == '*') {
    if(Serial.read() == '*') {
      qik.setSpeeds(recvInt(),recvInt());
} } }
```

Code Listing 4.2: Motor Speed Message Receive Loop

The wireless link between the server and a given microUSV needs to encode much more data than the wired link and should be scalable to accommodate future additions to the system's simulated sensor suite. This makes the simple, nonscaling communication scheme written for the wired link unfeasible for long term use in the wireless link. It must also be able to transfer data between applications written in two different languages; C++ and Python. Google's Protocol Buffer (protobuf) library [8], specifically protobuf build 3.7.1, was chosen as a suitable open source, language-neutral tool for this task. It serializes data structures in a compact way that requires less data than more traditional solutions such as Extensible Markup Language (XML) which reduces the system's wireless bandwidth requirement, allowing for the operation of more vessels simultaneously and potentially lower communication latency. These messages are transmitted over WiFi using the Transmission Control Protocol (TCP). TCP was chosen over User Datagram Protocol (UDP) since the maximum expected number of vessels operating simultaneously was relatively low, in the eight to twelve range. The error checking offered by TCP was deemed more important than the slight improvement in speed offered by UDP with such low network traffic and the higher risk of packet loss using UDP was unacceptable for robot navigation.

The wireless link communication protocol works as follows: The server computer running CVSS maintains an open network socket awaiting messages from any microUSV. The microUSVs all have access to the server's socket address and can send it a message requesting their most recently updated sensor values. The request message definition can be seen in Code Listing 4.3. The request message contains the AprilTag ID number of the sending microUSV which allows the server to identify the vessel which sent the request and pull the data associated with that vessel. The server responds to the request message by sending a SensorData message, defined in Code

Listing 4.4, which contains the requesting vessel's simulated sensor values. Details of the message format and syntax used can be found in the protobuf documentation [8].

```
message RequestData {
  int32 tag_id = 1;
  bool request_waypoints = 2; }
```

Code Listing 4.3: Protobuf RequestData Message Definition

4.3 CVSensorSimulator Implementation

CVSS is responsible for tracking all AprilTag-marked microUSVs in the operating environment and calculating their sensor values. To achieve this, it stores a list of Robot objects, one for each microUSV, which contain the current sensor values for that vessel, as shown in Figure 4.2. These sensor values must be updated each time the system receives a new frame from the overhead camera. Sensor updates are calculated based on the data provided by the PoseDetector object; a wrapper class around the AprilTag library [68]. This object pulls camera frames from the FrameBuffer and calculates the pose of any visible AprilTags. The pose estimates are then combined with location estimates for any colored targets in the operating area which are extracted from the camera frame using Hue, Saturation, Value (HSV) color thresholding methods in the OpenCV library [21]. Floating colored targets, specifically magenta ping-pong balls, were used as a generic stand-in for surface contaminants in marine environments for the purpose of experiments. The combined data provides CVSS

```
message SensorData {
  message Pose2D {
    float x = 1;
    float y = 2;
    float yaw = 3; }
  message Waypoint {
    float x = 1;
    float y = 2; }
  Pose2D pose = 1;
  repeated Pose2D nearby_vessel_poses = 2;
  repeated int32 target_sensors = 3 [packed=true];
  google.protobuf.Timestamp timestamp = 4;
  repeated Waypoint waypoints = 5;
  bool loop_waypoints = 6; }
```

Code Listing 4.4: Protobuf SensorData Message Definition with an estimate of each vessel's pose in the environment as well as the values of its "target sensors" which detect floating colored targets near each vessel.

4.3.1 Concurrency

Robots rely on frequent, real-time updates from their sensors to autonomously navigate their environments. Since CVSS is serving as a substitute for the microUSV's key navigation sensors, it is crucial that the application exhibit as little latency as possible with an acceptable update frequency. The sequential execution of each step needed to update the sensor values of all vessels was far too slow to enable real-time operation: Parallelizing as much of the application as possible was essential.

CVSS runs five concurrent threads:

- the main/server thread,
- the video capture thread,
- the AprilTag detector thread,
- the target detector thread,
- and the detection processor thread.

The application's main thread doubles as the thread handling socket requests and responses over the wireless network. After initializing the application, it maintains an open socket awaiting request messages from a microUSV. When a request message is received, the thread identifies which Robot object corresponds to the requesting vessel, pulls the sensor values currently stored in that Robot object and puts them

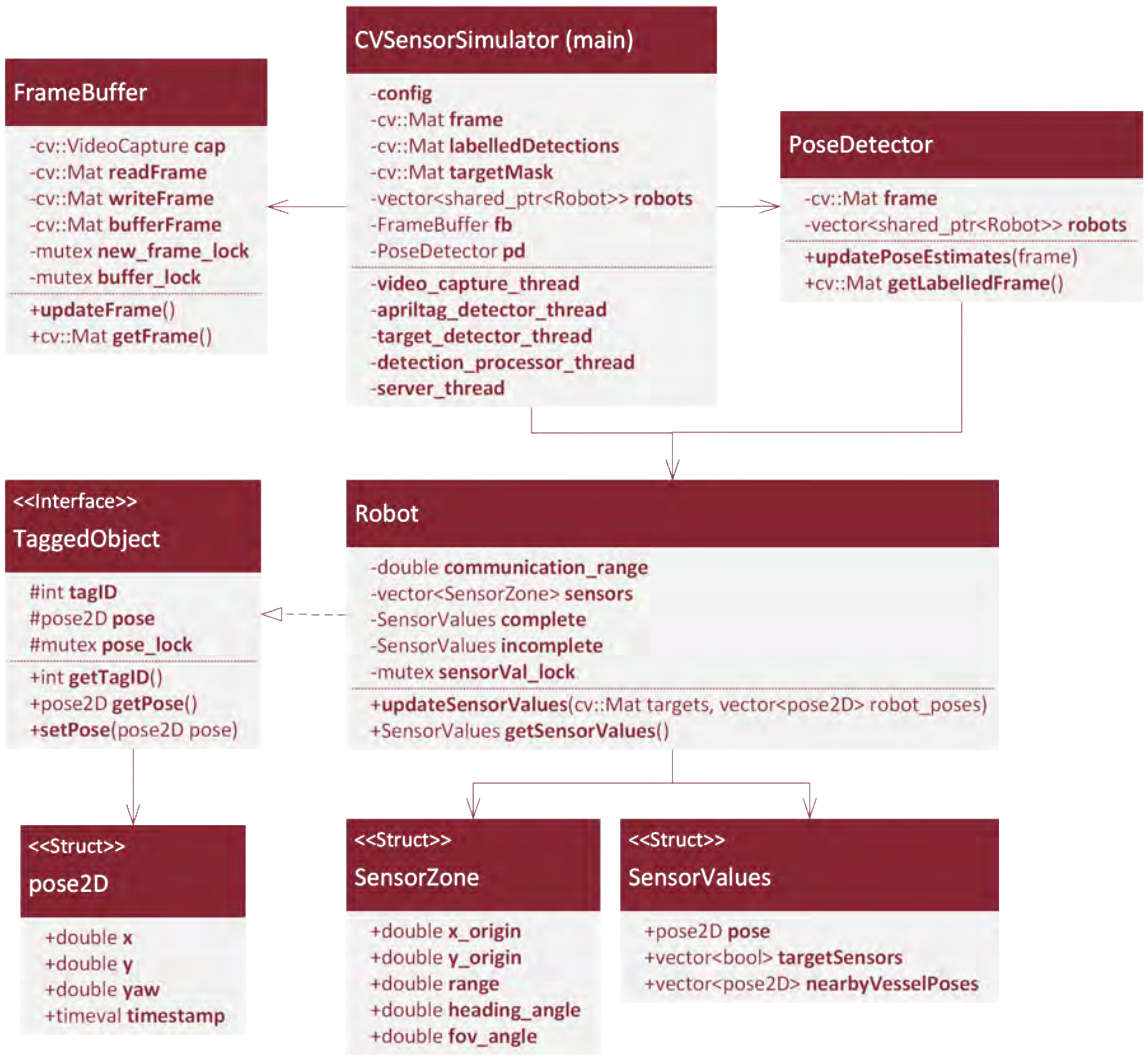


Figure 4.2: CVSensorSimulator Simplified Class Diagram

in a SensorData message which it sends back to the vessel. Since this operation can occur at any time, it is entirely likely the server thread will attempt to pull data from a Robot object while that object is being updated with new data by another thread. This means means the Robot class must be thread-safe.

To that end, each Robot object contains two instances of its SensorValues; **complete** and **incomplete**. The server's pull operation can only access the **complete** set of sensor values, locking them during use, while the update operation can access both instances. When an update occurs, the updating thread iterates through each value in the **incomplete** SensorValues instance before locking the **complete** values and copying over the now-updated values from the **incomplete** instance, as seen in Code Listing 4.5.

```
void Robot::updateSensorValues() {
  // update each value in SensorValues incomplete...
  std::lock_guard<std::mutex> lock(sensorVal_lock);
  complete = incomplete; }
SensorValues Robot::getSensorValues() {
  std::lock_guard<std::mutex> lock(sensorVal_lock);
  return complete; }
```

Code Listing 4.5: Robot Class SensorValue Update and Get Methods Pseudocode

The video capture thread has a single purpose: to pull frames from the overhead camera into memory as soon as they are available. This thread has access to the application's FrameBuffer object and executes the **updateFrame** method continuously in a loop. The FrameBuffer's data can also be accessed by the AprilTag detector thread which retrieves the most recent frame for processing, so much like the Robot class, the FrameBuffer class must be thread-safe.

The FrameBuffer object is functionally a queue of length one which decouples the acquisition and accessing operations associated with camera frame data. Since CVSS is trying to provide real-time sensor values, each new frame must immediately supersede the previous one: saving old frames for processing is redundant since the data they contain is out of date as soon as a new frame is available. The FrameBuffer stores only the most up to date data without introducing a backlog of redundant frames.

The FrameBuffer class contains three image variables: a **readFrame**, a **writeFrame**, and a **bufferFrame**. The class' get method can access the **readFrame** while the update method can access the **writeFrame**. Both methods have access to the **bufferFrame**. When called, the update method overwrites the **writeFrame** with the newest camera frame data then locks the buffer. It then moves the frame it stored in the **writeFrame** into the buffer. Since the data in the **bufferFrame** is redundant as soon as a new **writeFrame** is acquired, the method uses pointer swapping instead of a move operation for better speed: The redundant data in the **bufferFrame** is moved into the **writeFrame** where it can safely be overwritten during the next update operation. The update method then unlocks the **new_frame_lock** to indicate that a new frame is in the buffer.

The FrameBuffer's get method starts by trying to acquire the **new_frame_lock** which is initialized in a locked state. If no new frame is in the buffer the method call waits there for an update operation to complete and the lock to be released. Once the **new_frame_lock** is acquired, the get method also locks the buffer and swaps the **bufferFrame** into the **readFrame** and returns its value. These methods can be seen in Code Listing 4.6.

```
void FrameBuffer::updateFrame() {
  cv::VideoCapture >> *writeFrame;
  std::lock_guard<std::mutex> lock(buffer_lock);
  writeFrame.swap(bufferFrame);
  new_frame_lock.unlock(); }
Mat FrameBuffer::getFrame() {
  new_frame_lock.lock();
  std::lock_guard<std::mutex> lock(buffer_lock);
  readFrame.swap(bufferFrame);
  return *readFrame; }
```

Code Listing 4.6: FrameBuffer Update and Get Method Pseudocode

The AprilTag detector thread, target detector thread, and detection processor thread all operate on the same data from the FrameBuffer so they must be synchronized. These threads use a set of three barriers to ensure they are working on, at most, two subsequent frames. The AprilTag thread starts by acquiring the most recent camera frame from the FrameBuffer and stores it in a global variable **frame**. It then reaches the **frameAcquisitionBarrier**. The target detector thread starts its execution by waiting at that barrier so once it is reached by the AprilTag detector thread both threads can begin processing. The AprilTag detector thread proceeds to run the **frame** through the AprilTag algorithm via a PoseDetector object while the target detector thread runs HSV color thresholding on the **frame**. Once each of these threads has finished their work they arrive at the first of a pair of detector barriers.

The detection processor thread performs post-processing on the data acquired by the AprilTag detector and target detector threads. It starts its execution waiting at the first detector barrier. Once the other two threads arrive there, it pulls the data they have been working on from global variables into its local variables and then arrives at the second detector barrier. The AprilTag detector and target detector threads do not have any instructions between the first and second detector barriers so they arrive immediately and are synchronized with the detection processor thread. At this point the AprilTag detector and target detector thread have finished their processing responsibilities for frame **n** and have handed their data to the detection processor thread. They return to the start of their respective loops and begin processing data on frame **n+1** while the detection processor thread finishes processing frame **n**. The detection processor thread proceeds to use the data from the other two detection threads to populate the sensor values for each Robot object before restarting its own loop to await their arrival once again.

This method of dividing pre and post processing tasks between multiple threads allows CVSS to be processing two frames simultaneously at all times. This sequence of thread interactions is illustrated in Figure 4.3. Figure 4.4 shows the same sequence of inter-thread communications while highlighting the concurrent processing of frames **n** and **n+1**.

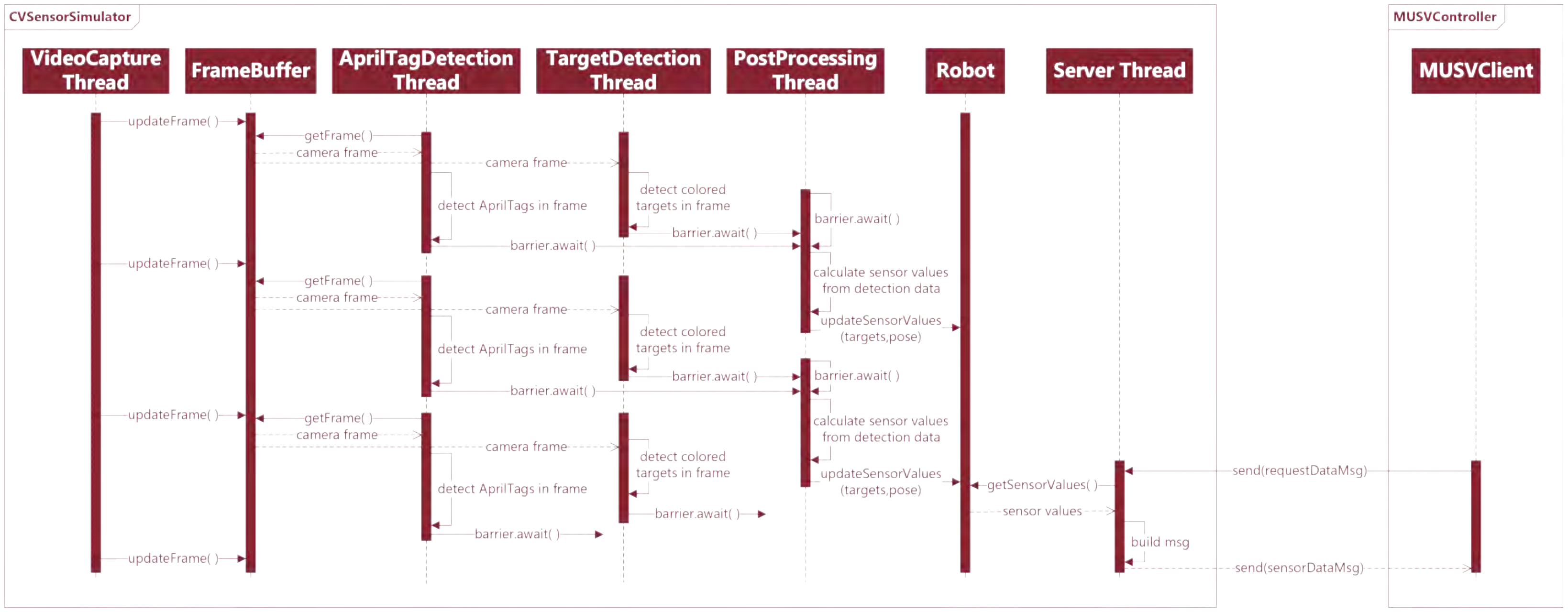


Figure 4.3: CVSensorSimulator Thread Sequence Diagram

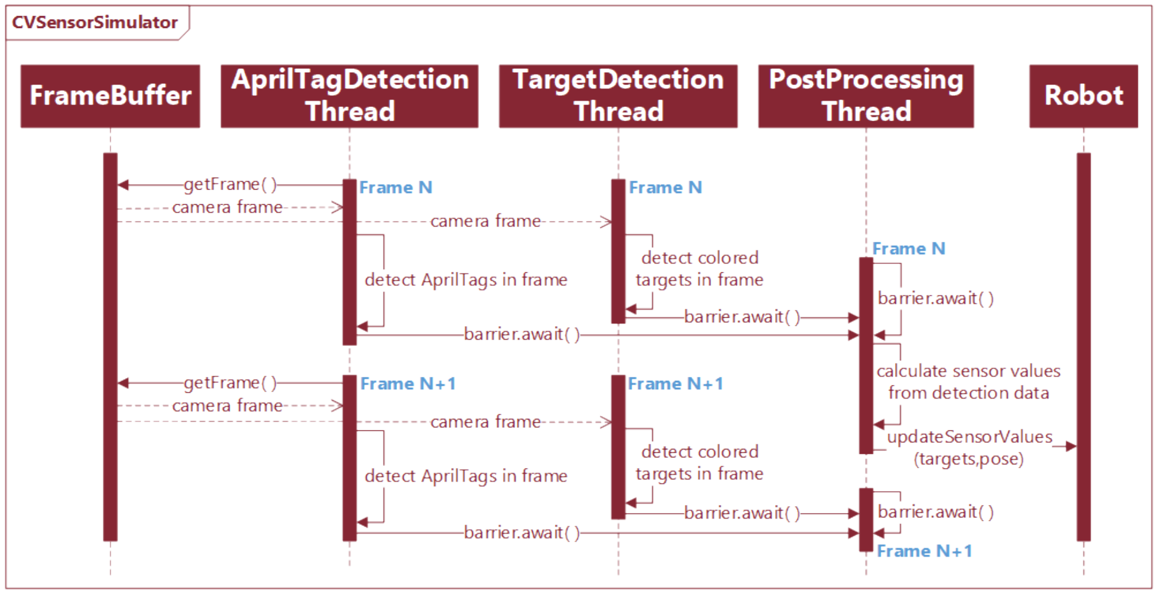


Figure 4.4: CVSensorSimulator Thread Sequence Diagram - Frame Handoff

Figure 4.5 shows that CVSS is able to maintain a stable sensor estimate update rate between 17 and 21 Hz while managing a fleet of up to 10 vessels. This update frequency is sufficient for simple navigation tasks even without utilizing the onboard IMU to estimate pose between updates.

4.3.2 Sensing and Communication

The computer vision-based nature of CVSS heavily influences the style of its virtual sensors: they are limited to those that can be estimated using camera data from a single perspective. Two types of sensors have been implemented in the current build: vessel pose estimation and colored target detection. The pose estimation sensor is a stand-in for a localization sensor such as Global Positioning System (GPS) or an Inertial Navigation System (INS). The colored target detection sensor represents a

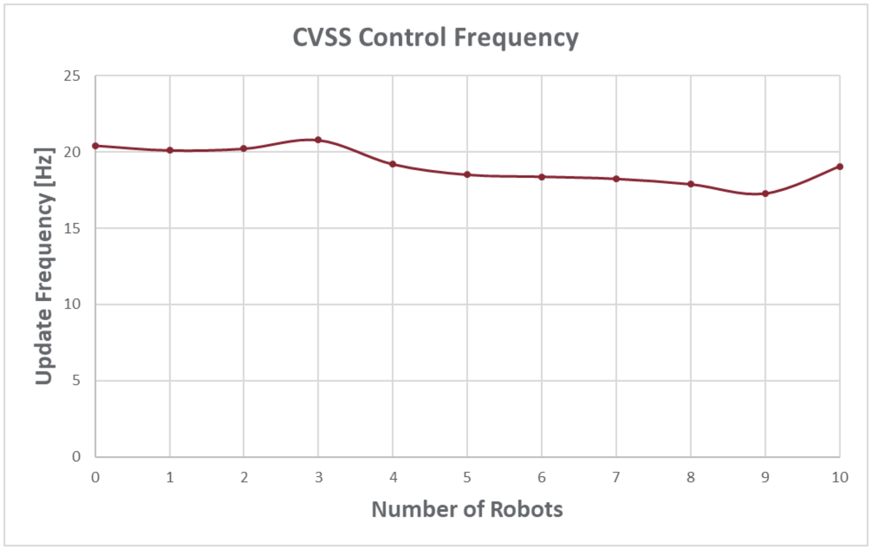


Figure 4.5: Server Computer Software Update Frequency vs Number of Robots on the Network

means of ranged environmental sensing such as LIDAR, radar, a standard or infrared camera, etc. Since the microUSV is meant as a general purpose research platform, the implementation of its target detection system is not specific to a particular sensor or method. Sensors used to detect icebergs are not necessarily suited to tracking floating plastics or oil slicks and vice versa. It is assumed that in any real-world implementation based on microUSV testing the specific sensor(s) will be chosen based on the target application but will be capable of producing similar types of ranged detection data.

The SensorData message type used to transmit data from CVSS to a microUSV, shown in Code Listing 4.4, defines all sensor types currently implemented including localization and environmental sensors. Each message field represents the data from a particular sensor. It also defines two sub-message types which are treated as objects on either end of the transmission to simplify the message construction and parsing code; Pose2D and Waypoint. Since USVs operate in an effectively planar environment, the application extracts each vessel's two dimensional pose from its AprilTag detection data for localization: x position, y position, and heading angle (yaw). Waypoints are also represented as 2D coordinates. The values in the **waypoints** message field are not sensor values, but instead used for pathing in the waypoint following controller discussed in Chapter 5.

Each microUSV is marked with a unique AprilTag attached to its lid. These fiducial tags allow the position of each vessel to be tracked when in view of an overhead camera using the AprilTag detection algorithm [68]. These tags must be kept in view of the camera in order to receive up-to-date pose. The AprilTag detection algorithm can detect tags with a large degree of skew but to ensure the most reliable performance the microUSV design attempts to minimize roll motion and so must be very stable; a dominant factor in the vessel's hull design. AprilTags have several families of tags with different levels of pattern intricacy: more complex tag families can have more pattern variations and combinations and thus more family members while simpler families can generally be perceived at greater distances. The detection distance was of greater importance to the microUSV system than the number of possible tags so a simple tag family, 25h9, was chosen. This tag family is the second lowest in resolution, ahead of the 16h5 family which was found to be susceptible to noise during testing.

The CVSS coordinate system, shown in Figure 4.6, is based on the data produced by the AprilTag algorithm. It follows a right handed coordinate system with its origin placed at the center of the camera frame with heading angles being measured clockwise relative to the negative y axis.

AprilTag detection and processing is handled in CVSS by the PoseDetector class.

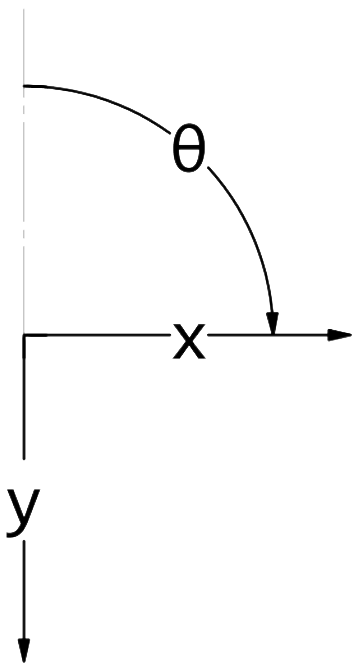


Figure 4.6: CVSS Coordinate Frame

This wrapper class searches a given camera frame for the AprilTags of microUSVs stored in the system. It intentionally neglects any detections of tags not associated with a microUSV as these spurious detections clutter the interface and waste processing time. The pose estimates it creates are stored in CVSS' Robot objects before being bundled into a SensorData message, specifically the **pose** field, and sent to the appropriate microUSV. This class is also responsible for drawing tag labels onto the video feed shown to the operator as seen in Figure 4.7.

A microUSV's target sensor, represented in the **target_sensors** field of a SensorData message, detects the presence of objects of a specific color near the vessel. This sensor is divided into six zones arranged to the port-forward section of the vessel with an additional small capture sensor zone near the vessel's midpoint as seen in Figure

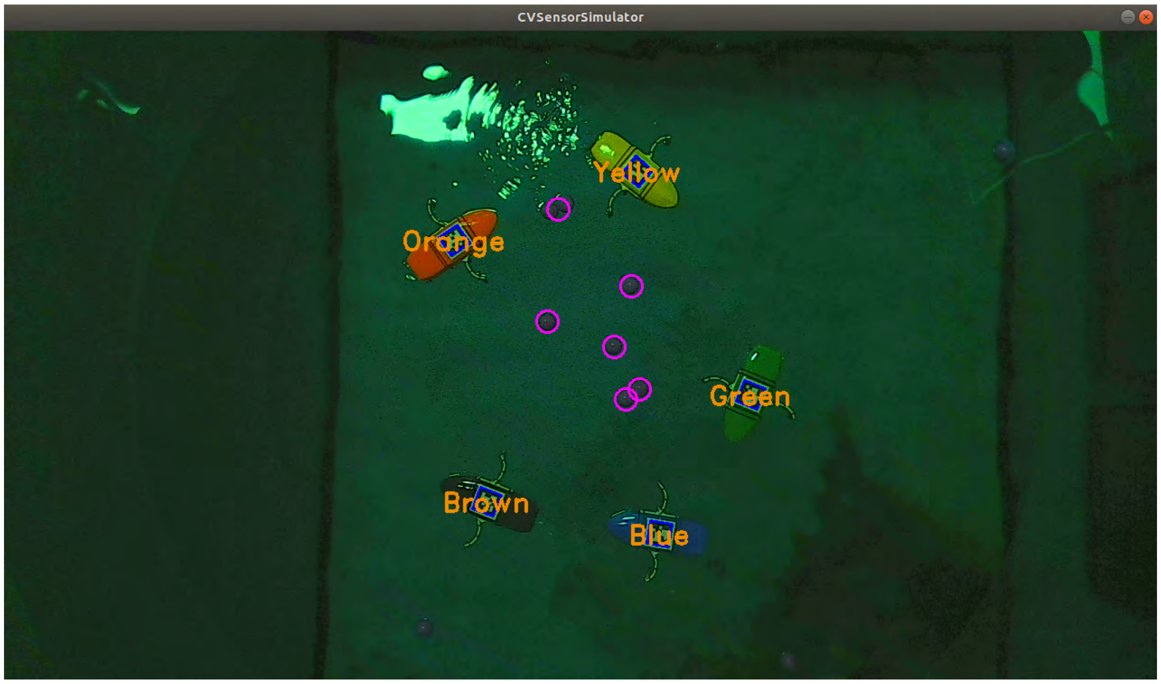


Figure 4.7: CVSS Application Window

4.8. Each triangular zone represents a 10° sub-arc of a 60° wide sensor field of view extending ahead of the vessel. Detections in each sub-arc are calculated and reported independently so a vessel can have coarse information about the relative heading of a detected target. Detections in the rectangular capture sensor zone are also calculated and reported separately from the other zones. This virtual sensor is positioned to solely detect when the microUSV has captured a target in one of its side-mounted nets.

This sensor layout was tailored specifically to the Orbital Retrieval algorithm discussed in Chapter 6 which was only concerned with detecting targets on each vessel's port side and those it had captured. This focus is reflected in the sensor zone layout. The sensor geometry described is only one example of how the the system could be configured. It can be easily rearranged to suit different target applications

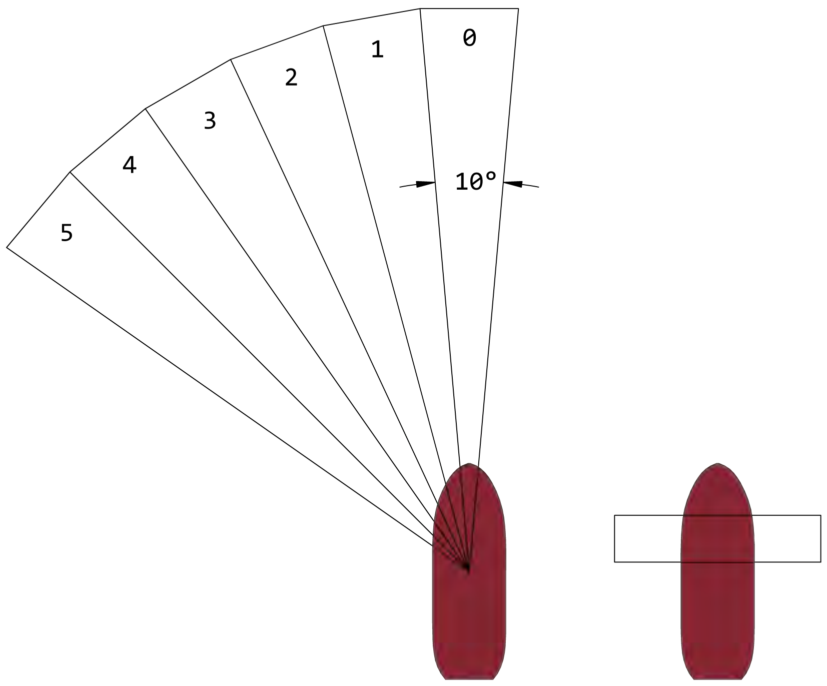


Figure 4.8: Colored Target SensorZone Geometry

without modifying the underlying hardware.

The target sensors are a computer vision-based system: Their values are calculated by CVSS. First a vessel's pose is estimated by the AprilTag detector thread. Concurrently the target detector thread is performing color thresholding on the same frame. This process creates a binary image where all pixels in the specified color range are set to one while the rest are set to zero. The system was tuned to detect magenta as ping-pong balls of this color were used as targets for the experiments in Chapter 6. Both results are then passed to the detection processor thread.

The detection processor thread iterates through the system's list of Robot objects, passing to each the poses of all detected vessels and the binary image of detected targets. The Robot objects then create a series of sensor zone masks based on their current pose estimate and the SensorZone geometry shown in Figure 4.8. The sensor zone masks are binary images with the same dimensions as the binary image of detected targets which are initialized with zeros at each pixel location. Onto each mask a different sensor zone sub-arc or capture sensor zone is projected from the vessel's pose in the form of pixel values set to one. These masks are then each compared against the binary targets image using a bitwise AND operation. If any non-zero pixels from the binary targets image overlap with the area of non-zero pixels defined in a sensor zone mask, that sensor zone is assigned a detection value of true. If no target pixels overlap with a sensor zone mask, that zone has a value of false. The binary detection values are then stored as an ordered list in the Robot object's **complete** SensorValues to be later retrieved by the server thread.

The quality and stability of both AprilTag and colored target detections depends heavily on the lighting conditions of the testing environment. In general, a brightly lit environment is better for detections however this can sometimes introduce areas with bright reflections which may impair the detection of any vessels or objects inside them. This can be seen near the top edge of Figure 4.7 where the testing tank's overhead fluorescent light is reflecting off of the water's surface. Diffusing light in the testing environment can help mitigate this.

The detection processor thread also uses the AprilTag detection data to simulate inter-vessel communication at limited range. The system, knowing the pose of every detected vessel, can calculate the distance between each of them. Any vessels which fall within a specified range of each other are considered able to communicate and so can share their location. This is simulated in CVSS by adding the pose data of any microUSV within range of a given vessel to that vessel's SensorData message in the **nearby_vessel_poses** field. This communication scheme was designed for the collision avoidance behavior in the orbital retrieval algorithm but can be augmented or changed to simulate alternate communication schemes transmitting any type of data.

4.4 MUSVController Implementation

The MUSVController application is comparatively simple, consisting of only a main client with a Controller object, shown in Figure 4.9. The client code, running on the microUSV's Raspberry Pi, is responsible for connecting to the CVSS socket and sending a RequestData message. It will then receive a SensorData message in response which it passes to its Controller object.

The Controller object can be an instance of one of several concrete classes implementing the Controller interface: WaypointController, OrbitalConstructionController, or OrbitalRetrievalController. The specific controller type can be chosen during initialization. Additional implementations can also be added easily. The Controller interface defines several key properties often critical for controller design and a single method: **get_motor_speeds**. This method is expected to accept the vessel's sensor data as an argument, perform its control logic internally, and return a pair of integer motor speeds. The client code calls this method in a loop, passing it new sensor values each time and then forwarding the motor speed values to the PeripheralController application over serial running on the vessel's onboard Arduino.

Of the three controller types implemented, two will be discussed in detail during later chapters: the WaypointController is discussed in Chapter 5 and the OrbitalRe-

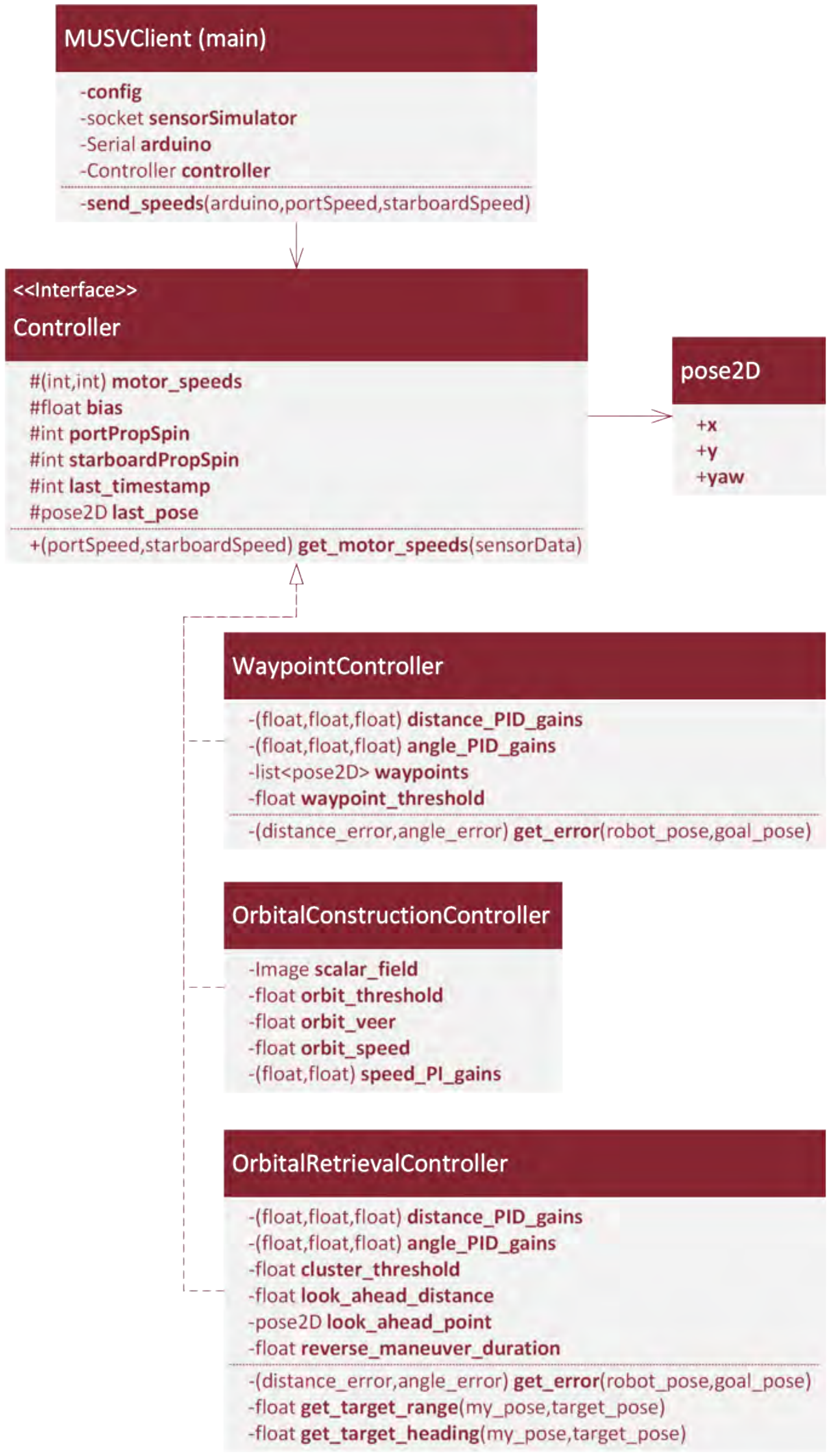

MUSVClient (main)
-config
-socket sensorSimulator
-Serial arduino
-Controller controller
-send_speeds(arduino,portSpeed,starboardSpeed)
<<Interface>>
Controller
#(int,int) motor_speeds
#float bias
#int portPropSpin
#int starboardPropSpin
#int last_timestamp
#pose2D last_pose
+(portSpeed,starboardSpeed) get_motor_speeds(sensorData)
pose2D
+x
+y
+yaw
WaypointController
-(float,float,float) distance_PID_gains
-(float,float,float) angle_PID_gains
-list<pose2D> waypoints
-float waypoint_threshold
-(distance_error,angle_error) get_error(robot_pose,goal_pose)
OrbitalConstructionController
-Image scalar_field
-float orbit_threshold
-float orbit_veer
-float orbit_speed
-(float,float) speed_PI_gains
OrbitalRetrievalController
-(float,float,float) distance_PID_gains
-(float,float,float) angle_PID_gains
-float cluster_threshold
-float look_ahead_distance
-pose2D look_ahead_point
-float reverse_maneuver_duration
-(distance_error,angle_error) get_error(robot_pose,goal_pose)
-float get_target_range(my_pose,target_pose)
-float get_target_heading(my_pose,target_pose)

Figure 4.9: MUSVController Simplified Class Diagram

trievalController is discussed in Chapter 6. The OrbitalConstructionController was an implementation of the algorithm described by Vardy in [89]. This algorithm was implemented as one of the first swarm applications tested on the microUSV platform.

During testing it was found that this algorithm's purely reactive controller style may not be suitable for the marine environment. The vessels' momentum causes them to drift on the surface of the water, introducing an effective latency of motion. This algorithm was developed using terrestrial robots where the controller issuing a turn command would result in an instantaneous turn. This does not happen when a USV controller issues a turn command which can influence output thrust, not speed. This means the vessels need time to accelerate and decelerate into their desired heading. As written, the orbital construction algorithm does not handle the motion latency well: the vessels tend to oscillate weakly along a mostly linear path leading out of the operating area instead of in the desired circular orbit. The lessons learned through implementing orbital construction heavily influenced the development of the orbital retrieval algorithm.

4.5 PeripheralController Implementation

The PeripheralController application, written in C, serves a single purpose: forwarding motor speed messages to the vessel's motor controller. It acts solely as a pass-through application, receiving and parsing serial messages, performing simple error checks, then passing the values to the motor controller via the PololuQik Arduino library.

Chapter 5

System Configuration and Testing

This chapter describes the configuration steps needed to set up the microUSV system and the laboratory environment in which the vessels will operate. This includes the installation requirements and procedures for the system's peripheral hardware as well as discussion of server and vessel software configuration parameters. It also discusses the methods and results from a series of waypoint following experiments used to validate the system's functionality.

5.1 Laboratory Environment Configuration

To configure an indoor body of water, such as a laboratory tank, for use as a microUSV testing environment, the system's peripheral devices must first be set up. As discussed in Chapter 3, the microUSV system requires a server computer, wireless router, and Universal Serial Bus (USB) camera in addition to the vessels themselves.

The camera must be positioned above and facing directly toward the water's surface and connected to the server computer like the setup shown in Figure 5.1.

There, the camera is suspended from the overhead I-beam using a flexible clamping arm. With the specific computer and camera used during testing, a 720p video resolution was found to generate the best results; offering a good compromise between detection distance and latency. The camera lens should ideally be positioned between 1.75 and 1.85 meters from the water's surface to maximize its field of view without exceeding the detection range. This maximizes the effective area in which AprilTags and colored targets can be detected. The visible, and therefore detectable, region of water is referred to as the vessels' operating area. Using the Intel RealSense D435 camera at that distance, with a 94° diagonal field of view and a 16:9 aspect ratio, the operating area should have approximate dimensions of 2.6×1.5 m.

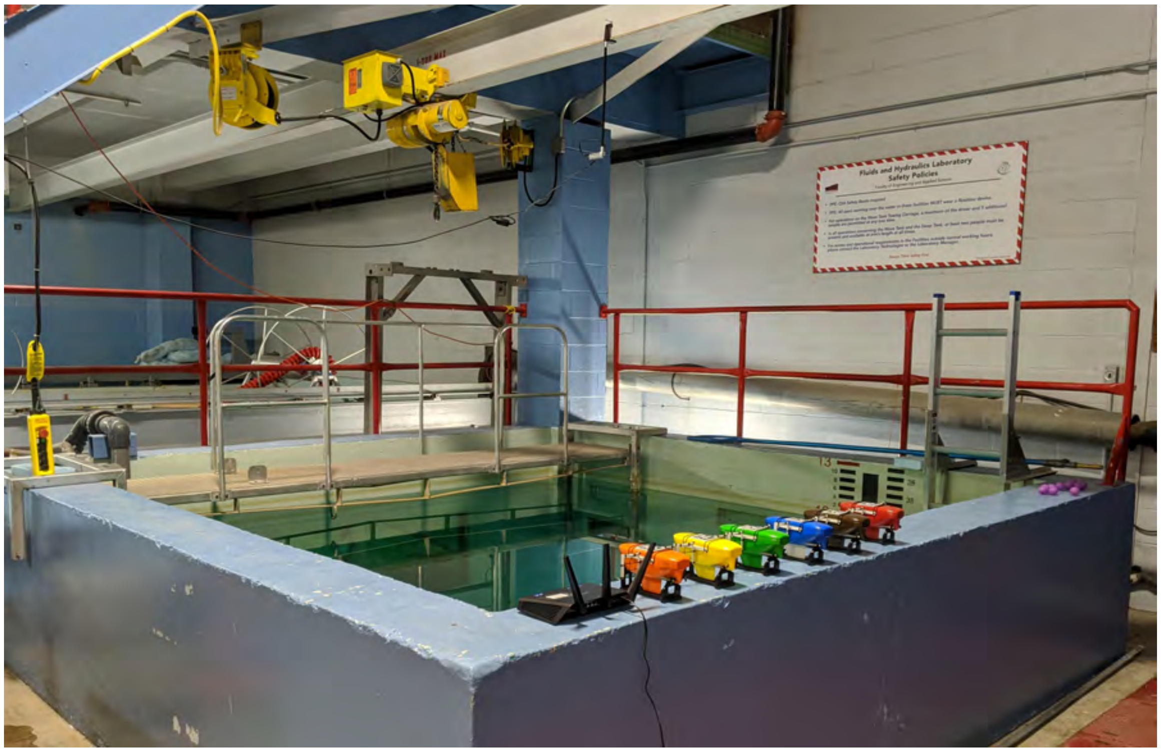


Figure 5.1: Laboratory Setup for microUSV Testing and Experiments

The camera must also be tuned to the lighting conditions of the environment. A brightly lit tank is beneficial for CVSensorSimulator (CVSS) detections but risks introducing surface reflections as discussed in Chapter 4. Since CVSS was developed to run on the Linux operating system, it is assumed that a user will have access to the v4l-utils package [11]. This package provides a terminal interface to directly modify the internal parameters of most types of digital cameras which can be connected to the server computer. The camera's exposure time should be reduced as much as is reasonable to reduce blurring and delay between frames. This will result in the camera producing much darker images. To compensate, parameters such as the camera's gain and gamma should be increased where available to increase the image brightness. The AprilTag algorithm [68] depends on the ability to detect clear edges in an image, so any automatic camera sharpness parameters should also be maximized. Finally, if CVSS is using color thresholding to simulate data for its virtual sensors, as is the case for the orbital retrieval algorithm discussed in Chapter 6, the camera's color saturation should also be maximized to increase the contrast between colored targets and the background.

The wireless router should be positioned near the tank, ideally with a clear line of sight to all vessels in the operating area. In the Figure 5.1 setup, the router was placed on top of the walls surrounding the tank to avoid being obstructed by them. The server computer and all microUSVs must be connected to the router's network on the same subnet. Configuring a static Internet Protocol (IP) address for the server computer is a suggested optional step. The IP address of each individual microUSV does not matter for system functionality; they can be dynamically assigned and changed between runs without consequence. Knowledge of the server computer's

IP address, however, is critical. The microUSVs need the server computer's IP address in order to connect to CVSS and receive sensor updates: each of them must know the address before communication can begin. The server IP address must be included in the config file of each vessel in the system.

5.2 Server and Vessel Software Configuration

The MUSVController application running onboard each microUSV begins its initialization process by reading values from a JavaScript Object Notation (JSON) configuration file. This file contains information necessary for the microUSV to function that may be unique to each vessel or may change between runs. The server computer's IP address is one such piece of information: each vessel's config file includes the IP address of the server computer which can be updated for different users or network setups without modifying the controller's source code. The microUSV's configuration file also includes controller and vessel-specific tuning parameters.

The vessel configuration file contains a variable used to select the type of controller the vessel will run as well as the tuning parameters for each type. For instance, the WaypointController discussed in Section 5.3 of this chapter is built on a pair of Proportional-Integral (PI) controllers. The gains for these controllers are stored in the vessel config file and can be easily edited during the tuning process without modifying the controller's source code.

Any parameters unique to a single vessel are also stored in its vessel configuration file to be retrieved during initialization. These include the vessel's AprilTag identification number, propeller spin direction flags, and motor bias. Each microUSV's

AprilTag must be unique to allow CVSS to properly identify it. Each vessel must therefore have its own unique tag identification number which it sends to CVSS in RequestData messages. The propeller spin direction flags are a pair of values, one for the port and one for the starboard motor, which indicate if the propellers need to spin clockwise or counterclockwise to drive the vessel forward. Since vessels are not guaranteed to have identical propeller configurations, this parameter must be unique to each vessel. Due to small imbalances in motor efficiency, friction, wear, or any number of other factors outside the control of software, a vessel's motor output speed is never guaranteed to be the same for both sides. Even when the same command is issued simultaneously to each motor, the port and starboard sides will deliver slightly different amounts of thrust, resulting in the vessel turning when a forward heading is requested. This bias in motor speed can be corrected in the control software by artificially reducing the output of one motor and increasing the other after a speed command is issued. The vessel configuration file's bias parameter controls how strong a correction to apply and must be unique to each vessel.

CVSS also has a JSON configuration file which it reads during its own initialization process. This file contains data unique to a specific set of hardware and variables which change frequently between experiment runs. The server configuration file includes the camera's calibration parameters which can be obtained using OpenCV [21] and a calibration target. It also contains a list of all microUSV's in the system and their associated AprilTag identification numbers. This list allows CVSS to filter out any spurious AprilTag detections not associated with a microUSV and avoid wasting time processing them. The HSV threshold values and list of waypoints are experiment-specific configuration file parameters which can easily be tuned between runs. The HSV threshold values define a color range to search for in each camera frame to detect colored targets. Including them as a tunable parameter makes changing the type of targets much simpler. Each test using the WaypointController may require a different set of waypoints. Including them in the configuration file allows them to be changed quickly without re-compiling CVSS.

5.3 Waypoint Following Experiment

Three test scenarios were used to validate the functionality of the microUSV system: a linear path following test, an elliptical path following test, and a multi-vehicle test. Each test case had the vessel(s) operate autonomously using a simple waypoint following algorithm based on PI controllers: WaypointController. The vessels were initialized with a list of waypoints which they steer toward sequentially, their PI controllers outputting motor speed commands which aim to minimize the vessel's distance-to-waypoint error and heading-angle error. This simple controller was chosen over more elaborate modern controllers because it was simple to implement and was familiar to most potential users, offering them a common frame of reference to compare against existing systems: It was not intended as an efficient or novel solution to the waypoint following problem.

5.3.1 WaypointController Algorithm

The **getSpeeds** function is called in a loop as long as the MUSVController application is running. The specifics of the **getSpeeds** function implementation depends on the type of controller object that was instantiated: for the WaypointController class, the algorithm is outlined in Algorithm 1. The algorithm starts by requesting a list of waypoint coordinates from CVSS by toggling the request_waypoints flag in its RequestData message to true. This flag is only toggled to true in a vessel's first message to CVSS. The controller proceeds to steer the vessel toward the first set of waypoint coordinates in the list. Once the vessel is within 50 mm of the waypoint, it has arrived within tolerance: The waypoint is removed from the head of the list and the vessel proceeds toward the next one until no waypoints remain.

Algorithm 1: WaypointController

```
input  : vessel pose, waypoints list
output: port motor speed, starboard motor speed
Function getSpeeds()
    if waypoints in list then
        calculate distance and heading angle errors
        apply distance PI gains $\Rightarrow$ calculate forward motor speeds
        apply angular PI gains $\Rightarrow$ calculate speed turn correction
        applyTurnCorrection()
        if vessel reached waypoint then
            remove waypoint from list
        end
        return motor speeds
    else
        return zeros
    end
Function applyTurnCorrection()
    port motor speed -= turn correction
    starboard motor speed += turn correction
```

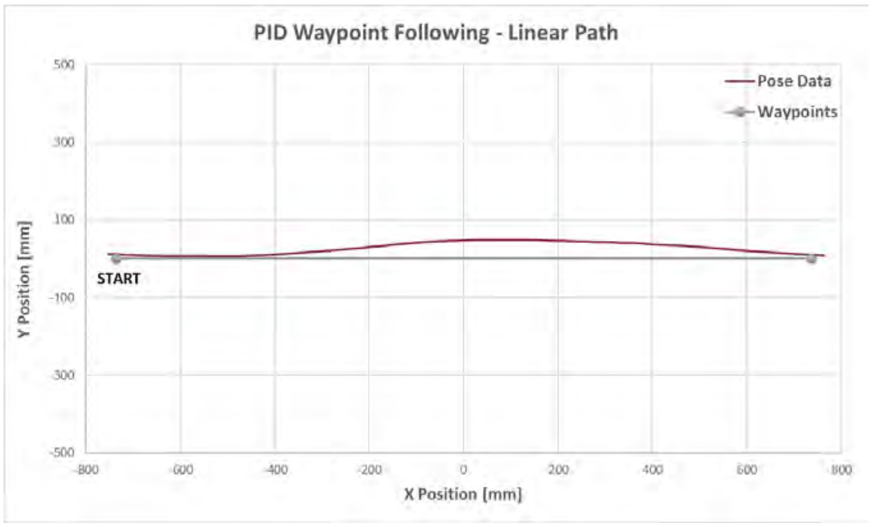


Figure 5.2: Linear Path Test Trajectory Plot

5.3.2 Linear Path Test

For the linear path following test, the microUSV was placed in a test tank and instructed to travel between two waypoints positioned 1.47 m apart along the camera's x-axis from left to right. The vessel's trajectory can be seen in Figure 5.2 along with the test's waypoints and the expected trajectory between them.

This test demonstrated the microUSV's ability to operate under its own power and follow a pre-defined path using onboard control logic, including fine error correction. The CVSensorSimulator system functioned as intended with an average update rate of 9.5 frames per second[1] which was an acceptable update rate for navigation purposes.

The vessel followed the expected trajectory very well with some small deviations. It reached the first waypoint with a small heading error and so when its waypoint

[1]The WaypointController tests were performed using an early software build and a different camera: a Logitech C920. This frame rate value does not reflect the update frequency observed in the final system configuration of 17 to 21 Hz as seen in Figure 4.5.

target was updated the vessel overcompensated and overshot the target heading before correcting itself to arrive on target. Better controller tuning may have improved this behavior, or it may simply be due to the limitations of this control scheme: since the controller only has access to the position of its next goal and not the subsequent ones, it does not know what heading angle will be needed after reaching its next waypoint. This creates a tendency to start the next leg of its path with a non-zero heading error. This pattern is also apparent in the elliptical path test.

Even so the vessel only deviated from the expected trajectory by a maximum of 49mm; a small margin for a 1.47 m long path. This error was calculated as the perpendicular distance from the vessel's position to the expected trajectory line.

5.3.3 Elliptical Path Test

The elliptical path following test used a setup identical to that of the linear path test but with a different set of waypoints. The vessel was instructed to follow an elliptical path, or more accurately, an octagonal path whose vertices intersect an ellipse with a major diameter of 1.36 m and minor diameter of 0.92 m. The vessel started at the left edge of the tank and proceeded through the waypoints counterclockwise. The vessel's trajectory can be seen in Figure 5.3 with the expected trajectory and waypoints shown as well.

This test demonstrated the microUSV's ability to handle more aggressive maneuvers: It can perform tight turns while keeping its AprilTag in view of the overhead camera.

This trajectory deviated from its expected trajectory much more than what was

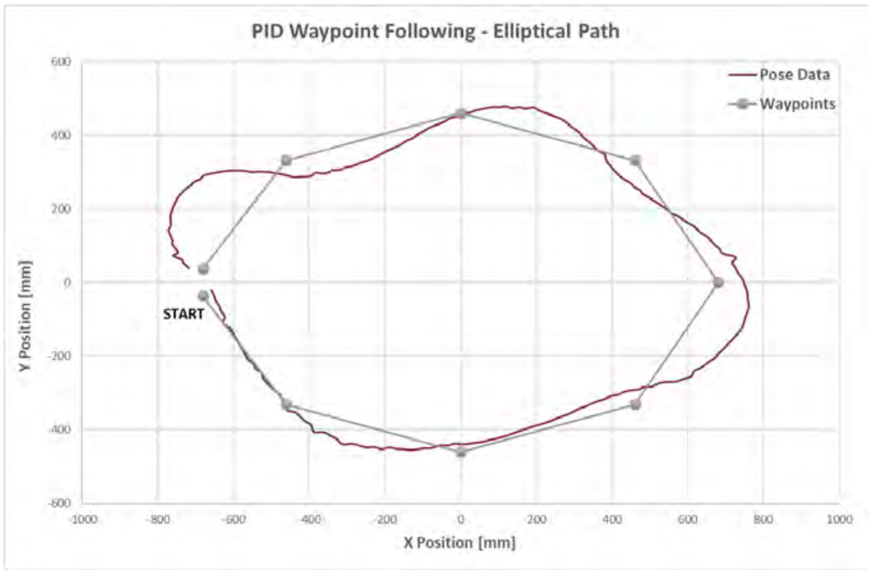


Figure 5.3: Elliptical Path Test Trajectory Plot

observed in the linear path test, but the vessel still reached each waypoint in sequence. Here the influence of sudden changes in heading error due to waypoint handoff is more obvious. While traveling between waypoints the vessel's only concern is reaching the next target, so its heading is directed at the next waypoint. Only once that waypoint is reached is the next waypoint considered with the vessel's current heading significantly off target. This leads to considerable overshoot in the vessel's trajectory, which is particularly noticeable on sharp turns like that observed at the fifth waypoint on the far right of Figure 5.3. The vessel still manages to achieve a reasonable approximation of the intended trajectory. Path following performance could be improved substantially by implementing a more robust control scheme such as the method described in [54].

The error, as seen in Figure 5.4, was calculated as the perpendicular distance from the vessel's position to the expected trajectory line. Due to the two-dimensional

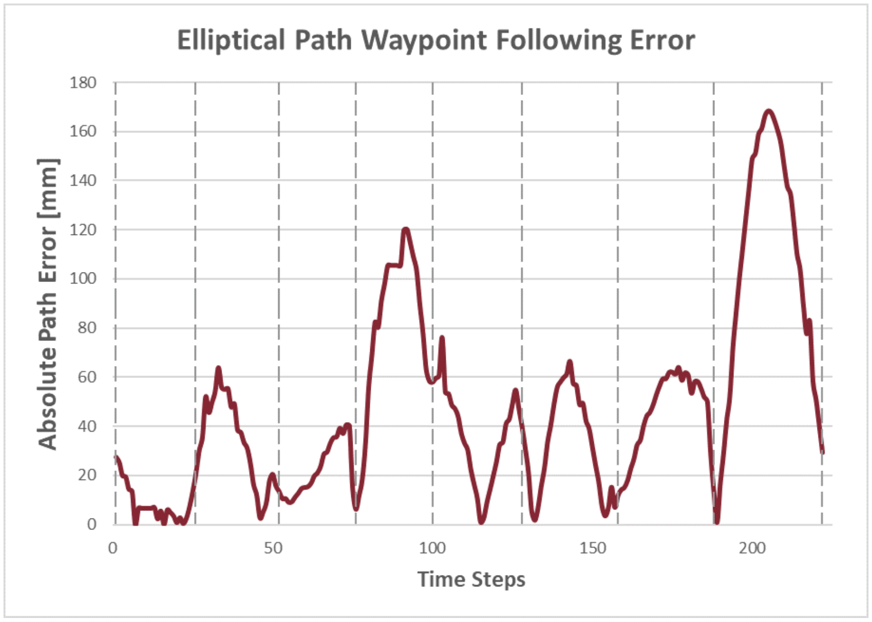


Figure 5.4: Elliptical Path Test Trajectory Error Plot

nature of the trajectory there were multiple expected trajectory line segments. The error calculation was therefore treated as a piecewise function, changing the expected trajectory line equation each time the vessel was considered to have reached its next waypoint. These transition points are denoted in Figure 5.4 by the vertical dotted lines. Note how the error tends to rise sharply after reaching each waypoint before decreasing as the vessel compensates for the sudden change in goal position.

5.3.4 Multi-Vehicle Test

The multi-vehicle test used a nearly identical trajectory to the elliptical path following test where each vessel is given the same set of eight waypoints arranged in a roughly elliptical shape to follow. Unlike the elliptical test, once the vessels reached the final waypoint in their trajectory, they were instructed to repeat that trajectory again ad

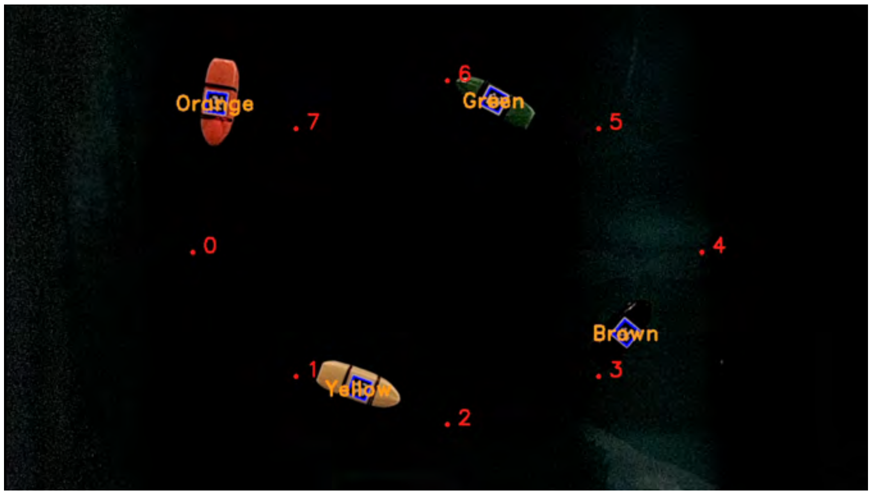


Figure 5.5: Multi-Vehicle Test Overhead Camera View

infinitum. Four vessels were launched sequentially from the same position, shown in Figure 5.5, with a roughly even spacing between their start times. Each vessel completed several laps of the trajectory with the first vessel successfully completing three laps of the trajectory while the fourth vessel had time for just over two laps in the 1-minute, 40-second-long test with the first and last vessel launching 28 seconds apart.

Five and six vessel configurations were also tested and were managed by CVSS without issue. Due to the limited space in the operating environment and the lack of a collision avoidance strategy in this control scheme, these tests quickly resulted in vessel collisions and pileups.

This test demonstrates the ability of CVSS to handle the simultaneous requests of multiple vehicles with enough speed to allow for real-time navigation of each. It also shows the microUSV's are sufficiently stable to survive the disturbances introduced by other vessels. The microUSVs appeared unperturbed by the wakes created by the other vessels in the tank and the multiple inter-vehicle collisions which occurred during testing. None of the collisions resulted in a vessel overturning or the overhead camera losing view of their AprilTags.

Chapter 6

Clustering Floating Marine Contaminants

This chapter discusses a novel method for gathering together dispersed surface contaminants in marine environments using a swarm of Unmanned Surface Vehicles (USV): the orbital retrieval algorithm. The proposed method could be used to aggregate contaminants such as floating plastics into clusters to assist cleanup efforts, expanding the effective range of a manned vessel offshore or automating debris collection in smaller semi-enclosed environments like harbors. The chapter includes a brief discussion of the impacts of marine plastics and existing approaches to object clustering and aggregation using swarms of robots. It then describes the orbital retrieval algorithm as well as the testing procedures and experimental results, performed on the microUSV platform, used to validate it. At the time of writing there are no known existing studies which investigate this topic in the marine domain.

6.1 Marine Plastics

Man-made pollution in marine environments is detrimental to life both above and below the water's surface. The damage caused by anthropogenic contaminants, particularly plastics and other petroleum-based products, being dumped into rivers and oceans is not limited to a population or ecosystem localized at the dumping site, but to all oceans and coastal regions on Earth. While large macroplastic debris, pieces over 5 mm in length, are most concentrated in coastal regions near dumping sites, ocean currents and other transport dynamics will carry contaminants thousands of kilometers away to affect the farthest reaches of the oceans as it fragments over time and is consumed by marine organisms [82]. Microplastics, plastic particles under 5 mm in length, have been demonstrated to work their way up a food chain, jumping between organisms as the particles are consumed and re-consumed through natural feeding and predation [52, 80]. They cause harm to organisms of all sizes, from zooplankton to fish to humans: they are known to obstruct digestive tracts and leach dangerous chemicals absorbed by the particles into the organisms [36, 75].

Marine plastics have a negative economic impact as well. From tourism and shipping to fisheries and aquaculture, multiple disparate marine industries suffer from the presence of plastics and other pollutants. Those working in such industries end up responsible for the repair and cleanup costs caused by the abundance of marine plastics rather than the producers (or consumers) of plastic products [66]. Most people will never see the build up of anthropogenic pollutants first hand and so the public in general is content to dismiss or forget the problem's existence while it is out of sight [65]. These petroleum-based contaminants, however, are non-biodegradable and can linger in the environment for centuries, cycling through an ecosystem over and over before finally decaying [18]. Waiting for these problems to resolve themselves is not an effective solution; an active approach is required.

Removing plastics from marine environments is a complex and costly endeavor. A 2014 study by Eriksen et al. [33] estimated the amount of floating plastic in the oceans at 268,940 tons and growing. If humanity were to stop dumping plastic waste into marine environments immediately the amount of material remaining in the oceans would still represent a monumental effort to redress. The optimization of marine cleanup methods is, therefore, paramount. Ship-based solutions are the most frequently proposed for this task, typically by towing a dragnet through or around large patches of contaminants [87]. Ships carry a high operating cost due to their fuel consumption and crewing needs. They also offer limited mobility and are best suited to managing large patches of pollutants offshore. The floating barrier concept implemented by the Ocean Cleanup Project [4] has shown some positive preliminary results but is only effective at large scales and when deployed at specific locations, namely the ocean gyres. It is not a suitable solution for cases such as capturing contaminants near shore in harbors and estuaries without heavily disturbing those environments. An alternative solution for such an environment is the use of marine robots such as USVs.

USVs have already shown success in acting as a force multiplier to manned missions performing other tasks such as bathymetric surveying [12]. They should prove to be a valuable tool for marine cleanup operations as well. Deploying a swarm of cooperative USVs to work in tandem with a manned mother ship could greatly improve the range and effectiveness of a localized offshore cleanup operation while minimizing the time required of and risk to human operators. USVs also offer a more nuanced solution for routine maintenance and cleanup of smaller regions like harbors and estuaries when paired with a passive contaminant collection tool like the Seabin [9]: they can operate in shallow waters while avoiding nearby ship traffic autonomously.

At the time of writing, USV-based plastic cleanup operations must be restricted to macroplastics due to the limited ability of available sensor technologies. Modern detection techniques for micro and nano plastic particles in aquatic environments requires the use of mass-spectrometry equipment [50]. These sensors cannot produce the real-time feedback or high sampling frequencies required to control a robot. Until such a real-time sensor exists, applications in this area are constrained to managing the detectable macroplastic debris.

6.2 Overview of Swarm Clustering and Foraging

Clustering and foraging tasks in the context of swarm robotics research are very similar: both seek to gather together objects initially dispersed throughout the environment using a swarm of simple agents [20]. These objects could be simulated food particles, victims in a search and rescue scenario, or pieces of waste to be gathered and removed, among many others. These methods also typically share several basic assumptions about their agents such as their ability to pick up and drop objects, a limited detection range, short term memory, some form of inter-agent communication, and often limited localization capabilities. The details of clustering and foraging methods vary as they are trying to achieve subtly different goals but both approaches are worth evaluating as inspiration for a swarm-based marine environmental cleanup method.

Swarm clustering involves robots placing objects near other objects to form clusters of objects. The classical definition of a swarm clustering task does not require a set gathering location; most existing approaches to this problem take a probabilistic approach where the probability of an agent picking up or dropping an object under different conditions determines the swarm's clustering behavior as the agents explore their environment randomly [42]. These solutions will typically form several smaller clusters at random locations which tend to merge into a single cluster as time progresses, also in a random location [84, 88]. Such an approach would be suitable for marine cleanup operations assuming the use of a mobile mother ship which could move to the location of the final cluster to extract the contaminants.

Foraging tasks are very similar to clustering but differ in that foraging assumes a predetermined location for the cluster. These algorithms seek to replicate the foraging behavior of ants and other social animals. Foraging agents explore an environment and gather targeted objects. Unlike clustering agents, foragers seek to form clusters specifically at a central nest location [20]. In the case of marine cleanup operations, the nest location would be a static contaminant collection point or tool such as a Seabin. Foraging methods rely heavily on the ability of agents to leave and return to their nest: they must have some form of localization. Studies in this area frequently assume foraging agents have very limited localization capability so they have developed many creative solutions such as stigmergy [85] and pheromones [43]. Most of these solutions are impractical to implement in a marine environment and are largely unnecessary: all known USV platforms are equipped with some means of localization such as Global Positioning System (GPS). Vessels in a swarm-based marine cleanup operation can be assumed to have localization capabilities, rendering most existing foraging solutions at least partially redundant.

There are no known swarm object clustering or foraging implementations on water. A swarm, or multi-agent, approach has been occasionally proposed for marine cleanup before, mostly focused on addressing oil spills, and evaluated in simulation [47, 35, 96] but these solutions have never been implemented using hardware. The collection of a liquid contaminant like oil poses different challenges than collecting solid contaminants such as plastic.

Marine contaminants, be they oil, plastic, or otherwise, have a tendency to drift and disperse due to the motion of waves and currents. A swarm performing marine contaminant clustering must conduct cluster maintenance constantly in addition to the exploration and retrieval tasks of cluster formation to avoid losing progress to entropy. Additionally gripper mechanisms often used in terrestrial clustering and foraging solutions are not suitable for gathering marine plastics as their size and shape is not consistent enough for a single gripper design. Skimmer contaminant capture mechanisms such as the solution discussed in [96] are a common alternative. These tools rely on a vessel's forward momentum to keep any contaminant it captures contained and so marine clustering agents cannot maneuver as freely as terrestrial agents.

With these constraints in mind, the orbital retrieval algorithm, discussed in subsequent sections, was predominantly inspired by the work of Gauci et al. [37] and Vardy [89]. These solutions utilize very simple, forward motion only, controllers to gather dispersed objects. The agents circle, or orbit, a fixed cluster area and push objects they find inward. Instructing all agents to orbit in the same direction minimizes inter-agent interference, simplifying or eliminating the need for collision avoidance strategies. This orbiting behavior is particularly interesting for marine contaminant clustering as any dispersing contaminants will be recaptured by the orbiting agents without the need for an explicit cluster maintenance instruction.

6.3 Orbital Retrieval Algorithm

The orbital retrieval algorithm was designed to form and maintain a cluster of floating target objects at a designated position using a swarm of USVs. Conceptually, the target objects represent pieces or patches of marine plastic or some equivalent floating solid contaminant but for development and testing purposes, a target is an individual object the algorithm is seeking. The vessels are assumed to have an onboard passive capture device such as a forward facing net or skimmer. This device must be able to capture a target object when the vessel moves toward and intercepts it, retain the target by moving forward continuously, and release a captured target by moving in reverse. In the case of testing using the microUSV, the vessels were equipped with side mounted nets which capture targets to either side as shown in Figure 3.8. Targets struck by the vessel's bow will roll to one side and also be captured by one of the nets.

Much like the methods discussed in [37] and [89] which inspired it, the orbital retrieval algorithm is a purely reactive controller: the controller does not perform mapping, path planning, or coordination with other vessels. When it receives a sensor input it simply reacts, producing a single output. This type of controller is ideal for swarm robotics applications as it can be implemented successfully on very simple and cheap hardware. This approach requires minimal processing, simple inter-vessel communication, and three sensors.

The first essential sensor is a means of global localization. As discussed above, all known USV platforms are equipped with a localization system such as GPS so this was an assumed to be an easy requirement for a platform implementing the algorithm to meet. This also allows a cluster point to be predefined using the localization system's global coordinate frame and provided to each vessel in the swarm on initialization. For the microUSV implementation of orbital retrieval, the role of localization sensor is filled by CVSensorSimulator (CVSS) pose estimates, simulating GPS data. The second essential sensor is a ranged target sensor: vessels must be capable of detecting nearby targets with coarse resolution. The final essential sensor is a capture sensor to indicate when the vessel has successfully captured a target. The sensor layout for the microUSV implementation can be seen in Figure 4.8. The capture sensor zone is positioned just ahead of the vessel's side-mounted nets to detect when a target rests in one or both of them. The target sensor zone is a roughly circular arc aimed toward the vessel's forward-port side. The zone is subdivided into six sub-arcs to simulate a course resolution sensor reading. This sensor allows a vessel to detect a target within the designated sensor zone and its relative heading within 10° but not the distance between vessel and target. Since the vessels will be orbiting exclusively in a clockwise direction, this sensor does not need to extend to the vessel's starboard side. The target sensor zone was, therefore, cut off at the centerline to reduce processing time.

The algorithm itself is outlined in Algorithm 2 and visualized in Figure 6.1. As with the WaypointController algorithm discussed in Chapter 5, the controller is implemented inside a **getSpeeds** function which is called in a loop, taking sensor data as an input and returning a pair of integer motor speeds. The controller processes the sensor data to deduce its current state, then reacts accordingly.

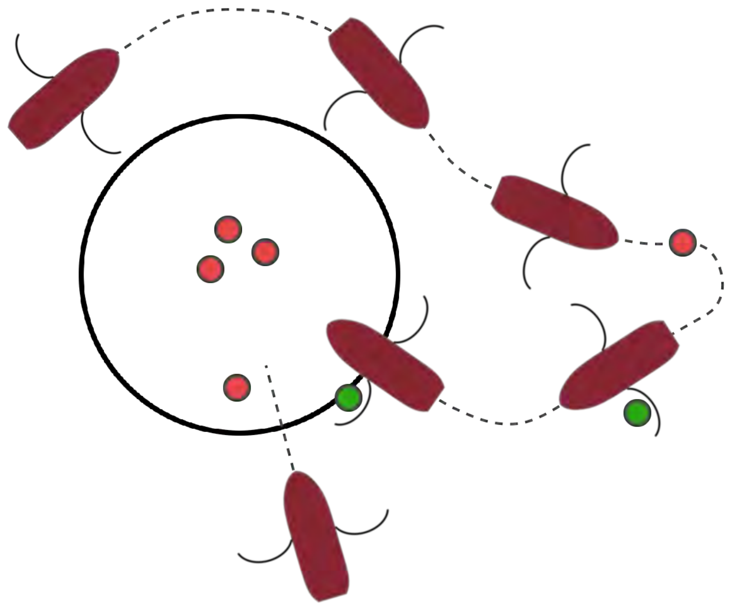


Figure 6.1: Steps of the Orbital Retrieval Algorithm - Retrieval Maneuver

6.3.1 Instruction Hierarchy

The controller performs one of several potential actions in descending order of priority. The highest priority action is to add a captured target to the cluster. If the vessel has captured a target, and the capture sensor is triggered, the vessel will attempt to drive toward the center point of the designated cluster area. The cluster area is a circular region defined by its origin coordinates and a **cluster threshold** radius, shown in Figure 6.1. Targets inside this region are considered part of the cluster. When a vessel carrying a captured target passes the **cluster threshold**, it moves

Algorithm 2: Orbital Retrieval

```
input  : vessel pose, cluster area coordinates, colored target sensor data,
         inter-vessel communication data
output: port motor speed, starboard motor speed
Function getSpeeds()
    if carrying captured target then
        if outside cluster threshold then
            // drive toward cluster area center
            find vector from vessel to cluster area center // Eqn. 6.1 and 6.2
            find heading angle error
            set distance error to 1.5 * (standard look ahead point distance)
            apply PI gains ⇒ calculate motor speeds
        else
            set motor speeds to full reverse
            reject new controller inputs for the next 1.5 seconds
        end
    else if other vessel ahead then
        // veer left and decelerate
        port motor speed = 66% of full reverse
        starboard motor speed = 0
    else if target detected ahead or to port then
        // veer left toward target
        set heading angle error to the sensor zone sub-arc detecting target(s)
        set distance error to standard look ahead point distance
        apply PI gains ⇒ calculate motor speeds
        // The vessel will often overshoot, losing sight of the target. It
        // will then veer back to the right, regaining sight of the target.
        // This pattern results in the vessel oscillating left and right,
        // following a roughly straight path toward the target.
    else
        motor speeds = orbitCW()
    end
    return motor speeds
```

in reverse at full speed for a fixed period of time. The target's forward momentum continues to carry it forward into the cluster and the vessel leaves the cluster area, returning to its orbiting behavior in search of targets.

If a vessel is not carrying a captured target, its next priority is to avoid collisions. The orbital retrieval algorithm includes a very simple collision avoidance strategy: if there are any other vessels within collision range and they are positioned in the path of this vessel then run the port propeller in reverse. This will result in the avoiding vessel decelerating and turning slightly to port. This simple strategy handles most cases of vessel congestion since all vessels should be orbiting in the same clockwise direction and the obstructing vessel should be moving away from the avoiding vessel. In the case of prolonged obstruction, the avoiding vessel will end up oscillating back and forth between its collision avoidance and standard orbiting behavior: veering to port will eventually lead to the obstructing vessel no longer being ahead of the avoiding vessel, causing it to change behaviors and begin veering back to starboard where it will again see the obstructing vessel and try to veer to port to avoid it, staying roughly in place until the obstructing vessel moves clear.

The detection of nearby vessels is handled in CVSS by simulating short range inter-vessel communication. If two vessels are within a given distance of each other, they are considered within communication range and can share their pose data with each other. This allows each vessel to calculate the relative distance and heading between them. If one vessel is being obstructed by the other, its collision avoidance behavior is triggered.

The next priority in the vessel's instruction hierarchy is to seek any targets outside the cluster it can detect. If a vessel's target sensor reports a detection, that vessel will steer towards the target's last known relative heading and accelerate to avoid obstructing any vessels behind it. Since the target sensor zone can only perceive the vessel's port side, this behavior will also result in controller state oscillations: If a target is detected the vessel will veer to port towards it, likely overshooting and losing sight of the target. At this point the default orbiting behavior takes over, turning the vessel to starboard and subsequently reacquiring detection of the target. This oscillation results in a the vessel taking a roughly straight path toward the target which will eventually be captured triggering the controller to return it to the cluster. The combination of a vessel leaving orbit, capturing a target, and returning it to the cluster is called a retrieval maneuver. Figure 6.1 illustrates all the steps of a retrieval maneuver with red circles indicating uncaptured targets and green indicating captured targets.

The lowest priority instruction for each vessel is to orbit the cluster area in a clockwise direction. This default behavior is critical for two reasons: First, completing a clockwise orbit will result in a vessel sweeping its port-facing target sensor through the broadest area outside the cluster area. This behavior effectively performs a sensor sweep of the area surrounding the cluster with each orbit. Secondly, since the targets are floating on the surface of the water, they tend to disperse over time, leaving the cluster area. If a target leaves the cluster area, there is a high probability that it will cross the path of an orbiting vessel which will detect it and return it to the cluster. This behavior maintains the integrity of the cluster of existing captured objects while other members of the swarm can periodically leave orbit to acquire new targets. This orbiting behavior is outlined in Algorithm 3.

The orbit algorithm is based on the same PI controller waypoint seeking algorithm

Algorithm 3: Orbit Clockwise

```
input  : vessel pose, cluster area center coordinates, orbit threshold
output: port motor speed, starboard motor speed
Function orbitCW()
    if outside orbit threshold then
        // calculate heading angle of vector tangent to orbit circle
        find vector from vessel to cluster area center // Eqn. 6.1 and 6.2
        find tangent orbit vector direction // Eqn. 6.3 and 6.4
        place new look ahead point along tangent orbit vector
    else
        use look ahead point from previous timestep
    end
    // navigate toward look ahead point
    find distance and heading angle errors
    apply PI gains ⇒ calculate motor speeds
    return motor speeds
```

as the WaypointController discussed in Chapter 5. Instead of a fixed list of waypoints, however, the algorithm calculates its own waypoint, or **look ahead point**, at each timestep. The **look ahead point** is a point located a fixed distance from the vessel along a vector tangent to the orbit circle. Recalculating this tangent vector and **look ahead point** each timestep results in the vessel following an approximately circular orbit path around the cluster area at constant speed. Once the **look ahead point** has been calculated, the algorithm applies the same PI controller logic to derive motor output speeds from the vessel's distance and heading angle errors.

If the vessel is inside the specified range from the cluster area origin designated as the orbit threshold, the geometry used to calculate the new **look ahead point** breaks down: it is impossible to derive a vector tangent to a circle from a point inside that circle. When this occurs, the vessel simply skips calculating a new **look ahead point** at that timestep and uses the **look ahead point** calculated prior, continuing toward the last known tangent point outside the orbit circle.

Figure 6.2 shows the geometry the tangent vector calculation is derived from[1]. The absolute heading angle from the vessel's position to the vector tangent to the orbit circle which will result in a clockwise orbit direction can be calculated as shown in equations 6.1 through 6.4:

$$D_{ctr} = \sqrt{\Delta x^2 + \Delta y^2} \tag{6.1}$$

$$\theta = \arctan_2(\Delta x, -\Delta y) \tag{6.2}$$

$$\phi = \arcsin(R/D_{ctr}) \tag{6.3}$$

$$\text{clockwise tangent orbit heading} = \theta - \phi \tag{6.4}$$

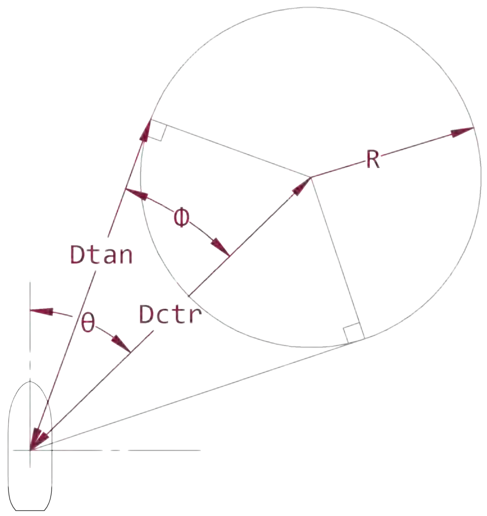


Figure 6.2: Orbit Tangent Vector Calculation Geometry

[1]Note that the CVSS coordinate frame defines positive angles as clockwise relative to the negative y axis as shown in Figure 4.6. This may make the subsequent trigonometry calculations appear incorrect if angles were assumed as defined counterclockwise relative to the positive x axis.

The vessel's current heading angle can then be subtracted from this absolute heading angle to find the heading angle error; the relative heading angle change the vessel must make to follow the tangent vector direction.

6.4 Algorithm Testing

The orbital retrieval algorithm was tested using the microUSV platform in the Memorial University deep tank, shown in Figure 5.1. Six vessels were available for testing, all equipped with side-mounted nets. Each vessel 3D printed in a unique color to make it easier for the user to distinguish them. Magenta ping-pong balls were used as the targeted objects the vessels were intended to gather representing marine plastic debris. The color of the ping-pong ball targets was chosen to maximize the color separation between the targets and vessels when performing HSV color thresholding in CVSS. Magenta was chosen as the color with the greatest separation in the HSV color space from the vessels' colors of orange, yellow, green, blue, brown, and red. This color distinction in CVSS can be seen in Figure 4.7 where only the targets, not the vessels, are outlined with magenta circles. The targets were also injected with water, filling approximately half their internal volume, to ballast them. This makes them rest lower in the water with their center of gravity at or below the waterline, similar to marine plastic debris. It also makes them harder to move accidentally, either by unintentionally adhering to a vessel and being dragged along its path or being pushed around by currents produced by a vessel's thrusters.

These experiments were meant to test the orbital retrieval algorithm's effectiveness at collecting floating contaminants. This was evaluated using two metrics of performance: the average distance between all visible targets and the center of the designated cluster area, or average target distance, as well as the number of targets detected in the cluster area, or cluster size. An ideal solution should minimize the average target distance while maximizing the cluster size. A low average target distance is an indicator not only of a dense cluster but that there are few outliers remaining in view outside the cluster area. A large cluster size indicates that the vessels were able to capture and retrieve the majority of contaminants in the operating area. CVSS was able to detect the positions of all objects in view of the overhead camera, both vessels and targets, for the duration of each experiment run. From this data, it calculated and recorded the raw performance metric data for each time step of an experiment run. It also recorded each vessel's pose history; a vessel's 2D pose at each time step during the run.

The orbital retrieval experiment was conducted using different numbers of vessels and targets deployed during each test run. Each run deployed a number of targets ranging from five to 40 in multiples of five (5, 10, 15, etc.) and between zero and six microUSVs. Testing each permutation of these two variables resulted in 56 total test cases. Test cases with zero vessels deployed were included to serve as a reference for performance. Contaminants floating in water have a tendency to disperse over time. The zero-vessel test cases provided data on how quickly this dispersion occurs for the targets used. The vessels' controller parameters were kept identical for each test case: The controller gains, the cluster area position and size, and the orbit threshold were all constant. The only variables being evaluated were the number of vessels and the number of targets deployed at the start of each test run.

It was hypothesized that any number of vessels running the orbital retrieval algorithm would produce better results than the zero-vessel test cases and that this performance would improve as more vessels were added. This trend of adding vessels improving performance was expected to reverse with the addition of the fifth or sixth vessel as inter-vessel interference was likely to begin impeding performance due to the limited size of the operating area.

6.4.1 Experimental Methods

Each test run began with an empty tank. The water was given time, typically two to three minutes, to become still between each run to avoid introducing currents or waves from previous runs as additional variables. Each test started with the launch of CVSS which would begin recording data from an empty tank with a circular cluster area defined at the center of the camera's field of view with a radius of 250 px. Next the targets for that run would be deployed. The appropriate number of targets for the run would be placed in a swimming pool net on the end of a long pole. This net was then held over the tank, approximately 80 mm above the surface of the water, and aligned with the center of the cluster area. The net was then quickly rotated 180° to face the surface of the water, dropping all targets simultaneously, before being removed. The targets would all enter the water at approximately the same time, in approximately the same place, and begin to disperse, spreading out toward the edges of the operating area. This dispersion was allowed to continue uninterrupted for ten seconds to produce a random distribution of targets throughout the environment. After the dispersion period, the first vessel would be launched from the left side of the tank: the vessel was placed in the water in view of the overhead camera and released, allowing the onboard controller to take over. In test cases with multiple vessels, the next vessel would then be launched from the same place after a five second delay. This process was repeated until all vessels had been launched. Each test run concluded three minutes after the launch of the first vessel: The vessels and targets were retrieved, CVSS was reset, and the water was allowed to settle before the next test began.

This testing procedure is slow; each run could take 15 to 20 minutes not including any extra time required to organize the data recorded by CVSS or change the batteries on one or more vessels between runs. Due to limited time and facility availability, performing replications of all test cases was not feasible. Each of the 56 test cases was run only once. To demonstrate the repeatability of the experiment's results a single test case was chosen and ran multiple times; specifically the test case deploying three vessels and ten targets. This repeatability test case was run seven times in total and the results from each run compared.

CVSS calculates the performance metric data for each run based on the data acquired by applying color thresholding to the camera feed. It uses OpenCV's [21] connected components functions to estimate the number and position of colored targets in each frame. This is a fast, reasonably accurate, but noisy means of counting the number of visible targets at each time step: it frequently underestimates the number of targets in view of the camera as several targets very close to each other may appear as a single blob to CVSS and only be counted once. It also provides a good estimate of the average target distance. Using the raw thresholded pixel data, however, provides a better average target distance estimate.

Instead of using connected components to calculate pixel blobs and estimating the average target distance from the blob centroids, the average target distance can instead be calculated by averaging the distance of each pixel belonging to a target. Evaluating the average distance on a pixel level bypasses the issue of multiple targets being classified as a single blob in CVSS, reducing their impact on the calculation of average distance. It also is better able to track targets near the periphery of the operating area where a target mostly out of view would not be counted by the connected components method as its blob would not be large enough to be counted. These factors result in the average target pixel distance producing a marginally better estimate than the average target distance with less noise. Both the average target distance calculated using connected components and the average target pixel distance are shown in Figure 6.3 for comparison.

The two average distance data sets in Figure 6.3 were both calculated for the same test run; zero vessels and 15 targets deployed. As expected, the targets disperse over time, beginning the test an average distance of 75 pixels from the cluster area center and ending over 400 pixels away after two minutes. The two measurement methods agree strongly with a correlation coefficient of 0.985 between them. The average target distance data, however, appears slightly more spread out. This was confirmed by fitting a fifth order polynomial to each data set and estimating the R^2 for each curve. The average target pixel distance data had a marginally better R^2 value of 0.987 compared to the R^2 of 0.976 of the average target distance curve. This confirmed that the pixel level measurement resulted in less noisy data. The average target pixel distance was therefore used as a performance metric instead of the average target distance in all other tests.

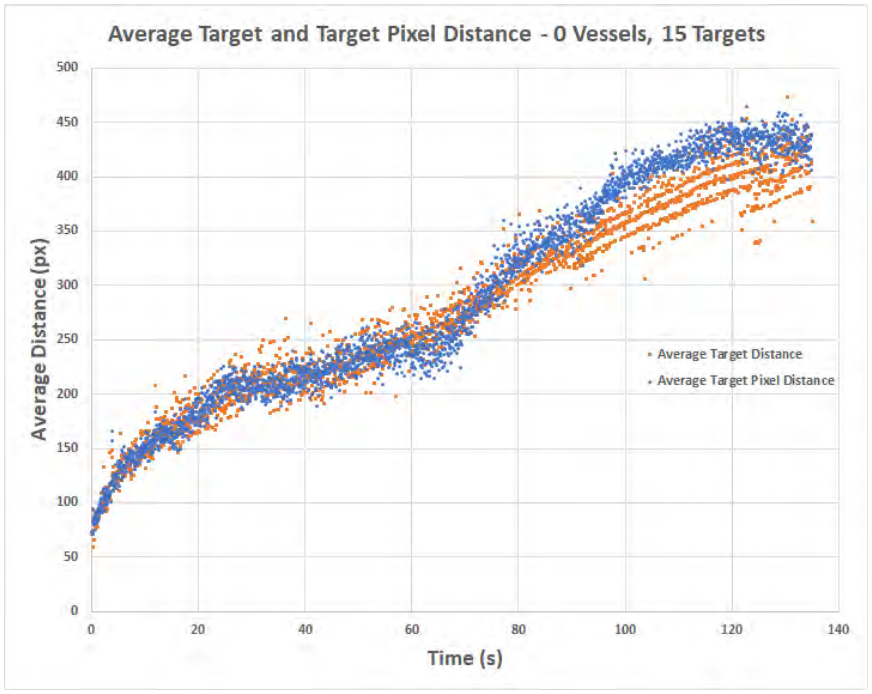


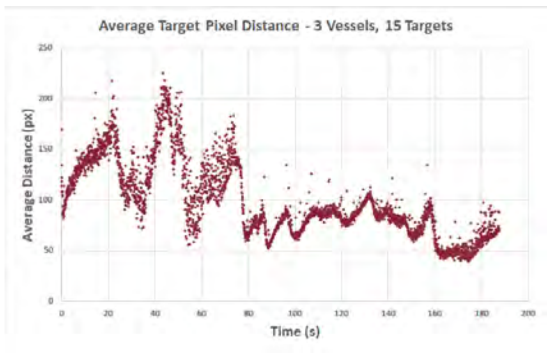
Figure 6.3: Comparison Plot Between the Measured Average Target Distance and the Average Target Pixel Distance from the 0 Vessels - 15 Targets Test Run

6.4.2 Results

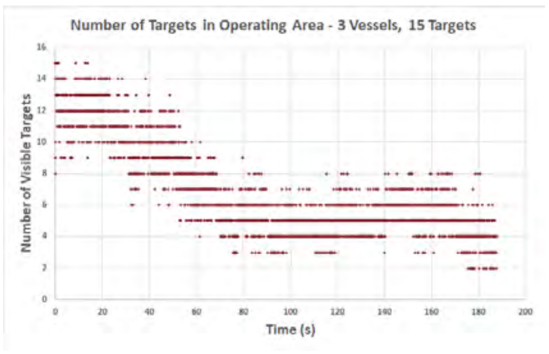
6.4.2.1 Single Test Performance

Figure 6.4 shows both performance metrics plotted over time for an example test case, specifically the case with three vessels and 15 targets. The trends shown in these plots reflect the general behavior observed across all test cases.

The average target pixel distance starts low and begins rising as the targets disperse, shown as a positive slope on the graph from zero to twenty seconds. This behavior is identical to what was seen in the zero vessel, 15 target test case. The dispersion is cut short at 20 seconds when the first vessel captures a target and starts pushing it toward the cluster area, resulting in a sharp negative slope in the graph as the captured target approaches the cluster. The uncaptured targets continue dispersing during this time, which can be observed at the end of the downward sloping section when the captured target is released in the cluster and the average target pixel distance stops decreasing. Several subsequent downward spikes coincide with the three vessels retrieving other targets near the cluster area and working their way outward to those on the periphery. The cluster stabilizes around 80 seconds when all targets still in the operating area have been added to the cluster. These clustered targets still tend to disperse over time but are recaptured by the orbiting vessels, shown as a sequence of low amplitude spikes in the distance plot from 80 to 180 seconds. This agrees with the expected result: the addition of three vessels greatly reduced the average distance targets moved from the center point of the cluster compared to the zero vessel, 15 target case. The cluster size results, however, do not agree with expectations.



(a) Average Target Pixel Distance vs Time



(b) Cluster Size vs Time

Figure 6.4: Performance Metric Data Plots from the 3 Vessels - 15 Targets Test Run

The cluster size plot shows that the number of targets visible to CVSS decreases steadily during the first 80 seconds of the test then stabilizes with an estimated 4.8 targets in the final cluster[2]. This is lower than expected: The three vessel swarm kept fewer targets inside the operating area than the zero vessel, 15 target control test which ended with a CVSS estimate of 6.7 targets in the operating area. Some frequent vessel behavior observed during the test accounts for this discrepancy: while performing their initial retrieval maneuvers, vessels were observed to force targets other than the one they had captured away from the cluster point. Unlike in terrestrial clustering applications, the floating targets are free to move and drift, requiring very little interference to be forced away from the goal. Vessels would often bump into targets or aim their thrusters outward as they turned toward the cluster area, catching targets in their wake and pushing them out of the operating area. This resulted in

[2]Note that the cluster size estimate is based on noisy connected components data. The noise in this data is heavily biased towards underestimating the number of visible targets. The CVSS estimate for cluster size is used instead of manual observations for consistency across test cases but for this test seven targets were counted in the final cluster; still fewer than expected.

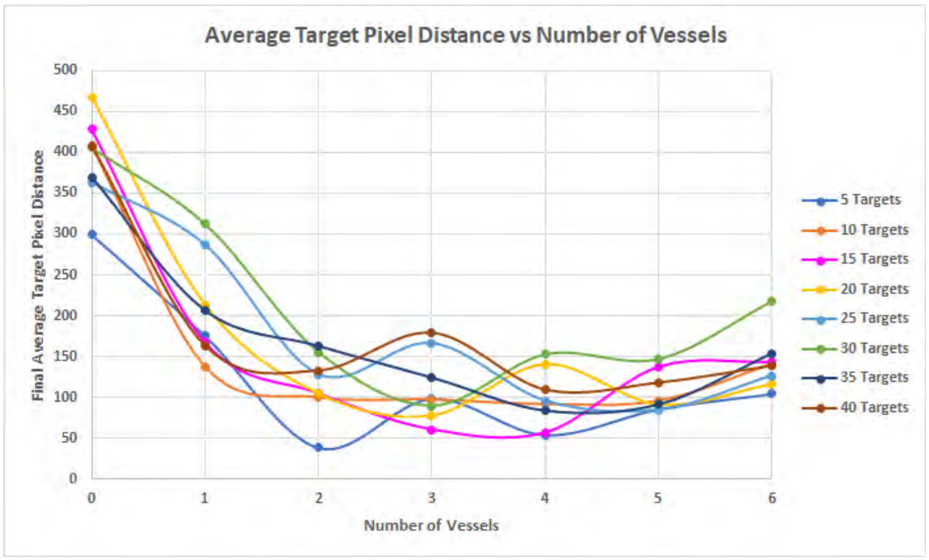
the count of visible targets being consistently lower when any number of vessels were introduced to the system compared to the control test, particularly when the number of deployed targets was high. This behavior was unexpected but consistent across all test cases.

Both average target pixel distance and cluster size results were reinforced during the repeatability test. The three vessel, ten target test was run seven times producing an average final target pixel distance of 97.17 pixels with a standard deviation of 22.71 pixels and an average final cluster size of 5.38 targets with a standard deviation of 1.28 targets. The consistency across replications is encouraging for the validity of all test case results.

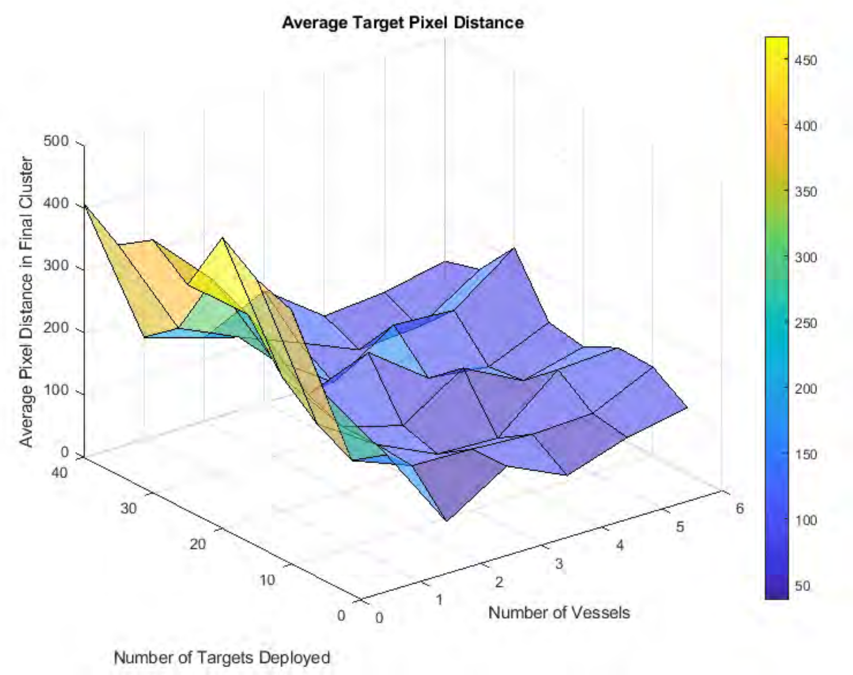
6.4.2.2 Aggregate Experimental Performance

Figure 6.5 shows two different visual representations of the same data: a line graph and a surface plot of the average target pixel distance aggregated from the end of all 56 test cases. The trend is particularly obvious in the line graph: regardless of the number of targets deployed, the average target pixel distance improved with the addition of one or two vessels but did not benefit substantially from the third or fourth where the curve level off. The addition of the fifth and sixth vessel introduced some inter-vessel interference and so there the distance curve begins to slowly rise again but remains much lower than the zero vessel control cases.

The cluster size aggregate data in Figure 6.6, similarly plotted in two variations, was very noisy, often performing worse than the corresponding zero vessel control test case. Aside from a minor inconsistent jump in performance in two vessel tests, the addition of extra vessels appears to have made little to no impact on the size



(a) Aggregate Data Line Graph



(b) Aggregate Data Surface Plot

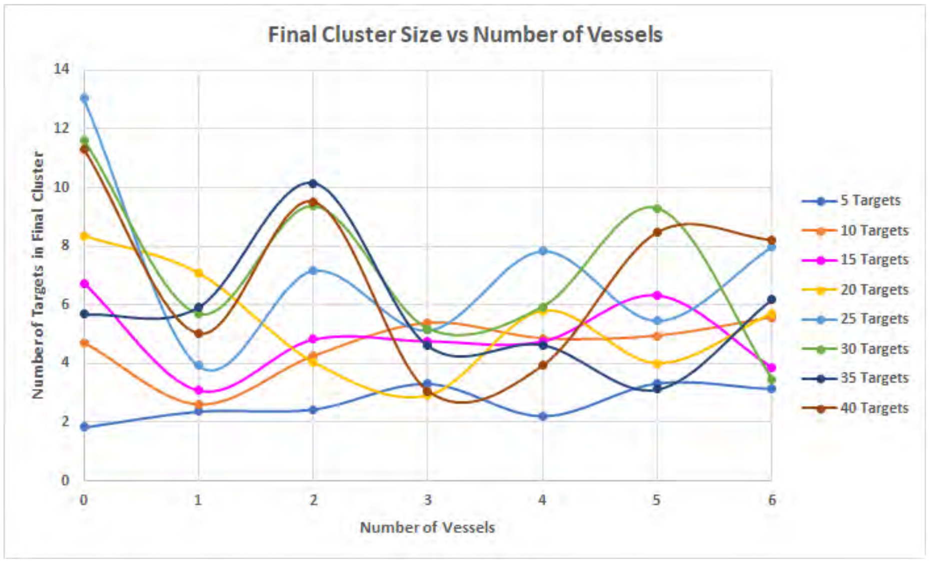
Figure 6.5: Average Target Pixel Distance Aggregate Data Plots

of the final cluster. In fact the final cluster size appears to be very similar in all test cases, about 5.2 targets, regardless of how many targets were initially deployed. This is most visible in the Figure 6.6 surface plot where the resulting surface appears mostly flat along the “Number of Targets Deployed” axis[3]. It was theorized that this poor performance may have been caused by the spatial limitations of the microUSV experimental setup.

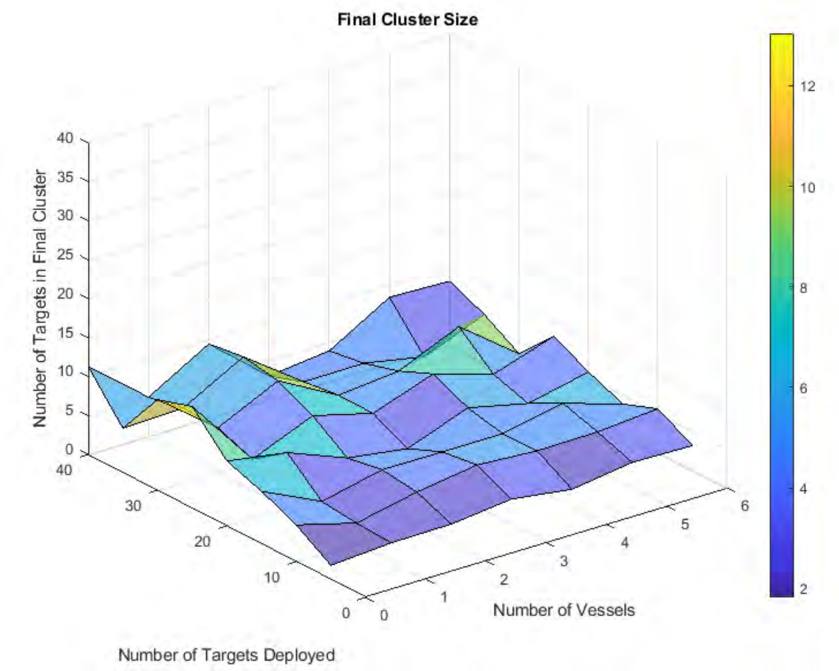
The size of the microUSV operating area is limited by the overhead camera’s field of view. For these experiments, the camera was positioned approximately 1.85 m from the surface of the water outputting a 720p video feed. This was found to be the maximum range and resolution the server computer could handle while still producing reliable AprilTag detections at an acceptable frequency. This configuration only results in a 2.6×1.5 m operating area. The vessel’s maneuvering room, particularly in the y-axis was fairly constrained.

Figure 6.7 shows CVSS operating area with an example of the path taken by a vessel when performing a retrieval maneuver. The vessel starts in the top right orbiting the cluster area, shown as a red circle, and detects a target in the bottom right sector of the operating area. It veers toward it, capturing the target around position (x:800,y:200) and turns back toward the cluster area. It carries the target toward the cluster, veering sharply toward the center to deposit the captured target before reversing sharply as shown by the sharp curvature change and subsequent

[3]It is worth noting that although the zero vessel test cases had more targets visible to the overhead camera across all test cases, these targets were spread throughout the environment and not truly in a cluster. Although the presence of vessels ended up pushing many targets further away from the cluster, those they did capture were kept close together.



(a) Aggregate Data Line Graph



(b) Aggregate Data Surface Plot

Figure 6.6: Final Cluster Size Aggregate Data Plots

linear motion away from the cluster starting at (x:-100,y:200). It then continues clockwise, detecting another target in the top left quadrant, and begins a second retrieval maneuver.

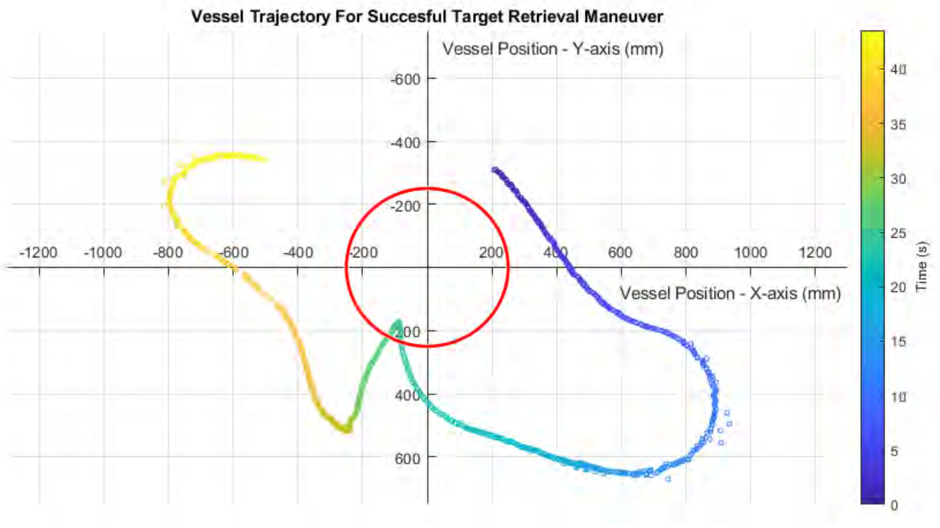


Figure 6.7: 2D Vessel Trajectory Plot - Successful Retrieval Maneuver

Although the sequence of events depicted in Figure 6.7 was successful, the vessel moved very close to the edge of the operating area at (x:700,y:625) as a result. Occasionally, a vessel attempting to retrieve a target near the periphery of the operating area would overshoot and move out of view of the overhead camera. It would fail to retrieve the target, frequently pushing it out of the operating area in the process, and lose localization capabilities. Sometimes the vessel could not recover from such a failed maneuver, remaining lost outside the operating area for the remainder of the test. Such a failed retrieval maneuver attempt can be seen in Figure 6.8.

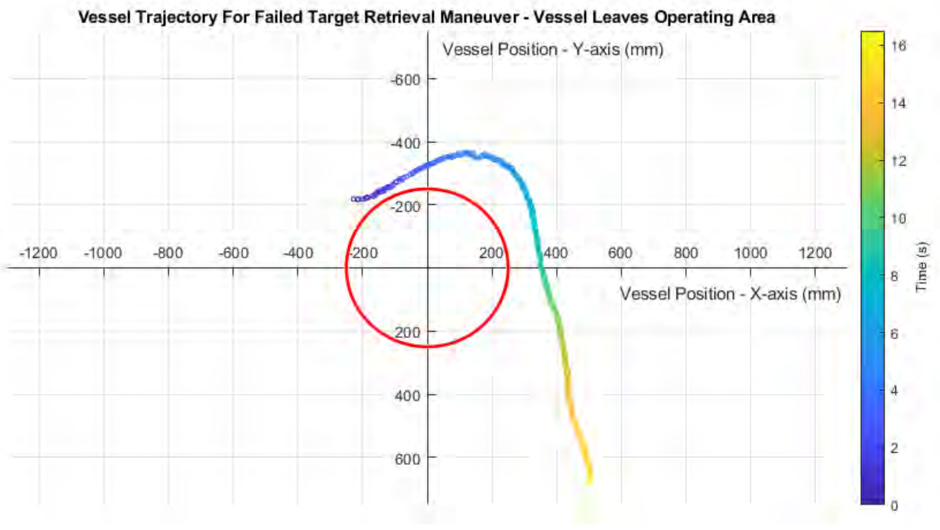


Figure 6.8: 2D Vessel Trajectory Plot - Failed Retrieval Maneuver

In Figure 6.8, the vessel again detects a target in the lower right quadrant and veers out to retrieve it. Unlike in Figure 6.7, the overhead camera loses sight of the vessel at (x:450,y:650) and the vessel fails to retrieve the target. The vessel leaves the operating area and stops moving.

This spatial limitation is an unrealistic shortcoming of the current microUSV system. A full-scale USV fleet which would be equipped with onboard target sensors and localization systems such as GPS would not encounter an invisible boundary the vessels could not observe or move beyond. In the Figure 6.8 example, the vessel would have simply captured the object and returned it to the cluster as normal. Similarly, the targets pushed away from the cluster immediately following vessel deployment would not move beyond detection range. The swarm might be able to reacquire them later despite having initially pushed them away. This limitation must be addressed before the orbital retrieval algorithm can be adequately tested.

The algorithm has shown some initial promising results but also demonstrated a significant shortcoming: disrupting the outermost targets worse than entropy, negatively impacting the clustering results. Addressing the testing system's spatial constraint issue may greatly improve the algorithm's cluster size metric performance.

Chapter 7

Conclusion

This thesis discusses the design of a novel USV platform for marine swarm robotics research applications. The microUSV fills a gap among existing commercial and open source Unmanned Surface Vehicle (USV) platforms: It was designed from the ground up to operate in indoor laboratory environments. The microUSV was intended as an intermediate hardware testing platform, bridging the gap between simulated testing and full-scale validation of marine swarm robotics algorithms on expensive open water USVs.

The platform was designed to be as small and inexpensive as possible while still offering properties critical to swarm robotics research such as good stability and onboard autonomy: It leverages predominantly off-the-shelf hardware and hobbyist electronics. The vessel's handful of custom components were designed for fabrication using a Fused Deposition Modeling (FDM) 3D printer; the most common and lowest cost type of 3D printer currently available. The design was made open source and as simple to fabricate as possible to allow other researchers, regardless of their level of familiarity with hardware assembly methods, to repurpose the platform for their own work.

As part of the cost and size reduction measures, the microUSV system design eschews physical sensors in favor of virtual ones. The vessels operate using simulated sensor data generated by a server computer connected to an overhead camera and running the AprilTag-based CVSensorSimulator (CVSS) application. This setup is able to provide updates for multiple simulated sensors at a frequency of 17 to 21 Hz to each of several vessels simultaneously. It provides pose estimates for localization and simulates nearby target detection using image color thresholding.

The microUSV's hardware and software were validated by testing the system's ability to perform waypoint following tasks and serving as the platform for initial development of the orbital retrieval algorithm. The platform proved watertight, stable, and easily capable of following set waypoints using a simple PI controller implementation during these experiments even with multiple vessels operating simultaneously. The results from testing the orbital retrieval algorithm were less encouraging.

Orbital retrieval is a novel algorithm for clustering floating marine contaminants using a swarm of USVs. This reactive control scheme guides vessels in a circular orbit around a central cluster area, performing a sensor sweep of the surrounding water. Any targets detected are retrieved and added to the cluster. The vessel's orbiting behavior helps maintain the integrity of the cluster as unlike objects in terrestrial clustering applications, floating marine targets have a tendency to disperse naturally over time. This approach was expected to yield larger and tighter clusters with the addition of more vessels. This proved to be untrue.

The vessels executing the orbital retrieval algorithm were able to successfully form and maintain a tight cluster of objects at the designated location but disrupted targets near those they retrieved in the process. Vessels unintentionally bumping into or pushing targets away from the cluster with thrust from their propellers was detrimental to the system's performance. This often resulted in fewer targets remaining in the operating area at the end of each test than the control test cases where zero vessels were deployed and the targets were allowed to disperse naturally. The algorithm consistently produced smaller clusters than expected.

The poor performance during the orbital retrieval tests may not have been the fault of the algorithm itself but of a testing system limitation: Because of its reliance on a single overhead camera for sensor data, the microUSV system's operating area is restricted to that camera's field of view. The current setup can achieve an operating area of 2.6×1.5 m which proved to be too small for this experiment. The targets vessels unintentionally pushed away from the cluster would frequently leave the operating area, becoming undetectable and thus could never be retrieved. Without the artificially limited operating area size, the vessels would be free to move further afield to retrieve these targets at a later time. Vessels were also occasionally observed to leave the operating area in pursuit of a target, losing localization sensor data in the process and becoming unable to navigate. This limitation must be addressed before development of the orbital retrieval algorithm can continue.

7.1 Future Work

In addition to the spatial limitations of its operating area, the microUSV system has some flaws which could be addressed in a future design revision. The known design issues are listed below with one or more potential solutions proposed for each.

7.1.1 Hardware Improvements

- The microUSV's thruster efficiency is too easily influenced by slight misalignments in its the drive train subassemblies. Not all potential users have the patience or experience to properly align each motor to its drive shaft and stuffing tube. Small angular misalignment between the motor and shaft are handled by a universal joint coupler but the addition of a second universal joint would also mitigate small axial misalignments. The drive train subassemblies could also be made removable so they could be properly aligned and tuned before installation.

- Similarly, modifying a vessel's drive shafts requires the use of a lathe; a potentially intimidating tool that is likely unfamiliar to any users without a background in mechanics or machining. An alternate means of mounting the propellers to the drive shaft without component modification would be ideal. The microUSV hardware article [41] proposes bonding the propellers to the shafts with adhesive as a possible alternative to threading them. This solution works but is permanent and may not be ideal for long term maintenance purposes. A shaft collar tensioned with setscrews similar to the drive dog may be a suitable non-permanent alternative.

- The power switch on the voltage bus is difficult to access by hand. A tool narrower than a user's fingers such as a small screw driver is often needed to turn the vessels on. This can be addressed by altering the voltage bus board layout, moving the switch into a more accessible location.

- Installing and removing the onboard electronics bracket involves connecting or disconnecting four jumper cables; one for each motor and one for each battery. Bundling these four cables into a single umbilical cable between the hull devices and electronics bracket devices would greatly simplify this process. Connecting and disconnecting a single cable instead of four would reduce the time needed to change batteries.

- On the topic of jumper cables, the various cables connecting devices on the onboard electronics bracket could be replaced by a single custom Printed Circuit Board (PCB) to simplify the wiring process, greatly reducing the risk of connecting a cable to the wrong pins. This solution is, however, more expensive and does not allow for easy expansion like the jumper cable system.

- Alkaline nine volt batteries were chosen to power the microUSV over more energy-dense alternatives such as Lithium Polymer (LiPo) batteries. Nine volt battery are readily available and battery change times are much shorter than LiPo charging times. These benefits were not found to be sufficient justification for choosing them over LiPo batteries. Changing the microUSV's power system to use LiPo batteries would improve run times and reduce waste. This change would require a careful reworking of the battery mounting configuration and general arrangement to keep the vessels stable. Another possible improvement to the onboard power system would be to replace the 9 V to 5 V voltage regulator with a 9 V to 5 V buck converter circuit. Such a module would consume less power than the voltage regulator in the current design and provide a modest improvement to vessel operation time without significant alterations to its electrical system.

- The vessel's motors were connected directly to the batteries while all other onboard devices were powered using the regulated five volt bus. The vessel's five volt devices, running on regulated power, would function the same way regardless of the level of battery discharge. The motors, however, would produce lower torques over time as the batteries drained, even when sent the same motor speed command. The motors should run on a regulated power supply to give consistent outputs independent of the slowly changing battery voltage. The motors could be replaced with a similar model requiring a lower voltage and be run off of the same regulated five volt bus as the other onboard devices.

7.1.2 Software Improvements

- A simulation environment should be developed which models the kinematics and dynamics of the microUSV platform. The slow hardware experiment and controller iteration process severely hampered development of the orbital retrieval algorithm: The total number of experiment runs, and their associated opportunities for controller tuning, were limited due to the time investment required to set up and execute each individual test scenario. A simulator would accelerate algorithm development considerably; allowing a greater number of controller and environmental parameters to be considered and varied simultaneously with near-instantaneous experimental setup. Pairing the microUSV with a swarm robotics simulator was the original intent during the platform's development: Testing on the microUSV platform was meant to serve as an intermediate step between simulation and full-scale hardware testing. Developing a microUSV simulator will make the hardware platform a more useful tool for swarm robotics development efforts.

- A future microUSV simulator should also incorporate dynamics modeling for target interactions. Many of the unanticipated behaviors observed during testing can be attributed to the interactions between the currents introduced by a vessel's wake or thrust path and the floating targets in the nearby environment. These behaviors, not being present in the ground-based swarms used when testing the algorithms which inspired orbital retrieval, were unexpected and hampered collection efforts. Modelling these behaviors will allow future algorithm development to better predict and avoid such issues, potentially even finding some useful interactions to exploit.

- Control algorithms running on the microUSV might produce more consistent results if they could control the motor's rotation speed instead of its input voltage. Attaching a rotary encoder to each motor would allow controllers to perform closed loop control on the propeller speed rather than the motor torque.

- As discussed above and in Chapter 6, the microUSV system's operating area is too constrained to effectively test some algorithms. The use of a more powerful computer to run CVSS may allow the camera resolution to be increased, increasing the maximum distance at which AprilTags can be detected, but this solution is not scalable. Replacing the AprilTags with a different visual pose detection system with greater detection ranges may prove more successful. The

WhyCode [55] tag system is worth investigating as they function similarly to AprilTags at longer range and would require almost no hardware modifications. Using an infrared sensitive camera to detect unique patterns of infrared-emitting diodes mounted to the top of each vessel may also be considered. This is likely harder to implement than repurposing an existing a commercial system such as the HTC Vive trackers [13].

- The reactive control scheme implemented in the orbital retrieval algorithm may not be suitable for marine environments. Because vessels cannot turn as quickly as terrestrial differential drive robots but must accelerate and decelerate over several seconds, the oscillatory behavior used in [37] and [89] is much slower. This will need to be tested once the microUSV system has been updated to address the spatial limitation issues. Using a trajectory planning approach instead of a reactive controller may prove more effective in this environment.

- Expanding the microUSV's target sensor range and allowing it to differentiate between near and far targets may improve the performance of the orbital retrieval algorithm. Where the current implementation only detects the presence of nearby targets, not the distance to them, it will always attempt to capture the nearest target first, often pushing away those farther from the cluster point in the process. If it were able to identify the farthest target and seek that one first a vessel would travel from the outside of the operating area inward rather than the current scheme which works from the inside outward. This may reduce the number of lost targets due to wave action and interactions with nearby vessels.

Bibliography

[1] Boat autopilot - based on the arduino. `https://www.instructables.com/id/Boat-Autopilot/`. Accessed: 2017-10-08.

[2] C-Worker 8 product information. `https://www.asvglobal.com/product/c-worker-8/`. Accessed: 2019-05-30.

[3] CAT-Surveyor. `https://www.subsea-tech.com/cat-surveyor/`. Accessed: 2019-05-30.

[4] Cleaning up the garbage patches. `https://theoceancleanup.com/oceans/`. Accessed: 2020-03-11.

[5] The correl lab swarm robotics platform - droplets. `https://code.google.com/archive/p/cu-droplet/`. Accessed: 2020-03-08.

[6] Heron unmanned surface vehicle. `https://www.clearpathrobotics.com/heron-unmanned-surface-vessel/`. Accessed: 2019-05-30.

[7] Laurent's multi tug - dundrum bay! `https://www.modelboatmayhem.co.uk/Modellers/Laurent/1ndex.htm`. Accessed: 2019-08-28.

[8] Protocol buffers. `https://developers.google.com/protocol-buffers/`. Accessed: 2019-09-10.

[9] Seabin v5. `https://seabinproject.com/the-seabin-v5/`. Accessed: 2020-03-06.

[10] Swarmdiver - micro swarming USV/UUV. `https://www.aquabotix.com/swarmdiver.html`. Accessed: 2020-03-19.

[11] V4l-utils. `https://linuxtv.org/wiki/index.php/V4l-utils`. Accessed: 2020-04-14.

[12] ASV global and terrasond mark industry first for unmanned hydrographic survey. `https://www.asvglobal.com/asv-global-terrasond-mark-industry-first-unmanned-hydrographic-survey/`, Aug 2016.

[13] *HTC Vive Tracker Developer Guidelines*, Jul 2017.

[14] n3m0 the autonomous boat. `https://hackaday.io/project/25508-n3m0-the-autonomous-boat`, Jun 2017.

[15] M. Allwright, W. Zhu, and M. Dorigo. An open-source multi-robot construction system. *HardwareX*, 5:e00050, 2019.

[16] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox. Mona: an affordable open-source mobile robot for education and research. *Journal of Intelligent & Robotic Systems*, 94(3):761–775, 2019.

[17] F. Arvin, J. Murray, C. Zhang, and S. Yue. Colias: An autonomous micro robot for swarm robotic applications. *International Journal of Advanced Robotic Systems*, 11(7), 2014.

[18] D. K. A. Barnes, F. Galgani, R. C. Thompson, and M. Barlaz. Accumulation and fragmentation of plastic debris in global environments. *Philosophical Transactions of the Royal Society B*, 364(1526):1985–1998, 2009.

[19] M. Barnes, N. Graham, C. Gregory, A. Randell, and W. Whitby. Sunken history of grand lake. *Journal of Ocean Technology*, 12(4):19–26, 2017.

[20] L. Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[21] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[22] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[23] R. G. Budynas and J. K. Nisbett. *Shigley's mechanical engineering design*. McGraw-Hill series in mechanical engineering. McGraw-Hill, New York, 9th ed.. edition, 2011.

[24] R. Campos, N. Gracias, and P. Ridao. Underwater multi-vehicle trajectory alignment and mapping using acoustic and optical constraints. *Sensors*, 16(3):387, 2016.

[25] D. F. Carlson, A. Fürsterling, L. Vesterled, M. Skovby, S. S. Pedersen, C. Melvad, and S. Rysgaard. An affordable and portable autonomous surface vehicle with obstacle avoidance for coastal ocean monitoring. *HardwareX*, 5:e00059, 2019.

[26] A. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

[27] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[28] T. Chung, M. Clement, M. Day, K. Jones, D. Davis, and M. Jones. Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs. volume 2016-, pages 1255–1262. Institute of Electrical and Electronics Engineers Inc., 2016.

[29] D. Cook, A. Vardy, and R. Lewis. A survey of AUV and robot simulators for multi-vehicle operations. In *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–8. IEEE, 2014.

[30] V. Costa, M. Duarte, T. Rodrigues, S. M. Oliveira, and A. L. Christensen. Design and development of an inexpensive aquatic swarm robotics system. *OCEANS 2016 - Shanghai*, pages 1–7, 2016.

[31] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*, volume 3342 of *Lecture Notes in*

Computer Science, pages 10–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[32] M. Duarte, J. Gomes, V. Costa, T. Rodrigues, F. Silva, V. Lobo, M. Monteiro Marques, S. Moura Oliveira, and A. Lyhne Christensen. Application of swarm robotics systems to marine environmental monitoring. *OCEANS 2016 - Shanghai*, pages 1–8, 2016.

[33] M. Eriksen, L. C. M. Lebreton, H. S. Carson, M. Thiel, C. J. Moore, J. C. Borerro, F. Galgani, P. G. Ryan, and J. Reisser. Plastic pollution in the world's oceans: More than 5 trillion plastic pieces weighing over 250,000 tons afloat at sea. *PLoS ONE*, 9(12):e111913, 2014.

[34] J. H. Evans. Basic design concepts. *Journal of the American Society for Naval Engineers*, 71(4):671–678, 1959.

[35] D. Fritsch, K. Wegener, and R. D. Schraft. Control of a robotic swarm for the elimination of marine oil pollutions. *IEEE Swarm Intelligence Symposium*, pages 29–36, 2007.

[36] T. S. Galloway. *Micro- and Nano-plastics and Human Health*, pages 343–366. Springer International Publishing, Cham, 2015.

[37] M. Gauci, J. Chen, W. Li, T. Dodd, and R. Groß. Clustering objects with robots that do not compute. pages 421–428, 01 2014.

[38] A. Gautam and S. Mohan. A review of research in multi-robot systems. In *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5, Aug 2012.

[39] P. S. Gonçalves, P. D. Torres, C. O. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering, 2009.

[40] C. Gregory. microUSV project repository. `https://doi.org/10.17605/OSF.IO/7FQ6U`, Mar 2020.

[41] C. Gregory and A. Vardy. microUSV: A low-cost platform for indoor marine swarm robotics research. *HardwareX*, 7:e00105, Apr 2020.

[42] H. Hamann. *Swarm Robotics: A Formal Approach.* Springer International Publishing, Cham, 2018.

[43] J. Hecker, K. Letendre, K. Stolleis, D. Washington, and M. Moses. Formica ex machina: Ant swarm foraging from physical to virtual and back again. volume 7461, pages 252–259, 2012.

[44] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems*, 28(2):51–64, 2008.

[45] E. N. Jacobs, K. E. Ward, and R. M. Pinkerton. The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel. Technical report, 1933.

[46] W. Jo, Y. Hoashi, L. L. Paredes Aguilar, M. Postigo-Malaga, J. M. Garcia-Bravo, and B.-C. Min. A low-cost and small usv platform for water quality monitoring. *HardwareX*, 6:e00076, 2019.

[47] N. M. Kakalis and Y. Ventikos. Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials*, 154(1):880–887, 2008.

[48] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl. Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259, 2009.

[49] P. Kimball, J. Bailey, S. Das, R. Geyer, T. Harrison, C. Kunz, K. Manganini, K. Mankoff, K. Samuelson, T. Sayre-McCord, F. Straneo, P. Traykovski, and H. Singh. The WHOI jetyak: An autonomous surface vehicle for oceanographic research in shallow or dangerous waters. *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–7, 2014.

[50] A. A. Koelmans, E. Besseling, and W. J. Shim. *Nanoplastics in the Aquatic Environment. Critical Review*, pages 325–340. Springer International Publishing, Cham, 2015.

[51] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3. IEEE, 2004.

[52] S. Kühn, E. L. Bravo Rebolledo, and J. A. van Franeker. *Deleterious Effects of Litter on Marine Life*, pages 75–116. Springer International Publishing, Cham, 2015.

[53] D. Larkin, M. Michini, A. Abad, S. Teleski, and M. A. Hsieh. Design of the multi-robot coherent structure testbed (mCoSTe) for distributed tracking of geophysical fluid dynamics. Volume 5B: 38th Mechanisms and Robotics Conference, Aug 2014.

[54] Z. Li, R. Bachmayer, and A. Vardy. Path-following control for unmanned surface vehicles. *IEEE International Conference on Intelligent Robots and Systems*, 2017-:4209–4216, 2017.

[55] P. Lightbody, T. Krajník, and M. Hanheide. An efficient visual fiducial localisation system. *ACM SIGAPP Applied Computing Review*, 17(3):28–37, 2017.

[56] Z. Liu, Y. Zhang, X. Yu, and C. Yuan. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control*, 41:71–93, 2016.

[57] I. Lončar, A. Babić, B. Arbanas, G. Vasiljević, T. Petrović, S. Bogdan, and N. Mišković. A heterogeneous robotic swarm for long-term monitoring of marine environments. *Applied Sciences*, 9(7), 2019.

[58] J. Manley. Unmanned surface vehicles, 15 years of development. In *OCEANS 2008*, pages 1–4. IEEE, 2008.

[59] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello,

A. Finzi, B. Siciliano, A. Sala, and N. Tomatis. The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–4. IEEE, 2012.

[60] J. McLurkin, A. J. Lynch, S. Rixner, T. W. Barr, A. Chou, K. Foster, and S. Bilstein. *A Low-Cost Multi-robot System for Research, Teaching, and Outreach*, pages 597–609. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[61] K. McTaggart, L. Boudet, and S. Oakey. Validation of a distributed simulation of ship replenishment at sea with model tests. *Journal of Marine Science and Technology*, pages 1–14, 2018.

[62] A. F. Molland. *The Maritime Engineering Reference Book: A Guide to Ship Design, Construction and Operation*, pages 75–115. Elsevier Science, 2008. ch.3 Flotation and stability.

[63] A. F. Molland. *The Maritime Engineering Reference Book: A Guide to Ship Design, Construction and Operation*, pages 484–577. Elsevier Science, 2008. ch.7 Seakeeping.

[64] F. Mondada, G. Pettinaro, A. Guignard, I. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. Gambardella, and M. Dorigo. Swarm-bot: A new distributed robotic concept. *Autonomous Robots*, 17(2):193–221, 2004.

[65] C. Morris, P. Sargent, D. Porter, R. Gregory, D. Drover, K. Matheson, T. Maddigan, C. Holloway, and L. Sheppard. Garbage in newfoundland harbours. *Journal of Ocean Technology*, 11(2):18–26, Jul 2016.

[66] S. Newman, E. Watkins, A. Farmer, P. t. Brink, and J.-P. Schweitzer. *The Economics of Marine Litter*, pages 367–394. Springer International Publishing, Cham, 2015.

[67] C. O'Donnell. World's cheapest foil chart for NACA section profiles. `http://www.boat-links.com/foilfaq.html`, Jan 1997. Accessed: 2020-03-27.

[68] E. Olson. AprilTag: A robust and flexible visual fiducial system. pages 3400–3407, 2011.

[69] O. Parodi, V. Creuze, and B. Jouvencel. Communications with thetis, a real time multi-vehicles hybrid simulator. 07 2008.

[70] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[71] C. Powers, R. Hanlon, and D. G. Schmale. Tracking of a fluorescent dye in a freshwater lake with an unmanned surface vehicle and an unmanned aircraft system. *Remote Sensing*, 10(1):81, 2018.

[72] J. Preiss, W. Honig, G. Sukhatme, and N. Ayanian. Crazyswarm: A large nano-quadcopter swarm. pages 3299–3304. Institute of Electrical and Electronics Engineers Inc., 2017.

[73] A. Prorok, M. A. Hsieh, and V. Kumar. Formalizing the impact of diversity on performance in a heterogeneous swarm of robots. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5364–5371, 2016.

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

[75] C. M. Rochman. *The Complex Mixture, Fate and Toxicity of Chemicals Associated with Plastic Debris in the Marine Environment*, pages 117–140. Springer International Publishing, Cham, 2015.

[76] E. Rohmer, S. P. N. Singh, and M. Freese. CoppeliaSim (formerly V-REP): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[77] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975, 2014.

[78] J. Rubinovas. Hydrostatics calculator for solidworks. `https://www.floatsoft.net/`. Accessed: 2019-05-13.

[79] C. Sarraf, H. Djeridi, S. Prothin, and J. Billard. Thickness effect of naca foils on hydrodynamic global parameters, boundary layer states and stall establishment. *Journal of Fluids and Structures*, 26(4):559–578, 2010.

[80] N. Seltenrich. New link in the food chain? marine plastic pollution and seafood safety. *Environmental health perspectives*, 123(2):A34–A41, 2015.

[81] D. Sousa, D. Hernandez, F. Oliveira, M. Luís, and S. Sargento. A platform of unmanned surface vehicle swarms for real time monitoring in aquaculture environments. *Sensors (Basel, Switzerland)*, 19(21):4695, Oct 2019.

[82] M. Thiel, G. Luna-Jorquera, R. Álvarez Varas, C. Gallardo, I. A. Hinojosa, N. Luna, D. Miranda-Urbina, N. Morales, N. Ory, A. S. Pacheco, M. Portflitt-Toro, and C. Zavalaga. Impacts of marine plastic pollution from continental coasts to subtropical gyres—fish, seabirds, and other vertebrates in the se pacific. *Frontiers in Marine Science*, 5:238, 2018.

[83] T. Tosik, J. Schwinghammer, M. J. Feldvoß, J. P. Jonte, A. Brech, and E. Maehle. Mars: A simulation environment for marine swarm robotics and environmental monitoring. *OCEANS 2016 - Shanghai*, pages 1–6, 2016.

[84] D. Tsankova and V. Georgieva. From local actions to global tasks: Simulation of stigmergy based foraging behavior. In *2004 2nd International IEEE Conference 'Intelligent Systems' - Proceedings*, volume 1, pages 353–358, 2004.

[85] D. Tsankova, V. Georgieva, F. Zezulka, and Z. Bradac. Immune network control for stigmergy based foraging behaviour of autonomous mobile robots. *International Journal of Adaptive Control and Signal Processing*, 21(2-3):265–286, 2007.

[86] J. Tu. *Computational fluid dynamics : a practical approach*. Elsevier/Butterworth-Heinemann, Amsterdam ; Boston, 2nd ed.. edition, 2013.

[87] S. Van Schie, K. Laura Stack, and B. Slat. *Alternative Cleanup Concepts*, pages 68–73. Mar 2014.

[88] A. Vardy. Accelerated patch sorting by a robotic swarm. In *2012 Ninth Conference on Computer and Robot Vision*, pages 314–321. IEEE, 2012.

[89] A. Vardy. Orbital construction: Swarms of simple robots building enclosures. *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pages 147–153, 2018.

[90] A. Vasilijevic, P. Calado, F. Lopez-Castejon, D. Hayes, N. Stilinovic, D. Nad, F. Mandic, P. Dias, J. Gomes, J. C. Molina, A. Guerrero, J. Gilabert, N. Miskovic, Z. Vukic, J. Sousa, and G. Georgiou. Heterogeneous robotic system for underwater oil spill survey. In *OCEANS 2015 - Genova*, pages 1–7. IEEE, 2015.

[91] R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

[92] M. Waibel, B. Keays, and F. Augugliaro. Drone shows: Creative potential and best practices. *Verity Studios*, Jan 2017.

[93] A. Weinstein, A. Cho, G. Loianno, and V. Kumar. Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors. *IEEE Robotics and Automation Letters*, 3(3):1801–1807, Jul 2018.

[94] J. Werfel, K. Petersen, and R. Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science (New York, N.Y.)*, 343(6172):754–758, 2014.

[95] Z. Yan, N. Jouandeau, and A. A. Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12), 2013.

[96] E. M. H. Zahugi, M. M. Shanta, and T. V. Prasad. Oil spill cleaning up using swarm of robots. In N. Meghanathan, D. Nagamalai, and N. Chaki, editors, *Advances in Computing and Information Technology*, pages 215–224, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Appendix A

Bill of Materials

Table A.1: Detailed Bill of Materials

ITEM NO.	DESCRIPTION	SOURCE	QTY	PKG COST	PKG SIZE	ORDER PKGS	Total Cost	PART NUMBER
1	Hull Rev 1.6	3D Print	1	$46.41	1	1	$46.41	MUSV-01
2	Lid Rev 1.6	3D Print	1	$18.71	1	1	$18.71	MUSV-02
3	Keel	3D Print	1	$13.51	1	1	$13.51	MUSV-03
4	Electronics Bracket	3D Print	1	$10.96	1	1	$10.96	MUSV-04
5	Stuffing Tube Bracket	3D Print	2	$0.23	1	2	$0.45	MUSV-05
6	Gearmotor Bracket	3D Print	2	$0.20	1	2	$0.39	MUSV-06
7	8GB Micro SD Card Class 10	Amazon	1	$10.99	1	1	$10.99	
8	9V Battery	Amazon	2	$13.98	8	1	$13.98	
9	UVPOXY 500mL Kit	Amazon	20	$66.85	500	1	$66.85	
10	8" Mini to Micro USB Cable	Amazon	1	$8.21	1	1	$8.21	
11	Vaseline Petroleum Jelly	Amazon	1.6	$3.67	433.5	1	$3.67	
12	2mm Hair Elastic	Amazon	2	$5.99	29	1	$5.99	
13	Arduino Nano	BuyaPi	1	$12.95	1	1	$12.95	
14	Raspberry Pi Zero W	BuyaPi	1	$12.99	1	1	$12.99	
15	2x20 Male Header Pin Strip	BuyaPi	1	$0.95	1	1	$0.95	
16	Loctite Blue 242	Canadian Tire	1	$8.99	6	1	$8.99	067-0036-8
17	Marine Silicone Sealant	Canadian Tire	5	$5.49	82	1	$5.49	067-0842-8
18	LM7805 5V Fixed Voltage Regulator	Digikey	1	$2.32	1	1	$2.32	LM7805
19	Male Header Pin	Digikey	44	$2.95	32	2	$5.90	732-2671-ND
20	Voltage Bus Breadboard	Digikey	1	$1.65	1	1	$1.65	SBBTH1506-1-ND
21	22 AWG Stranded Wire (multiple colors)	Digikey	5	$0.85	1	5	$4.25	22759/32-22-0-DS-ND
22	Slide Switch SPDT	Digikey	1	$3.62	1	1	$3.62	563-1388-ND
23	9V Battery Snap Connector	Digikey	2	$0.74	1	2	$1.48	36-235-ND
24	Crimp Connector 22-24AWG Female	Digikey	38	$0.17	1	3	$6.31	WM2510-ND
25	Crimp Connector 22-24AWG Male	Digikey	4	$0.41	1	4	$1.64	WM2517-ND
26	2 Pos. Conn Housing	Digikey	12	$0.47	1	1	$5.62	WM2800-ND
27	3 Pos. Conn Housing	Digikey	4	$0.67	1	4	$2.68	WM2801-ND
28	4 Pos. Conn Housing	Digikey	1	$0.68	1	1	$0.68	WM2802-ND
29	2 Pos. Conn Housing w/ Latch	Digikey	2	$0.38	1	2	$0.76	WM2900-ND
30	2 Pos. Conn Housing Male	Digikey	2	$0.74	1	2	$1.48	WM2533-ND

Table A.1 – continued from previous page

ITEM NO.	DESCRIPTION	SOURCE	QTY	PKG COST	PKG SIZE	ORDER PKGS	Total Cost	PART NUMBER
31	0.1 μF Capacitor	Digikey	4	$0.28	1	4	$1.12	478-3188-ND
32	28mm Diameter Propeller	Hobby King	2	$2.17	5	1	$2.17	017000471-0
33	U-Joint Shaft Coupler	Hobby King	2	$10.24	5	1	$10.24	017000469-0
34	#2 Flat Washer 18-8SS	Mcmaster-Carr	8	$9.38	500	1	$9.38	98017A601
35	#2-56 Locknut 18-8SS	Mcmaster-Carr	4	$5.17	50	1	$5.17	91831A002
36	#2-56x0.188" Heat Set Insert	Mcmaster-Carr	25	$14.69	100	1	$14.69	94180A312
37	#2-56x1/4" PHMS 18-8SS	Mcmaster-Carr	15	$5.88	100	1	$5.88	91772A077
38	#2-56x1/4" PHMS Nylon	Mcmaster-Carr	10	$7.52	100	1	$7.52	94735A707
39	#2-56x7/16" PHMS 18-8SS	Mcmaster-Carr	4	$9.83	100	1	$9.83	91772A080
40	#2x1/8" Unthreaded Nylon Spacer	Mcmaster-Carr	4	$10.99	25	1	$10.99	94639A460
41	#5 Washer UHMW	Mcmaster-Carr	4	$18.01	25	1	$18.01	95649A120
42	#5-40 Locknut 18-8SS	Mcmaster-Carr	2	$6.69	100	1	$6.69	91831A006
43	1/8" Flanged Sleeve Bearing	Mcmaster-Carr	4	$0.88	1	4	$3.51	6338K562
44	1/8x2" Bar Stock 304SS	Mcmaster-Carr	1	$6.47	5	1	$6.47	8992K781
45	1/8x3" Drive Shaft 316SS	Mcmaster-Carr	2	$5.99	1	2	$11.97	1263K133
46	5/16" 316SS Smooth-Bore Tube	Mcmaster-Carr	2	$11.34	6	1	$11.34	89785k828
47	1/8" Brass Drive Dog	Offshore Electronics	2	$3.26	1	2	$6.52	ose-80252
48	Pololu 5:1 Micro Metal Gearmotor HPCB 12V	Pololu	2	$23.51	1	2	$47.03	3036
49	Pololu MinIMU-9	Pololu	1	$20.89	1	1	$20.89	2738
50	Pololu Qik 2s9v1	Pololu	1	$32.68	1	1	$32.68	1110

Total Cost (Single Vessel): $522.00