

| <b>REPORT DOCUMENTATION PAGE</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                    |                                         |                  |                                          | <i>Form Approved</i><br>OMB No. 0704-0188                        |   |   |   |                                             |  |                                       |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|-----------------------------------------|------------------|------------------------------------------|------------------------------------------------------------------|---|---|---|---------------------------------------------|--|---------------------------------------|--|
| <p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p> |                    |                                         |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>1. REPORT DATE (DD-MM-YYYY)</b><br>05/14/2019                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                    | <b>2. REPORT TYPE</b><br>Masters Thesis |                  |                                          | <b>3. DATES COVERED (From - To)</b><br>08/2017 - 05/2019         |   |   |   |                                             |  |                                       |  |
| <b>4. TITLE AND SUBTITLE</b><br>Design and Testing of an Autonomous Mothership for Surface Vehicle Swarm Docking                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                    |                                         |                  | <b>5a. CONTRACT NUMBER</b><br>N/A        |                                                                  |   |   |   |                                             |  |                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                    |                                         |                  | <b>5b. GRANT NUMBER</b><br>N/A           |                                                                  |   |   |   |                                             |  |                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                    |                                         |                  | <b>5c. PROGRAM ELEMENT NUMBER</b><br>N/A |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>6. AUTHOR(S)</b><br>Andrew Freeman                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                    |                                         |                  | <b>5d. PROJECT NUMBER</b><br>N/A         |                                                                  |   |   |   |                                             |  |                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                    |                                         |                  | <b>5e. TASK NUMBER</b><br>N/A            |                                                                  |   |   |   |                                             |  |                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                    |                                         |                  | <b>5f. WORK UNIT NUMBER</b><br>N/A       |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b><br>Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                    |                                         |                  |                                          | <b>8. PERFORMING ORGANIZATION REPORT NUMBER</b><br>N/A           |   |   |   |                                             |  |                                       |  |
| <b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b><br>N/A                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                    |                                         |                  |                                          | <b>10. SPONSOR/MONITOR'S ACRONYM(S)</b><br>N/A                   |   |   |   |                                             |  |                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                    |                                         |                  |                                          | <b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b><br>N/A             |   |   |   |                                             |  |                                       |  |
| <b>12. DISTRIBUTION/AVAILABILITY STATEMENT</b><br>DISTRIBUTION A. Approved for public release: distribution unlimited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                    |                                         |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>13. SUPPLEMENTARY NOTES</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                    |                                         |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>14. ABSTRACT</b><br>This thesis elucidates the essential parameters for the design of an autonomous surface vehicle swarm mothership and considerations for docking the two vehicles together autonomously. A theoretical analysis is presented for extending the energy capacity of the vehicles through mothership recharging, resulting in mothership design specifications. A computational analysis of mother-daughter relative motions in waves follows, validating the hullform suitability for docking. Finally, a prototype developed by the author using the proposed methodology is detailed and experimental results are presented which highlight important considerations contributing to the success of autonomous mothership docking.                                                                                                                                                                                                                   |                    |                                         |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>15. SUBJECT TERMS</b><br>Autonomous Surface Vehicle, Mothership, Swarm, Autonomous Docking                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                    |                                         |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>16. SECURITY CLASSIFICATION OF:</b><br><table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%; padding: 2px;"><b>a. REPORT</b></td> <td style="width: 33%; padding: 2px;"><b>b. ABSTRACT</b></td> <td style="width: 33%; padding: 2px;"><b>c. THIS PAGE</b></td> </tr> <tr> <td style="text-align: center; padding: 2px;">U</td> <td style="text-align: center; padding: 2px;">U</td> <td style="text-align: center; padding: 2px;">U</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                    |                                         | <b>a. REPORT</b> | <b>b. ABSTRACT</b>                       | <b>c. THIS PAGE</b>                                              | U | U | U | <b>17. LIMITATION OF ABSTRACT</b><br><br>UU |  | <b>18. NUMBER OF PAGES</b><br><br>131 |  |
| <b>a. REPORT</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <b>b. ABSTRACT</b> | <b>c. THIS PAGE</b>                     |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| U                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | U                  | U                                       |                  |                                          |                                                                  |   |   |   |                                             |  |                                       |  |
| <b>19a. NAME OF RESPONSIBLE PERSON</b><br>Andrew Freeman                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                    |                                         |                  |                                          | <b>19b. TELEPHONE NUMBER (Include area code)</b><br>617-253-4339 |   |   |   |                                             |  |                                       |  |

THIS PAGE INTENTIONALLY LEFT BLANK

# Design and Testing of an Autonomous Mothership for Surface Vehicle Swarm Docking

by

Andrew Freeman

B.S., University of South Carolina (2006)

Submitted to the Department of Mechanical Engineering  
in partial fulfillment of the requirements for the degree of

Naval Engineer's Degree and Master of Science in Mechanical  
Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

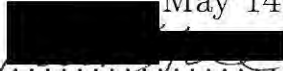
© Andrew Freeman, 2019. All rights reserved.

The author hereby grants to the U.S. Government and MIT permission  
to reproduce and to distribute publicly paper and electronic copies of  
this thesis document in whole or in part in any medium now known or  
hereafter created.

Author ..... 

Department of Mechanical Engineering

May 14, 2019

Certified by ..... 

Dick K. P. Yue

Philip J. Solondz Professor of Engineering

Thesis Supervisor

Accepted by ..... 

Nicolas Hadjiconstantinou, Chairman  
Committee on Graduate Students



# Design and Testing of an Autonomous Mothership for Surface Vehicle Swarm Docking

by

Andrew Freeman

Submitted to the Department of Mechanical Engineering  
on May 14, 2019, in partial fulfillment of the  
requirements for the degree of  
Naval Engineer's Degree and Master of Science in Mechanical Engineering

## Abstract

The field of marine autonomous systems has blossomed over the past several decades yet much remains to be explored. One important focus area gaining increased attention in this domain is the understanding and development of cooperative groups of vessels, known as swarms. Using simple behavioral rule sets, these systems of marine vehicles can work together in many ways including traveling in a “flock” or forming a perimeter around an area of interest. In many cases, however, the vehicles encounter limitations with respect to their capacity to carry a certain resource. This can manifest itself in the form of finite electrical energy, cargo-carrying capacity, or the need to replenish a swarm vehicle’s deployable payloads. In all of these situations, there are many useful scenarios where the addition of a “mothership” can improve overall system performance.

This thesis elucidates the essential parameters for the design of such a mothership and considerations for docking the two vehicles together autonomously. Focus is placed on maintaining the mothership lightweight, compact, and low cost to ensure the results are usable by a broad audience and that the principles are shown to apply to the widest array of designs. A methodology and tools are presented herein to prepare future designers for creating similar systems. These include a theoretical analysis of the finite resource problem and how to leverage parameter relationships to determine design specifications for the mothership. A computational analysis of mother-daughter relative motions in waves follows which validates that the hullform characteristics support reduced complexity at the docking bay. Finally, a prototype developed by the author using the proposed methodology is detailed and experimental results are presented which highlight important considerations contributing to the success of autonomous mothership docking.

Thesis Supervisor: Dick K. P. Yue

Title: Philip J. Solondz Professor of Engineering



# Acknowledgments

I am infinitely grateful to my wife, Rebecca, and to my kids for their tremendous love and support. I also owe a debt of gratitude to my thesis advisor, Professor Yue, and to Dr. Grgur Tokić who conferred their wisdom and support upon me throughout this thesis. Finally, thanks to MIT Sea Grant, the MIT Sailing Pavilion, and my colleagues in the Naval Construction and Engineering program for their support and advice.



# Contents

|          |                                                                       |           |
|----------|-----------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                   | <b>17</b> |
| 1.1      | Background and Related Research . . . . .                             | 18        |
| 1.2      | Thesis Focus . . . . .                                                | 20        |
| 1.3      | Scenario Under Consideration . . . . .                                | 20        |
| 1.4      | Assumptions . . . . .                                                 | 21        |
| 1.5      | Design Philosophy . . . . .                                           | 22        |
| <br>     |                                                                       |           |
| <b>2</b> | <b>Swarm Mothership Energy Analysis</b>                               | <b>23</b> |
| 2.1      | Problem Formulation . . . . .                                         | 24        |
| 2.2      | Energy Endurance Simulations . . . . .                                | 29        |
| 2.3      | Energy Factor Exploration . . . . .                                   | 30        |
| 2.4      | Analysis of Energy Factor Relationships . . . . .                     | 32        |
| 2.5      | Analysis of Specific Parameters . . . . .                             | 33        |
| 2.6      | Conversion of Design Parameters to Specific Design Specifications . . | 35        |
| 2.7      | Design Tool . . . . .                                                 | 36        |
| 2.8      | Summary . . . . .                                                     | 36        |
| <br>     |                                                                       |           |
| <b>3</b> | <b>Minimizing Relative Motions</b>                                    | <b>39</b> |
| 3.1      | Background . . . . .                                                  | 39        |
| 3.2      | Overview of Wave Interactions . . . . .                               | 40        |
| 3.3      | Computation of Wave Forces . . . . .                                  | 41        |
| 3.4      | Vehicle Modeling . . . . .                                            | 43        |
| 3.5      | Computation of Relative RAO: Approximate Method . . . . .             | 44        |

|          |                                                                |           |
|----------|----------------------------------------------------------------|-----------|
| 3.6      | Computation of Relative RAO: WAMIT Multi-Body Method . . . . . | 46        |
| <b>4</b> | <b>Prototype Design and Construction</b>                       | <b>49</b> |
| 4.1      | Daughter Vessel Description . . . . .                          | 49        |
| 4.2      | Application of Mothership Design Process . . . . .             | 51        |
| 4.3      | Hull Structure . . . . .                                       | 52        |
| 4.4      | Electronics . . . . .                                          | 54        |
| 4.5      | System Code . . . . .                                          | 56        |
| 4.6      | Propulsion System . . . . .                                    | 57        |
| <b>5</b> | <b>Docking System</b>                                          | <b>61</b> |
| 5.1      | Process Overview . . . . .                                     | 62        |
| 5.2      | Camera . . . . .                                               | 63        |
| 5.3      | Video Frame Stabilization . . . . .                            | 64        |
| 5.4      | Open Source Computer Vision . . . . .                          | 66        |
| 5.5      | Controller . . . . .                                           | 68        |
| 5.6      | Vision System Summary . . . . .                                | 69        |
| <b>6</b> | <b>Testing and Results</b>                                     | <b>71</b> |
| 6.1      | Lab Testing . . . . .                                          | 71        |
| 6.2      | Small Tank . . . . .                                           | 72        |
| 6.3      | Tow Tank . . . . .                                             | 74        |
| 6.4      | Conduct of Field Tests . . . . .                               | 76        |
| 6.5      | Field Test Observations . . . . .                              | 78        |
| <b>7</b> | <b>Conclusion</b>                                              | <b>83</b> |
| 7.1      | Key Takeaways . . . . .                                        | 83        |
| 7.2      | Recommendations for Future Work . . . . .                      | 86        |
| 7.2.1    | Improvements to Prototype Mothership . . . . .                 | 86        |
| 7.2.2    | Related Research . . . . .                                     | 88        |
| 7.3      | Conclusion . . . . .                                           | 89        |





# List of Figures

|      |                                                                                                                                                                                                    |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2-1  | Representative Examples of Energy Factors and Calculated Results . . .                                                                                                                             | 29 |
| 2-2  | Representative Simulation . . . . .                                                                                                                                                                | 29 |
| 2-3  | Simulation of an Overwhelmed Mothership . . . . .                                                                                                                                                  | 30 |
| 2-4  | Effect of Different Energy Factors on $q$ . . . . .                                                                                                                                                | 31 |
| 2-5  | Effect of Different Energy Factors on Total Swarm Endurance in Days                                                                                                                                | 31 |
| 2-6  | Effect of Different Energy Factors on Average Fraction of Daughters<br>Operational (scaled from 0 to 1) . . . . .                                                                                  | 31 |
| 2-7  | $\kappa$ vs. $\pi$ Effect on Swarm Extension . . . . .                                                                                                                                             | 32 |
| 2-8  | Recharge Setpoint at 40% Energy Capacity . . . . .                                                                                                                                                 | 34 |
| 2-9  | Recharge Setpoint at 60% Energy Capacity . . . . .                                                                                                                                                 | 34 |
| 2-10 | Height Parametrics . . . . .                                                                                                                                                                       | 35 |
| 2-11 | Length Parametrics . . . . .                                                                                                                                                                       | 35 |
| 2-12 | Width Parametrics . . . . .                                                                                                                                                                        | 36 |
| 2-13 | Weight Parametrics . . . . .                                                                                                                                                                       | 36 |
| 2-14 | Swarm Mothership Design Tool - Graphical User Interface . . . . .                                                                                                                                  | 37 |
| 2-15 | Summary of Energy Analysis . . . . .                                                                                                                                                               | 37 |
| 3-1  | Panel Vertices Used for Daughter and Catamaran . . . . .                                                                                                                                           | 43 |
| 3-2  | Wave Spectra for Massachusetts Bay and Cape Cod Bay, where $S_\omega$<br>is the spectral density, $f$ is the wave frequency, and $f_{MB}$ is the peak<br>frequency for Massachusetts Bay . . . . . | 45 |

|     |                                                                                                                                                                                            |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3-3 | WAMIT Single Body Analyses, where $RAO_3$ is the mothership RAO in heave, $RAO_5$ is the mothership RAO in pitch, and $RAO_d$ is the daughter RAO in heave . . . . .                       | 46 |
| 3-4 | WAMIT Approximate Method, where $RAO_r$ is the relative RAO between the daughter and mother, $f$ is the wave frequency, and $f_{MB}$ is the peak frequency for Massachusetts Bay . . . . . | 47 |
| 3-5 | WAMIT Multi-Body Method, where $RAO_r$ is the relative RAO between the daughter and mother, $f$ is the wave frequency, and $f_{MB}$ is the peak frequency for Massachusetts Bay . . . . .  | 48 |
| 4-1 | Daughter Vessel Developed by Zoss et al., 2018 [1] . . . . .                                                                                                                               | 50 |
| 4-2 | Daughter Vessel Thruster Configuration [1] . . . . .                                                                                                                                       | 50 |
| 4-3 | Simulation Used to Determine Design Parameters . . . . .                                                                                                                                   | 52 |
| 4-4 | Modular Mothership Hull from ExpandaCraft . . . . .                                                                                                                                        | 53 |
| 4-5 | Electronics Configuration . . . . .                                                                                                                                                        | 55 |
| 4-6 | Thruster Configuration . . . . .                                                                                                                                                           | 58 |
| 4-7 | Final Mothership . . . . .                                                                                                                                                                 | 60 |
| 5-1 | Docking Paradigm . . . . .                                                                                                                                                                 | 62 |
| 5-2 | Vision System Operational Window . . . . .                                                                                                                                                 | 63 |
| 5-3 | No Correction . . . . .                                                                                                                                                                    | 65 |
| 5-4 | Video Processing . . . . .                                                                                                                                                                 | 65 |
| 5-5 | 3-Axis Gimbal . . . . .                                                                                                                                                                    | 65 |
| 5-6 | 3-Axis Gimbal Mounted to Mothership . . . . .                                                                                                                                              | 66 |
| 5-7 | Mothership Computer Vision Detection Process . . . . .                                                                                                                                     | 67 |
| 5-8 | Mothership Computer Vision Tracking Process . . . . .                                                                                                                                      | 68 |
| 6-1 | Initial Testing: Small Square Tank . . . . .                                                                                                                                               | 73 |
| 6-2 | Second Phase of Testing: Tow Tank . . . . .                                                                                                                                                | 74 |
| 6-3 | Correct Detection . . . . .                                                                                                                                                                | 75 |
| 6-4 | Extent of Visual Range . . . . .                                                                                                                                                           | 75 |

|     |                                               |    |
|-----|-----------------------------------------------|----|
| 6-5 | Baseline . . . . .                            | 76 |
| 6-6 | Medium . . . . .                              | 76 |
| 6-7 | Large . . . . .                               | 76 |
| 6-8 | Initial Charles Testing . . . . .             | 77 |
| 6-9 | Successful Docking in Charles River . . . . . | 78 |



# List of Tables

|     |                                                                    |    |
|-----|--------------------------------------------------------------------|----|
| 4.1 | Design Specifications . . . . .                                    | 52 |
| 4.2 | Graphical User Interface for Buoy and Mothership Control . . . . . | 57 |
| 6.1 | Estimation of Detection Range in Tow Tank . . . . .                | 76 |



# Chapter 1

## Introduction

Great strides have been made in the development and proliferation of autonomous marine vehicles over the past several decades. Autonomous Undersea Vehicles (AUVs) and Autonomous Surface Vehicles (ASVs) have both progressed from research curiosities to deployable systems providing concrete benefit to users. Their applications span the spectrum from oceanographic research, marine oil operations, and naval warfare [2].

Researchers have recently begun to explore using multiple instances of these vessels in the form of swarms [3, 4]. This thesis builds off of one such system described by Zoss, et al. [5, 1] in which a scalable swarm of autonomous mobile buoys was developed. The buoys use the implementation of simple behavioral rules to achieve complex collective functions. For instance, a group of vehicles could use the positions and velocities of its neighbors to traverse together in an organized flock. These vessels could similarly form a perimeter around a region of interest while maintaining position to continuously monitor it. The applications for these marine swarm systems is vast with a great deal of their potential still largely unexplored.

Despite the great capabilities provided by these systems, one of the biggest challenges they face are their finite energy capacities. Space and weight are limited on these platforms, resulting in limited energy stores which reduce useful operational times. Battery capacities can sometimes be increased, but they remain finite and limited below what is often required. Although solar charging [6] and wave energy

harvesting [7] are sometimes viable options for extending autonomous marine vehicle endurance, these solutions are not always available or able to fully compensate for the required energy consumption.

One potential solution to this problem is the inclusion of a mothership vessel to bring more energy capacity into the overall system. Mothership technology has begun to gain attention from this perspective as well as from the desire for automated launch & recovery, but the topic still requires a great deal of research. This thesis seeks to contribute to the study of that problem.

## 1.1 Background and Related Research

The design of a swarm mothership for energy replenishment sits at the intersection of many fields. First, as an ASV, it builds upon the general concepts of those types of vessels which pull from naval architecture, mechanical engineering, electrical engineering, computer science, and more. Its classification as a mothership for swarm vehicles adds additional nuance to the discussion since the scalability of the system will have a large effect on the design of the mothership. This not only requires the mothership to carry a corresponding multiple of battery recharge capacities, but sufficient energy to traverse the distances between the vessels, speed to avoid the complete discharge of daughters before docking can occur, and an appropriate spread of daughter vessels consistent with the capabilities of the swarm units and the mothership.

ASV technology dates back several decades with one of the earliest examples coming from MIT Sea Grant in 1993 [8]. Their vehicle ARTEMIS was used to study some of the basic problems in unmanned surface vehicles including control, navigation, and data collection. Since then, many platforms from researchers around the globe have built upon that early foundation to investigate more advanced topics such as optimal ASV hull design, sensor employment, and docking [2].

A number of steps are required to achieve a successful mating. These are described in the 2016 Spring Handbook for Ocean Engineering in which Bellingham breaks down the process into five phases: (1) En route; (2) Approach setup; (3) Approach;

(4) Terminal Homing; and (5) Capture [9]. Each step has been investigated by a number of researchers, but challenges remain and different scenarios require different strategies. This thesis was primarily focused on the approach and terminal homing phases.

Sensor and control solutions have been proposed for directing the movement of both AUVs and ASVs during the approach and homing phases. These have taken the form of vision, electromagnetic, radio frequency, and sonar techniques, exploiting them in a variety of ways. Computer vision, which was used in this thesis, has been explored from a number of different of dimensions including vision-based tracking of an AUV by an ASV for docking [10, 11], the challenges of camera field of view and limited degrees of freedom [12], and combining visual with sonar information [13].

Within the broad view of ASV or AUV autonomous docking to include stationary docks, an example of such a system actually developed is the Autonomous Ocean Sampling Network (AOSN) described by Frye, et al. [14]. AOSN demonstrated a sensor system with autonomous docking and recharging of AUVs at a stationary dock. A few researchers have explored the concept of docking with mobile vehicles. Notably, Brizzolara at the MIT Sea Grant Program explored the hull design of such a vessel [15]. Their study focused extensively on efficient hydrodynamics and reducing vessel motions, noting the dependency on vessel size given a certain sea state and the difficult motions imposed on a surfaced AUV.

Autonomous surface vehicle docking is impeded by a number of obstacles including environmental disturbances and object tracking accuracy. Designers likely do not know the exact conditions that a vehicle will face and must incorporate features which eliminate or mitigate these challenges. Kim et al. explored this issue with their demonstration of docking using homogeneous ASVs to form a containment boom for oil spills [16]. They describe the successful use of visual feature extractors optimized to improve latency and reinforcement learning techniques to address the lack of a priori information.

Finally, this thesis builds very specifically off of the work done by Zoss et al. described above. In addition to facilitating the conceptual scenario this thesis explores,

the swarming buoys were utilized in this work’s testing to provide a daughter for the mothership to mate with. The electronics and vehicle control code from Zoss’ vehicles were also used as the baseline from which the prototype mothership development started. This enabled a greater focus on the autonomous docking challenges instead of re-plowing the field of generating common ASV components.

## 1.2 Thesis Focus

Despite the vast nature of related research in the fields surrounding swarm mothership design, there remain many interesting areas ripe for investigation. This thesis explores the essential concepts for design of a swarm mothership and key aspects governing autonomous docking between the two vehicles. Specifically, a design methodology is proposed which connects the naval architecture, swarm scalability, and energy limitation aspects of the system to facilitate the development of mothership design parameters. The mothership hullform analysis is further developed by a discussion and computational analysis of relative motions in waves between the mothership and daughter. Finally, to provide a practical view and validate the suitability of a novel docking paradigm, the design, construction, and testing of a prototype mothership docking system is described.

## 1.3 Scenario Under Consideration

A large set of meaningful needs can be met by the utilization of these vessel swarms. For instance, the sustained collection of environmental data would benefit greatly by the use of a marine swarm. This system provides the capability of characterizing the water body in both temporal and spatial variations. The swarm could leverage the spread of its sensors and result in meaningful sensing patterns enabled by the vehicles’ behavioral rulesets.

Another pragmatic example is the use of a swarm to sense for harbor surveillance. This could occur in a country’s own harbor or from a system deployed overseas.

The vehicles would be depended on to traverse the surrounding area with its sensors engaged, following an appropriate behavioral ruleset to survey the scene. Its energy status would be of great concern to the user since the supply would be finite and thus the introduction of a mothership would be beneficial, especially when deployed away from home shores.

Innumerable other scenarios can be derived which use the principles described above. The outline of each situation follows a similar pattern:

- A scalable swarm of autonomous vessels needs to operate for an extended period of time
- Each swarm asset has some finite resource to be replenished periodically (i.e., trash-holding capacity, electrical energy, etc.)
- A mothership is employed which has greater capacity to carry this finite resource

## 1.4 Assumptions

The scenario and prototype developed in this thesis are rooted in a number of assumptions which shape the resulting system and manage the scope of this work.

1. Two ASVs require docking together
2. All interactions between the ASVs are autonomous
3. Docking is defined to mean approaching from a distance approximately equal to the error of GPS until the smaller vessel is contained within a docking bay provided by the mothership
4. The mothership will be assigned to support at least two daughters during a scenario, though only required to dock with one vessel at a time
5. To maintain swarm vessels on station for a longer period of time and to reduce their energy loss to transiting, the mother will traverse to the daughter for docking

6. However, once within docking range, the daughter will maneuver into the bay due to greater maneuverability and a desire to have the mother provide a lee against wave action

## 1.5 Design Philosophy

The buoy swarm described by Zoss and the thesis described herein have characteristics which are based upon foundational principles which complement its role as a swarm and ensure the system is as usable and deployable as possible. These are:

1. **Smart:** Despite being governed by simple behavioral rulesets, collective behaviors can be quite complex. Vehicles operate autonomously.
2. **Scalable:** The addition of new vehicles minimally impacts the rest of the system in terms of cost and complexity.
3. **Attritable:** The swarm is robust to unit losses. Individual unit cost is not so great that the loss of a unit is prohibitive.
4. **Deployable:** Units are lightweight and compact to allow ease of launch, recovery, and operations.

## Chapter 2

# Swarm Mothership Energy Analysis

Although a large body of work has been undertaken on the topic of ASVs, little has been done to propose a design process for an ASV mothership to sustain a swarm of other vessels. This distinction introduces many interesting problems that bear investigation since the selection of design parameters can have a significant impact on performance. Due to the scalable nature of swarms and the potentially large variations in energy demands or the spread of assets in physical space, a mothership's role is highly dependent on these interactions.

Ultimately, the design of such a system will depend on the operational situation it must fulfill. To build upon and complete the description from Chapter 1, the scenario assumed for this analysis is that there is a swarm of  $N$  daughters and one mothership. The daughters have a finite battery capacity and are depleting their energy reserves over time. There is a desire for the system to last longer than the design battery capacity of the daughters allows and thus a mothership is introduced to sustain the system over a longer period. The mothership is expected to travel to the daughters and the two are expected to dock in support of transferring electrical energy. Based upon these assumptions, the ability of the system to be prolonged is dependent on a number of factors which include:

- Daughter energy capacity  $E_d$
- Daughter energy usage  $P_d$
- Mothership energy capacity  $E_m$
- Mothership energy usage  $P_m$

- Spread of the daughters in space  $\delta$
- Number of daughters  $N_d$
- Number of motherships  $N_m$
- Speed of the mothership  $v$
- Time required to dock  $t_d$
- Rate of charging  $P_{ch}$
- Setpoint to initiate recharging  $\psi$

All of these will affect the extension of the swarm energy endurance  $q$ , defined as:

$$q = \frac{\text{Swarm Energy Endurance with Mothership}}{\text{Swarm Energy Endurance without Mothership}}$$

## 2.1 Problem Formulation

The goal of the analysis is to determine how best to maximize the energy endurance extension  $q$ . This is governed by the equations below which determine the state of electrical energy for the mother and daughter vessels at each point in time. The problem begins by breaking the scenario into three states which result in different energy consumption relationships shown below.

**Assumptions** The “hotel load” power consumption of the mother is segregated from this analysis since it is assumed to be relatively independent of the daughter vessels. This assumption reduces the complexity of the equations by not having to include an arbitrary constant or unrealistically modeling the hotel load as a proportional relationship. Instead, the energy analysis below can be conducted and the hotel load of the mothership can be added later to the required mothership capacity.

To reduce the number of dimensions in order to limit the computational complexity, a few parameters were chosen to be constants. However, each of these could be added back in as variables for greater fidelity. First, the recharging setpoint and charging rate were given assumed values which were consistent with Lithium Polymer battery use. These batteries are constrained by a lower discharge boundary and by a recharge rate proportional to their capacity. The mothership speed was also given an assumed value, 1 m/s. This was consistent with the speeds employed by the daughters and also was a conservatively slow speed which meant the swarm vehicles would

discharge more of their batteries as the mothership was transiting. Of note, the power required for the mothership to transit was not dependent on this value. Finally, the time to dock was neglected due to its relatively small time period compared to transit and operations.

To begin our analysis, we first define the relevant terminology and then generate the base equations for mother and daughter energy capacity at time  $t_n$  given the condition at time  $t_{n-1}$  in the three states.

Definitions:

$$E_{d_{n-1}} = \text{Energy of the daughter at time } n - 1$$

$$E_{d_n} = \text{Energy of the daughter at time } n$$

$$E_{m_{n-1}} = \text{Energy of the mother at time } n - 1$$

$$E_{m_n} = \text{Energy of the mother at time } n$$

$$P_d = \text{Power consumption of the daughter due to operations}$$

$$P_m = \text{Power consumption of the mother due to transiting}$$

$$P_{ch} = \text{Power transfer from charging}$$

$$t = \text{Time increment}$$

The first state is characterized by the daughter vessels operating while the mothership idles, waiting to be called to recharge. During this time period, the daughter is expending energy determined by the daughter power consumption due to operations,  $P_d$ . The mothership is using only its hotel load and does not reduce its energy capacity available for the swarm system.

Next, when the recharge sequence is initiated, the daughter vessel continues to expend energy dictated by  $P_d$  while the mothership begins to expend energy governed by the power consumption due to transiting,  $P_m$ . The mothership's power consumed will result from its velocity and hull resistance characteristics as well as the spread of

the daughters in physical space. The velocity and spread of assets will also determine the duration the transit state.

Finally, once the two vessels have docked and recharging has commenced, the daughter vessel's energy is increasing based on the charging power,  $P_{ch}$ , while the mothership is decremented the same amount. This energy transfer is idealized in the analysis and does not account for energy losses to heat or other inefficiencies.

### Base Equations:

1. Daughter operating, mother idling

$$E_{d_n} = E_{d_{n-1}} - P_d t \quad (2.1)$$

$$E_{m_n} = E_{m_{n-1}} \quad (2.2)$$

2. Daughter operating, mother transiting for recharge operations

$$E_{d_n} = E_{d_{n-1}} - P_d t \quad (2.3)$$

$$E_{m_n} = E_{m_{n-1}} - P_m t \quad (2.4)$$

3. Mother recharging daughter

$$E_{d_n} = E_{d_{n-1}} + P_{ch} t \quad (2.5)$$

$$E_{m_n} = E_{m_{n-1}} - P_{ch}t \quad (2.6)$$

We next eliminate the dimensional dependence by employing five non-dimensional factors:

**Swarm Energy Factors:**

$$\textbf{Scalability Factor}(N) : \frac{\text{Number of Daughters}}{\text{Number of Motherships}} = \frac{N_d}{N_m} \quad (2.7)$$

$$\textbf{Capacity Factor}(\kappa) : \frac{\text{Energy Capacity of Daughter}}{\text{Energy Capacity of Mothership}} = \frac{E_d}{E_m} \quad (2.8)$$

$$\textbf{Time Factor}(\tau) : \frac{\text{Time to Transit Between Daughters}}{\text{Time to Fully Discharge a Daughter}} = \frac{t_{transit}}{t_{full\ discharge}} \quad (2.9)$$

$$\textbf{Rate Factor}(\rho) : \frac{\text{Power Consumption of the Daughter}}{\text{Charging Power}} = \frac{P_d}{P_{ch}} \quad (2.10)$$

$$\textbf{Power Factor}(\pi) : \frac{\text{Power Consumption of the Daughter}}{\text{Power Consumption of the Mothership}} = \frac{P_d}{P_m} \quad (2.11)$$

The energy factors, assumed values from above, and a characteristic value for daughter energy capacity are then incorporated into the equations. These modifications allow us to look at a generalized set of relationships which are not dependent upon specific values. Once it is desired to apply this analysis to a specific situation, dimensional quantities can take the place of the daughter energy capacity and practical values will then be formulated for all the remaining quantities.

**Base Equations with Energy Factors Incorporated:**

Initial Conditions:

$$E_{d_0} = E_{char} \quad (2.12)$$

$$E_{m_0} = \frac{E_{char}}{\kappa} \quad (2.13)$$

Operational State:

$$E_{d_n} = E_{d_{n-1}} - P_{ch}\rho t \quad (2.14)$$

$$E_{m_n} = E_{m_{n-1}} \quad (2.15)$$

Transiting State:

$$E_{d_n} = E_{d_{n-1}} - P_{ch}\rho t \quad (2.16)$$

$$E_{m_n} = E_{m_{n-1}} - \frac{P_{ch}\rho}{\pi} t \quad (2.17)$$

Recharging State:

$$E_{d_n} = E_{d_{n-1}} + P_{ch}t \quad (2.18)$$

$$E_{m_n} = E_{m_{n-1}} - P_{ch}t \quad (2.19)$$

Next, the time steps for each state are calculated with the following relationships:

Operational State Duration:

$$t_{dis} = \frac{E_{char}}{P_{ch}\rho} \quad (2.20)$$

Transiting State Duration:

$$t_{tr} = \frac{E_{char}}{P_{ch}\rho} \tau \quad (2.21)$$

Recharging State Duration:

$$t_{chg} = \frac{E_{char}}{P_{ch}} \quad (2.22)$$

Although not included in the equations above, the Scalability Factor,  $N$ , affects the analysis by dictating the number of daughters instantiated in the simulation. This is important because while the aggregate energy values of the daughters compared to the mother can give a rough idea of the endurance of the system, the number of daughters affects the transit time of the mother and can contribute to a queue of daughters waiting to be charged resulting in lost operational time even with remaining mothership capacity.

Figure 2-1 shows two typical results of this formulation. Representative values are chosen for the energy parameters and the relative values fall out of the relationships described above. Once the characteristic daughter energy is replaced with a real world

| Example 1: Sensor Buoy Swarm<br>Low Power Expenditure Scenario |            |       |                             | Example 2: Larger Capacity Vehicles<br>More Dynamic Operations |                              |
|----------------------------------------------------------------|------------|-------|-----------------------------|----------------------------------------------------------------|------------------------------|
| Dimensional Example                                            | Units      | Value | Notes                       | Value                                                          | Notes                        |
| Daughter Energy                                                | kiloJoules | 216   | Equiv to 5 Amp-Hrs at 12V   | 864                                                            | Equiv to 20 Amp-Hrs at 12V   |
| Mother Energy                                                  | kiloJoules | 10800 | Equiv to 250 Amp-Hrs at 12V | 43200                                                          | Equiv to 1000 Amp-Hrs at 12V |
| Daughter Power                                                 | Watts      | 15    | Equiv to 1.25 Amp at 12V    | 120                                                            | Equiv to 10 Amp at 12V       |
| Mother Power                                                   | Watts      | 75    | Equiv to 6.25 Amp at 12V    | 240                                                            | Equiv to 20 Amp at 12V       |
| Charging Power                                                 | Watts      | 300   | Equiv to 25 Amp at 12V      | 1200                                                           | Equiv to 100 Amp at 12V      |
| Transit Time                                                   | Seconds    | 720   | Equiv to 12 min             | 360                                                            | Equiv to 6 min               |
| Charging Time                                                  | Seconds    | 720   | Equiv to 12 min             | 720                                                            | Equiv to 12 min              |
| Discharge Time                                                 | Seconds    | 14400 | Equiv to 4 hr               | 7200                                                           | Equiv to 2 hr                |
| Number Daughters                                               |            | 5     |                             | 5                                                              |                              |
| Extension (q)                                                  |            | 5     |                             | 8                                                              |                              |
| Number Mothers                                                 |            | 1     |                             | 1                                                              |                              |
| Pi                                                             |            | 0.2   |                             | 0.5                                                            |                              |
| N                                                              |            | 0.2   |                             | 0.2                                                            |                              |
| Rho                                                            |            | 0.05  |                             | 0.1                                                            |                              |
| Kappa                                                          |            | 0.1   |                             | 0.1                                                            |                              |
| Tau                                                            |            | 0.05  |                             | 0.05                                                           |                              |

|            |
|------------|
| Given      |
| Calculated |

Figure 2-1: Representative Examples of Energy Factors and Calculated Results

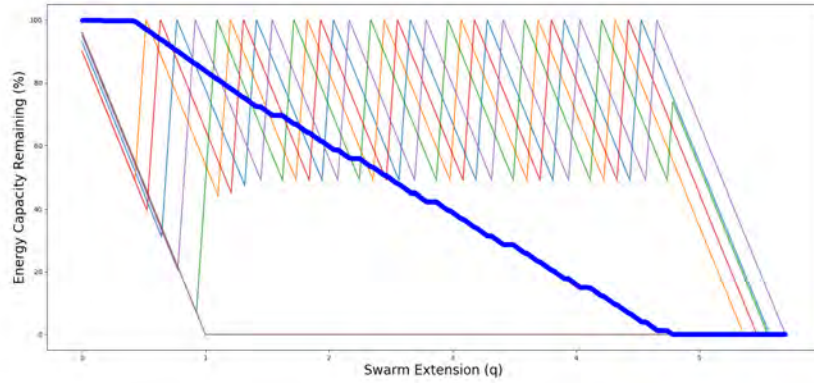


Figure 2-2: Representative Simulation

value, the rest of the parameters are determined.

## 2.2 Energy Endurance Simulations

Using the formulation of the equations above, simulations were conducted to determine the mothership's effect on  $q$ . After selecting energy factors, the simulation ran until the mothership and all daughters were depleted of energy. The energy status of each vehicle was tracked throughout the simulation as shown in Figure 2-2 along the y-axis and the energy extension factor,  $q$ , was plotted along the x-axis. The green, blue, red, orange, and purple lines represent the energy status of daughters and the

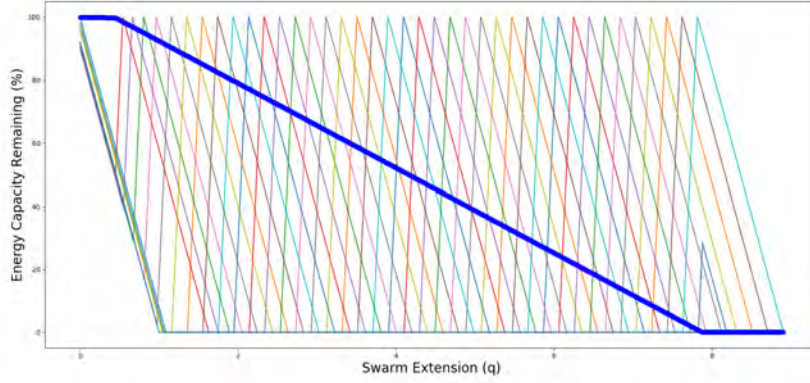


Figure 2-3: Simulation of an Overwhelmed Mothership

thicker blue line represents the status of the mothership. The brown line that quickly goes to zero and rides along the bottom represents the status of a daughter with no mothership. Figure 2-3 shows the result of a mothership less suitable for its swarm since the energy status lines for each of the daughters are routinely at zero while the daughters wait in the queue to be recharged.

## 2.3 Energy Factor Exploration

To further explore the effects of the energy factors, thousands of simulations were run with different permutations of factors. This allowed the five dimensional space to be fully understood in terms of how to improve swarm endurance with the mothership. These results were then cut into two dimensional slices for analysis and plotted as shown in Figure 2-4. This analysis showed that the selection of energy factor parameter combinations has a significant effect on swarm energy extension,  $q$ . The contour values shown are how much energy factor combinations improve  $q$ . It is important to note that this does not mean the factors that result in greater extension are necessarily the factors that result in the longest-lasting swarms. It just means that these factors have a more significant effect on *extending* the endurance compared to the same system with no mothership. This is demonstrated by comparing Figure 2-4 to Figure 2-5 which shows which systems perform best in terms of dimensional param-

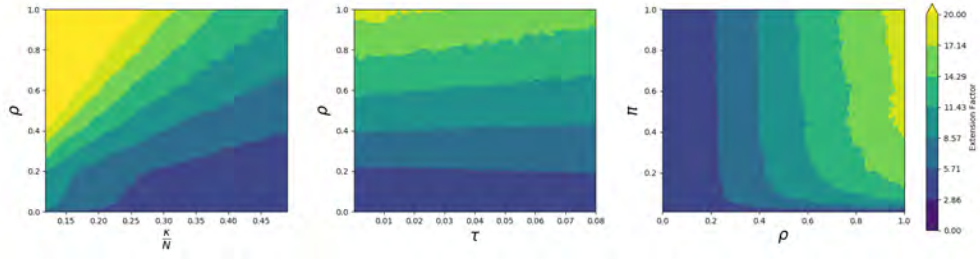


Figure 2-4: Effect of Different Energy Factors on  $q$

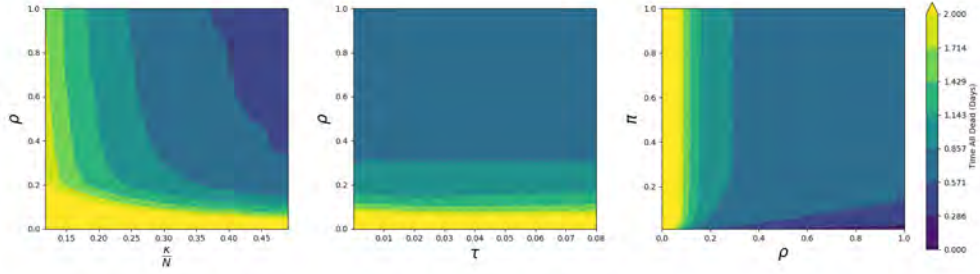


Figure 2-5: Effect of Different Energy Factors on Total Swarm Endurance in Days

eters. One can see that despite the same combinations of energy parameters being examined, the areas which are highlighted as “best” on the two figures is different.

Finally, the extension of the swarm endurance is necessary but not sufficient to provide a useful system. There are scenarios where the swarm endurance is extended by a significant factor but upon closer examination it is obvious that the individual daughters are dead while they are awaiting a mothership to come available. Thus, the average operational availability of daughters is calculated, shown in Figure 2-6.

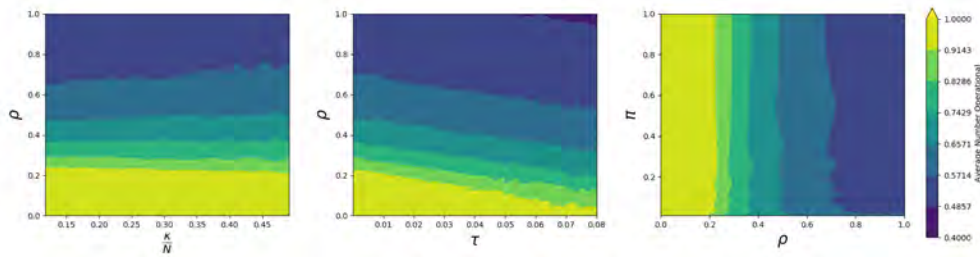


Figure 2-6: Effect of Different Energy Factors on Average Fraction of Daughters Operational (scaled from 0 to 1)

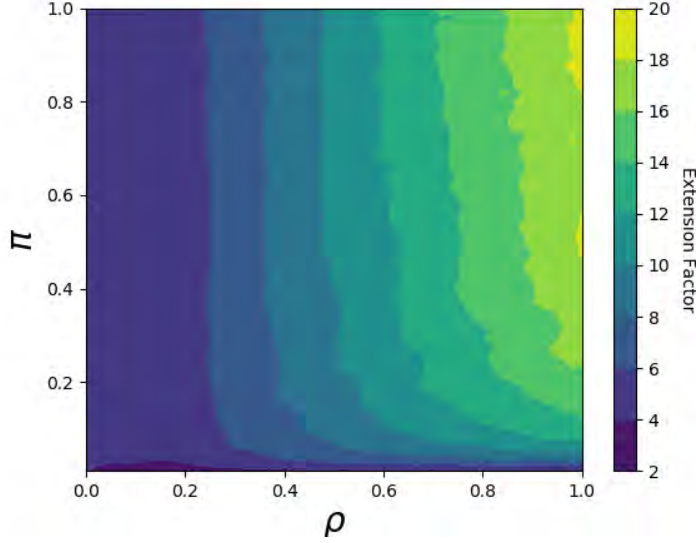


Figure 2-7:  $\kappa$  vs.  $\pi$  Effect on Swarm Extension

## 2.4 Analysis of Energy Factor Relationships

The value of several of the parameters is intuitive individually, but the combination of parameters provides insights into their combinatorial impact and the thresholds at which greater gains are made. By examining the various contour plots, it is clear that some parameters have a significant impact and others are not as critical. Figure 2-7 shows how the interactions between  $\rho$  and  $\pi$  result in an extension of swarm endurance. There are key thresholds to avoid and regions to focus on for a designer interested in this parameter. For instance, a designer hoping to extend the swarm by  $q = 6$  should avoid  $\rho$  values below 0.3 based on this collection of energy factors.

It is important to note that these are purely theoretical analyses, however, and they must be bounded within realistic terms. One prime example of this is  $\kappa$ , which measures the ratio of the daughter energy capacity to the mother's. It is clear having a very small value for  $\kappa$  is ideal, but this could quickly result in a value outside the practical range of a system. A 0.02 value for  $\kappa$  combined with a 9.2 Amp-Hour battery on the daughter results in a 460 Amp-Hour battery on the mother. This obviously gets even more drastic as the initial daughter battery capacity goes up and the fraction gets smaller. In some cases, these large mothership batteries are practical

but in others they are not. In any case, the designer should recognize the importance of placing practical limitations on energy factors. To achieve this goal in the design tool, a filter was incorporated which sets the top and bottom bounds of energy factors to be analyzed.

## 2.5 Analysis of Specific Parameters

**Battery Capacity  $E_d$  and  $E_m$**  Daughter capacity is obviously an important parameter with respect to swarm durations. Implicit in the necessity to have a mothership, though, is that the daughter battery capacity will be limited by some constraint and smaller than the mothership capacity ( $E_d < E_m$ ). Thus, while ranges of daughter energy capacities should be examined, this parameter can serve as a useful constraint on the system that can be used to derive the rest of the design parameters.

**Mothership Speed  $\nu$**  As the spread  $\delta$  of the daughters or their discharge rates increase ( $\tau$  or  $\rho$  goes up), the swarm benefits from the mothership going faster to sustain the swarm. This means, however, that the power consumed by the mothership due to transiting will increase and places emphasis on the importance of reducing hull resistance. By narrowing in on a range of suitable  $\tau$  and  $\rho$  values, one can deduce hull form characteristic limitations and propulsion system parameters for a realistic design.

**Charge Setpoint  $\psi$**  The recharge setpoint  $\psi$  can change results of the system significantly. For instance, Figure 2-8 and 2-9 depict the same combination of parameters with the exception of beginning charging at 40% battery in the first scenario and 60% in the second. Waiting to begin charging the first daughter until 40% results in a delay which cascades to the other daughters causing the system to have a non-operational daughter for a portion of time. It is also important to consider that certain battery types will suffer damage if discharged below a threshold and so keeping a buffer to the bottom limit is useful. Consideration should be given to idling vessels which approach such a limit.

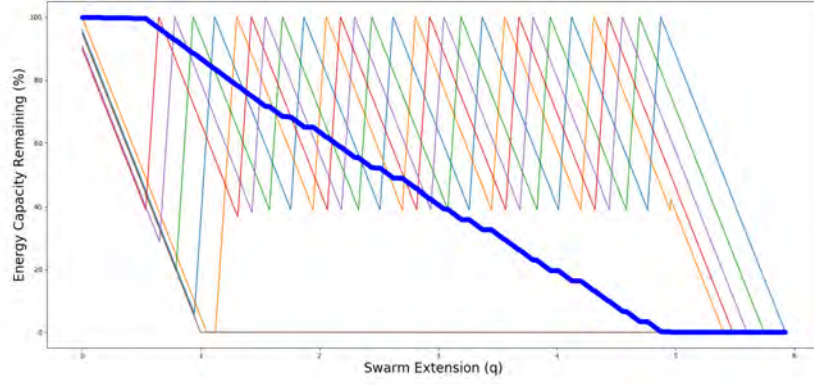


Figure 2-8: Recharge Setpoint at 40% Energy Capacity

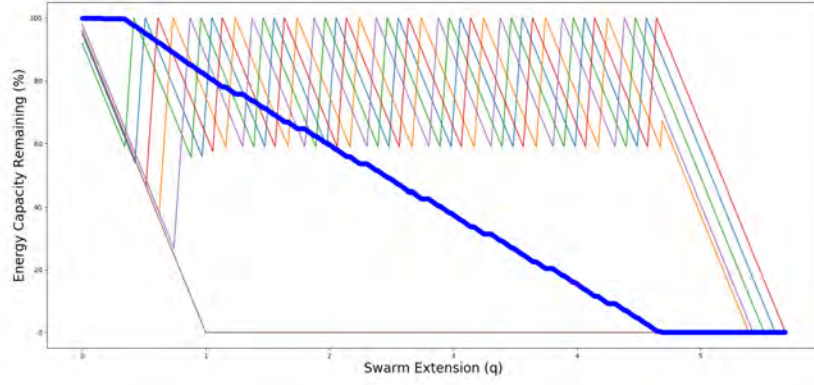


Figure 2-9: Recharge Setpoint at 60% Energy Capacity

**Charge Rate  $P_{ch}$**  The charging rate must be carefully managed to sustain the swarm for a long period of time. Although this analysis is generalized for any energy source, the prototype used in this thesis employed Lithium Polymer (LiPo) batteries which encounter significant problems if charged at too high of a rate [17]. Many LiPo batteries are designed to be charged at a rate of 1C to 5C, where C is the capacity of the battery. Thus, if the battery will be discharged at a fixed rate regardless of its capacity, one can affect the discharge/charge balance  $\rho$ . However, the total charge time of the battery will be the same since the battery charging rate scales at the same rate as battery capacity.

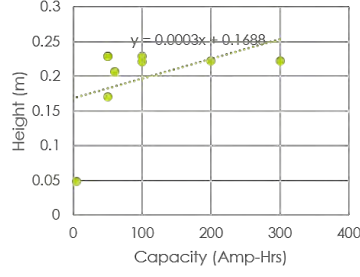


Figure 2-10: Height Parametrics

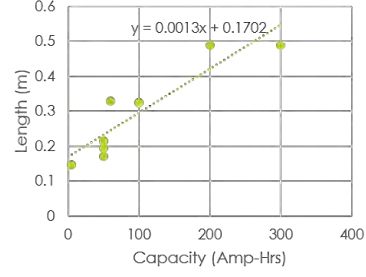


Figure 2-11: Length Parametrics

**Spread of Daughters in Space  $\delta$**  The physical spread of the daughters  $\delta$  affects the transit time of the mothership which is related to  $\tau$ . If the daughters are spread out further, either the mothership velocity  $\nu$  will need to increase (which will increase  $P_m$ ) or the daughter vessel discharge rate  $P_d$  must go down, governed by  $\pi$  and  $\tau$ . If neither of these occurs, the extension  $q$  will go down.

## 2.6 Conversion of Design Parameters to Specific Design Specifications

Once suitable ranges of energy factors are selected based on the requirements of the system, simulations can then be run by assigning specific values to energy factors and adding in dimensional constraint. The remaining mothership parameters will be calculated from the previously described mathematical relationships. This analysis can be carried further by relating the resultant mothership energy capacity to parametric equations depending on the energy type used. As an example, multiple commercially available LiPo batteries were surveyed to determine their Amp-hr capacity, weight, and volumetric dimensions. This allows a designer to translate the required mothership battery capacity derived in the energy analysis into a weight and dimensions the mothership will be required to support. The parametric data used are shown in Figures 2-10 to 2-13.

Other mothership design requirements can be derived from this analysis. One of the output parameters is the required mothership velocity  $\nu$  which can be weighed

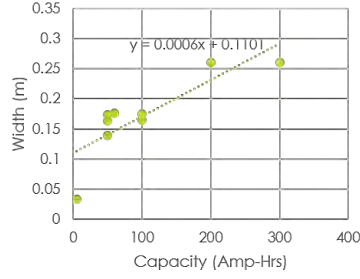


Figure 2-12: Width Parametrics

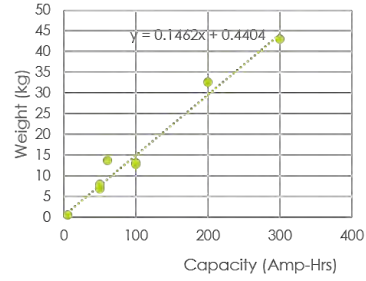


Figure 2-13: Weight Parametrics

against the hullform resistance and power output of the mothership to determine required thruster output. If a hull has already been selected, its resistance values will be essential in determining the required thrust output to achieve the velocity requirement. Conversely, if thrusters have been selected, the hullform resistance will need to be targeted based on the available thrust and required speed.

## 2.7 Design Tool

A design tool was developed that implements this analysis technique. The back-end code includes the mathematical relationships previously described and goes through iterative combinations of energy factor values to evaluate the impact of parameter combinations on improving swarm performance. A GUI, shown in Figure 2-14 is provided to explore the data and to select specific combinations of energy parameters to generate a simulation. One can also explore contour plots for ranges of factors to see which regions are most suitable for a specific design application. Lastly, the parametric equations were incorporated to provide specifications for the mothership load capacity based on the energy requirements.

## 2.8 Summary

The energy extension analysis process provides a useful way to convert the multidimensional endurance problem faced by the swarm into realistic mothership values to facilitate a design. The process is summarized in Figure 2-15.

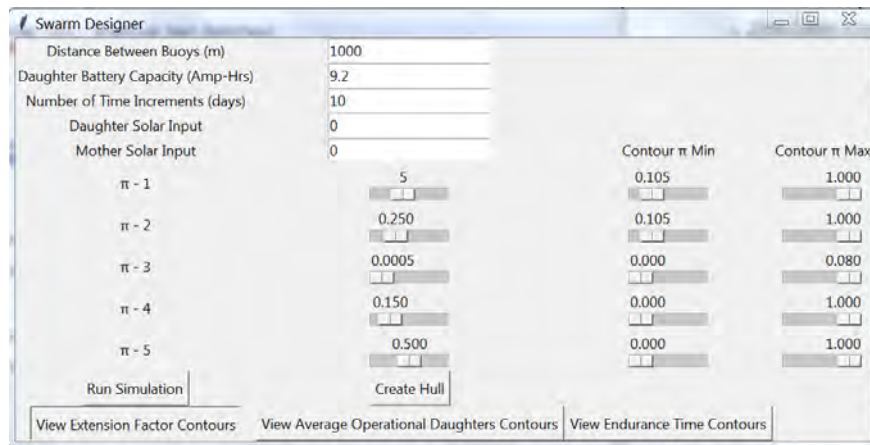


Figure 2-14: Swarm Mothership Design Tool - Graphical User Interface

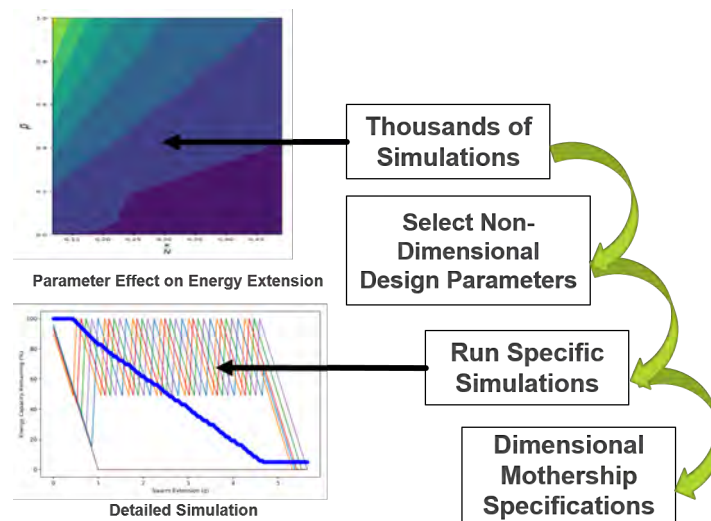


Figure 2-15: Summary of Energy Analysis



# Chapter 3

## Minimizing Relative Motions

### 3.1 Background

Seakeeping analysis is an essential component of any boat or ship design. For traditional manned vessels, standards apply to ensure motions are satisfactory for personnel or essential evolutions such as aviation operations. Unmanned vessels benefit from not having to worry about the effects of motions on personnel and are typically designed to withstand wetting or submersion, but the movement of the vessel is often still of concern. In the case of inter-vehicle docking, excessive motions between the two vessels complicates the docking & mating processes.

The problem of motion analysis for autonomous marine vehicle docking is discussed in detail by a 2015 paper by Brizzolara at the MIT Sea Grant Program [15]. In the study, a novel SWATH hull design with improved resistance and seakeeping performance is proposed for an ASV mothership conducting AUV launch & recovery. In particular, they attempt to minimize the relative vessel response to waves and note the improvement of the SWATH hull across a large range of  $\frac{L}{\lambda}$ , where  $L$  is the length of the vessel and  $\lambda$  is the wavelength. The analysis presented in this chapter seeks to investigate a similar problem, except for smaller vessels, with two ASVs, a different daughter shape, and focused exclusively on a catamaran mothership. The ASV to ASV connection is of concern based on their conclusion that excitation forces of surface waves resulting in large drift motions are significant enough that AUV recovery

should be conducted below the surface. ASVs unfortunately do not have this option and so we must seek to reduce this effect as much as possible. Docking systems must then be designed to reduce and mitigate the effects of these motions or will be limited by sea state conditions.

## 3.2 Overview of Wave Interactions

The key environmental concern is the effect of wave forces on the two bodies. This matters both in terms of their motions individually and relative to one another. The importance of wave motions and forces is dependent upon their  $\frac{L}{\lambda}$  as well as the relative positions of the two bodies [18]. Although the prototype vessels under consideration in this work were small compared to the expected wavelength, this thesis is designed to be more broadly applicable and so more generalized analysis techniques were used. Larger vessels must be analyzed for these interactions since their effect on the water due to scattering may result in additional forces which will impact docking success. The motion analysis below has been non-dimensionalized for the benefit of use on different scales.

There are three phases of docking operations worth noting related to the motion characteristics. The first is the approach of the two vessels toward one another. Large motions by either or both of the vessels can complicate the process of aligning the daughter into the docking bay. Second, once within the docking connection region, unacceptable movement can complicate or prevent connection between the two vessels. Finally, once connected, excessive stress on the connection can result in mechanical failure or disruption of electrical or data transfer. The concerns relating to the approach are mainly related to the sensor which will be described in Chapter 5 or due to the daughter, which is not within the purview of the mothership designer. In this chapter, we are concerned with the motions within the docking bay and how to design the mothership to minimize this effect.

### 3.3 Computation of Wave Forces

Since the prototype described in Chapters 4 and 5 is relatively small compared to the expected wavelength, its impact on the surrounding waves is minimal and therefore the diffraction force can be neglected. In this case, the Froude-Krylov approximation is valid and we only need to compute the incident wave force to estimate the total force [18]. However for vessels where  $\frac{L}{\lambda} = O(1)$ , more detailed analysis is required. Since this thesis was focused on a broader range of vessel sizes, we explore the topic using a more general approach.

To find the effect of waves on the two bodies, Wave Analysis MIT (WAMIT) was employed which solves the radiation and diffraction potentials of the wave problem by using the Boundary Integral Element Method (BIEM), also known as the ‘panel’ method [19]. This technique uses panels which describe the surface of the body to compute hydrodynamic coefficients including added mass, damping, forces, and Response Amplitude Operators (RAOs). The velocity potential on each panel is assumed to be constant, thus the more fine of a mesh used, the more accurate the results. The results below are based on linearized solutions, but the program does have the capability of conducting higher order analyses if continuous b-splines are used.

To model the hydrodynamic environment, WAMIT assumes a linear plane progressive wave given by the velocity potential:

$$\phi_0 = \frac{igA}{\omega} \frac{\cosh[k(z+H)]}{\cosh kH} e^{-ikx \cos \beta - iky \sin \beta}$$

With the dispersion relationship:

$$\omega^2 = gk \tanh kH \approx gk \text{ (Deep Water)}$$

Where:

$\phi_0$  is the velocity potential

$g$  is the acceleration due to gravity

$A$  is the wave amplitude

$z$  is the body position in the vertical direction

$\omega$  is the wave frequency

$k$  is the wavenumber

$H$  is the water depth

$\beta$  is the incident wave angle

**Computational Approach** Two iterations of analyses were run to determine the relative motions between the bodies. The first method was an approximation which calculated each of the bodies individually and then combined the results by offsetting the daughter to the mothership's docking bay and taking the complex difference between the two results. The second method leveraged WAMIT's built-in multi-body analysis which provides a more accurate result since it accounts for the effect of the body on the surrounding fluid which impacts the results of the other vehicle in close proximity. It was expected that for  $\frac{L}{\lambda} \ll 1$ , the two methods would provide similar results since the effects due to diffraction would be minimal.

**WAMIT Usage** WAMIT uses several input files to calculate the hydrodynamic values. First, a body shape must be described in a Geometric Data File (GDF) by specifying the four-point panels for all submerged surfaces. Symmetry can be leveraged in the  $x$  or  $y$  directions if appropriate to reduce computation time. Care was taken to make the panel size as similar as possible along different surfaces of the body and discretization size was varied to confirm the values converged toward a final result. Figure 3-1 shows representative points provided for the buoy and catamaran vertices, respectively.

The moments of inertia and VCGs for the bodies were input into a Force Control File (FRC) based on 3D models created in SolidWorks. Although variations of length were created to more accurately describe the vehicles, variations in pontoon separation were not modeled due to the large number of permutations analyzed. As a result, the VCGs and moments of inertia for different pontoon length and separation distance

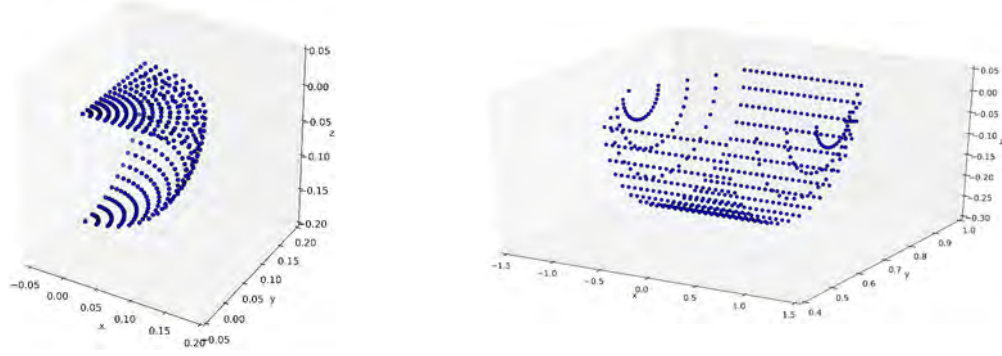


Figure 3-1: Panel Vertices Used for Daughter and Catamaran

combinations is an approximation but is representative of the process by which a mothership catamaran could be analyzed.

The final input file of note specifies the environmental parameters in a Potential Control File (POT). These include items such as the wave headings, wave periods, modes to analyze, etc. For this exercise, heave and pitch were considered from headings of 000, 045, and 090.

### 3.4 Vehicle Modeling

Minimum specifications for hull parameters can be derived from the energy analysis described in Chapter 2. The buoyancy requirements result from the energy payload required. When applied to a catamaran design, one can look at a combination of pontoon dimensions to determine the minimum vessel length. For the motion analysis, these minimum requirements were used as the basis for analyzing a range of pontoon length and separation combinations for vessel response.

First, the daughter vessel's surface panels were modeled using Python and saved into a '.gdf' text file. The hemispherical shape was relatively simple to model due to its straightforward geometry. Since WAMIT's symmetry feature was leveraged in two dimensions to reduce computation, only a quarter of the vessel was modeled. By running WAMIT first on this model, results were able to be compared against previously validated models for a submerged hemisphere. The added mass and damping for a hemisphere in heave were compared to the results found in Hulme's 1982 paper

[20] and found to be in agreement.

This method was then replicated using the same procedure, but instead producing a cylinder with cones on one end to represent the mothership pontoons. The cylinder and cone were given an offset equivalent to half the separation from the other pontoon and the shape was mirrored by using the symmetry feature in WAMIT across the long axis of the boat. Sets of pontoon lengths and separations were combined with an assumed diameter of 0.25 m into permutations to generate a selection of possible hullforms based on these shapes. Usage of Python scripts to prepare input files and execute WAMIT made looking at many permutations relatively straightforward.

### 3.5 Computation of Relative RAO: Approximate Method

The first analysis was conducted using the approximate method described above. This method, which ignores the effects due to other bodies, was used to provide a comparison against the WAMIT multi-body method in order to highlight the importance of two-body interaction in docking. It also provided an easy way to analyze different positioning of the bodies using post-processing calculations rather than having to run WAMIT again for each new position. After calculating the motions for the catamaran variants and daughter individually, results were compared to one another by finding the difference at each frequency of the complex RAOs between the daughter and each mothership variant. Heave calculations for the daughter were conducted with it offset to the mothership's docking bay. To compare the results of the hemispherical body with those of the catamaran, the absolute value of difference between the coupled complex pitch and heave RAOs was computed as shown below. Of note, the pitch analysis for the hemisphere was not included due to the axisymmetric nature of its geometry. Final calculations were made using the following formulation:

$$RAO_{rel} = RAO_{cat3} + RAO_{cat5} \cdot x - RAO_{hemi3} \cdot e^{ikx} \quad (3.1)$$

Where:

$$RAO_{rel} = \text{Relative RAO}$$

$RAO_{cat3}$  = Complex Catamaran RAO in Heave

$RAO_{cat5}$  = Complex Catamaran RAO in Pitch

$RAO_{hemi3}$  = Complex Hemisphere RAO in Heave

$x$  = Distance from origin to docking bay

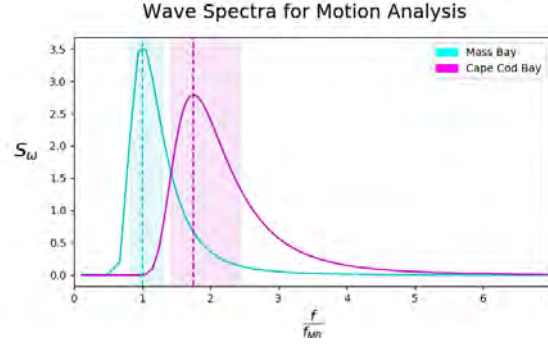


Figure 3-2: Wave Spectra for Massachusetts Bay and Cape Cod Bay, where  $S_\omega$  is the spectral density,  $f$  is the wave frequency, and  $f_{MB}$  is the peak frequency for Massachusetts Bay

**Wave Spectra** Ultimately the relative difference between the hemisphere and the catamaran are most impactful where the energy of the environment's wave spectrum is large. Thus, representative spectra were added to the analysis to determine these regions of interest. Since the prototype in this thesis was focused on a coastal environment, the Bretschneider Spectrum with a characteristic wave height for Massachusetts Bay and Cape Code Bay was used. The spectra are seen in Figure 3-2 with bands that indicate the width of the half-peak for each spectrum.

**Results** Individual heave and pitch response for the mothership as well as heave for the daughter are shown in Figure 3-3. Variations in response are found between the different pontoon lengths and separation distances. In particular, the shorter variants exhibit a resonant response close to the Massachusetts Bay and Cape Cod Bay wave spectra indicating their unsuitability for those environments. The relative RAO using the approximate method is shown in Figure 3-4. This analysis further

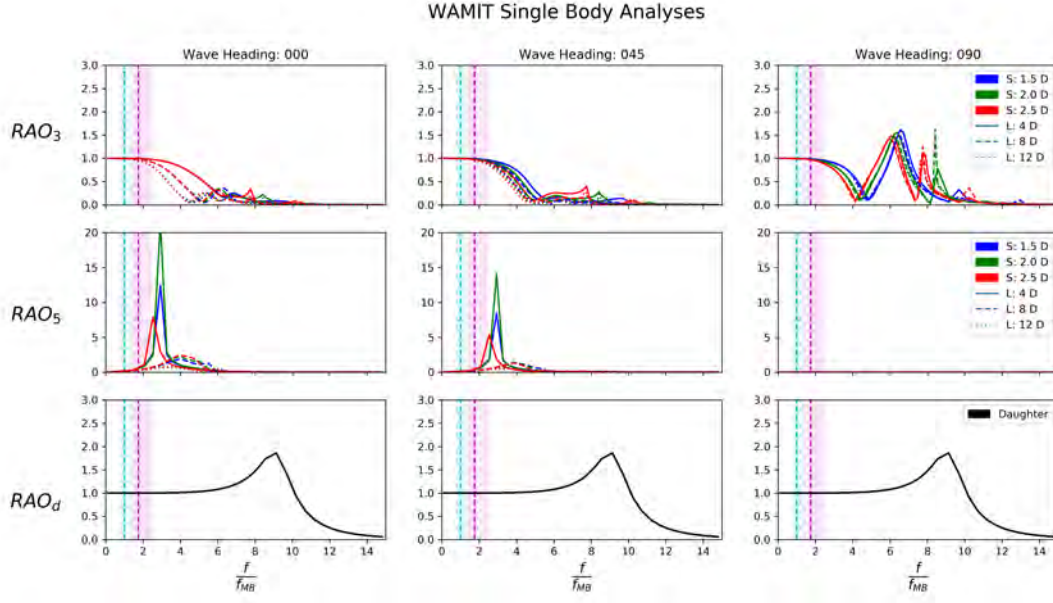


Figure 3-3: WAMIT Single Body Analyses, where  $RAO_3$  is the mothership RAO in heave,  $RAO_5$  is the mothership RAO in pitch, and  $RAO_d$  is the daughter RAO in heave

clarifies permutations of hull parameters which are more optimal for minimizing vessel motions.

### 3.6 Computation of Relative RAO: WAMIT Multi-Body Method

WAMIT includes the ability to calculate hydrodynamic effects with multiple bodies. This allows the interaction of one body with the surrounding water to be taken into account for another object close by. The methodology for conducting this analysis is similar to the approximate method except that global gdf and frc files are used to describe the individual bodies and their relationship to one another. In addition to the modes to be analyzed, the global gdf file defines the global coordinate system and how the other bodies relate to it. The global frc file contains VCG and moment of inertia data for both bodies. The end result are hydrodynamic coefficients for both bodies which can then be compared in the same manner as the approximate method

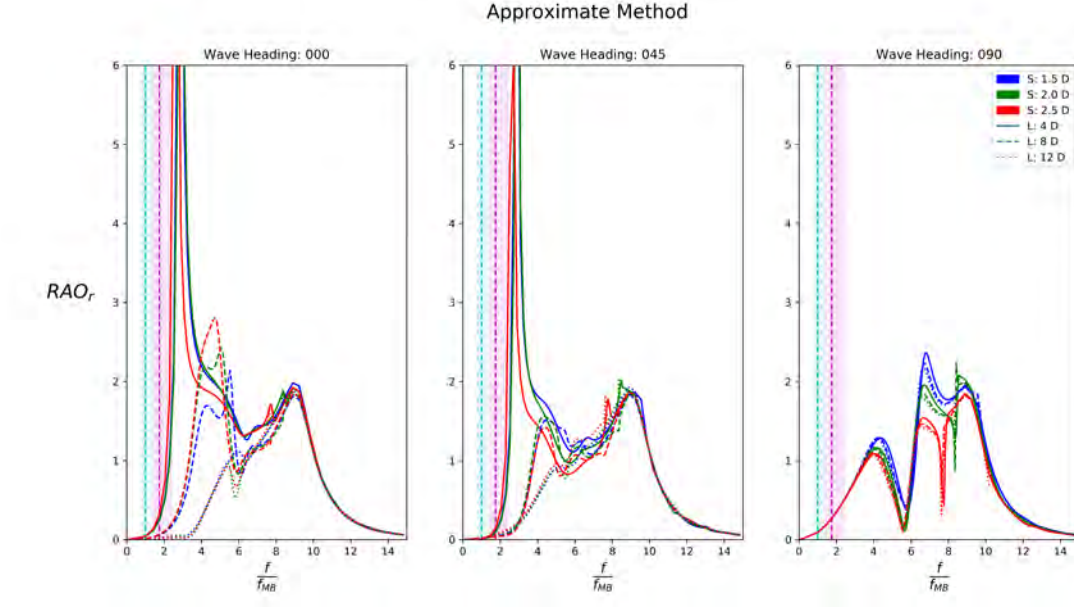


Figure 3-4: WAMIT Approximate Method, where  $RAO_r$  is the relative RAO between the daughter and mother,  $f$  is the wave frequency, and  $f_{MB}$  is the peak frequency for Massachusetts Bay

except that the daughter's position is already accounted for via WAMIT. The primary difference between the two calculation outputs is that the WAMIT multi-body method incorporates the effects of one body on the other.

**Results** As seen in Figure 3-5, the shorter variants again exhibit a resonant peak due to pitch near the spectra. As expected, the multi-body results follow the approximate method somewhat closely for the lower frequencies, but are more complex and often larger at the higher frequencies. The mothership is largely unaffected when the multi-body analysis is included, but due to the daughter's smaller size relative to the wave and the mothership, it exhibits a much greater response at higher frequencies. The results of the analyses led the selection of a longer mothership to prevent excessive motions during docking operations in these environments. Less emphasis was placed on the pontoon separation since the response in the two wave spectra was minimally impacted by this spacing.

Motion analysis is an essential step in designing an autonomous maritime docking system. The comparison of the two calculation techniques above highlights the

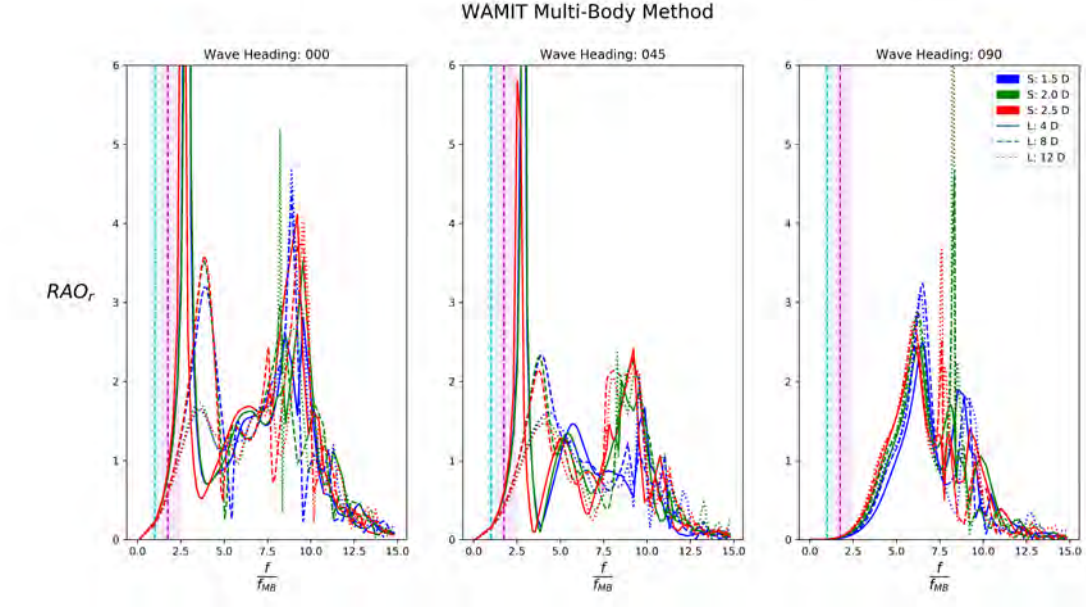


Figure 3-5: WAMIT Multi-Body Method, where  $RAO_r$  is the relative RAO between the daughter and mother,  $f$  is the wave frequency, and  $f_{MB}$  is the peak frequency for Massachusetts Bay

importance of analyzing for these motions and taking into consideration the inter-vehicle effects. Although the responses were somewhat consistent for low frequencies, there are significant differences at the higher frequencies which can affect system performance.

# Chapter 4

## Prototype Design and Construction

This chapter details the design and construction of a prototype to illustrate the feasibility, challenges, and areas of potential future research interest for autonomous marine swarm motherships. The discussion below includes a description of the hull, support structure, propulsion system, and a majority of the electronics. The docking system including camera, computer vision processing, and docking Proportional, Integral, Derivative (PID) controller are all described in the next chapter.

The prototype design process focused on producing a mothership at the lower end of size, weight, and cost in accordance with the guidance outlined in Chapter 1 and followed the processes which were described in Chapters 2 and 3. To quantify the Chapter 1 philosophy, the dimensions were limited to that which could fit in a standard U.S. van, the Ford Transit, which has a cargo space of approximately 3.2 m long, 1.3 m wide, and 1.4 m tall [21]. Weight was limited to what could be reasonably carried by three to four individuals, roughly 80 kg. Finally, cost was limited to \$4,000 for the prototype with the understanding that follow-on production would be cheaper.

### 4.1 Daughter Vessel Description

For design and testing purposes, one of the small marine swarm vehicles described by Zoss et al. [1] was used as a daughter vessel, shown in Figure 4-1. These hemispherical mobile buoys are approximately 30 cm in diameter and have about 10 cm in freeboard.

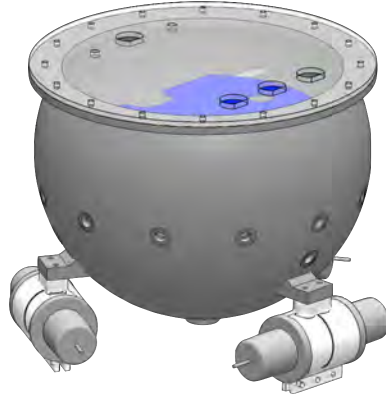


Figure 4-1: Daughter Vessel Developed by Zoss et al., 2018 [1]

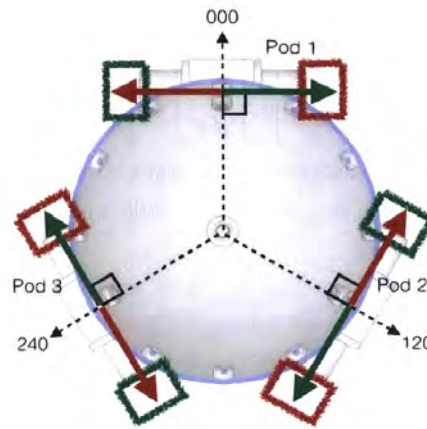


Figure 4-2: Daughter Vessel Thruster Configuration [1]

The buoy is propelled by six podded propulsors which allow it to turn with a very small turning radius and field tests have shown it to operate with a mean speed of 0.7 m/s. A weight was suspended from the bottom to lower the center of gravity and improve pitch/roll motions. Control of the daughter leverages GPS and compass inputs combined with swarm behavior rule sets for automatic control or a Graphical User Interface (GUI) for manual control. Regardless of control method, a goal bearing is compared to the compass heading to determine thruster output.

The electronics suite on the daughter consisted of a single board computer in the form of a BeagleBone Black with connections to an Adafruit Ultimate GPS, LSM330 compass & accelerometer, and XBee Pro wireless communications device. All electronics were mounted on a custom PCB for convenience. Two 5000 mA-hr

Lithium Polymer batteries supplied approximately 12V to the propulsion system as well as 3.3V and 5V voltage regulators which fed the other electronic components. XBee communications facilitated inter-vehicle or user control and status updates. Commands were passed in the form of strings and decoded by the receiver using a dictionary of words with their associated actions.

## 4.2 Application of Mothership Design Process

In order to provide a representative demonstration of the energy and motion analyses, a mothership design was chosen which could extend five 20 Amp-hour capacity daughters spread an average distance of approximately 500 m by a factor of  $q = 3$ . By selecting values on the Energy Factor contour plots discussed in Chapter 2, a simulation was generated which modeled the desired outcome. After running multiple simulations to tune parameters, final energy requirements were generated for the mothership. The 300 Amp-hour capacity was compared to the parametric data for batteries to determine the required payload weight and geometric dimensions. The input and output parameters of the energy analysis are shown in Table 4.1. The final simulation is shown in Figure 4-3. After validating that these were consistent with size, weight, and cost design philosophy criteria, these values were established as the baseline mothership design specifications.

After using the energy analysis to derive the weight of the payload and including the additional weight of structural and electronic components, combinations of pontoon length and separation were considered which could provide the required buoyancy while ensuring adequate motions in accordance with Chapter 3. In the end, pontoons were selected based on what was easily commercially available, but the length and separation combination was found to be suitable for the scenario based on the Chapter 3 analyses.

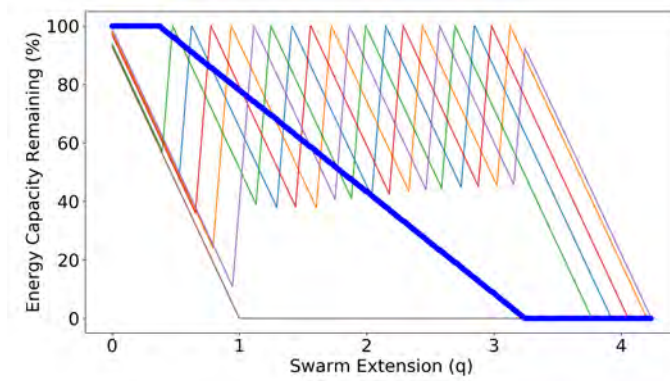


Figure 4-3: Simulation Used to Determine Design Parameters

Table 4.1: Design Specifications

| Specification                           | Value                |
|-----------------------------------------|----------------------|
| <b>Given</b>                            |                      |
| $N$                                     | 5                    |
| $\kappa$                                | 0.066                |
| $\tau$                                  | 0.024                |
| $\rho$                                  | 0.175                |
| $\pi$                                   | 0.325                |
| Average Distance Between Daughters      | 500 m                |
| Daughter Battery Capacity               | 20 Amp-hours         |
|                                         |                      |
| <b>Calculated Values for Mothership</b> |                      |
| Mothership Speed                        | 1 m/s                |
| Average Daughter Power Draw             | 3.5 A                |
| Average Mother Power Draw               | 10.76 A              |
| Battery Capacity                        | 300 Amp-hours        |
| Battery Weight                          | 132 kg               |
| Battery L / W / H                       | 0.56 / 0.29 / 0.26 m |

### 4.3 Hull Structure

To achieve the design specifications for hull shape, modest materials were planned for the prototype consisting of PVC pipes connected with T-slotted framing. The PVC pipes were to be of a large diameter, approximately 0.2 to 0.3 m, in order to provide the buoyancy required to support the catamaran and would have fairings at the ends to reduce hydrodynamic resistance. However, a modular kayak company was found named ExpandaCraft which produces pontoons that can be separated into smaller (1.2 m) pieces as seen in Figure 4-4. Since this pontoon hull structure very closely approximated the original design and the modularity features supported the goal of transporting the mothership within the back of a van, ExpandaCraft components were chosen for the mothership pontoons. These pontoons provided several additional



Figure 4-4: Modular Mothership Hull from ExpandaCraft

advantages including being relatively inexpensive, lightweight, and easy to construct. Two center and two bow pieces were connected via T-slotted framing to construct the 2.4 m catamaran hull.

The pontoons of the catamaran provided the structure for the docking bay, located at the forward end of the mothership. The pontoons were separated by approximately 0.75 m at their forward tips and about 0.45 m at their closest point in the center of the vessel. The wide opening allowed the 0.3 m diameter daughter vessel enough room for error to come into the entrance while funneling it into a narrower dimension further into the mothership. A panel was placed at the end of the docking bay to stop the daughter from being swept underneath the mothership. The panel was of a mesh construction to allow waves and wind to pass without excessive interference. The docking camera mount was installed directly over the mesh to align the daughter into the bay.

**Support Structure and Battery Enclosure** To connect the pontoons together and provide structure to support the battery box, one inch T-slotted framing was used.

Four risers connected to the pontoons provided an attachment point for the framing. The one inch frames were made of aluminum which provided excellent strength and rigidity while reducing weight and facilitating modularity. In addition to ensuring the structural integrity of the vessel, cross beams provided a platform to mount the battery box. The box utilized was a marine grade battery box sized at 0.62 x 0.38 x 0.33 m, more than enough to hold the expected battery and provide reasonable protection for the electronics. Bolts with wing nuts connected the battery box to the framing and made the box easy to remove and reattach for transportation. The enclosure was not waterproofed since there was sufficient freeboard to make wetting unlikely, but a final design should incorporate greater protection from the elements.

## 4.4 Electronics

The baseline for the mothership's power and electronics suite was derived from Zoss' buoys[1]. Maintaining this configuration was useful in that it allowed focus to remain on the central tenets of the thesis and also ensured that the replication of these systems ensured that the entire system of mothership and daughters was more maintainable by using redundant components and code. The electronic contributions from this thesis were those that enabled the capability for the two vessels to dock. This required the addition of a camera as a sensor, new thrusters, modifications to the power system, and code to handle the docking process.

As with Zoss' buoys, the electronics suite for the mothership was centered around a BeagleBone Black. These devices are similar to an Arduino or Raspberry Pi but with slightly different capabilities. Two of its biggest advantages are its large array of input/output options and ability to run Linux [22]. In addition, usage of a BeagleBone provided sufficient computational power to handle the image processing algorithms required for the mothership. Additionally, the presence of a USB port and Linux's Video for Linux (V4L) protocol enabled a simple plug and play capability for video streaming.

The BeagleBone also presented several challenges throughout the development

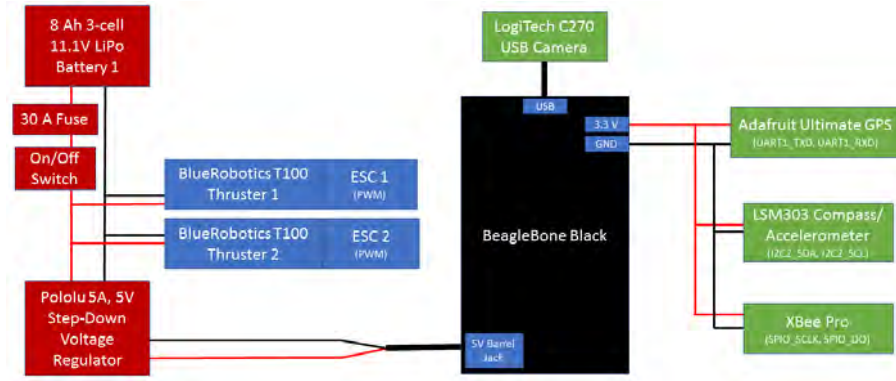


Figure 4-5: Electronics Configuration

process. Since the daughters were used as the baseline electronic configuration, it was unclear at the outset of the thesis whether it was preferable to attempt to maintain its original state including Linux and Python versions or to update everything. Efforts were made along both vectors and difficulties were experienced from attempts to keep the old configuration and those to make a new one. Examples of these problems were difficulty enabling pins in certain BeagleBone Debian Linux versions, differences in processing latency, and availability of older or newer Python packages. Ultimately, the older configuration was maintained because of a latency issue experienced with the newer Debian Linux distribution. It is likely the latency could be improved which would provide more capabilities from newer Python packages, but the older versions were acceptable to accomplish the objectives of the thesis.

As shown in Figure 4-5, a Digi XBee Pro wireless device, Adafruit Ultimate GPS, and LSM303 Compass & Accelerometer were connected via the indicated pins on the BeagleBone. The camera, a LogiTech C270 webcam, was connected to the BeagleBone's USB port.

Although the mothership was sized for a 300 Amp-hour battery, power to the BeagleBone was provided through a 5V regulator connected to a 8 Amp-hour, 3S Lithium Polymer 11.1 V battery. The 8 Amp-hour battery capacity imposed a very limited endurance on the system due to the current draw of the thrusters, but was acceptable for testing purposes by limiting the range of thruster output and only using the thrusters as necessary for docking tests rather than continuous operations.

Connected in parallel to the BeagleBone were two additional electrical loops to provide power to the thrusters via Electronic Speed Controllers (ESCs). Since each thruster was rated for 12 A, these loops were designed to withstand greater current flow than the BeagleBone loop. A 30 A fuse was connected in line with the outlet of the battery to provide over current protection and a switch was positioned next to the fuse to control power for the whole system.

## 4.5 System Code

The mothership’s electronics and docking system were programmed using Python version 2.7. The baseline control code for the mothership was taken directly from the daughter control code since the majority of its functionality was the same as the daughter. As with the BeagleBone configuration, Python packages included on the original daughter were largely maintained to avoid configuration errors. Major differences included the thruster control, XBee message dictionary, and the inclusion of a camera docking control class. Details of the daughter code are described in [5] and differences for the mothership can be found in Appendix A.

To handle the different functions of the mothership than what the daughter required, additional control words and actions were added to the XBee dictionary to be processed on the mothership. These new functions, shown in Table 4.2, facilitated docking and tried to enable the change of parameters during testing as much as possible. For instance, PID values could be changed and passed via the XBee during docking tests to avoid having to SSH back into the mothership’s Beaglebone and change hard-coded values. The GUI developed by Zoss was modified to reflect the different functionality. In cases where the mothership and daughter both required a function but needed to be controlled separately, duplicate functions were made which were interpreted by the appropriate vehicle and ignored by the other. For instance, a signal to turn was not desired to control both the mothership and daughter at the same time. Some features used by the buoy swarm but not required docking were removed to improve usability, but should be restored for an operational system. Others

Table 4.2: Graphical User Interface for Buoy and Mothership Control

| Function             | Use                               |
|----------------------|-----------------------------------|
| GUI Dock             | Commence docking using laptop     |
| Daughter Dock        | Commence docking using mothership |
| Mother Thrusters On  | Arm mothership thrusters          |
| Mother Thrusters Off | Disarm mothership thrusters       |
| Kp                   | Set docking Kp value              |
| Ki                   | Set docking Ki value              |
| Kd                   | Set docking Kd value              |
| Auto Heading On      | Enable auto heading               |
| Auto Heading Off     | Disable auto heading              |
| Kpt                  | Set auto heading Kp value         |
| Kdt                  | Set auto heading Kd value         |

similarly were simply added for testing purposes such as “GUI Dock” which uses the laptop to dock the buoy.

## 4.6 Propulsion System

Two BlueRobotics T100 thrusters were used which each provided 23 N of forward thrust and 18 N reverse thrust [23]. These output sufficient force to overcome the Charles River current and allow the mothership to station-keep for docking operations.

Control of the thrusters was through the BeagleBone, which provided PWM signals to each ESC using the Adafruit\_BBIO.PWM library. A signal was first sent to start the thruster in the stopped position. Subsequently, the duty cycle was updated to enable the thruster to move in the forward or reverse direction. To stop the thrusters temporarily, the duty cycle was reset to the neutral position and to stop the thrusters for upon completion of operations, a deliberate stop signal was sent and the PWM pins were reset. If the thrusters were still running when power was lost, the PWM pins required reset. Thus, the code was designed to stop, cleanup, start, and set the duty cycle for each new program run.

The thrusters were mounted using a double piece of 1 in T-slotted framing attached to the ends of T-slotted framing as shown in Figure 4-6. The double piece provided additional rigidity to prevent excessive bending of the thruster struts. The T-slotted framing provided the additional advantage of allowing the thrusters to be raised or lowered for storage or operations. Thrusters were placed to sit outside of the pontoons which allowed a better moment arm to assist turns and also facilitate greater water



Figure 4-6: Thruster Configuration

flow over the propellers than if they were mounted inboard.

Two different thruster control algorithms were implemented. The first method allowed for manual control of the vessel to facilitate docking testing. Using a command from the GUI, an XBee signal was sent to the mother with instructions to either increase the speed of both thrusters, decrease the speed of both thrusters, or to decrement one side while increasing the other to enable it to turn. Each forward, backward, or turn command would result in another increase or decrease of the thruster output until a limit was reached at the top or bottom of the allowable duty cycle band. For testing purposes, the output band was limited to avoid drawing excessive current which would reduce battery endurance and testing time.

The second propulsion method implemented was an auto-heading-keeping feature which allowed the mothership to remain pointed in the same direction while enabled. This greatly improved the docking process in the field and was essential to dock successfully in the presence of wind, waves, and current. To implement this, the compass heading at the time of initiation was set to be the goal bearing and subsequent compass readings were compared to determine heading drift. Based on the relative heading difference and a proportional coefficient, a controller directed a left or right turn to return the vessel to its desired heading. Derivative control ensured the mothership slowed its turn as it approached its goal to minimize the overshoot or increased its turn back toward the goal if turning too far in the wrong direction.

**Construction** Due to the modular design, a significant portion of the physical construction was relatively straightforward. Use of T-slotted framing made it easy to try different structural configurations and mounting techniques for the electronics, camera, and thrusters. Due to the ability to leverage preexisting work done by Zoss, the power system, baseline electronics, and mothership software framework was simplified. The only custom modification required for the structural components were holes drilled in an L-bracket to mount the thrusters. The bulk of time and effort during the prototype development phase of the thesis was focused on improving the computer vision, docking control code, and thruster management.

**Final Design Summary** Ultimately, a suitable design was completed which met the dimension, weight, and expense criteria of Chapter 1. The mothership pontoons can support approximately 75 kg which correlates to around a 300 A-hr battery when all other weights are accounted for. By disconnecting the front sections of the pontoons, the completed mothership is 1.2 x 1.2 x 0.5 m and can fit comfortably in the back of a standard U.S. van. Wheels were added to the top of the frame to improve transportability. Once the electronics box wing nuts are removed, the box slides off, allowing the mothership to be turned over to ride on its wheels. Although this was extremely convenient, care had to be taken not to lose bolts during the transit due to vibrations. The final cost of the prototype was approximately \$1,600. If a full size battery, docking connection, and recharge system were incorporated, the estimated cost would be \$2,000 to \$2,500. The constructed prototype is shown in Figure 4-7.



Figure 4-7: Final Mothership

# Chapter 5

## Docking System

Several different ASV docking techniques have been employed by previous researchers including those based on sonar, vision, radio, and electromagnetism. For this thesis, computer vision algorithms were used due to the wide availability of inexpensive cameras and the expansive collection of open source resources for computer vision processing. Focusing on vision also provided an opportunity to exploit an advantage ASVs hold over AUVs by being able to leverage vision at longer distances. To track the daughter vessel, Open Computer Vision (OpenCV) was used in its Python formulation.

The docking system on board the mothership follows a unique paradigm whereby the mothership tracks the relative position of the daughter vessel and sends steering orders to guide the daughter into the docking bay. This use of offboard sensors to guide the daughter vessel stands in contrast to more common techniques employed to dock marine vehicles which leverage organic sensors for docking. For instance, the ASVs ‘ROAZ’ described by Martins, et al. and ‘Starship’ by Dunbabin, et al. both use vision tracking from an ASV camera to guide themselves to a surfaced AUV for docking [10, 11]. Maki et al. and Bellingham both describe AUVs with computer vision capability which home in on light sources to dock [13, 9]. The mother-controlled model was employed to take advantage of the daughter vessel’s greater maneuverability while ensuring that as much system complexity was shifted to the mothership as possible. This paradigm also allowed the mothership to remain

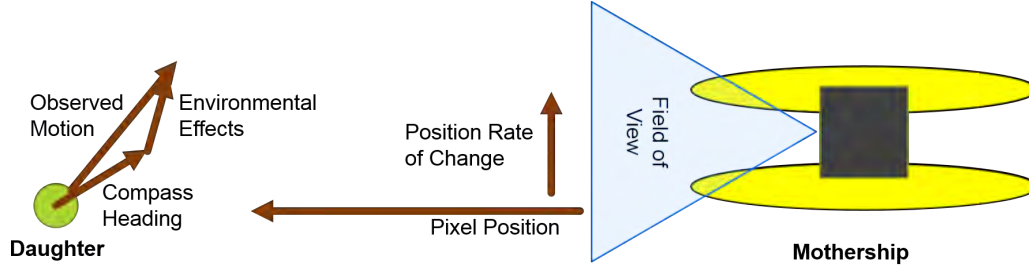


Figure 5-1: Docking Paradigm

positioned in a sheltering orientation for the daughter vessel to enter the docking bay.

## 5.1 Process Overview

Upon reaching a low-power setpoint, a daughter vessel requests recharge from the mothership via the swarm communications network. The mothership then uses its compass heading along with both vehicles' GPS locations to travel to the daughter's location. This allows the daughter vessel to continue to operate while taking advantage of the mothership's greater long distance mobility. Once within 10 m of each other, the mothership begins heading-keeping and the daughter vessel begins its approach towards the docking bay using a heading calculated from its heading and the GPS locations of the vehicles.

As the daughter vessel approaches, the mothership's camera tracks its location on the screen using OpenCV detection and tracking techniques. Based on the tracking box's lateral offset from the center of the screen and its rate of movement across the screen, the mothership communicates steering orders to the daughter vessel to guide it into the bay as depicted in Figure 5-1. The steering orders are generated using PID control to account for vessel movement and constant errors over time. This process continues until the vessel is within the pontoons and a command is sent to end the docking process. For the conduct of the thesis, no automatic docking completion signal was implemented so user input was required to observe the completion of docking and end the process from the GUI. To implement the docking system, the mothership was equipped with a webcam, gimbal mount, computer vision code, a

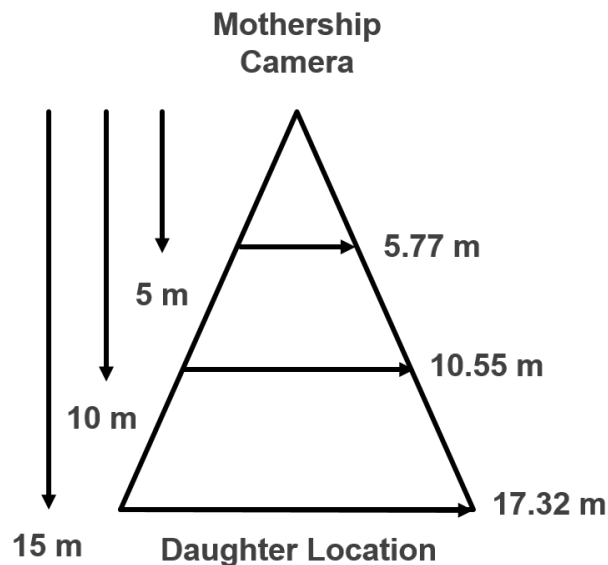


Figure 5-2: Vision System Operational Window

PID controller, and an XBee wireless communications device.

## 5.2 Camera

The LogiTech C270 was chosen for the prototype due to its compatibility with BeagleBone and Linux as well as its very low cost of less than \$30.00. Its limitations included not being designed for the marine environment, a limited field of view, and a lower resolution than some other cameras on the market. These disadvantages were deemed to be acceptable for the construction of the prototype since significant wetting was not expected and meaningful testing could occur with the existing field of view. The camera was connected directly to the BeagleBone via its USB connection and the camera frames were easily accessed using basic Python OpenCV tools.

The C270 has a resolution of 720 p, 30 fps, 60 degree field of view and a fixed focus. The limited field of view meant that if the daughter vessel were crossing ahead of the mothership at 90 degrees to the camera's view, the maximum distance available that the daughter would remain in the frame was 5.77, 10.55, and 17.32 m for ranges of 5, 10, and 15 m each as depicted in Figure 5-2. This meant that for shorter acquisition

ranges where the daughter was approaching at a large angle, the mothership needed to exercise sufficient control in a short period of time or risk losing the daughter from its view.

## 5.3 Video Frame Stabilization

As discussed in Chapter 3, the movement of the mothership's sensors complicates the docking procedure. In this case, a camera mounted to the hull structure is subject to some of the worst motions anywhere on the vessel due to the increased moment arm as a result of it being mounted high. This is obviously unacceptable in a vision system where the position of the daughter in the frame will be used to determine control system actions (Figure 5-3).

The issue of camera stabilization is of concern for multiple fields including general photography and videography as well as other related fields like UAVs. Some of the methods employed to mitigate the camera motion are:

1. Passive Mechanical Stabilization. Mounts designed to damp out motions (not tested).
2. Electro-mechanical Stabilization. Mounts can also be used to actively maintain the original orientation of the camera. Sensors detect the motion of the mount and turn servo motors to maintain the planes in their original orientations (Figure 5-5).
3. Video Processing. By finding key points in a video frame and matching them to later frames, one can rotate the image to maintain its original orientation. If part of an image is missing because it has rotated out of the frame, that section is cropped while the rest of the image is rotated to remain along its original axis (Figure 5-4 based on implementation by Spannbaauer [24]).
4. Accelerometer Corrected. Inputs from an accelerometer on the movement of the camera can be used to rotate the camera frame based on the actual movement observed (not tested).

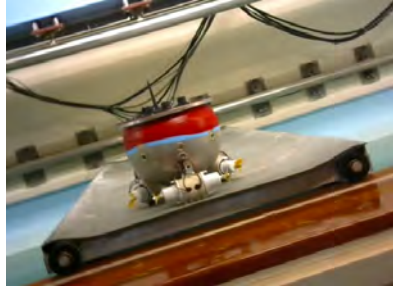


Figure 5-3: No Correction

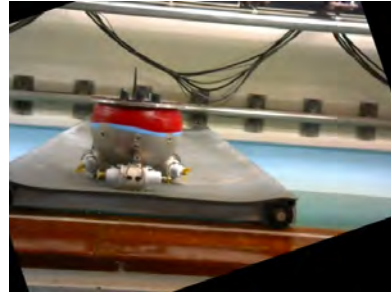


Figure 5-4: Video Processing

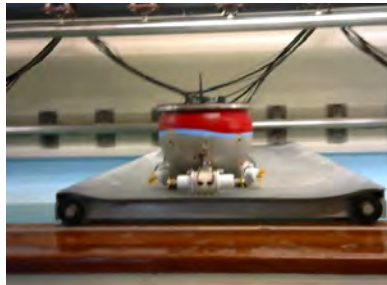


Figure 5-5: 3-Axis Gimbal

For the prototype, a FeiyuTech G6 3-axis gimbal was employed since the video processing and accelerometer correction methods would introduce undesired latency while post-processing the video frame and would also still result in uneven sampling of the desired field of view, as demonstrated in the cropped image in Figure 5-4. The passive gimbal would have insufficiently mitigated the motion and so an electro-mechanical gimbal was installed on the prototype, shown in Figure 5-6. Pitch and roll motions were greatly reduced, though not eliminated, by use of the gimbal. Although a 3-axis gimbal was purchased, a 2-axis gimbal would have been more appropriate since the camera should remain aligned with the long axis of the mothership during yaw motions. The gimbal had a setting to only compensate for pitch and roll, but sometimes it would wander in yaw because the mount was not physically locked in that orientation. The end of the gimbal handle was pre-bored with a standard bolt fitting, simplifying fixation to the mothership hull structure.

The gimbal represented one of the more expensive purchases for the prototype and was less aligned with the inexpensive philosophy. Since the mothership is already equipped with an accelerometer, it one might consider fabricating a 2-axis gimbal



Figure 5-6: 3-Axis Gimbal Mounted to Mothership

with two servos for a more inexpensive solution.

## 5.4 Open Source Computer Vision

OpenCV contains tools to input, evaluate, manipulate, and output visual information. These capabilities were extremely useful to enable the mothership to determine the error in position of the daughter vessel in order to determine the required control actions. The computer vision processes required for docking included both detection and tracking of objects. Baseline code for the vision system was derived from tutorials presented by Rosebrock at PyImageSearch.com [25, 26]. The final mothership code is provided in Appendix A.

Detection and tracking in computer vision differ in technique and results. Detection is useful when one desires to initially find an object in an image. While you could continue to detect the image in each new video frame, detection algorithms are often computationally expensive, resulting in greater resource usage and latency [27]. Additionally, detection algorithms fail to take advantage of the information gained in the previous frame. This means that dynamic parameters are ignored, which could otherwise be leveraged to anticipate where the object is in the next frame. Tracking tools take advantage of this information and are generally employed after the initial object detection.

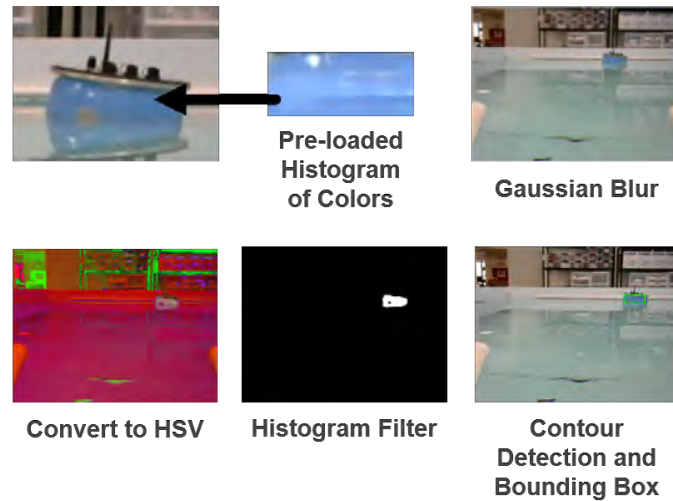


Figure 5-7: Mothership Computer Vision Detection Process

**Visual Detection Design** To find the daughter upon initiation of docking, the design employed a relatively simple technique which used color segmentation applied to the frame. The first step taken in the method was to pre-load an image of the daughter's most prominent color features. This image was then processed to calculate a histogram of colors into a series of color bins.

Once the video stream began, the frame was loaded and pre-processed to prepare it for object detection. The pre-processing steps included a Gaussian Blur to eliminate noise in the image and conversion to the Hue, Saturation, Value (HSV) color space. The previously-calculated histogram was then compared to the video frame to determine regions of greatest overlap. This eliminated everything from the frame except that which most matched the color histogram filter. A contour detection algorithm was then applied to the frame and the largest contour found was determined to be the daughter vessel. After computing the bounding rectangle for the object, the detection phase was complete. The detection process is shown in Figure 5-7.

**Visual Tracking Design** For the tracking phase, a region of interest was passed using the bounding rectangle calculated in the detection phase. This provided the initial tracking window for a visual tracking algorithm called CamShift. CamShift

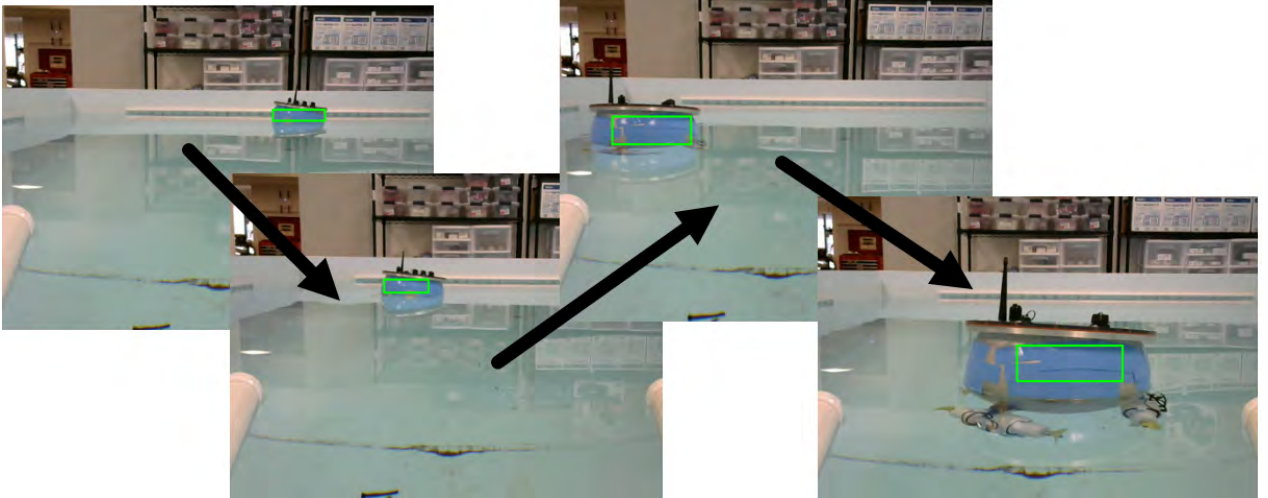


Figure 5-8: Mothership Computer Vision Tracking Process

again used the color histogram of the frame compared to the pre-loaded image and attempted to find the greatest concentration of matching pixels from frame to frame [28]. In addition to tracking the movement horizontally and vertically, CamShift attempted to expand and contract the track box to compensate as the image got larger or smaller, depending on range. Figure 5-8 shows the process of tracking the daughter over time.

## 5.5 Controller

A Proportional, Integral, Derivative (PID) controller was implemented to provide steering directions from the mothership to the daughter. The proportional control used the x-position of the computer vision tracker box centroid and calculated its difference from the centerline of the video frame. This lateral offset was scaled with a proportional coefficient into required turning amounts for the daughter. Motion of the daughter was taken into account using a derivative input based on the change in pixel position from one frame to the next. To mitigate noise seen in small tracker box errors from one frame to the next, derivative errors were sampled based upon a sample rate. This ensured that calculated motion of the daughter vessel was due to physical positioning rather than sensing error. Too large of a sample rate risked mis-

identification of movement, however, if the daughter made a turn in the intervening time frame. For integral control, the daughter goal heading maintained historical information since the mothership was only interested in controlling the relative motion between the two vehicles. The control signal was defined by the following formula:

$$C = K_p \times e + K_i \times \int e dt + K_d \times \frac{e_{t_2} - e_{t_1}}{t_2 - t_1},$$

where  $C$  is the control,  $e$  is the lateral center pixel offset of the daughter's track box from the center of the frame,  $t_1$  is the time of the last sample,  $t_2$  is the time of the current sample, and  $K_p$ ,  $K_i$ ,  $K_d$  are the PID coefficients.

The control coefficients were given default values that made the proportional value small relative to the derivative value. This was done to ensure that when the daughter is far away from the mothership, a large lateral offset should be met with a small response since the daughter has time to respond before getting within close range. Also at the far range, the derivative value perceived by the mothership will be small even if the daughter is moving quickly. Thus, both proportional and derivative responses will be muted which is appropriate for the longer distance. At close range, lateral offset of the daughter and movement across the screen will result in very large errors which combine with the relatively smaller proportional value to provide a moderate response. The larger derivative value allows movements to be recognized quickly and the vehicle to respond fast enough to close range errors to arrive within the docking bay. In any case, the total control value per sample period was capped to prevent excessive steering orders.

## 5.6 Vision System Summary

The techniques employed in the docking system were implemented with relatively straightforward computer vision processing techniques. The color segmentation methods were chosen as a way to test the docking concept but were expected to impose limitations in terms of detection range and accuracy. As will be discussed further in Chapter 6, the system worked suitably well to dock in the field but also experienced

challenges consistent with expectations. This docking system design provides a useful foundation for a future design to build upon since the primary limitations in the design are able to be addressed modularly in the code.

# Chapter 6

## Testing and Results

Autonomous docking tests were conducted in multiple phases which allowed the performance of subsystems and the overall vessel to be characterized at different levels of complexity. The various tests also provided opportunities for incremental development of capabilities. As much as possible, systems were developed in a modular fashion to allow for the independent analysis of different components.

Testing and development occurred in four stages:

1. In lab bench testing
2. Small square tank ( $4 \times 3 \times 1$  m)
3. Tow tank ( $30 \times 2.4 \times 1$  m)
4. Charles River

### 6.1 Lab Testing

Due to the nautical nature of the thesis, in-lab testing was only suitable for subsystem analysis. Lab tests occurred throughout the thesis and primarily leveraged a laptop for executing algorithms rather than the mothership to allow for easier developmental changes. However, as the prototype grew closer to completion, a computer monitor, mouse, and keyboard were attached to the mothership's BeagleBone via a 4-port

USB hub in the lab to make the BeagleBone setup more usable for developmental purposes. This configuration was essential to validate that the code which was written on a Windows laptop would work sufficiently well on a single board computer running Debian Linux.

During lab testing periods, qualitative tests were conducted to determine the suitability of the computer vision detection and tracking algorithms, gimbal performance, wireless connectivity, GUI functionality, and electrical testing. This sort of testing was particularly effective for the vision algorithm since the cluttered background of the lab provided more than ample false detection opportunities to understand the performance of the system.

In-lab testing also confirmed the ability of the peripheral devices to function properly. This step turned out to be non-trivial since some configurations of Debian Linux on the BeagleBone were found to be more difficult to access the input/output pins. Also complicating the pin interface were differences in the Python package and Linux distribution configuration from the baseline daughter. Certain packages had since been changed and pin setup was different than when the daughters were developed. These were not insurmountable issues, but they required decisions to be made on which configuration of Linux and Python to use to provide the best overall configuration management for the system. Ultimately, the original daughter configuration was maintained to avoid too many differences throughout the swarm. Once the pins were set up, the XBee, GPS, and accelerometer/compass were tested in lab to verify correct response.

## 6.2 Small Tank

A  $4 \times 3 \times 1$  m tank at MIT Sea Grant was used for initial in-water testing, shown in Figure 6-1. This provided an excellent opportunity to get acquainted with the daughter controls, test the vision algorithm on a moving vessel, and adjust the coefficients of the PID controller. During this phase, the vision detection algorithm was still in its nascent stages and routinely acquired the wrong object initially. This

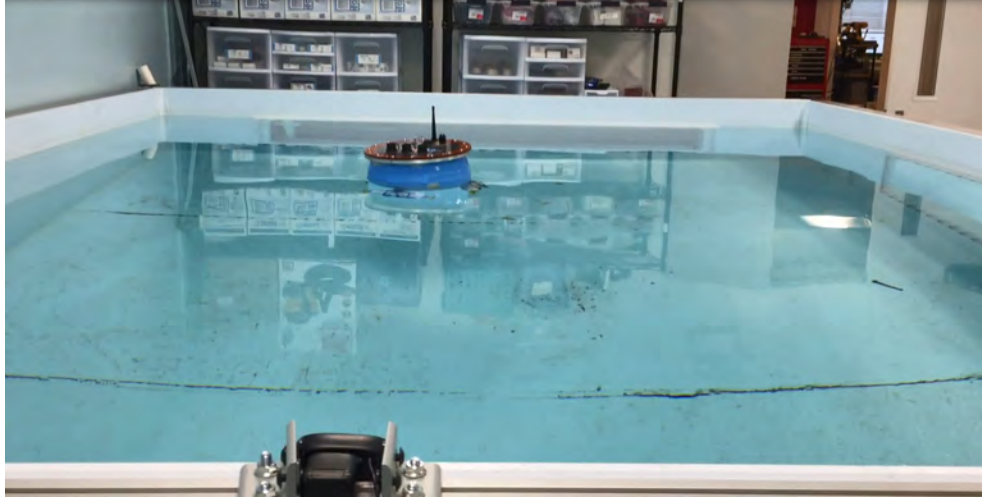


Figure 6-1: Initial Testing: Small Square Tank

proved to be only a small inconvenience at this point since when it occurred, obscuring the camera's view with the exception of the daughter would result in the tracking box moving from the false detection to the daughter. Once the tracking box was on the daughter, it never lost track of it unless the daughter went out of the camera frame. This verified the detection algorithm required further improvement and that the tracking algorithm was ready for the next stage of testing.

Since the tracking algorithm worked sufficiently well throughout this phase, modification and testing of the PID control system was facilitated. By refining the control coefficients, the system was able to dock from a variety of initial position offsets and approach angles. The daughter was covered with blue tape during this test period to provide a color for the vision system to track. The blue color was chosen since it was an available material in the lab and ultimately worked better for tracking than anticipated. In subsequent test phases, the daughter was covered in red for even greater vision recognition.

Although the small tank was extremely useful for initial testing, it had several limitations. The small size meant the daughter could only be operated at reduced speeds which were not representative of what would be required for docking in the operational environment. Detection range was also trivial because the maximum distance was only about 4 m. The dimensions also made positioning the daughter

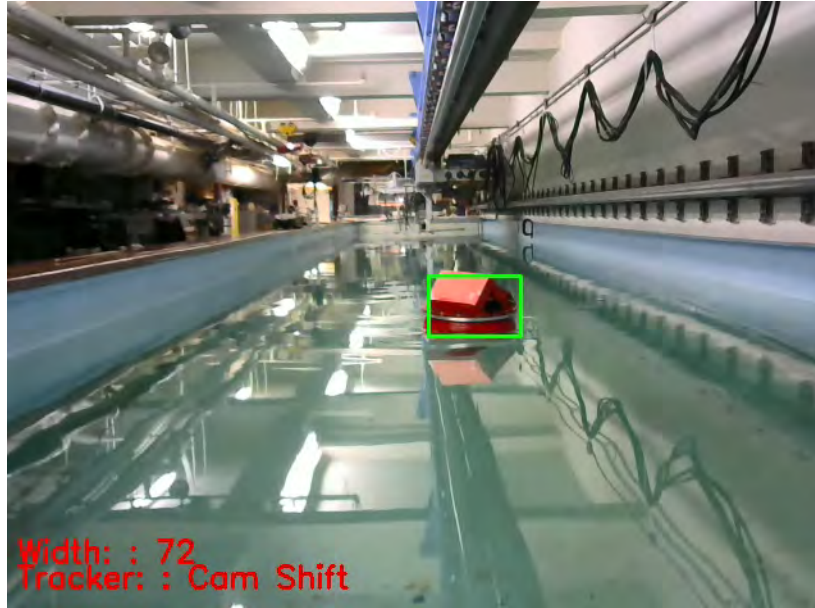


Figure 6-2: Second Phase of Testing: Tow Tank

difficult for test runs because there was little room to maneuver. Lastly, the calm water simplified docking but was not representative of the ultimate goal of open-water testing.

## 6.3 Tow Tank

As seen in Figure 6-2, the tow tank is much larger than the square tank, measuring 30 m long and 2.4 m wide. These increased dimensions allowed the use of greater daughter speeds, larger initial ranges, and tracking for a longer period of time. This meant the speed could match what was expected to be used for docking in the open water testing. The longer distances allowed for a better characterization of vision detection range and also allowed the daughter to align itself to dock over a longer period of time, requiring fewer last minute maneuvers to achieve docking success. The larger space also meant there were more opportunities for false detections in the vision algorithm, which provided a good opportunity to refine the system before deployment in the field.

One of the remaining limitations from this phase was the inability to test a wide

range of initial lateral offsets from the mothership or approach angles by the daughter since the tank was relatively narrow. Only initial positions relatively directly ahead of the mothership were feasible. To somewhat mitigate this, initial range was often reduced to require the daughter to maneuver more dynamically in order to test the limits of docking performance.

Since the distances and speeds were more representative of the ultimate goal, the tests conducted in the tow tank were much more similar to those that would be conducted in the field, with the exception of environmental disturbances. Tests were initially conducted with the camera connected to a laptop computer. Halfway through the phase, the mothership's electronics were positioned on a platform over the tank connected to a monitor, keyboard, and mouse, controlling the docking operations. This allowed a verification of mothership performance with video and SSH outputs which led to changes such as reducing the video resolution on the mothership to improve processing latency. Finally, by the end of tow tank testing, the mothership was floating in the tow tank controlling the docking operation. The only user interaction was positioning of the daughter vessel to its pre-docking position and velocity as well as to commence the docking sequence through the user GUI.

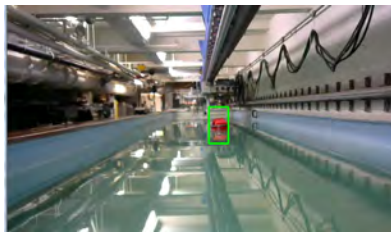


Figure 6-3: Correct Detection

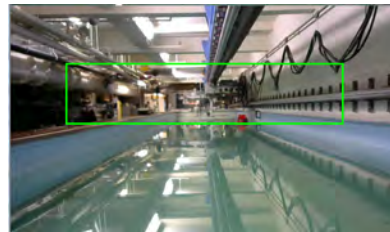


Figure 6-4: Extent of Visual Range

In addition to characterizing docking success, the an estimation of daughter detection range was conducted as shown in Figures 6-3 and 6-4. Using a laptop connected to the C270 Logitech webcam, the daughter was positioned progressively farther away until either no or a false detection resulted. This procedure was repeated with the configurations shown in Figures 6-5 to 6-7 which included a baseline daughter, minor additional above-water surface area, and a large above-water surface area. Results are shown in Table 6.1.



Figure 6-5: Baseline



Figure 6-6: Medium



Figure 6-7: Large

| Visible Red Surface Area ( $cm^2$ ) | Detection Range ( $m$ ) |
|-------------------------------------|-------------------------|
| 230                                 | 3                       |
| 450                                 | 8                       |
| 817                                 | 10                      |

Table 6.1: Estimation of Detection Range in Tow Tank

Data was logged during the testing process to provide the opportunity to analyze test runs after completion. These included daughter center pixel offset, pixel motion, control inputs, and more. Additionally, video was recorded on the BeagleBone to help analyze docking success or failure. The complete list of data was composed of the following items:

- $X$ -offset of the daughter from center pixel on the camera frame
- Derivative of the daughter's  $x$ -position on the camera frame
- Control value provided by the PID algorithm
- Tracking box height and width as an analog for range
- Meta data including the vision tracking technique,  $K_p$ ,  $K_i$ , and  $K_d$
- Time in seconds

## 6.4 Conduct of Field Tests

Initial field tests were conducted with just the daughter vessel to validate the vision and control technique, as was done with the tank testing. A laptop initially controlled the process before incorporating the mothership's electronics. The first tests were



Figure 6-8: Initial Charles Testing

located in a sheltered area between the dock of the MIT Sailing Pavilion and the river wall, shown in Figure 6-8. This allowed validation of the system with some environmental disturbances but not the full range of wind and waves. The water conditions in this phase were with minimal wind and wave disturbances.

Later incremental steps included testing the daughter with the laptop in the main river flow path and then finally testing the daughter and mothership together in the main flow path. Final testing culminated in successful dockings in the main river flow where the mothership and daughter were subject to a steady drift and small waves. Further characterization of the system in higher sea states was not conducted and would be beneficial as a next step towards fully understanding the system.

The composition of individual tests consisted of moving the two vehicles apart approximately 10 m, driving the daughter in the general direction of the mothership, and initiating the docking process. All commands were given via a laptop using the GUI described in Chapter 4. Once the docking command was issued, all subsequent



Figure 6-9: Successful Docking in Charles River

actions were automated until the test was complete.

A representative docking for the final tests is shown in Figure 6-9 where the average wind speed was approximately 6 mph with 22 mph peak. Wave period was approximately 1 s and significant wave height 0.15 m. The two vessels were subject to a steady drift and the mothership was employing automatic heading-keeping within a  $\pm 5$  degree band. The daughter vessel approached at an initial angle of approximately 45 degrees about 10 m away. The entire process shown was automated, with the mothership visually tracking the daughter and issuing steering commands based on the location and movement of the tracking box. The process lasted for about one minute and resulted in a successful docking.

## 6.5 Field Test Observations

Over one hundred docking tests were conducted in the Charles River to determine the capability of the system in the presence of wind and waves. A series of challenges were overcome until the system demonstrated reliable docking success in the main Charles River flow. The challenges experienced and solutions implemented are described below. Remaining limitations and future modifications are discussed in Chapter 7.

**Daughter Control** Initial difficulty was experienced due to an inability to properly control the daughter vessel. In the presence of waves, the daughter would not respond

properly to turn commands and would frequently enter into uncontrolled spins. Although the daughters have been successfully deployed to the field on many occasions in similar conditions, the version used in this thesis was an older model without several improvements that have been made to the current batch. Upon closer examination, the source of the trouble was found to be the daughter's heading compensation algorithm. A heading compensation algorithm was inappropriately reporting heading with a greater error than the baseline compass. As the daughter pitched and rolled, increasingly wrong headings were read resulting in incorrect thruster outputs. These incorrect thruster outputs led to the uncontrolled spins. The heading compensation algorithm was removed and testing with the baseline compass input demonstrated very good control for the daughter.

This error highlighted two key points. First, this docking technique is very dependent upon the daughter vessel's ability to properly navigate and propel itself. Some other docking paradigms rely strictly on the mothership and are not subject to this limitation. The other observation was that the compass headings provided by the LSM330 on the daughter vessel were sufficiently accurate to navigate the daughter without correction.

**Vision System False Detections** A conscious decision was made in the design to deploy the mothership without the most advanced vision system. This was done to limit the scope of the vision system development to allow enough time to develop and test other systems given the limited time available for the thesis. Thus, when the system was deployed on the Charles River, false detections were unsurprisingly experienced. These manifested themselves in several ways. First, as demonstrated in the tow tank, detection range for the daughter was only approximately 10 m. Attempts to dock outside this range would generally result in a track box being generated around a much larger object such as the sky or water. Second, other reddish objects would sometimes be tracked instead of the daughter. For instance, the wall along the side of the Charles River is reddish-brown and would sometimes be tracked instead of the daughter.

To mitigate the false detections, a number of steps were taken. First, since the top of the frame was never useful for tracking in this paradigm, the frame was cropped at a certain height to eliminate the sky and buildings. Second, the largest tracking box that the daughter could generate was known and so if the width of the track box grew beyond that size, the tracking system was programmed to attempt to search for the daughter again. The trouble with this mitigation was that occasionally the daughter vessel would be within a larger track box that was still generally following the daughter's path. Finally, the last action taken was to adjust the angle of the camera since it would often be too high or low, resulting in poor tracking results. Too much reflection from the water or too much of the sky in the picture resulted in false detections. To mitigate this, care was taken to angle the camera properly before deployment. Occasionally, the mothership camera was unplugged from the BeagleBone and attached to the laptop while still mounted on the mothership to gain an understanding of what it was perceiving.

**Controller Tuning** As testing shifted from the square tank to the tow tank to the sheltered Charles River to the main river flow, the controller coefficients and sample rate needed to be adjusted to ensure docking success. Within the tank and calm river settings, lower coefficients were required for control. Within the main body of the river, larger PID coefficients were required to ensure the appropriate response from the system. Sample rate also had to be increased to allow previous control signal responses to be interpreted by the mothership. Otherwise, too many control actions would be sent to the daughter resulting in large overshoots and frequently docking failure.

**Communications Saturation** The initial design for the docking system leveraged a discrete turn command amount rather than a custom amount. To ensure fine enough control was available, the turn increment was set to '1' and for turns larger than that, left or right commands were sent until their effect added to the required amount. This worked sufficiently well in the tank setting, but in the more dynamic environment of

the river, the mothership would frequently end up sending a near continuous stream of turn commands. This large flux of XBee communications resulted in a completely saturated XBee which prevented other important commands from being processed including orders to adjust mothership thrusters or to end the docking process. Both vehicles would become inoperable until the mothership was reset to discontinue the XBee transmissions. This required physically towing the mothership back to flip its on/off switch.

To correct communications saturation, the turn command was reformulated to provide a custom amount which resulted in just one transmission per control loop rather than tens of transmissions. The sample rate was also decreased which spread the time between communications out further.

**Mothership Heading** The mother-controlled docking paradigm was fairly resilient to yaw and sway motions during the docking process. As the mother turns, the position and derivate error with the daughter grows, causing the daughter to turn more. Thus, movements of the mother are taken into account and can be overcome in situ. However, if the mother turns enough such that the daughter exits the camera's field of view, the docking process fails. Early testing in the main flow of the Charles almost exclusively resulted in failure due to this limitation. Attempts were made to maintain the heading of the mothership with manual GUI commands, but even this was insufficient to overcome the problem. Later, automatic heading control was implemented on the mothership as described in Chapter 4 and the docking failures due to this issue were resolved.

**Large Approach Angle at Close Range** Due to the damping effect from the river on the motion of the daughter, its movements were relatively slow during field tests. This meant that even when the daughter was approaching the mothership from a large initial approach angle, it was still usually able to dock successfully. This was not true, though, when the daughter approached at close range from a large initial angle. Although the system would begin to work fairly well, the daughter would

typically exit the camera frame on the other side, resulting in docking failure. To alleviate this, the approach angle of the daughter was limited slightly. Somewhat large angles were still acceptable, but those approaching 90 degrees were sometimes too large.

**Testing Summary** The incremental testing steps and varied environments were essential to developing a working prototype. Each step contributed to a greater understanding of the system and enabled moving to the next level of capability. Field testing validated this docking method as a feasible alternative to traditional docking schemes and highlighted the important factors which contribute to success.

# Chapter 7

## Conclusion

This thesis presented a novel swarm mothership design process which highlighted essential characteristics impacting system performance. The focus on extension of the swarm energy endurance led to specifications which facilitated an improved mothership design. By analyzing variations in hullform parameters, the relative motions were minimized. Computational results indicated large variations in motion response are possible depending on mothership characteristics and that the interactions of the bodies upon one another is significant, emphasizing the importance of this analysis.

The prototype which was developed validated the feasibility of a unique mother-controlled docking technique. This concept demonstrated both advantages and disadvantages relative to conventional paradigms, and should be considered as a viable method for future swarm designers. Observations from the research including successes, failures, and future work are described below.

### 7.1 Key Takeaways

**Mothership Design Process** The energy and motion analyses provided a meaningful basis to design the hull of the mothership. Due to the multidimensional nature of the swarm energy endurance problem and its dependence on mothership design parameters, this analysis is essential to mothership design. Similarly, the hullform characteristics of the vessels were found to result in significant relative motions, de-

pending on the design. Without these two processes, much of the design would have been arbitrary and led to suboptimal performance.

**Mother-Controlled Docking Technique** This novel paradigm worked successfully and demonstrated its ability to be applied in a realistic environment. With some of the additional improvements identified below as future work, designers should consider use of this technique as a means of reducing complexity on the daughter vessel or in any general scenario where an external controller has better navigation and steering information than the daughter vessel.

**Communications Management** Despite the success of the technique, it was apparent on several occasions how important it was to manage communications between the two vessels. Under this paradigm, the daughter has no way to dock in the bay without external assistance. If the mothership is giving excessive turn commands or other control actions, the daughter and mother will be unable to receive and process other messages. This can result in erratic turn behaviors during docking and also complete failure of the vehicles since they will be unable to receive new propulsion commands.

**Camera Field of View** Although the camera was specifically chosen to be low end, one of the most frequent causes of docking failure was the daughter going out of the camera frame. A wider field of view would mitigate this problem and would be one of the highest return on investment improvements for near term modifications to the prototype.

**BeagleBone** While the BeagleBone is equipped with significant capability for a low cost, it does not come without challenges. The Linux distribution for the system often made it difficult to install packages via the normal processes. This resulted in reduced OpenCV functionality, eliminating the ability to use some very capable vision trackers for testing. Luckily, CamShift worked sufficiently well to dock the daughter vessel, but the other trackers offered greater tracking box stability that would have improved

results. These issues can be overcome, but it was decided to spend time resources elsewhere since the docking system was able to be demonstrated successfully. Future applications should leverage these additional tools.

**Motion Stabilization** The three-axis gimbal sufficiently mitigated pitch and roll motions on the mothership to allow the camera to properly track the daughter. In retrospect, a two-axis gimbal should have been purchased since the camera should remain aligned with the long axis of the ship during yaw motions to ensure it is always directing the daughter toward the docking bay. The gimbal acquired for the thesis had a mode to fix its heading in yaw, but it unfortunately drifted which required periodic manual correction between tests.

**Tank vs. Open Water Testing** Using the indoor square and tow tanks was essential in the development of this system, particularly in a cold area like Cambridge, MA. Much of the prototype development and daughter testing occurred in the winter months and would have been impossible otherwise. It was also helpful to develop the system without environmental disturbances and with the ability to connect the mothership to a monitor, keyboard, and mouse. Testing the mothership's electronics along the side of the tank with a full view of system outputs via video monitor and SSH was essential to identify faults and undesirable results such as latency.

**Dependence on Daughter Vessel Performance** Throughout the tank and open water testing, it became clear that many of the results were directly dependent on the ability of the daughter vessel to perform in a specific way. Even though the daughter demonstrated itself to be a very capable and reliable platform, there were occasions where it had a fault which prevented successful dockings. Since the mothership was completely reliant upon the daughter to have it drive into the docking bay, no amount of improvement to the mothership would have improved the results in the mother-controlled docking paradigm. If instead the mothership were responsible for the docking approach, the emphasis on daughter performance would be reduced or eliminated.

## 7.2 Recommendations for Future Work

To build off of this thesis, a number of items are recommended for future research which would further enhance the understanding of autonomous surface vehicle swarm motherships and the docking process. The items below are composed of those specifically related to the previously described prototype and those related to other aspects of ASV swarm mothership docking.

### 7.2.1 Improvements to Prototype Mothership

There are several modifications that could be made to the mothership which are not necessarily novel but would improve the capability of the system. Several decisions were made throughout the thesis to keep the scope of work within the time frame allotted which left deliberate shortcomings in a number of areas.

**Computer Vision** First, the vision system should be improved by using more advanced detection and tracking algorithms. A detection system based on deep learning principles is reasonable to implement and would improve detection range as well as reduce false detections. Similarly, more improved trackers should be implemented which would only require updating OpenCV packages on the BeagleBone. The reason this was not done for this thesis was that the baseline system was found to be suitable and the preparation for using these additional tools was more involved than the thesis scope supported.

**Station-Keeping** The heading-keeping capability demonstrated in this thesis was key to enabling docking success in the open water. The next step in positioning for the mothership would be to also remain in roughly the same location during the docking process regardless of current or wave action. This would prevent the two vehicles from being swept far from their desired location while coming together for docking. To achieve this, the mothership would likely need thrusters besides those fixed astern port and starboard or it would need the two installed thrusters to be trainable.

**Cooperative Docking** The docking system worked well throughout a range of conditions, but as the environmental parameters worsen, the accuracy and success rate will go down. To improve success as this happens, a cooperative docking mode should be implemented whereby the mothership turns in place to orient the docking bay towards the daughter.

**Daughter Range Discrimination** No method was implemented to determine the range of the daughter during the docking process. Range discrimination could be helpful in a number of ways including validation of docking success at the end of the process or as a means to modify the docking process to improve control inputs. For instance, range could be used to change the sample rate as the daughter gets closer to allow the docking system controller to respond quicker as the daughter approaches. When the daughter is off course in close proximity, a large time between samples does not allow enough time to compensate before the docking fails.

**Docking Failure Management** No means to detect or compensate for docking failures was implemented. Docking failure detection would be dependent upon the tracking algorithm to recognize an exit from the camera frame or false detection. Several computer vision techniques employ a probability to determine whether an object has been appropriately classified, which could be leveraged to trigger a failed docking. One could also use a dynamic lighting methodology where the daughter changes its light configuration based on mothership commands to confirm that it is the object being tracked.

Once docking has failed, the daughter vessel should be directed to reposition for another attempt at docking. This could be another attempt from the same position or attempts could be made to find a more optimal path. Differences in current, wave, or wind direction could be leveraged to try to improve success on the next attempt.

### 7.2.2 Related Research

Other topics would contribute in new ways to the field of autonomous mothership docking and deserve further attention. In many cases, these were considered during this thesis but left unanalyzed or unimplemented to keep the scope manageable.

**Physical Docking Connection** Several AUVs have demonstrated docking connections with stationary recharge systems, but the author is not aware of any ASV to ASV docking connections to facilitate recharging. Investigation should be undertaken to develop a connection which could simply yet reliably connect the two vehicles, provide protection against environmental effects, and be resilient to the motions between the two vehicles.

**Recharge System** As with the physical connection, there are related applications which conduct recharging operations, but none the author is familiar with requiring such continuous ASV to ASV recharge capability for so many assets. For Lithium Polymer batteries, as were used in this thesis, consideration should be taken to understand the safety implications and mitigations for autonomously recharging at sea.

**Mother-Controlled Docking Variations** There may be additional improvements to the mother-controlled docking technique by including additional information in the process such as GPS speed over ground, vehicle headings, and range estimation. Rather than the daughter pointing directly at the mothership to dock, it may approach by a more optimal trajectory.

**Recharge Queue Management** The energy simulations in Chapter 2 assumed a static recharge setpoint and strictly followed a First-In, First-Out queue management approach. Further consideration should be given to the decision-making process of determining how to fulfill recharge requests for the swarm. For instance, one might decide to begin recharging a daughter at 70% rather than a lower setpoint to allow the mothership to get ahead on its responsibilities. The mothership may also benefit from recharging the next physically closest daughter rather than the one with the

least remaining energy. Simulations and subsequent field tests should be run with different paradigms to determine how changes in strategy, initial conditions, and system parameters and result in improved results.

## 7.3 Conclusion

As interest and development of marine swarms continues to grow, an increased emphasis will be placed on the design of motherships and autonomous docking. The thesis described herein provides a first step towards this progression by presenting a novel swarm energy analysis model, analysis of minimizing relative motion between the two marine vehicles, and by constructing a feasible prototype capable of docking with a daughter vessel using an unconventional control technique. The system tested in this work demonstrated capability that could be used in a relevant system today with minimal additional change but that also has the potential to be much greater with future research.



# Appendix A

## Mothership Code

The mothership leveraged a majority of the code presented in [5]. New files and major modifications are included in this appendix.

```

1  '''
2  @author: Andrew Freeman
3  MIT          2016-2019
4  Based on code derived from
5  tutorials by Adrian Rosebrock
6  at PyImageSearch.com
7  '''
8
9  import numpy as np
10 import cv2
11 import time
12 from threading import Thread
13 import imutils
14 from os.path import exists
15
16
17 class CameraDocker(Thread, object):
18     def __init__(self, **kwargs):
19         Thread.__init__(self)
20         self.vs = cv2.VideoCapture(0)
21         self.W = 320
22         self.H = 240
23         self.vs.set(3, self.W)
24         self.vs.set(4, self.H)
25         self.frame_y_max = int(self.H * 0.55) #
26         self.max_width = 120 #
27         self.sample_time = 1.5
28         self.color_limit = 5
29         self._start_thread('main')
30         self.coop_turn_increment = 0.1
31         global XB
32         XB = kwargs.pop('XB')
33         self.dock = False
34         self.kp = .02
35         self.ki = 0
36         self.kd = .7
37         self.cam_shift_limit = 3
38         self.track = [0, 0, 0, 0]
39         self.daughter = False
40         self.end_dock = False
41         self.derive_time = 0.5
42         self.save_video = False

```

```

43         self.control_cap = 30
44         self.zero_control = False
45         self.zero_control_2 = False
46
47         fileTestString = "./trackerOutput/VideoOUT00.avi"
48         fileChar = list(fileTestString)
49         for i in range(0, 100):
50             fileChar[-6] = str(int(i / 10))
51             fileChar[-5] = str(int(i % 10))
52             fileIDstr = ''.join(fileChar)
53             if (exists(fileIDstr) == False):
54                 break
55             self.out = cv2.VideoWriter(fileIDstr, cv2.cv.
CV_FOURCC('M', 'J', 'P', 'G'),
56                                     10.0, (int(self.vs.get(3)),
int(self.vs.get(4))-self.frame_y_max))
57
58
59     def _start_thread(self, name):
60         if name == 'main':
61             self.main_loop = Thread(target=self.run_dock)
62             self.main_loop.daemon = True
63             self.main_loop.start()
64             print('Camera thread')
65
66     def run_dock(self):
67         while True:
68             if self.dock:
69                 if self.daughter:
70                     self.camera_guide_daughter(self.kp,
self.ki, self.kd, self.track)
71                     self.dock = False
72                     self.daughter = False
73
74     def set_dock(self, kp, ki, kd, dock, track, dock_mode)
:
75         self.dock = dock
76         self.kp = float(kp)
77         self.ki = float(ki)
78         self.kd = float(kd)
79         self.track = track
80         self.end_dock = False
81         if dock_mode == 'daug':
82             self.daughter = True
83

```

```

84     def camera_guide_daughter(self, kp, ki, kd, track):
85         time_started = time.time()
86         frame_count = 0
87         cam_shift_count = 0
88         x_data = []
89         y_data = []
90         w_data = []
91         h_data = []
92         d_error_data = []
93         control_data = []
94         metadata = []
95         time_data = []
96         time_data.append(0)
97         hdg_error_data = []
98
99         frame = cv2.imread("redbottle.PNG")
100        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
101        hist = cv2.calcHist([hsv], [0, 1], None, [180,
256], [0, 180, 0, 256])
102        cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX
)
103
104        control = 0
105        hdg_error = 0
106        d_error = 0
107        i_error = 0
108        last_x = 0
109        control_window = 1.5
110        w = 0
111        h = 0
112        x = 0
113        y = 0
114        data_sent = False
115
116        fileTestString = "/home/debian/Python/
trackerOutput/MotherOUT00.npy"
117        fileChar = list(fileTestString)
118        for i in range(0, 100):
119            fileChar[-6] = str(int(i / 10))
120            fileChar[-5] = str(int(i % 10))
121            fileIDstr = ''.join(fileChar)
122            if (exists(fileIDstr) == False):
123                break
124
125        time_moved = 0

```

```

126         time_moved2 = 0
127
128         metadata.append("CamShift")
129         metadata.append(control_window)
130
131         while True:
132             ret, orig_frame = self.vs.read()
133
134             # The webcam was mounted upside down, so flip
135             the image
136             orig_frame = cv2.flip(orig_frame, -1)
137
138             # Since the top of the frame (sky, buildings
139             , etc.) isn't useful, chop it out to reduce false
140             detections
141             frame = orig_frame[self.frame_y_max:, :]
142
143             if frame is None:
144                 break
145
146             # blur frame and convert it to the HSV color
147             space
148             blurred = cv2.GaussianBlur(frame, (11, 11), 0
149             )
150             hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV
151             )
152
153             # Use histogram from pre-loaded image to
154             generate mask
155             mask = cv2.calcBackProject([hsv], [0, 1],
156             hist, [0, 180, 0, 256], 1)
157
158             # The first several frames are an attempt to
159             initially find the buoy
160             # Then the bounding box is sent to CamShift
161             which tracks it for the remainder of the process
162             if frame_count < self.color_limit:
163                 frame_count += 1
164                 # find contours in the mask and
165                 initialize the current
166                 cnts = cv2.findContours(mask.copy(), cv2.
167                 RETR_EXTERNAL,
168                 cv2.
169                 CHAIN_APPROX_SIMPLE)
170                 cnts = imutils.grab_contours(cnts)

```

```

158         vis = frame.copy()
159         (H, W) = vis.shape[:2]
160         info = [("Tracker", "Color")]
161
162         # only proceed if at least one contour
163         was found
164         if len(cnts) > 0:
165             # find the largest contour in the
166             mask, then use
167             # it to compute the minimum enclosing
168             circle and
169             # centroid
170             c = max(cnts, key=cv2.contourArea)
171             ((x, y), radius) = cv2.
minEnclosingCircle(c)
172             # The bounding rectangle is used to
173             pass a track window to CamShift later
174             track_window = cv2.boundingRect(c)
175             try:
176                 # only proceed if the radius
177                 meets a minimum size
178                 if radius > 10:
179                     (x, y, w, h) = [int(v) for v
in track_window]
180                     # Check to make sure box
181                     width is reasonable--otherwise it is a false detection
182                     if w < self.max_width:
183                         cv2.rectangle(vis, (x, y)
, (x + w, y + h),
184                                     (0, 255, 0)
, 2)
185
186                     else:
187                         w = 0
188
189                     # If the track box was a
190                     false detection, we don't want a faulted control value
191                     self.zero_control = True
192                     self.zero_control_2 =
True
193             except:
194                 print("Error")
195
196         for (i, (k, v)) in enumerate(info):
197             text = "{}: {}".format(k, v)

```

```

191             cv2.putText(vis, text, (10, H - ((i *
20) + 20)),
192                             cv2.FONT_HERSHEY_SIMPLEX,
20) + 20)),
193                             0.5, (0, 0, 255), 2)
194
195         else:
196             cam_shift_count += 1
197             vis = frame.copy()
198             hsv = cv2.cvtColor(frame, cv2.
199             COLOR_BGR2HSV)
200
201         try:
202             mask = cv2.calcBackProject([hsv], [0,
203             1], hist, [0, 180, 0, 256], 1)
204             term_crit = (cv2.TERM_CRITERIA_EPS |
205             cv2.TERM_CRITERIA_COUNT, 10, 1)
206             track_box, track_window = cv2.
207             CamShift(mask, track_window, term_crit)
208             (x, y, w, h) = [int(v) for v in
209             track_window]
210             cv2.rectangle(vis, (x, y), (x + w, y
211             + h),
212                             (0, 255, 0), 2)
213             if w > self.max_width:
214                 frame_count = 0
215                 w = 0
216                 self.zero_control = True
217                 self.zero_control_2 = True
218                 camW = w
219
220             info = [("Dock Mode", "Daughter"),
221                     ("Tracker", "Cam Shift"),
222                     ("Data File", fileIDstr[40:]),
223                     ("Width", camW),
224                     ("Y", y)]
225
226             for (i, (k, v)) in enumerate(info):
227                 text = "{}: {}".format(k, v)
228                 cv2.putText(vis, text, (10, H - (
229                 (i * 20) + 20)),
230                             cv2.
231                             FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
232
233         except:
234             print(track_box)

```

```

226
227         if frame is not None:
228             frame2 = vis.copy()
229             self.out.write(frame2)
230
231         x_data.append(x)
232         y_data.append(y)
233         w_data.append(w)
234         h_data.append(h)
235
236         # Calculate PID values and provide control
237         input
238         if cam_shift_count > self.cam_shift_limit and
239         w < self.max_width:
240             if cam_shift_count == 11:
241                 XB.write(XB.rem_XB_msg("Started Cam
242                 Shift"))
243             hdg_error = x - (W / 2)
244             if time.time() - time_moved2 > self.
245             sample_time:
246                 if last_x != 0:
247                     d_error = x - last_x
248                 else:
249                     d_error = 0
250                     time_moved2 = time.time()
251                     last_x = x
252             i_error += hdg_error
253
254             if time.time() - time_moved > self.
255             sample_time:
256                 time_moved = time.time()
257                 if not self.zero_control and not self
258                 .zero_control_2:
259                     control = kp * hdg_error + kd *
260                     d_error + ki * i_error
261                 elif not self.zero_control_2:
262                     control = 0
263                     self.zero_control = False
264                 else:
265                     control = 0
266                     self.zero_control_2 = False
267                 print("Control value {ctl}".format(
268                 ctl=control))
269                 print("Heading error {hdg}".format(

```

```

262 hdg=hdg_error))
263             print("Derivative error {drv}".format
                (drv=d_error))
264             print("Frame Width {fmwd}".format(
                fmwd=W))
265             print("X Value {xval}".format(xval=x)
                )
266             print("kp {kpval}".format(kpval=kp))
267             print("kd {kdval}".format(kdval=kd))
268             print("kp signal {kpsig}".format(
                kpsig=(kp*hdg_error)))
269             print("kd signal {kdsig}".format(
                kdsig=(kd * d_error)))
270
271             if control < -control_window:
272                 control = max(-self.control_cap,
                control)
273
274                 if control > -10:
275                     XB.send_Command(['Turn', '-'
                + '0' + str(int(abs(control)))])
276                 else:
277                     XB.send_Command(['Turn', str(
                int(control))])
278             elif control > control_window:
279                 control = min(self.control_cap,
                control)
280
281                 if control < 10:
282                     XB.send_Command(['Turn', '+'
                + '0' + str(int(control))])
283                 else:
284                     XB.send_Command(['Turn', '+'
                + str(int(control))])
285
286             if self.end_dock:
287                 cv2.putText(vis, "DOCKED!", ((H / 2)
                - 220, (H / 2) - 60), cv2.FONT_HERSHEY_SIMPLEX,
288                 1.4, (0, 0, 255), 3)
289                 dock_success = "DockYes"
290
291             if True:
292                 XB.send_Command(['Motor', 'disarm
                '])
293
294             metadata.append(dock_success)
295             control_data.append(0)
296             hdg_error_data.append(0)

```

```

294         d_error_data.append(0)
295         data_sent = True
296         cv2.waitKey(1)
297         cv2.destroyAllWindows()
298         cv2.waitKey(1)
299         cv2.waitKey(1)
300         cv2.waitKey(1)
301         cv2.waitKey(1)
302         tracker_data = np.zeros((9, len(
x_data)), dtype=object)
303         tracker_data[0, :len(metadata)] =
metadata
304         tracker_data[1, :] = time_data
305         tracker_data[2, :] = x_data
306         tracker_data[3, :] = y_data
307         tracker_data[4, :] = w_data
308         tracker_data[5, :] = h_data
309         tracker_data[6, :] = control_data
310         tracker_data[7, :] =
hdg_error_data
311         tracker_data[8, :] = d_error_data
312
313         try:
314             np.save(fileIDstr,
tracker_data)
315
316         except:
317             print("oops")
318             time.sleep(5)
319             break
320
321         control_data.append(control)
322         hdg_error_data.append(hdg_error)
323         d_error_data.append(d_error)
324         time_data.append(time.time() - time_started)
325
326         if self.save_video:
327             try:
328                 self.out.release()
329             except:
330                 fileTestString = "./trackerOutput/
VideoOUT00.avi"
331                 fileChar = list(fileTestString)
332                 for i in range(0, 100):
333                     fileChar[-6] = str(int(i / 10))

```

```

334         fileChar[-5] = str(int(i % 10))
335         fileIDstr = ''.join(fileChar)
336         if (exists(fileIDstr) == False):
337             break
338         self.out = cv2.VideoWriter(fileIDstr, cv2
    .cv.CV_FOURCC('M', 'J', 'P', 'G'),
339                                     10.0, (int(
    self.vs.get(3)), int(self.vs.get(4)) - self.frame_y_max))
340
341         if not data_sent:
342             tracker_data = np.zeros((9, len(x_data)),
    dtype=object)
343             tracker_data[0, :len(metadata)] = metadata
344             tracker_data[1, :] = time_data
345             tracker_data[2, :] = x_data
346             tracker_data[3, :] = y_data
347             tracker_data[4, :] = w_data
348             tracker_data[5, :] = h_data
349             tracker_data[6, :] = control_data
350             tracker_data[7, :] = hdg_error_data
351             tracker_data[8, :] = d_error_data
352
353             try:
354                 np.save(fileIDstr, tracker_data)
355
356             except:
357                 print("oops")
358
359             # On the BeagleBone, multiple waitKey calls were
    required to properly close the window
360             cv2.waitKey(1)
361             cv2.destroyAllWindows()
362             cv2.waitKey(1)
363             cv2.waitKey(1)
364             cv2.waitKey(1)
365             cv2.waitKey(1)

```

```

1  '''
2  @author: Brandon Zoss
3  MIT          2015-2016
4  Modified by Andrew Freeman
5  MIT          2016-2019
6  to support mothership docking
7  with buoys
8  Original filename "Buoy.py"
9  '''
10
11 from Mother_def import *
12 from threading import Thread
13 from time import time, sleep
14 from numpy import trapz, arctan2, rad2deg, array, average
15 from numpy import meshgrid, argmin, linspace, sign
16
17 import Chebyshev as cheb
18 import Hermite as herm
19
20 class Mother(Thread, object):
21
22     def __init__(self, **kwargs):
23         Thread.__init__(self)
24
25         global XB, MD, MS, MP, GPS, compass, CD
26         XB = kwargs.pop('XB')
27         MD = kwargs.pop('MD')
28         MS = kwargs.pop('MS')
29         MP = kwargs.pop('MP')
30         GPS = kwargs.pop('GPS')
31         compass = kwargs.pop('compass')
32         CD = kwargs.pop('CD')
33
34 #=====
35 # Set Default Mother parameters
36 #=====
37
38     # Used for calculating Batt Remaining
39     self.disch, self.times = list(), list()
40     self.amps_out = 0.
41
42     # Steering Method
43     self.drive_Grad = False
44     self.drive_Cont = False

```

```
44         self.swarm = False
45
46         self.assemble = False
47         self.path_x, self.path_y = list(), list()
48         self.UL, self.LR = list(), list()
49         self.p_indx = 0
50
51         self.rem_drive = False
52         self.follow = [False, ' ']
53
54         # Goal and Compass Heading
55         self.lat, self.lon = 0., 0.
56         self.goal = [42.358412, -71.087675] # MIT Charles
57
58         self.left_thruster = 9.0
59         self.right_thruster = 9.0
60
61         self.goal_BRG = 000
62         self.goal_RNG = 10.
63
64         self.relh = 0.0
65         self.auto_heading = False
66
67         # Propulsive Parameters
68         self.arm = False
69         self.thrust = [0., 000]
70         self.max_spd = 100. # Max speed in percent
71         self.power = [0., 0.]
72         self.max_fwd = 7.0
73         self.max_aft = 11.0
74         self.last_time = 0
75
76         # Data Logging
77         self.DataLogRate = 10.
78         self.BattTimer = time()
79         self.DataTimer = time()
80
81
82         # Set Default Mother commands and actions
83         # Establish Command Dictionary
84         self.comm_dict = { 'Mother':self.set_arm,
85                             'MomSpeed':self.set_speed,
86                             'Drive':self.set_drive,
87                             'Goal':self.set_goal,
88                             'DataRate':self.set_rate,
```

```

89         'Turn':self.set_turn,
90         'XBee':self.set_XBee,
91         'Follow':self.set_leader,
92         'Assemble':self.set_contour,
93         'Pattern':self.set_pattern,
94         'Corner':self.set_corner,
95         'Swarm':self.set_swarm,
96         'Dock': self.dock,
97         'End':self.end_dock,
98         'Reset':self.reset_motors,
99         'Thrust':self.set_thrust,
100        'Derive':self.set_derive,
101        'AH':self.set_auto_heading,
102        'Save':self.save_video,
103        'Control':self.set_control_cap
104    }
105
106
107        # Establish Set Actions for Certain Commands
108        self.act_dict = {'Mother':{'arm':True, 'disarm':
False},
109                        'Drive':{'GPS':[False, False,
False],
110                                'grad':[True, False,
False],
111                                'cont':[False, True,
False],
112                                'rem':[False, False,
True]}},
113                        'Turn':{'left':-1, 'right':1, '
180':180, 'mothleft':-5, 'mothright':5, 'moth180':180},
114                        'XBee':{'reset':['00000000', '
0000FFFF']}}
115
116        self._start_thread('main')
117
118
119        #=====
120        # Define decorators
121        #=====
122
123        @property
124        def speed(self):
"""

```

```

125         Linearly reduce speed as the goal is
126         approached.
127         """
128         spd = self.max_spd
129
130         if self.goal_RNG < 4.0:
131             spd = self.max_spd * self.goal_RNG / 4.0
132
133         return spd
134
135     @property
136     def goal(self):
137         """
138         Returns the goal of the Mother.
139         If the mother is exhibiting a behavior of
140         following or assembling, the returned
141         goal is that defined by the particular
142         behavior.
143         """
144         if self.follow[0] and self.follow[1] in MD.swarm.
keys():
145             self._goal = [MD.swarm[self.follow[1]]['lat']
146
147             MD.swarm[self.follow[1]]['lon']
148
149         if self.assemble and (len(self.path_x) > self.
p_indx):
149             self._goal = [self.path_y[self.p_indx],
150                             self.path_x[self.p_indx]]
151
152         return self._goal
153
154     @goal.setter
155     def goal(self, lat_lon):
156         self._goal = lat_lon
157
158     @property
159     def goal_BRG(self):
160         return self._goal_BRG
161
162     @goal_BRG.setter
163     def goal_BRG(self, brg):
164         """
165         When setting the Mother goal bearing, and

```

```

166         ultimately thrust vector, this allows the
167         Mother to exhibit behaviors that exploit this
168         parameter in order to take control of the
169         mother - regardless of bearing to the
170         current goal.
171         """
172         self._goal_BRG = brg
173
174     @property
175     def goal_RNG(self):
176         return self._goal_RNG
177
178     @goal_RNG.setter
179     def goal_RNG(self, rng):
180         """
181         When setting the range to a given
182         goal, this function triggers events
183         within assembly type behaviors, such
184         as shifting to the next goal in a
185         string of waypoints.
186         """
187
188         self._goal_RNG = rng
189
190         if self.assemble and rng <= 4.0:
191             self.p_indx += 1
192
193             if self.p_indx >= len(self.path_x):
194                 self.p_indx = 0
195
196             if self.rem_drive or self.swarm:
197                 self._goal_RNG = 4.0
198
199     @property
200     def thrust(self):
201         return self._thrust
202
203     @thrust.setter
204     def thrust(self, vector):
205         """
206         When setting the thrust vector,
207         the thrust vector is immediately
208         converted to a true vector, so that
209         they thrust is always in terms of
210         True bearing to a given location.

```

```

211         """
212         mag, brg = vector
213
214         brg = get_True_BRG(brg, compass.tilt_comp_heading
215     )
216         self._thrust = [mag, brg]
217
218     @property
219     def cheb_grad(self):
220         """
221         Returns the gradient at the location of
222         the Mother, based on Chebyshev field
223         estimation. Chebychev field must be
224         calculated in Mother Data for this to
225         return a meaningful gradient.
226         """
227         if len(MD.temps) <= MD.c_mn.size:
228             return 000
229
230         kwargs = {'ax':array(MD.lons).min(), 'bx':array(
231             MD.lons).max(),
232             'ay':array(MD.lats).min(), 'by':array(
233             MD.lats).max()}
234
235         u, v = cheb.grad(self.lon, self.lat, MD.c_mn, **
236             kwargs)
237
238         angle = (90 - rad2deg(arctan2(v, u))) % 360
239
240         return angle
241
242     @property
243     def herm_grad(self):
244         """
245         Returns the gradient at the location of
246         the Mother, based on Hermite field
247         estimation. Hermite field must be
248         calculated in Mother Data for this to
249         return a meaningful gradient.
250         """
251         if len(MD.temps) <= MD.c_mn.size:
252             return 000
253
254         kwargs = {'_x':MD._x, '_y':MD._y,

```

```

252         'L_x':MD.L_x, 'L_y':MD.L_y,
253         '_ff':0}
254
255     u, v = herm.grad(self.lon, self.lat, MD.c_mn, **
256     kwargs)
257
258     angle = (90 - rad2deg(arctan2(v, u))) % 360
259
260     return angle
261
262     def monitor_batt(self, amps):
263         """
264         This function tabulates battery useage
265         and crudely monitors battery status.
266         Always assumes a full battery at initialization.
267
268         Later, this will be used to trigger return
269         home events if the battery status is critical
270         """
271         self.disch.append(sum(amps) + 0.200)
272         self.times.append(time())
273
274         self.power = amps
275
276         if len(self.disch) > 1000:
277             self.amps_out += trapz(self.disch, array(self
278             .times)/3600.0)
279             self.disch, self.times = list(), list()
280
281     def end_dock(self, filler=[]):
282         CD.end_dock = True
283
284         return "Ending the Docking Process"
285
286     def set_derive(self, derive_values=[]):
287         CD.sample_time = float(derive_values.pop(0))
288         return "New Sample Time Interval: " + str(CD.
289         sample_time)
290
291     def save_video(self, video_values=[]):
292         print("In save video")
293         CD.save_video = True
294         return "Video Saved"

```

```

294     def set_auto_heading(self, auto_values=[]):
295         auto_heading,self.Kpt,self.Kit,self.Kdt =
auto_values
296         if auto_heading == "Off":
297             self.auto_heading = False
298         elif auto_heading == "On":
299             self.auto_heading = True
300         self.Kpt = float(self.Kpt)
301         self.Kit = float(self.Kit)
302         self.Kdt = float(self.Kdt)
303         self.auto_head = compass.tilt_comp_heading
304         return 'Auto Heading ' + str(self.auto_heading) +
' ' + str(self.Kpt) + ' ' + str(self.Kit) + ' ' + str(
self.Kdt)
305
306     def set_control_cap(self, cap=[]):
307         control_cap = int(cap.pop())
308         CD.control_cap = control_cap
309         return "New Control Cap: " + str(control_cap)
310
311     def set_thrust(self, thrust_values=[]):
312         side = thrust_values.pop(0)
313         print("Side " + side)
314         amount = float(thrust_values.pop(0))
315         self.left_thruster = MP.thruster_duty_cycle[0]
316         self.right_thruster = MP.thruster_duty_cycle[1]
317
318         if not self.arm:
319             self.arm = True
320         if side == 'left':
321             self._turn(-amount)
322
323         elif side == 'right':
324             self._turn(amount)
325
326         elif side == 'fwd':
327             mean = (self.left_thruster + self.
right_thruster)/2.
328             if MP.thruster_duty_cycle[0] != MP.
thruster_duty_cycle[1]:
329                 self.left = mean
330                 self.right_thruster = mean
331             else:
332                 self.left_thruster -= amount
333                 self.right_thruster -= amount

```

```

334         elif side == 'aft':
335             mean = (self.left_thruster + self.
right_thruster) / 2.
336             if self.left_thruster != self.right_thruster:
337                 self.left_thruster = mean
338                 self.right_thruster = mean
339             else:
340                 self.left_thruster += amount
341                 self.right_thruster += amount
342
343         elif side == 'off':
344             self.left_thruster = 9.0
345             self.right_thruster = 9.0
346
347             MP.thruster_duty_cycle[0] = _in_range(self.
left_thruster, self.max_fwd, self.max_aft)
348             MP.thruster_duty_cycle[1] = _in_range(self.
right_thruster, self.max_fwd, self.max_aft)
349             print(MP.thruster_duty_cycle)
350             return "Left Thruster: " + str(MP.
thruster_duty_cycle[0]) + " Right Thruster: " + str(MP.
thruster_duty_cycle[1])
351
352     def _turn(self, amount):
353         self.set_arm(True)
354
355         self.left_thruster = MP.thruster_duty_cycle[0] -
amount
356         self.right_thruster = MP.thruster_duty_cycle[1] +
amount
357
358         MP.thruster_duty_cycle[1] += amount
359         MP.thruster_duty_cycle[0] -= amount
360         if _check_wait(MP.thruster_duty_cycle[0], self.
left_thruster):
361             MP.thruster_duty_cycle[0] = 9.0
362             print("Sleeping!")
363             sleep(1)
364
365         if _check_wait(MP.thruster_duty_cycle[1], self.
right_thruster):
366             MP.thruster_duty_cycle[1] = 9.0
367             print("Sleeping!")
368             sleep(1)
369

```

```

370         MP.thruster_duty_cycle[0] = _in_range(self.
left_thruster, self.max_fwd, self.max_aft)
371         MP.thruster_duty_cycle[1] = _in_range(self.
right_thruster, self.max_fwd, self.max_aft)
372
373         return
374
375     def dock(self, pid=[]):
376         kp, ki, kd, dock_mode = pid
377         CD.set_dock(kp, ki, kd, True, (1, 1, 100, 100),
dock_mode)
378
379         return "Docking via " + dock_mode
380
381
382     @property
383     def batt_rem(self):
384
385         # Assume a 5aH battery
386         remaining = 100.0*(4.6 - self.amps_out)/5.0
387
388         return remaining
389
390     def _start_thread(self, name):
391
392         if name == 'main':
393             self.main_loop = Thread(target=self.
_main_update)
394             self.main_loop.daemon = True
395             self.main_loop.start()
396
397     def _main_update(self):
398         """
399         Written to be part of it's own thread, this
400         function continuously updates main aspects of
401         Mother operation, such as battery logging,
402         command parsing, and heading evaluation.
403         """
404         while True:
405
406             try:
407                 self.monitor_batt(MS.pump_outputs)
408             except:
409                 MD.write_TEST(['Failed to monitor_batt'])
410

```

```

411         self.update_GPS_params()
412
413         try:
414             addr, command = XB.read()
415         except:
416             addr, command = None, None
417
418         if addr and command:
419             self.eval_message(addr, command)
420
421         self.motor_update() # issue is in
motor_update...
422
423         if time() - self.DataTimer >= self.
DataLogRate:
424             self.data_update()
425
426         if time() - self.BattTimer >= 60:
427             self.batt_update()
428
429     def update_GPS_params(self):
430         """
431         Updates GPS related parameters, such
432         as location and new bearing and range
433         to GPS goal coordinates.
434         """
435
436         if GPS.GGA.is_valid:
437
438             self.lat, self.lon = GPS.GGA.latitude, GPS.
GGA.longitude
439
440             rng_args = [self.lat, self.lon, self.goal[0],
self.goal[1]]
441             self.goal_BRG, self.goal_RNG = get_BRG_RNG(*
rng_args)
442
443             compass.declination = GPS.mag_var
444
445     def eval_message(self, addr, command):
446         """
447         Parse incoming message for action.
448         If message contains swarm data, update
449         the swarm dictionary. Otherwise send
450         message response to originator.

```

```

451         """
452
453         try:
454             msg = self.parse_com(bytearray.fromhex(
command).decode())
455         except:
456             msg = None
457
458         if msg:
459
460             if len(msg) == 8:
461                 MD.parse_msg(addr, msg)
462
463             else:
464                 print(msg)
465                 XB.write(XB.rem_XB_msg(msg))
466
467         def motor_update(self):
468
469             rel_BRG = get_rel_BRG(self.goal_BRG, compass.
tilt_comp_heading)
470             # Adjust heading to avoid swarm components
471             rel_BRG += get_mother_avoidance([self.lat, self.
lon], MD.swarm, self.thrust[1])
472
473             if self.auto_heading:
474                 self.set_arm(True)
475                 rel = heading_diff(self.auto_head, compass.
tilt_comp_heading)
476                 print("Motor Update", self.auto_head, compass
.tilt_comp_heading, rel)
477
478                 if abs(rel) > 2.:
479                     if time() - self.last_time > 0.2:
480                         self.last_time = time()
481                         amount = self.turn_controller(rel)
482                         print("Goal BRG: " + str(self.
auto_head) + ", " + "Heading: " +
483                             str(compass.
tilt_comp_heading)
484                             + ", Rel: " + str(rel) + ", Turning
Amount: " + str(amount))
485                         self._turn(sign(amount)*min(0.1, abs(
amount)))
486                 else:

```

```

487             MP.thruster_duty_cycle[0] = 9.0
488             MP.thruster_duty_cycle[1] = 9.0
489             self.set_arm(False)
490
491
492             msg = MP.set_Motors_duo(self.arm, self.speed,
rel_BRG)
493             MD.write_TEST([msg])
494
495             self.thrust = MP.Thrust_Vector
496
497     def turn_controller(self, rel):
498
499         deriv = (rel - self.relh)*self.Kdt
500         prop = rel*self.Kpt
501
502         print("Deriv = " + str(deriv) + "Prop = " + str(
prop))
503
504         self.relh = rel
505
506         amount = prop - deriv
507         return amount
508
509     def batt_update(self):
510         """
511         Written to be part of it's own thread, this
512         function periodically gathers and broadcasts
513         data related to battery monitoring of the mother
514         at prescribed intervals.
515         """
516         self.BattTimer = time()
517
518         msg = 'Battery Remaining: %.1f Percent' % self.
batt_rem
519         XB_msg = XB.rem_XB_msg(msg)
520
521         try:
522             XB.write(XB_msg)
523         except:
524             pass
525
526     def data_update(self):
527         """
528         Written to be part of it's own thread, this

```

```

529         function periodically gathers, broadcasts
530         and writes logged data streams on prescribed
531         intervals.
532         """
533         self.DataTimer = time()
534
535         for txt in [self.__str__(), '\n', compass, '\n',
GPS, '\n']:
536             print(txt)
537
538             if GPS.GGA.is_valid:
539
540                 temps = MS.temp_readings
541
542                 strm_args = [GPS.GGA, self.goal,
543                             [self.cheb_grad, average(temps)]
544                             ,
545                             self.thrust]
546
547                 stream = MD.pack_Stream(*strm_args)
548
549                 try:
550                     XB.write(XB.rem_XB_msg(stream))
551                 except:
552                     pass
553
554                 goal_BR = [self.goal_BRG, self.goal_RNG]
555
556                 mother_args = [GPS.GGA, goal_BR,
557                                temps, MP.pump_power,
558                                compass.tilt_comp_heading]
559
560                 MD.write_Mother(*mother_args)
561                 MD.write_GPS(GPS.GGA, GPS.RMC, self.goal)
562
563                 MD.lats.append(GPS.GGA.latitude)
564                 MD.lons.append(GPS.GGA.longitude)
565                 MD.temps.append(average(temps))
566         =====
567         =====
568         # Set the PRINT method for this class
569         =====
570         =====
571
572         def __str__(self):

```

```

570
571         temps = MS.temp_readings
572
573         data = ['\n',
574                 'Goal BRG: \t\t%.1f[deg]' % self.goal_BRG
575                 ,
576                 'Goal RNG: \t\t%.1f[m]' % self.goal_RNG,
577                 'Heading: \t\t%.1f[deg]' % compass.
578                 tilt_comp_heading,
579                 'Thrust BRG: \t\t%.1f[deg]' % self.thrust
580                 [1],
581                 'Mother Speed: \t\t%.1f[%%]' % self.speed
582                 ,
583                 'Probe Temps: \t\t' + ' '.join(['%.1f[K]'
584                 % t for t in temps]),
585                 'Avg Temp: \t\t%.1f[K]' % average(temps),
586                 'Motors Armed: \t\t%s' % self.arm,
587                 'Motor Output: \t\t' + ' '.join(['%.3f[A'
588                 ]' % p for p in self.power]),
589                 'Driving Gradient: \t%s' % self.
590                 drive_Grad,
591                 'Driving Contour: \t%s' % self.drive_Cont
592                 ,
593                 'Driving by rem: \t%s' % self.rem_drive,
594                 'Data Log Rate: \t\t%.1f[sec]' % self.
595                 DataLogRate,
596                 'Battery Remaining: \t%.1f[%%]' % self.
597                 batt_rem]
598
599         return '\n'.join(data)
600
601 #=====
602 #
603 #   Lookup Dictionary Actions to incoming Command
604 #=====
605
606     def parse_com(self, command):
607
608         command = command.replace('}', '')
609         com = command.split(',')
610
611         if len(com) == 8:
612             return com
613
614         if len(com) == 1:

```

```

603         return None
604
605         if 1 < len(com) < 8:
606             print("*****Command Received*****")
607             key, act = com[0:2]
608
609             if key in self.act_dict and act in self.
act_dict[key]:
610                 return self.comm_dict[key](self.act_dict[
key][act])
611
612             elif key in self.comm_dict:
613                 return self.comm_dict[key](com[1:])
614
615             return None
616
617         return 'Message ERROR'
618
619 #=====
=====
620 #   Define Standard Actions for Incoming Commands
621 #=====
=====
622     def set_arm(self, state = True):
623
624         self.arm = state
625
626         if self.arm:
627             XB.armed = True
628             return 'Motors Armed'
629         MP.go_forward = False
630         MP.go_back = False
631         XB.armed = False
632         return 'Motors DisArmed'
633
634     def set_drive(self, states = [False, False, False]):
635
636         self.drive_Grad, self.drive_Cont, self.rem_drive
= states
637         self.assemble, self.swarm = False, False
638         self.follow = [False, ' ']
639
640         if self.drive_Grad:
641             return 'Driving by Gradient'
642

```

```

643         elif self.drive_Cont:
644             return 'Driving by Contour'
645
646         elif self.rem_drive:
647             return 'Driving by Remote'
648
649         return 'Driving by GPS waypoint'
650
651     def set_turn(self, turn):
652
653         self._goal_BRG = (self.goal_BRG + turn) % 360
654
655         return 'Goal Bearing = %.1f' % self.goal_BRG
656
657     def reset_motors(self, filler=[]):
658         print("Made it to reset motors in Mother")
659         MP.reset_Motors()
660
661     def set_XBee(self, addr):
662
663         if len(addr) is not 2:
664             return 'XBee Address Input ERROR'
665
666         XB.params['DH'], XB.params['DL'] = addr
667
668         args = (XB.params['DH'], XB.params['DL'])
669         string = 'XBee RESET. DH: %s, DL: %s' % args
670
671         if XB.params['DL'] == '0000FFFF':
672             MD.reset()
673             args = (MD.Mother_file[-6:-4], MD.GPS_file[-6
674 :-4])
675             string += '\tMother_File: %s, GPS_File: %s'
676 % args
677
678         return string
679
680 #=====
681 # Define Non-Standard Actions for Incoming Commands
682 #=====
683
684     def set_rate(self, rate=[]):

```

```

684         if len(rate) is not 1:
685             return 'DataRate Input ERROR'
686
687         self.DataLogRate = float(rate.pop(0))
688
689         return 'Refresh Rate = %.1f seconds' % self.
DataLogRate
690
691     def set_speed(self, spd=[]):
692
693         if len(spd) is not 1:
694             return 'Speed Input ERROR'
695
696         self.max_spd = float(spd.pop(0))
697
698         return 'Max Motor Speed: %.1f' % self.max_spd
699
700     def set_goal(self, coords=[]):
701
702         if len(coords) is not 2:
703             return 'Coordinate Input ERROR'
704
705         self.goal = [float(coord) for coord in coords]
706
707         return 'New GPS Goal --> LAT: %.6f, LON: %.6f' %
tuple(self.goal)
708
709     def set_leader(self, mother_id=[]):
710
711         if len(mother_id) is not 1:
712             return 'MotherID Input ERROR'
713
714         mother_id = mother_id.pop(0)
715
716         if len(mother_id) != 8 or mother_id not in MD.
swarm:
717             return 'Invalid Address'
718
719         states = [False, False, False]
720         self.drive_Grad, self.drive_Cont, self.rem_drive
= states
721         self.assemble, self.swarm = False, False
722
723         self.follow = [True, mother_id]
724

```

```

725         return 'Following Mother: %s' % mother_id
726
727     def set_contour(self, S=[]):
728
729         if len(S) is not 1:
730             return 'Contour Input ERROR'
731
732         S = S.pop(0)
733
734         if len(MD.temps) < MD.c_mn.size:
735             return 'Insufficient Data to Assemble on %s'
736         % S
737
738         xi = linspace(min(MD.lons), max(MD.lons))
739         yi = linspace(min(MD.lats), max(MD.lats))
740         xx, yy = meshgrid(xi, yi)
741
742         herm_kw = {'_x':MD._x, '_y':MD._y,
743                   '_L_x':MD.L_x, '_L_y':MD.L_y,
744                   '_ff':MD._ff}
745
746         cheb_kw = {'ax':min(MD.lons), 'bx':max(MD.lons),
747                   'ay':min(MD.lats), 'by':max(MD.lats)}
748
749         T_field = cheb.eval_2d_array(xx, yy, MD.c_mn, **
750 cheb_kw)
751
752         try:
753             self.path_x, self.path_y = get_contour(xx, yy
754 , T_field, float(S))
755         except ValueError:
756             return 'No Good Contour Locations for %s' % S
757
758         states = [False, False, False]
759         self.drive_Grad, self.drive_Cont, self.rem_drive
760 = states
761
762         self.swarm, self.follow = False, [False, ' ']
763
764         self.assemble = True
765
766         rng = [get_BRG_RNG(self.lat,self.lon,y,x)[1] \
767               for x,y in zip(self.path_x,self.path_y)]
768
769         self.p_indx = argmin(rng)
770

```

```

766         return 'Aligning to Contour: %s' % S
767
768     def set_corner(self, points=[]):
769
770         if len(points) is not 3:
771             return 'Points Input ERROR'
772
773         lat = points.pop(0)
774         lon = points.pop(0)
775         corner = points.pop(0)
776
777         if corner == 'UL':
778             self.UL = [float(lon), float(lat)]
779             return 'Upper Left Corner - LAT: %s, LON: %s'
780 % (lat, lon)
781
782         if corner == 'LR':
783             self.LR = [float(lon), float(lat)]
784             return 'Lower Right Corner - LAT: %s, LON: %s'
785 % (lat, lon)
786
787         return 'Bad Corner Inputs'
788
789     def set_pattern(self, D=[]):
790
791         if len(D) is not 1:
792             return 'Pattern Input ERROR'
793
794         theta = float(D.pop(0))
795
796         if not len(self.UL) or not len(self.LR):
797             return 'Missing UL or LR Coords'
798
799         kw = {'rows':5, 'cols':5, 'rot':theta}
800
801         try:
802             self.path_x, self.path_y = lawn_mower(self.UL
803 , self.LR, **kw)
804         except ValueError:
805             return 'No Good Pattern Data'
806
807         states = [False, False, False]
808         self.drive_Grad, self.drive_Cont, self.rem_drive
809 = states
810
811         self.swarm, self.follow = False, [False, ' ']

```

```

807
808         self.assemble = True
809
810         rng = [get_BRG_RNG(self.lat,self.lon,y,x)[1] \
811                 for x,y in zip(self.path_x,self.path_y)]
812
813         self.p_indx = argmin(rng)
814
815         args = (self.path_y[self.p_indx], self.path_x[
self.p_indx], theta)
816
817         return 'Driving Pattern at %.6f, %.6f - Rotated
by %s[deg]' % args
818
819     def set_swarm(self, coef=[]):
820
821         if len(coef) is not 2:
822             return 'Coefficient Input ERROR'
823
824         if len(MD.swarm) < 2:
825             return 'Insufficient Swarm Members: %.0f' %
len(MD.swarm)
826
827         S = float(coef.pop(0))
828         H = float(coef.pop(0))
829
830         self.swarm_coef = (S, H)
831
832         states = [False, False, False]
833         self.drive_Grad, self.drive_Cont, self.rem_drive
= states
834         self.assemble, self.follow = False, [False, ' ']
835
836         self.swarm = True
837
838         return 'Swarming with S:%.1f, H:%.1f' % self.
swarm_coef
839
840
841     def _check_wait(old,new):
842         wait_time = False
843         if (old < 9.0 and new > 9.0) or (old > 9.0 and new <
9.0):
844             wait_time = True
845         return wait_time

```

```
846
847
848 def _in_range(val, vmin=7.4, vmax=10.2):
849     if val < vmin:
850         val = vmin
851     if val > vmax:
852         val = vmax
853     return val
854
855
856 def heading_diff(goal, head):
857     rel = head - goal
858     if abs(rel) > 180.:
859         rel = -sign(rel)*(360. - rel*sign(rel))
860     return rel
```

```

1  '''
2  @author: Brandon Zoss
3  MIT          2015-2016
4  Edited by: Andrew Freeman
5  Original filename "Buoy_Pumps.py"
6  MIT          2016-2019
7  '''
8  import Adafruit_BBIO.PWM as PWM
9
10 import time
11
12
13 class MotherPumps(object):
14
15     def __init__(self, mode = 'tri'):
16         PWM.stop("P9_14")
17         PWM.stop("P8_19")
18         PWM.cleanup()
19
20         self.Pin = {'P1':{'PWM':'P9_14'},
21                     'P2':{'PWM':'P8_19'}}
22
23     for pump in self.Pin:
24         PWM.start(self.Pin[pump]['PWM'], 9.0, 60)
25         time.sleep(7)
26         self.pump_power = [0., 0.]
27         self.turn_burst_left = False
28         self.turn_burst_right = False
29         self.go_forward = False
30         self.go_back = False
31         self.burst_time = 1.0
32         self.max_speed = 100.0
33         self.thruster_duty_cycle = [9.0, 9.0]
34         self.auto_heading = False
35
36     def reset_Motors(self):
37         print("Made it to reset motors in Motor Pumps")
38         PWM.stop("P9_14")
39         PWM.stop("P8_19")
40         PWM.cleanup()
41         PWM.start("P9_14", 9.0, 60)
42         PWM.start("P8_19", 9.0, 60)
43         time.sleep(3)
44         PWM.set_duty_cycle("P9_14", 9.0)
45         PWM.set_duty_cycle("P8_19", 9.0)

```

```

46
47     def set_Motors_duo(self, arm, max_spd, rel_BRG):
48         """
49         Calculate required pump output for a duo
arrangement
50         based on desired heading and speed.
51
52         Designed for BlueRobotics T100 Thrusters:
53         Max limits: 1100 - 1900 ms
54         Operationally Limit signal to 1230 - 1700 ms
55         Equivalent to a duty cycle of: 7.4 to 10
56         """
57
58         # P9_14 is the left thruster
59         # P8_19 is the right thruster
60
61         neut = 9.0
62         rev = ((10.2 - neut) * (max_spd/100.0)) + neut
63         fwd = neut - ((neut - 7.4) * (max_spd/100.0))
64
65         burst_aft = ((10.2 - neut) * (max_spd / 100.0)) +
neut
66         burst_fwd = neut - (max_spd / 100.0) * (neut - 7.4
)
67
68         if not arm:
69             self.pump_power = [0., 0.]
70
71         try:
72             PWM.set_duty_cycle("P9_14", 9.0)
73             PWM.set_duty_cycle("P8_19", 9.0)
74         except:
75             PWM.stop("P9_14")
76             PWM.stop("P8_19")
77             PWM.cleanup()
78             PWM.start("P9_14", 9.0, 60)
79             PWM.start("P8_19", 9.0, 60)
80             time.sleep(3)
81             PWM.set_duty_cycle("P9_14", 9.0)
82             PWM.set_duty_cycle("P8_19", 9.0)
83
84         return self.pump_power
85
86         if self.turn_burst_left:
87             print("Burst Time:")

```

```

88         print(self.burst_time)
89         try:
90             print("Burst left")
91             print(burst_aft)
92             print(burst_fwd)
93             print(max_spd)
94             PWM.set_duty_cycle("P9_14", burst_aft)
95             PWM.set_duty_cycle("P8_19", burst_fwd)
96             time.sleep(self.burst_time)
97             PWM.set_duty_cycle("P9_14", neut)
98             PWM.set_duty_cycle("P8_19", neut)
99         except:
100             print("Burst left exception")
101             PWM.stop("P9_14")
102             PWM.stop("P8_19")
103             PWM.cleanup()
104             PWM.start("P9_14", 9.0, 60)
105             PWM.start("P8_19", 9.0, 60)
106             time.sleep(3)
107             PWM.set_duty_cycle("P9_14", burst_aft)
108             PWM.set_duty_cycle("P8_19", burst_fwd)
109             time.sleep(self.burst_time)
110             PWM.set_duty_cycle("P9_14", 9.0)
111             PWM.set_duty_cycle("P8_19", 9.0)
112         self.turn_burst_left = False
113         return self.pump_power
114
115     elif self.turn_burst_right:
116         try:
117             print("Burst right")
118             PWM.set_duty_cycle("P9_14", burst_fwd)
119             PWM.set_duty_cycle("P8_19", burst_aft)
120             time.sleep(self.burst_time)
121             PWM.set_duty_cycle("P9_14", neut)
122             PWM.set_duty_cycle("P8_19", neut)
123         except:
124             print("Burst right exception")
125             PWM.stop("P9_14")
126             PWM.stop("P8_19")
127             PWM.cleanup()
128             PWM.start("P9_14", 9.0, 60)
129             PWM.start("P8_19", 9.0, 60)
130             time.sleep(3)
131             PWM.set_duty_cycle("P9_14", burst_fwd)
132             PWM.set_duty_cycle("P8_19", burst_aft)

```

```

133         time.sleep(self.burst_time)
134         PWM.set_duty_cycle("P9_14", 9.0)
135         PWM.set_duty_cycle("P8_19", 9.0)
136         self.turn_burst_right = False
137         return self.pump_power
138
139     elif self.auto_heading:
140         if rel_BRG >= 15 and rel_BRG <= 345:
141             if rel_BRG <= 180:
142                 self.turn_burst_right = True
143             else:
144                 self.turn_burst_left = True
145             return self.pump_power
146
147     try:
148         PWM.set_duty_cycle("P9_14", self.
thruster_duty_cycle[0])
149         PWM.set_duty_cycle("P8_19", self.
thruster_duty_cycle[1])
150     except:
151         PWM.stop("P9_14")
152         PWM.stop("P8_19")
153         PWM.cleanup()
154         PWM.start("P9_14", 9.0, 60)
155         PWM.start("P8_19", 9.0, 60)
156         time.sleep(3)
157         PWM.set_duty_cycle("P9_14", self.
thruster_duty_cycle[0])
158         PWM.set_duty_cycle("P8_19", self.
thruster_duty_cycle[1])
159
160     return self.pump_power
161
162
163 #=====
=====
164 #   Set Mother Speed based on Goal Approach
165 #=====
=====
166     def set_Speed(self, dist, max_spd):
167         """
168         Sets motor speed based on approach to goal in a
169         linearly decaying fashion.
170         """
171

```

```
172         spd = max_spd
173         self.max_speed = max_spd
174
175         if 1.0 < dist < 2.5:
176             spd = max_spd * dist / 5
177
178         elif dist < 0.5:
179             spd = 0
180
181         return spd
182
183     @property
184     def Thrust_Vector(self):
185         """
186         Returns resulting thrust vector based on
187         individual
188         pump inputs.
189
190         Used to qualify calculated pump outputs.
191         """
192
193         thrust_vector = [0., 0.]
194
195         return [0., 0.]
196
197     def get_Thrust_Vector_Quad(self, pump_power):
198
199         return [0., 0.]
200
```

# Bibliography

- [1] B. M. Zoss, D. Mateo, Y. K. Kuan, G. Tokić, M. Chamanbaz, L. Goh, F. Vallegria, R. Bouffanais, and D. K. Yue, “Distributed system of autonomous buoys for scalable deployment and monitoring of large waterbodies,” *Autonomous Robots*, vol. 42, no. 8, pp. 1669–1689, 2018.
- [2] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, “Unmanned surface vehicles: An overview of developments and challenges,” *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.
- [3] I. O’Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, and M. Yim, “Self-assembly of a swarm of autonomous boats into floating structures,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1234–1240, 2014.
- [4] J. Paulos, N. Eckenstein, T. Tosun, J. Seo, J. Davey, J. Greco, V. Kumar, and M. Yim, “Automated Self-Assembly of Large Maritime Structures by a Team of Robotic Boats,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 958–968, 2015.
- [5] B. Zoss, “Design and Analysis of Mobile Sensing Systems: an Environmental Data Collection Swarm,” Massachusetts Institute of Technology, Tech. Rep., 2016.
- [6] J. R. Higinbotham, P. G. Kitchener, and J. R. Moisan, “Development of a New Long Duration Solar Powered Autonomous Surface Vehicle,” *Oceans 2006*, pp. 1 – 6, 2006.
- [7] H. Joe, H. Roh, H. Cho, and S. C. Yu, “Development of a flap-type mooring-less wave energy harvesing system for sensor buoy,” *Energy*, vol. 133, pp. 851–863, 2017.
- [8] J. Manley, “Development of the autonomous surface craft ACES,” *Oceans ’97. MTS/IEEE Conference Proceedings*, vol. 2, 1997.
- [9] J. G. Bellingham, “Autonomous underwater vehicle docking,” in *Springer Handbook of Ocean Engineering*, X. N. I. Dhanak, Manhar R., Ed. Cham: Springer International Publishing, 2016, ch. 16, pp. 387–406.

- [10] A. Martins, J. M. Almeida, H. Ferreira, H. Silva, N. Dias, and E. Silva, "Autonomous surface vehicle docking manoeuvre with visual information," *Proc. ICRA07 IEEE International Conference on Robotics and Automation*, no. April, pp. 10–14, 2007.
- [11] M. Dunbabin, B. Lang, and B. Wood, "Vision-based docking using an autonomous surface vehicle," *2008 IEEE International Conference on Robotics and Automation*, pp. 26–32, 2008.
- [12] D. Amarasinghe, G. K. I. Mann, and R. G. Gosine, "Vision-based hybrid control strategy for autonomous docking of a mobile robot," *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, vol. 21, no. 8, pp. 1600–1605, 2005.
- [13] T. Maki, R. T. Shiroku, Y. Sato, T. Matsuda, T. Sakamaki, and T. Ura, "Docking method for hovering type AUVs by acoustic and visual positioning," in *IEEE International Underwater Technology Symposium*, 2013.
- [14] D. E. Frye, J. Kemp, W. Paul, and D. Peters, "Mooring developments for autonomous ocean-sampling networks," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 477–486, 2001.
- [15] S. Brizzolara, "The Second Generation of Autonomous Surface Vessels - Optimized Performance for AUVs Assistance at Sea," in *Oceans 2015: Genova*. IEEE, 2015.
- [16] Y. H. Kim, S. W. Lee, H. S. Yang, and D. A. Shell, "Toward autonomous robotic containment booms: Visual servoing for robust inter-vehicle docking of surface vehicles," *Intelligent Service Robotics*, vol. 5, no. 1, pp. 1–18, 2012.
- [17] L. Somerville, J. Bareño, S. Trask, P. Jennings, A. McGordon, C. Lyness, and I. Bloom, "The effect of charging rate on the graphite electrode of commercial lithium-ion cells: A post-mortem study," *Journal of Power Sources*, vol. 335, pp. 189–196, 2016.
- [18] J. N. Newman, *Marine Hydrodynamics*. MIT Press, 1977.
- [19] WAMIT Inc., "WAMIT User Manual Version 5.4," 2019. [Online]. Available: <https://www.wamit.com/techdescription.htm>
- [20] A. Hulme, "The wave forces acting on a floating hemisphere undergoing forced periodic oscillations," *Journal of Fluid Mechanics*, vol. 121, no. 57, pp. 443–463, 1982.
- [21] Ford, "2019 Ford® Transit Cargo Van Model Highlights," 2019. [Online]. Available: <https://www.ford.com/commercial-trucks/transit-cargo-van/models/transit-van/>
- [22] D. Molloy, *Exploring BeagleBone®*, 2nd ed. Wiley, 2018.

- [23] BlueRobotics, “T100 Thruster Technical Specifications,” 2019. [Online]. Available: <https://www.bluerobotics.com/store/thrusters/t100-t200-thrusters/t100-thruster/>
- [24] A. Spannbauer, “Module documentation: Python Video Stabilization with OpenCV,” 2018. [Online]. Available: [https://adamspannbauer.github.io/python\\_video\\_stab/html/module\\_docs.html](https://adamspannbauer.github.io/python_video_stab/html/module_docs.html)
- [25] A. Rosebrock, “Tutorial : Using CamShift to Track Objects in Video Using CamShift to Track Objects in Video,” pp. 2–3, 2014. [Online]. Available: [https://www.pyimagesearch.com/wp-content/uploads/2014/11/opencv\\_crash\\_course\\_camshift.pdf](https://www.pyimagesearch.com/wp-content/uploads/2014/11/opencv_crash_course_camshift.pdf)
- [26] —, “Ball Tracking with OpenCV,” 2015. [Online]. Available: <https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/>
- [27] —, “OpenCV Object Tracking,” 2018. [Online]. Available: <https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/>
- [28] G. Bradski, “Computer Vision Face Tracking as a Component of a Perceptual User Interface,” *Workshop on Applic. of Comp. Vision*, pp. 214–219, 1998.