Design, Development, and Modeling of a Novel Underwater Vehicle for Autonomous Reef Monitoring

by

Alex Goldman

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2020 by the
Graduate Supervisory Committee:

Jnaneshwar Das, Chair
Greg Asner
Hamidreza Marvi

ARIZONA STATE UNIVERSITY

December 2020

©2020 Alex Goldman

All Rights Reserved

ABSTRACT

A novel underwater, open source, and configurable vehicle that mimics and leverages advances in quad-copter controls and dynamics, called the uDrone, was designed, built and tested. This vehicle was developed to aid coral reef researchers in collecting underwater spectroscopic data for the purpose of monitoring coral reef health. It is designed with an on-board integrated sensor system to support both automated navigation in close proximity to reefs and environmental observation. Additionally, the vehicle can serve as a testbed for future research in the realm of programming for autonomous underwater navigation and data collection, given the open-source simulation and software environment in which it was developed. This thesis presents the motivation for and design components of the new vehicle, a model governing vehicle dynamics, and the results of two proof-of-concept simulation for automated control.

ACKNOWLEDGMENTS

Developing the uDrone was truly a group effort. First and foremost I would like to thank my committee chair, Dr. Jnaneshwar Das, for allowing me to realize one of my lifelong dreams and work on an underwater vehicle. His support was vital through the entirety of this project.

Fellow members of the DREAMS Lab put in great effort to create the uDrone. A.L.G Prasad and Harish Anand were key contributors of code and simulation development. Cole Brauer, Rodney Staggers Jr, and Devin Keating were responsible for the terrific physical design and construction of the uDrone. Zhiang Chen and Sarah Bearman were amazing sounding boards for the uDrone vision and science applications.

Dr. Greg Asner, a committee chair, came up with the inspiration for the creation of the uDrone. His team at GDCS, particularly Jiwei Li and Nick Vaughn, provided much helpful background.

Dr. Hamidreza Marvi, a committee chair, provided education and insight for the development of the mathematical model.

I would also like to thank all my teammates for class projects who often allowed us to work on tasks related to the uDrone. In particular, Evan Lamb and Jason Newton contributed to the optimal controls project that is featured chapter 5 of this thesis.

Finally, I would not be here today without the love and support of my wife, sisters, parents, and friends. Thank you all.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

The need for creating an autonomous underwater vehicle surfaced when the Distributed Robotic Exploration and Mapping Systems (DREAMS) Lab became a part of the Center for Global Discovery and Conservation Science (GDCS) at Arizona State University (ASU). DREAMS has experience building and deploying autonomous systems that aid in scientific studies. These systems are typically unmanned aerial vehicles (UAV) or drones, but boats and submersibles are also in operations. GDCS participates in a range of conservation sciences, but one of their main focuses is on imaging, both in the visible spectrum and beyond. Ecologically, they have focused on rain forests and coral reefs, among other biomes. Considering the capabilities of DREAMS and the needs of GDCS for methods of imaging and exploring coral reefs, the idea of building an underwater vehicle was born.

Autonomous Underwater Vehicles (AUV) and Remotely Operated Vehicles (ROV) have been exploring the oceans for decades. However, many of these vehicles have drawbacks and limitations. ROVs are typically used when precision control and navigation of close quarters are needed. To accomplish this they require an experienced operator on the surface connected to the vehicle via a tether. The need for a tether limits where ROVs can go and who can deploy them. AUVs run autonomously, not needing a tether or operator. They are typically used in open ocean environments, where there are no obstacles or impediments to following a predetermined flight path. Making matters more difficult, most commercially available ROVs and AUVs

come with proprietary software systems, limiting the ability to use state-of-the-art autonomous navigation tools.

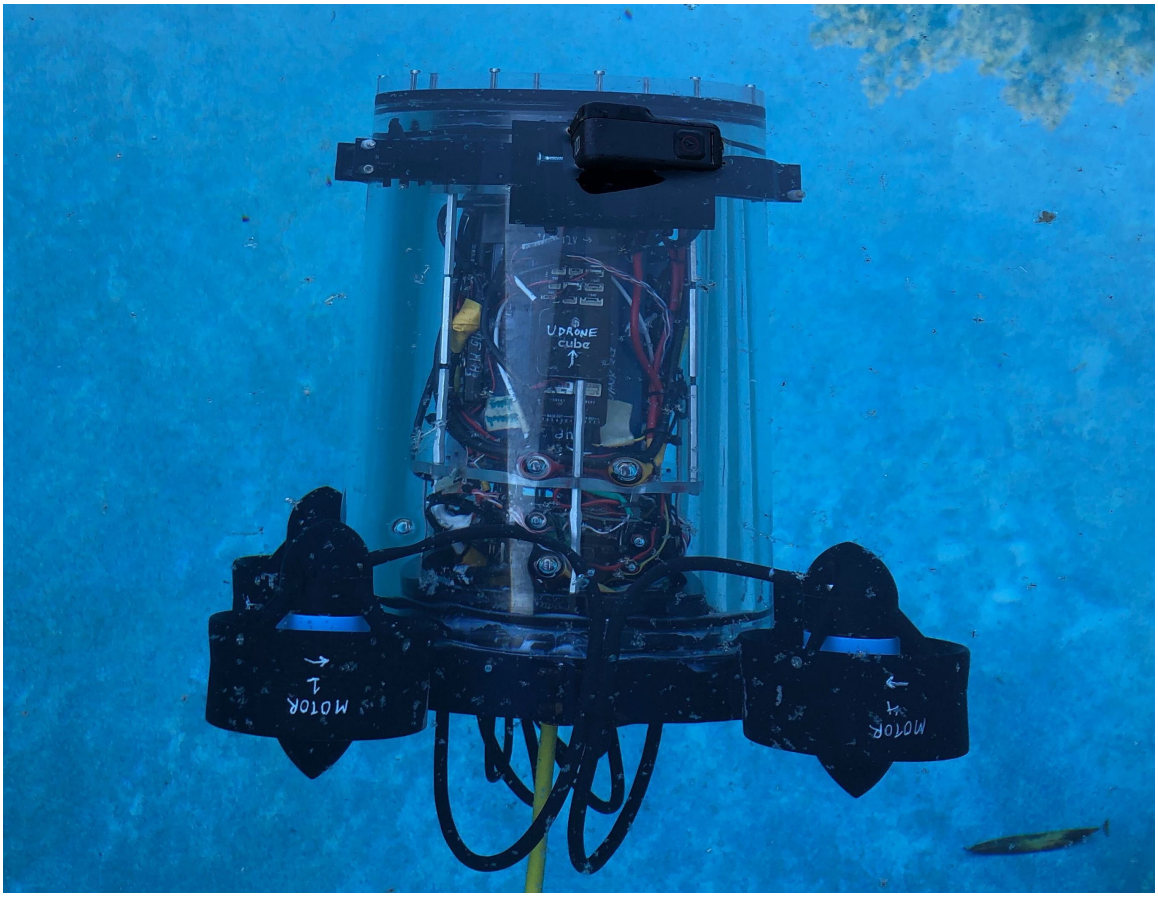

Figure 1. uDrone in Pool Test

Source: Photo courtesy of Jnaneshwar Das, used with permission.

For all these reasons, the DREAMS lab decided to build a new, customized autonomous underwater vehicle. Based on a motor configuration that mimics that of a quad-rotor aerial vehicle, colloquially referred to as "drones", this vehicle was named the uDrone. This chapter will outline the motivation for developing the uDrone. Subsequent chapters will go through the background of the underwater vehicle world before the uDrone, the technical details of the uDrone system and its development,

and a detailed mathematical model of the vehicle dynamics. Lastly, two methods of autonomously controlling the uDrone are explained, evaluated, and compared.

1.1 Thesis Statement

An autonomous underwater drone testbed that leverages state of the art avionics and machine vision algorithms will enable safe, cost-effective, and time-efficient coral reef exploration and monitoring at a global scale.

This statement can be broken down to better understand the project goals. Autonomous is defined as being able to operate without direct human control and supervision. Underwater drone is the vehicle itself, emphasizing the setting it is meant to explore. Testbed is a set of software and simulation tools to enable rapid development and testing. State of the are avionics refers to control methodologies and algorithms developed for quad-rotor aerial drones that can be applied to this project. Machine vision algorithms will be used to create and understanding for the vehicle of its surroundings, which is key to enabling autonomy. The uDrone seeks to be safer and less time consuming than the diving being done for related research currently while meeting the budgetary goals. And reef exploration and monitoring is the vehicles ultimate goal and purpose; all the other elements combine to enable this final goal.

1.2 Science Motivation

GDCS has several ways of obtaining large quantities of imaging data. They contract with Planet satellites and fly an aeroplane fitted with special equipment,

known as the Global Airborne Observatory (GAO). These methods have been used to do extensive research on South American rain forests whereas coral reefs are the current "frontier" of research for the GAO. Better imaging and monitoring of coral reefs will enhance researchers understanding of the effects of climate change on reefs and allow for more detailed coral models to be created.

The immediate goal is to develop a three-dimensional structure of coral reefs that contains full spectral data, from 400 to 800 nm (Li et al. 2019). This goal is made more difficult by the inherently complex physics of light moving through water. Several methods have been explored to essentially "see through" the water and retrieve the data from the coral alone. In one approach, the inherent properties of the water are used to calculate the effects of the water on the light spectra moving through it (Thompson et al. 2017). In another, bottom reflectance data is collected by scuba divers with specialized equipment and used to calibrate the data from satellites (Li et al. 2020).

This last method necessitates a human operating in a potentially dangerous environment to collect data. Specifically, Dr. Greg Asner, head of GDCS would use a ADS HandHeld 2 Spectrometer attached to a diver propulsion vehicle (DPV) to collect samples. These data collection trips would last up to four hours and very few researchers have the technical diving capability or constitution to complete them. This was the initial motivation for the uDrone project. Figure 2 shows the DPV and its use on the reef.

The risk of death from scuba diving is low, with an estimated annual death rate of 2 deaths per 100,000 recreational scuba divers in the United States. Many of these deaths are due to cardiovascular health and diver errors (Buzzacott 2016). These risks are mitigated in the scientific diving community by the increased requirements set

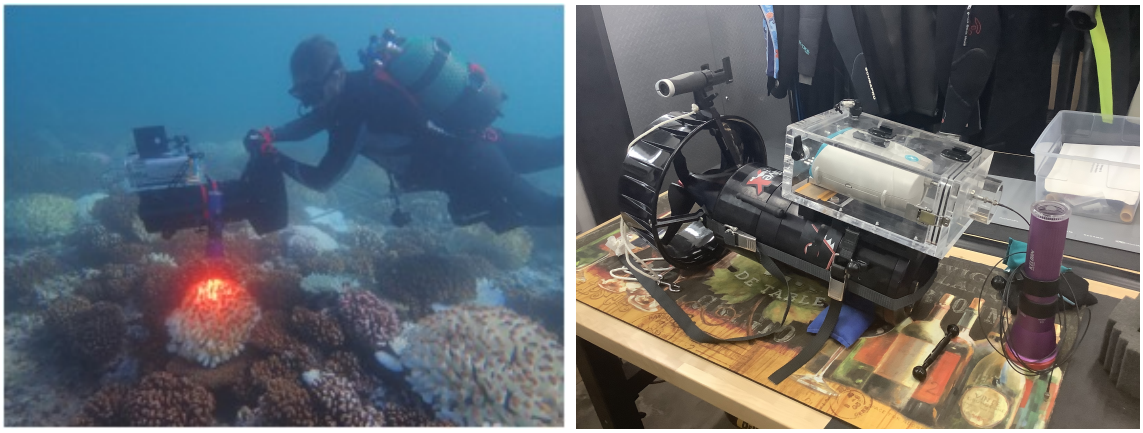

Figure 2. Dr. Greg Anser using a DPV for collecting data (left) and the DPV out of water with the sensor attached

Source: Left photos courtesy of Greg Asner, used with permission.

out by the American Association of Underwater Scientists (AAUS), which require additional training and medical checks (AAUS 2019). ASU and all associated labs follow these requirements for diving activities.

Dive injury, however, is significantly more common than fatality, with an estimated 15 occurrences per 10,000 dives. While there are many potential avenues for dive injury, including interactions with the marine environment, barotrauma, and gas contamination, one ailment in particular can strike divers at any skill level and is particularly exacerbated by long and repeat dives. This is decompression sickness. Decompression sickness is the second most common type of dive injury, accounting for 27% of all dive injuries according, to the Divers Alert Network (DAN), the world's largest scuba diving safety association. (Barotrauma is the most common accounting for 41% of injuries).

Decompression sickness, or DCS, is caused when gas bubbles absorbed by body tissue at higher pressures underwater are not able to properly "off-gas" or reabsorb. This can cause bubbles to form in various regions of the body. Most commonly this

occurs in joints or tissue which can impair motor function and have permanent effects. In some cases a bubble can form in the circulatory system, blocking blood flow, which can be fatal (Hall 2014).

In order to reduce the risk of DCS while diving, scuba divers use tools to monitor the dissolved gasses in their system. Historically, this has been done with tables before and after a dive but most modern divers use a dive computer which tracks their gas absorption. This risk can also be mitigated for especially deep or long dives by mixing inert gasses into the breathable air in order to limit the amount of any one gas that gets absorbed.

An AUV, such as the uDrone, can be deployed to totally eliminate the health risks associated with human-based coral reef data collection. This also opens up the possibility of significantly more data to be collected for the existing GDCS projects, along with finding new opportunities for collecting and using reef data. For example, researchers at the University of Hawaii, Hilo are collecting detailed imaging data of coral reefs and using them to create three-dimensional reconstructions of the reef. These reconstructions are then incorporated into other data sets for ecological monitoring (Fukunaga et al. 2019). The uDrone would allow for more data to be collected with less human health risk as it replaces divers who collect data.

1.3 Engineering Motivation

While the scientific need inspired its initial creation, the uDrone is useful beyond these specific implementations. It is quickly becoming a testbed for innovation in autonomous underwater navigation within the DREAMS Lab. One thesis has already

been written leveraging the system for multi-robot coordination and several others are in the works at this time.

The development of a testbed starts with two parallel tracks: hardware development and software in the loop (SITL) simulation. In the hardware development phase the needs of the vehicle are determined, currently available hardware is specified and sourced, and the vehicle itself is put together. SITL simulation is a method of simulation where all elements of the vehicle are simulated, including the flight controller, the internal computation, the vehicle itself, and the world in which it operates. These tracts combine to a third step: hardware in the loop simulation. In this step the vehicle motion is still simulated in a virtual world, but the flight controller and internal CPU of the actual vehicle interact with this simulation. This step leads to the final step in the chain, which is real world deployment.

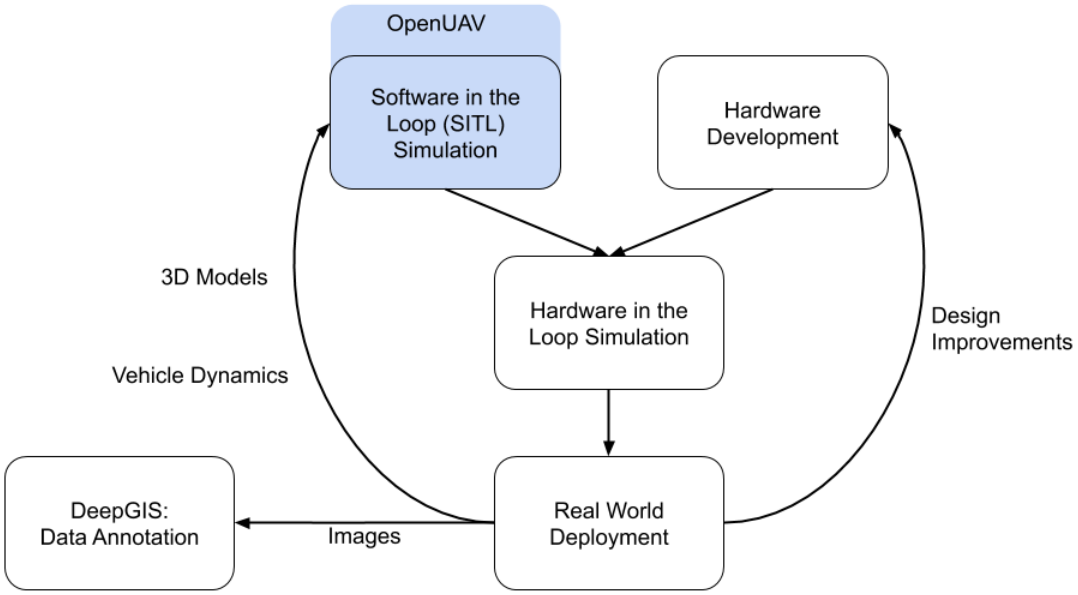

Figure 3. The testbed steps and cycle of development

This process, illustrated in Figure 3, is not a straight path but a cycle. The

real world deployment generates leanings which are put back into the first steps to improve the uDrone. Design improvements are gathered every time the vehicle is put together or deployed. In fact, the vehicle pictured in Figure 4 is the third iteration of the internal structure to hold the equipment. The real world deployments also generate more data about the vehicle's dynamics and images collected can be used to create additional 3D reef models. These can be used to make the simulation environment more realistic and accurate. This also feeds nicely into other DREAMS lab projects. The SITL simulation has become part of the OpenUAV stack, a tool for rapid prototyping and remote working. And the images collected from deployment can be fed into DeepGIS, a tool for annotating data for machine learning.

Chapter 2

BACKGROUND

There are many unique challenges to building autonomous vehicles for underwater environments. And, several other university labs and organizations have build vehicles for aquatic exploration. This section will discuss these challenges and several of the most relevant vehicles.

2.1 Challenges of Underwater

Working in underwater environments presents a number of challenges not experienced by aerial, ground, or surface vehicles. These include physical challenges, difficulty of communication, and lack of global positioning systems.

2.1.1 Physical Challenges

First and foremost is the need to seal the vehicle for water and pressure. The uDrone will need to maintain a watertight seal down to 30m of water. This is made more challenging by the need to have some components exterior to the enclosure which houses the electronics, necessitating the creation of watertight pass-throughs. As the vehicle goes to deeper depths this becomes increasingly harder. Every 10m of water adds an additional atmosphere of pressure to the vehicle. This pressure causes two problems. First, enclosures must be strong enough to resist the force of the pressure

change and not fail structurally. Second, o-rings and other seals shrink under pressure, which can lead to leaks.

In order to achieve this, the uDrone uses a Blue Robotics enclosure to keep the electronics dry inside. In order to handle the cable pass-throughs, and o-ring sealed pass-through is attached to the rear of the enclosure and then the potting compound is used to seal the area between the cable and pass-through.

An underwater vehicle also has to manage its buoyancy. If the vehicle is too heavy or too light it will end up moving through the water column at an angle, increasing drag and decreasing efficiency. Additionally, if the centers of buoyancy and mass do not line up then there will be a moment acting on the vehicle, complicating modeling and control. While this presents challenges, it can also lead to opportunities. By creating a vehicle that is slightly positively buoyant or by having a mechanism that allows a vehicle to drop ballast, a vehicle will float and can be recovered in the event of failure. Since the main camera on the uDrone is front-facing, the center of buoyancy can be set slightly behind the center of gravity. This will cause it to point down when hovering, but in motion, the thrusters can compensate for the moment.

Many autonomous vehicles are able to simplify certain aspects of the model. For example, a vehicle moving through the air does not need to worry about added mass from the layer of air near the vehicle surface. In fact, air drag is often ignored unless a drone is moving at high speeds. This is not possible underwater due to the greater viscosity and density of water compared to air. In order to get reliable system modeling, expensive and complicated fluid dynamic software must be used. Since this is often impractical for applications such as the uDrone, the values for added mass and drag are often estimated and then verified experimentally.

Similar to the effects of wind on an aerial vehicle, underwater vehicles must be able

to compensate for currents. This is made more challenging as the speed of the vehicle cannot be verified by GPS. Instead, visual odometry must be used to determine the effects of current on the vehicle and aid in compensation.

2.1.2 Communication Challenges

WiFi or radio signals do not penetrate to useful depths underwater. Radio waves in the MHz or higher frequency range typically penetrate less than 1 meter underwater. Since the radio controllers used on drones communicate at 900 MHz and WiFi uses 2.4 GHz or 5 GHz, none of these are viable communication options.

Optimal communication is possible but is limited by line of sight and water column turbidity (Hydromea 2020). Optimal communication has been implemented in a simulation for the uDrone, where it follows a boat that has a visual marker mounted underneath. This is reasonable for some use cases of the uDrone, but cannot be relied on for all scenarios.

The most common method of real-time communication with underwater vehicles is via a tether. This allows for fast and reliable connections but limits the movement of the vehicle based on cable length and cable management. While the uDrone is capable of functioning with a cable for testing purposes, it is designed to operate without one.

The only reliable method of wireless underwater communication is via acoustic transponders. However, these devices tend to be large, expensive, and power-hungry. While one day it might be useful to have wireless acoustic communications at specific sites of uDrone deployment, it is not practical in the initial use case.

2.1.3 Global Positioning and Navigation Challenges

GPS satellites transmit data at high frequencies, in the GHz range. For this reason, traditional, satellite-based GPS is not practical for use underwater. The uDrone is still outfitted with GPS so it can determine its location on the surface, but it cannot rely on this as a method of localization while underwater.

It is possible to use acoustic signals to localize an underwater vehicle. This requires three or more surface-based acoustic broadcasters. An underwater vehicle can then triangulate between these sources in order to determine its position relative to the sources. This is basically the equivalent to creating an underwater GPS system. While it is possible that this could be implemented at some operational sites in the future, it is not practical to build in the beginning and is therefore not a reliable way for the uDrone to localize.

Due to the lack of GPS, the uDrone must use visual and inertial based odometry to determine its location. LIDAR, which is the measurement tool of choice of most aerial and land-based robots, is severely limited underwater for two reasons. First, lower frequency light is absorbed quickly underwater, and since most LIDARs use these lower frequencies, the range is severely limited. Second, murky water greatly limits the ability of light to pass through water, and since it is very common to have many dissolved particles in the water column this makes LIDAR inaccurate. Therefore, the best type of sensor to measure distance underwater is with an echo-sounder.

Another tool typically used by land-based robotics to understand their surroundings is using depth cameras. There are two main types of depth cameras: stereo and infrared. Stereo cameras calculate the pixel depth of an image by looking at the difference in location between the images from two cameras placed a known distance

apart. Infrared cameras work by projecting a pattern in infrared light onto surfaces and using the deformations to calculate distance. Since lower frequency lights, such as red and infrared, are absorbed most quickly by water, infrared depth cameras do not work well underwater. For this reason, a stereo camera is the most practical type of depth camera for use underwater.

2.2 Previous Work

Existing Autonomous Underwater Vehicles (AUVs) and Remote Operate Vehicles (ROVs) tend to fall into 2 categories: Flight style and hover style. Flight style vehicles are similar to torpedoes. They typically have long, cylindrical bodies with a single thruster at the rear and control surfaces to control roll, pitch, and yaw while moving. These vehicles are typically used in the open ocean to complete larger area surveys, which necessitates their long range and higher speeds. The trade offs for speed, endurance, and range are less controllability, agility, and precision. Hover style vehicles have nearly opposite abilities. They are typically box shaped with multiple thrusters in various directions allowing for precise movement in any direction. Due to the challenge of controlling these highly maneuverable vehicles, they are typically ROVs, meaning there is a pilot on the surface connected via a tether and they are not autonomous.

In this chapter I will present four vehicles, one each in the flight and hover style and then two in a newer category of underwater vehicle, called hydrobatic. Hydrobatic vehicles are both fast and agile, balancing the trade-offs between the range and speed or flight style vehicles and the maneuverability of hover style vehicles. The two hydrobatic vehicles presented here served as inspiration for the uDrone. This section

is not meant to be a comprehensive review of underwater vehicles. Instead, it will serve to show some examples and give some background of the research that inspired the uDrone.

2.2.1 HippoCampus μUAV

The HippoCampus micro underwater vehicle was developed at the Institute of Mechanics and Ocean Engineering, Hamburg University of Technology in Germany. This small, agile, and inexpensive vehicle was developed for use in swarms. Similar to the uDrone thruster configuration, the HippoCampus drew on inspiration from multi-rotor aerial vehicles to develop a four propeller design (Hackbarth, Kreuzer, and Solowjow 2015). In fact, the open-source code developed for this vehicle has been a huge help in designing gazebo simulations and PX4 controllers for the uDrone. Robust models were derived for the HippoCampus which were used to show its ability to function as a submerged Furuta Pendulum (Duecker et al. 2018). Designed to be inexpensive and work in swarms, the HippoCampus has limited capacity for sensors. It uses a gyroscope, compass, and depth sensor, but does not make room for any visual or sonar sensors.

2.2.2 AQUA: an Aquatic Walking Robot

The AQUA robot was developed over 10 years ago by the Mobile Robotics Lab at McGill University. It is based on the RHex hexapod robot. Its original intent was to create an amphibious robot that could transition from walking on land to swimming in water. Instead of thrusters, which are used by the majority of underwater vehicles, the

AQUA robot uses six fins to propel itself through the water (Georgiades et al. 2004). More recently, this platform has been used with an on board GPU in order to handle vision based navigation (Manderson and Dudek 2018).

The AQUA, along with its simulation environment, is a good example of an underwater robotic testbed leading to innovation. The work has transitioned from hardware to computer vision and autonomous navigation. For example, using this system a new vision based underwater navigation tool called Nav2Goal was created. In this, the underwater vehicle moves to a goal defined by the user. Along the way, it not only avoids obstacles but also takes paths that bring it near to areas of greater interest. This is particularly useful in coral reefs when a vehicle can decide to follow a path along a coral reef instead of a sandy bottom on its way to its goal (Manderson et al. 2020).

While this vehicle satisfies most of the requirements of the uDrone project, it was mainly developed as a research vehicle and for amphibious implementations, making it not ideally suited to the project needs.

2.2.3 Light Autonomous Underwater Vehicle

The Light Autonomous Underwater Vehicle (LAUV) was developed in 2012 by the Laboratório de Sistemas e Tecnologia Subaquática (LSTS) at the Universidade do Porto in Protugal and is manufactured by OceanScan Marine Systems & Technology. This vehicle was designed to be small enough to be carried by one person while still having full capabilities for scientific and defence surveys. It is built in the 'flight' or 'torpedo' style of underwater vehicle, having a long cylindrical body with a single thruster at the rear and control surfaces to control roll, pitch, and yaw while moving.

The LAUV has an endurance of 6-8 hours at a speed of 1.4m/s, which would satisfy the same needs as the uDrone (Sousa et al. 2012). However, due to the flight style design it would not be able to reliably and precisely follow the terrain of the reef as necessitated by this project.

2.2.4 Blue Robotics ROV2

There are many ROVs exploiting the oceans, from the 2,400 kg ROV Minerva used for deep ocean research (NTNU 2020), to the 3.4 kg Sofar Trident consumer underwater drone (SofarOcean 2020). Because it uses many similar hardware components along with an open source software stack, the Blue Robotics ROV2 will be highlighted here. This vehicle is sold as a kit by Blue Robotics, utilizing many of its components. It is meant for both recreational enthusiasts and small scale research. The vehicle is from the surface via a tether. Onboard, it houses a PixHawk flight controller running ArduPilot, an alternative to PX4. Additionally, cameras, echo sounders, and other sensors can be added to the vehicle. It has a maximum speed of 1.5 m/s and a battery life of about 2 hours (Blue Robotics Inc. 2020c).

The Blue Robotics ROV2 has been used for autonomous underwater navigation experiments. One such set of experiments, conducted at the French-Mexican Laboratory on Computer Science and Control. The authors used the tether of the ROV2 to retrieve sensor data from underwater. This data was processed on a remote ground station and then sent back to the vehicle, again via the tether. The whole system operated with ROS and the MAVROS interface to ArduPilot. This allowed for real time and off board computer vision based autonomy (Manzanilla et al. 2019).

Chapter 3

TECHNICAL DETAILS OF UDRONE SYSTEM

This section will list the requirements for the uDrone system along with the chosen hardware components and software tools used to meet them.

3.1 System Requirements

Based on the science needs and goals of building a testbed, several design requirements emerged.

3.1.1 Science Requirements

- Maintain a constant speed of 1 m/s.
- Maintain a constant distance from the reef, ideally between 30 and 100 cm.
- Know the exact distance to the reef to calibrate spectrography readings.
- Operate for at least 4 hours.
- Be able to carry the necessary spectrography equipment and light source.
- Travel fully autonomously, with no tether or human control.
- Maintain a straight line path underwater.
- GPS tag the start and end locations in order to estimate the exact locations of all measurements in-situ.
- Be deployed and retrieved by one or two people from the shore or a small boat.

3.1.2 Testbed Requirements

- Use ROS and PX4 as open source frameworks.
- GPU to allow for on board machine learning and neural networks.
- Sensors to generate local awareness for navigation.
- Cost less than $10,000.
- Easy access to internal components.
- Each component replaceable or upgradable.
- Code can be tested in a robust simulation environment.

Figure 4. Side (left), front (center), and orthogonal (right) view of the uDrone

3.2 Hardware Components

The hardware components were chosen to be easy to obtain and mimic the construction of autonomous aerial vehicles as much as possible. The physical and logical connections of the uDrone system is diagramed in Figure 5. In this diagram the blue boxes represent physical components, the red boxes represent software frameworks, the solid lines represent physical connections, and the dashed lines represent logical

connections. The specific components listed here are discussed in the remainder of this chapter.

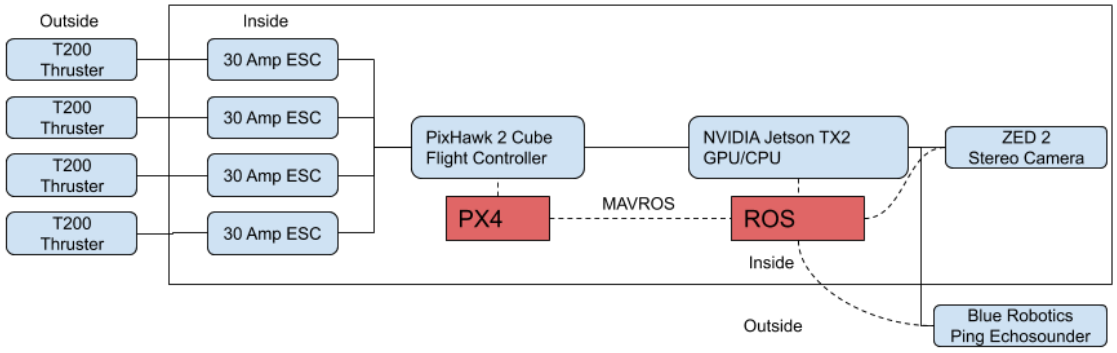

Figure 5. uDrone System Diagram

3.2.1 Blue Robotics T200 Thrusters

Blue Robotics is a leading source for marine robotics components. Arising from a kickstarter in 2014, they are DREAMS Lab's preferred vendor for many components. Their thrusters fit the uDrone's needs quite well, including their size, power usage, thrust capabilities, depth limits, and price. Additionally, as these are sold commercially it is very easy to have spares and order more on short notice. Given how perfectly these thrusters fit the requirements, no other thruster was seriously considered. One potential drawback to these thrusters is the expected lifetime of 600-1000 hours (Blue Robotics Inc. 2020b). While for some long range missions this would be a problem, with a mission time of only a few hours this poses no risks to the uDrone.

The thrusters are controlled via a 30 amp electronic speed controller (ESC). This device modulates the power to the device based on pulse width modulation (PWM) signals it receives from the flight controller.

3.2.2 PixHawk 2 Cube Black

The DREAMS Lab uses PixHawk flight controllers for most aerial and aquatic robotics applications. Due to the experience and expertise with this piece of equipment, the same type of flight controller is used for the uDrone. The cube was chosen over newer PixHawks, like the PixHawk 4 mini, because it is more widely used in the PX4 community and the smaller size was not needed for the uDrone implementation.

The PixHawk 2 Cube runs PX4 on the NuttX operating system. It has multiple built-in three-axis gyroscopes, accelerometers, and magnetometers along with barometers, forming redundant internal measurement units (IMUs). It communicates with the on-board computer via MAVLINK messages over a serial connection. These messages are generated on the on-board computer from the MAVROS package. The PixHawk is able to control the motors via PWM outputs.

These PixHawk devices typically run either PX4 or ArduPilot. Due to DREAMS Lab's expertise, along with the related work being done, PX4 is used for the uDrone. The PX4 software has several different flight modes, including position, altitude, acrobatic, and manual control. The uDrone uses the acrobatic mode controlled via the on-board computer. In this mode the controller receives the desired orientation or angular rate along with a thrust value. Utilizing the internal IMUs, a built in PID controller takes these inputs and sends the desired PWM command to each motor (PX4 Dev Team 2020).

3.2.3 Jetson TX2 with Connecttech Orbit Carrier

The on board computer is an NVIDIA Jetson TX2. This device has a 256-core NVIDIA Pascal™ GPU along with a Quad-Core ARM® Cortex®-A57 MPCore (NVIDIA Corporation 2020b). The ARM processor is used to run ROS while the GPU is used to run neural-networks and process imagery on board. This device is specifically made for embedded systems and robotic applications. The TX2 runs on a Linux kernel with NVIDIA's JetPack SDK (NVIDIA Corporation 2020a).

To make interfacing with the TX2 easier, it is purchased with a Connect Tech Orbitty Carrier. The board connects to the TX2 module and provides computer style connectors, such as HDMI and Ethernet (Connect Tech Inc. 2020). This allows for easy interfacing with the TX2 for development and testing purposes.

The NVIDIA Jetson TX2 is specifically built to be used with embedded robotic systems. The product's tagline is "The AI Platform for Autonomous Machines." This device has already been used for on board applications in autonomous underwater vehicles (Manderson and Dudek 2018). Given its form factor, capabilities, and previous uses in related fields, this was chosen over other on board AI products for use in the uDrone.

3.2.4 ZED 2

The StereoLabs ZED 2 was chosen as a stereo depth camera, and can be clearly seen in the center view of figure 4. StereoLabs specializes in AI-enabled stereo cameras. The ZED 2, their newest offering, has an internal IMU, barometer, and magnetometer to increase its visual servoing and depth-sensing abilities. The ZED 2 can also run neural

networks on board to accomplish things like person tracking (Stereolabs Inc. 2020). While this feature is not used at the moment on the uDrone, it could be used with the implementation of other neural networks, such as fish identification and tracking. ZED cameras also integrate directly into ROS, making incorporating them into the system easier. An example of the output from the ZED 2 stereo camera while the uDrone is undergoing a pool test can be seen in figure 6.

Figure 6. Stereo Camera Output from ZED 2 in uDrone Pool Test
Source: Photo courtesy of Jnaneshwar Das, used with permission.

3.2.5 Blue Robotics Ping Echosounder

The Blue Robotics Ping is a single beam echosounder for measuring distance underwater on the uDrone. It is attached in a downward facing orientation, allowing it to tell the distance of the uDrone to the reef. By aligning it in the same direction as the hyper-spectral sensor it is possible to know the distance to the reef at every moment of sensing. The Ping Echosounder has a range of 0.5 m to 30 m and a beam width of 30 degrees. This device connects to the on board compute via a serial connection and communicates over a ROS node that was developed using the Blue Robotics open-source interface (Blue Robotics Inc. 2020d).

3.2.6 Blue Roboitcs Enclosure

The main body of the uDrone needed to be a watertight enclosure to keep the electronic components dry. It also needed to have pass-throughs to communicate with the thrusters and external sensors. A blue robotics enclosure was used to accommodate all of this and reduce the need for precision machining within the lab. Standardizing on the blue robotics system allowed for the rapid prototyping of the uDrone. The 8 inch enclosure was used as it would accommodate all of the electronics inside that were required (Blue Robotics Inc. 2020e).

3.2.7 Battery

The Blue Robotics Lithium-ion Battery was chosen to minimize different suppliers. This 14.8 Volt, 18 Amp-hour battery comes in a cylindrical shape (Blue Robotics Inc. 2020a). Two of them can fit inside the enclosure with room to spare for other components. With two batteries, one battery will be dedicated to powering the thrusters while the other powers the compute and sensing equipment. The power consumption this latter group is listed in this section are shown in table 1. The total power for internal components is 0.82A at 14V. Adding a 50% safety factor for calculating run time, the equipment should pull 1.23A, which, with an 18Ah batter will allow for 14.6 hours of run time. A detailed discussion of the power consumption of the thrusters is discussed in section 4.6. From those calculations the thruster run time was determined to be 6 hours, which is more of a limiting factor than the internal equipment.

Table 1. Power usage of uDrone Internal components

Component	Power Usage at 14V
Tx2	0.54A
PixHawk	0.11A
ZED 2	0.14A
Ping	0.04A
Total	0.82A

Sources: NVIDIA Corporation (2020b), PX4 Dev Team (2020), Stereolabs Inc. (2020), Blue Robotics Inc. (2020b)

3.3 Software

In order to function as a testbed for innovation, the uDrone needed to be built using open source and easily accessible software. Additionally, it was desired to use the tools that the DREAMS lab had existing experience with.

3.3.1 ROS

All on board computers run Ubuntu Linux and the Robot Operating System (ROS). ROS is an open source library of tools that are commonly used for robotic systems. The architecture of ROS is similar to a network, consisting of nodes, processes that perform computation, and messages that communicate between notes (Open Robotics 2020; Open Source Robotics Foundation 2020a). All of the sensors, such as the ZED 2 and Ping Echosounder, have existing packages that can communicate with ROS. The sensor data is accessed via a node which generates messages that are used by other nodes. ROS has many additional libraries that are useful for autonomy and other uDrone needs.

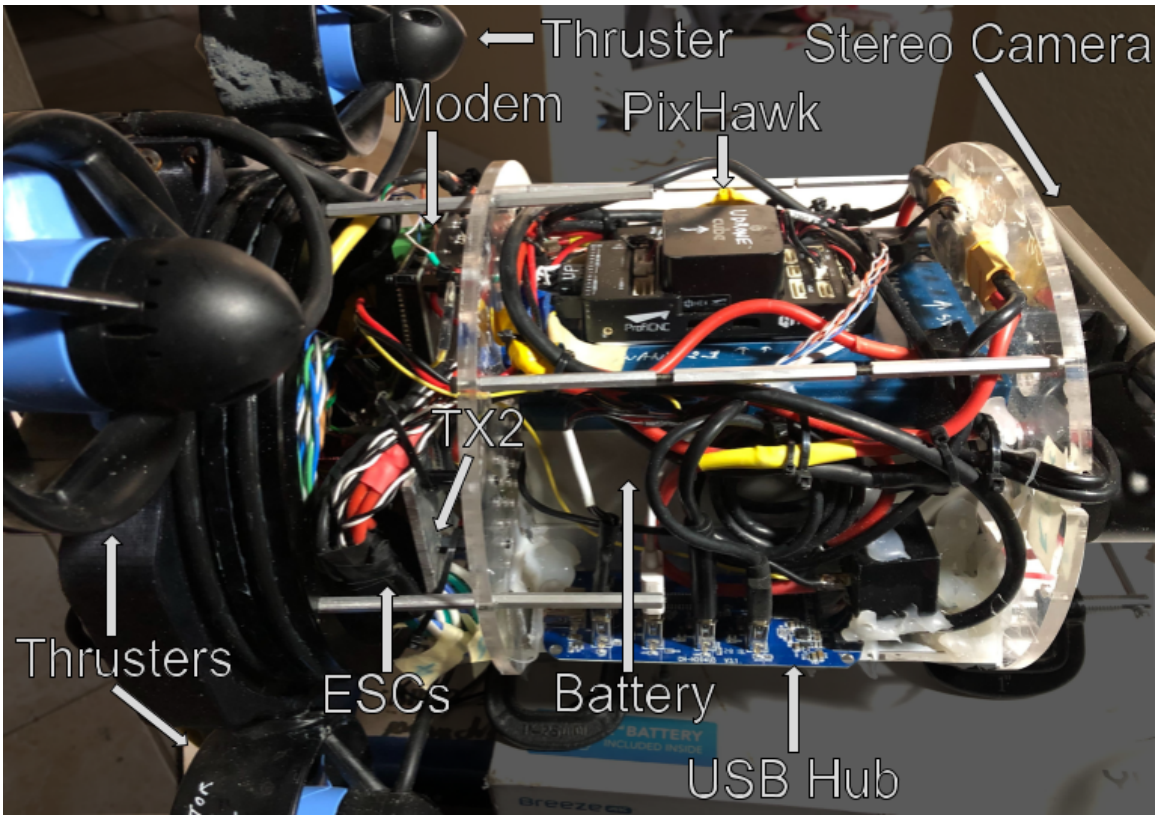

Figure 7. Annotated Picture of uDrone without Enclosure

Source: Photo courtesy of Jnaneshwar Das, used with permission.

3.3.2 PX4

The on board PixHawk flight controller runs PX4 on a nuttX operating system. PX4 is an open source autopilot for drones and other vehicles. This software is used and maintained by a series of academics and other drone developers. There is already a strong knowledge of PX4 from Dr. Das and other members of the DREAMS lab. At its most basic level, PX4 translates desired control inputs, i.e. go to a certain point or rotate clockwise, into specific motor actuation. It uses internal PID controllers to reach set points and mixer files to relate control inputs uniquely for different vehicles. Existing vehicle templates can be used to speed up development of

a controller for the uDrone. Specifically, the HippoCampus micro AUV has a similar thruster configuration to the uDrone and was used as a starting point for development (Dronecode Project, Inc. 2020).

3.3.3 MAVLink and MAVROS

PX4 communicates using a protocol called MAVLink. There is a ROS package called MAVROS which allows for the translation between ROS messages and MAVLink messages. This package allows for users to write programs with ROS and communicate them directly to the flight controller without needing to write interpreters. This architecture was implemented for the uDrone to allow for fast and seamless development (Ermakov 2018).

3.3.4 QGround Control

In order to interact with the vehicle while it is running, a ground control software is used. For this project QGround Control is used. This software communicates with the flight controller running PX4 using MAVLink and can be used in both simulation and field trials. It can be used to test manual control, set various types of mission vehicles, and observe the state of the vehicle and all its components. In autonomous mode the vehicle is controlled directly from the on board ROS computer, but QGround Control allows for a window into its operation and provides control overrides when necessary (Dronecode Project Inc. 2019).

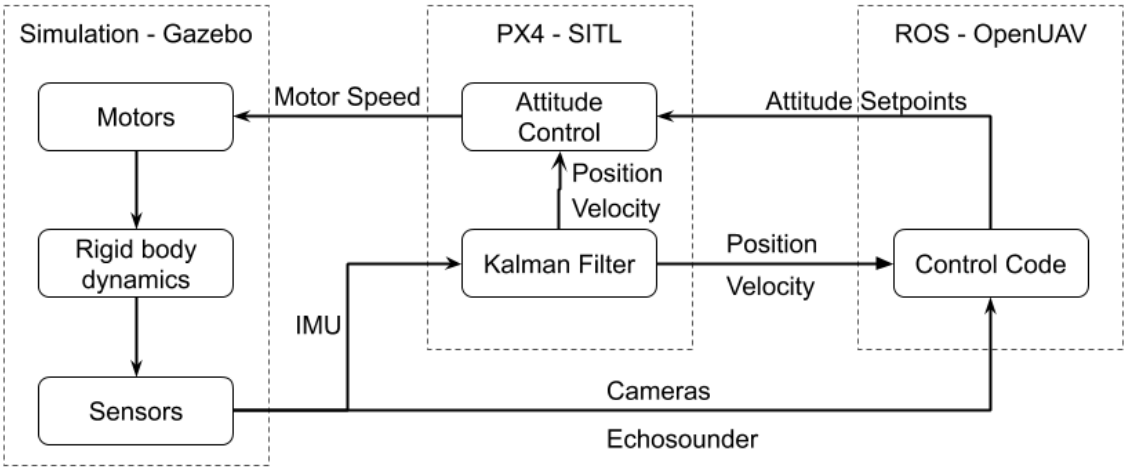

Figure 8. System diagram showing the interaction between the software components in simulation

3.4 Simulation

A core component of the whole system is the simulation environment. This allows for testing perception and control methodologies and forms the foundation of all the work that is eventually used on the uDrone. The goal of the simulation environment is to be as close to real life. A diagram showing how all the components of the simulation environment fit together can be seen in figure 8.

This particular setup is used because of its close resemblance to real world deployment. In simulation the control code, which runs with ROS on a desktop or OpenUAV container. In real world deployment this is all run on the internal CPU/GPU on-board the uDrone. The PX4 based flight controller is simulated using SITL while it is run on the PixHawk in deployment. And the simulated world in Gazebo is made to mimic the underwater world and vehicle dynamics that will be experienced in deployment.

3.4.1 Gazebo

A core component of the entire system is the simulation framework used for development and testing. The simulation of the uDrone is done using Gazebo. Gazebo is a simulator for robotics allowing for the testing of systems in real world like physical environments. It integrates directly with ROS, making development easier. Gazebo allows for the creation of custom worlds and robots, allowing for simulations of the uDrone on a coral reef. PX4 also has a library to work directly with Gazebo, which includes 3D and physics models for many of its vehicle templates. For most development, ROS is run and PX4 is emulated on the host system. Using this setup it is also possible to run hardware-in-the-loop simulations, allowing for tests of the performance of the TX2 and PixHawk Cube (Open Source Robotics Foundation 2020b).

3.4.2 UUV Simulator

To make the Gazebo environment as real as possible, a package called the Unmanned Underwater Vehicle Simulator (UUV Sim) was used to simulate the effects of a vehicle moving underwater. UUV Sim was specifically designed to aid in the simulation of underwater vehicles. This packages main contribution and reason it was used on this project is the implementation of the equations of motion detailed by Thor Forsson in the Handbook of Marine Hydrodynamics. It also has improved thruster models, several controllers purpose built for AUVs, aquatic world files, and several vehicle models (Manhães et al. 2016).

3.4.3 Reef Data

One of the main challenges to creating a simulation environment to test the uDrone was in finding a model of a reef to test with. To obtain a realistic reef model I took video footage while recreational scuba diving in Kona, Hawaii. This was a particularly good spot since the dives sites were only a few miles away from the planned initial site of the uDrone deployment. I built a special rig that allowed me to hold two GoPro cameras while I dove. The goal of the two cameras was to get a wider spread and possibly use them together as a stereo pair. Because the cameras were not synced, the stereo pair plan did not work.

The 3D model was generated from the footage using Agisoft Metashape structure from motion software. The videos, originally shot at 30 frames per second, needed to be down sampled to 6 frames per second. This allowed for some overlap between each image but not so much that processing took too long. Next, Agisoft found feature correspondence between the overlapping images. With this information, the software was able to generate a mesh and full 3D model with a photographic overlay, seen in figure 9.

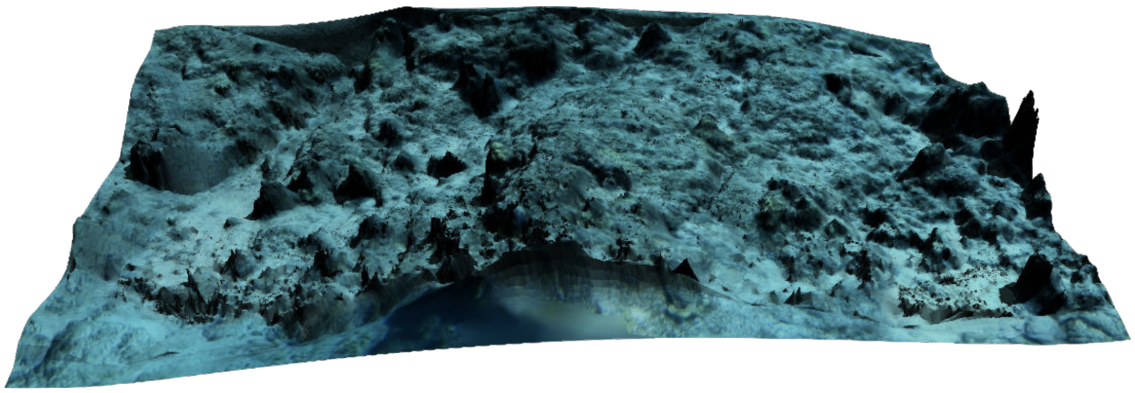

Figure 9. Three Dimensional Reef Model for Simulation

This process was not without significant challenges. Since there are many suspended particles in the water, structure from motion would often mistake these for features and incorporate them into the surface. For this reason, although many videos were shot, only one was able to be turned into a reliable reef model. The 3D model came from a section of the Kaloko Arches dive site near Kona, Hawaii. While the exact dimensions cannot be determined using this method, the model is estimated to be 16 by 8 meters.

3.5 Construction

The internal frame that supported the batteries and other electrical equipment and the external coupling for the thrusters were 3D printed based on designs. The whole vehicle was then assembled and prepared for testing. By leveraging 3D printing the team was able to test our and rapidly prototype design ideas. Also, since the structural components are easier to source as they can be 3D printed almost anywhere.

One major change that came about during this construction process involved the design of the interior structure. This structure went through several iterations in order to make physically working on the uDrone easier. Specifically, the compute cluster was repositioned so it would be closer to the camera in the front and easier to access without removing the batteries. Additionally, the entire structure was built to be removed as one solid piece, allowing for easy access when inserting into the enclosure.

Chapter 4

MATHEMATICAL MODEL OF VEHICLE DYNAMICS

This chapter will detail the mathematical model that was developed to describe the motion of the uDrone.

4.1 Reference Frames and State Space

Conventions for creating this model will follow those laid out by Thor Fossen in the 2011 Handbook of Marine Craft Hydrodynamics and Control. The fixed world frame is defined using the North-East-Down (NED) coordinate system, represented as $\{n\} = (x_n, y_n, z_n)$. in this system, x_n points North, y_n points East, and z_n points down.

The body-fixed frame $\{b\} = (x_b, y_b, z_b)$ is fixed its origin point, o_b, to the uDrone. For purposes of this model, o_b is set at the point on the plane of the motors at the back of the uDrone that is along its center-line. x_b runs along the center-line of the vehicle, pointing from the aft (back) of the uDrone to the fore (front). z_b runs from top to bottom and, following the right-hand rule, y_b runs towards the starboard (right). Further, following convention, roll is defined as rotation about x_b, pitch as rotation about y_b, and yaw as rotation about z_b, with counter-clockwise (CCW) being the positive direction.

This choice of body frame origin reduces the complexity of modeling the forces produced by the motors. The drawback is some added complexity caused by the moments of the center of buoyancy and center of gravity relative to the chosen origin,

but this is more straight forward than the complexity of calculating motor forces about a different point.

The position of the vehicle, or the body-fixed frame $\{b\}$, with respect to the world frame $\{n\}$ is expressed as $\boldsymbol{p} = [N, E, D]^T$. The attitude of the vehicle is expressed as $\boldsymbol{\Theta} = [\phi, \theta, \psi]^T$ using Euler angles. Together these create the 6-dimensional position/orientation vector $\boldsymbol{\eta} = [\boldsymbol{p}, \boldsymbol{\Theta}]^T$. The linear and angular velocities are expressed with respect to the vehicles fixed frame $\{b\}$. The linear velocity is $\boldsymbol{v} = [u, v, w]^T$ and angular velocity $\boldsymbol{\omega} = [p, q, r]^T$. when combine, these form the combine 6-dimensional velocity vector $\boldsymbol{\nu} = [\boldsymbol{v}, \boldsymbol{\omega}]^T$. All together, $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ represent the 12-dimensional state space describing the state of the uDrone (Fossen 2011a).

4.2 Equations of Motions

The general equation of motion for the uDrone can be derived as a function of the combine linear and angular velocity vector $\boldsymbol{\nu}$ (Fossen 2011b). This is shown in equation 4.1.

$$\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\boldsymbol{M}_{A}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{A}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{hydrodynamic forces}} + \underbrace{\boldsymbol{g}(\boldsymbol{\eta})}_{\text{hydrostatic forces}} = \boldsymbol{\tau} \tag{4.1}$$

With the terms defined as:

$$\boldsymbol{\eta} = \text{combine position and orientation}$$

$$\boldsymbol{\nu} = \text{combine linear and angular velocity}$$

$$\dot{\boldsymbol{\nu}} = \text{combine linear and angular acceleration}$$

$$\boldsymbol{M}_{RB} = \text{rigid-body system inertia matrix}$$

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \text{rigid-body Coriolis matrix}$$

$$\boldsymbol{M}_{A} = \text{added mass system inertia matrix}$$

$$\boldsymbol{C}_{A}(\boldsymbol{\nu}) = \text{added mass Coriolis matrix}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \text{damping matrix}$$

$$\boldsymbol{g}(\boldsymbol{\eta}) = \text{gravitational and buoyant forces}$$

$$\boldsymbol{\tau} = \text{control inputs}$$

4.3 Rigid-Body Forces

The exact values of the system inertia ($\boldsymbol{M}$), Coriolis ($\boldsymbol{C}$), and Drag ($\boldsymbol{D}$) matrices can be determined either through intensive hydrodynamic modeling or experimentation. Due to the complexity of hydrodynamic modeling, experimentation is typically used to determine these values in underwater vehicles similar to the uDrone (Duecker et al. 2018). It is possible, however, to estimate the rigid body matrix values using information about the vehicle generated from the 3D model. This section details these estimations and explains how they were calculated.

The rigid body system inertia matrix about the center of origin can be calculated

using equation 4.2 which is found in Fossen 2011c.

$$\boldsymbol{M}_{RB}^{CO} = \begin{bmatrix} m\boldsymbol{I}_{3\times 3} & -m\boldsymbol{S}\left(\boldsymbol{r}_g^b\right) \\ m\boldsymbol{S}\left(\boldsymbol{r}_g^b\right) & \boldsymbol{I}_g - m\boldsymbol{S}^2\left(\boldsymbol{r}_g^b\right) \end{bmatrix} \tag{4.2}$$

In this equation, m is the mass, $\boldsymbol{I}_{3\times 3}$ is the three-by-three identity matrix, $\boldsymbol{I}_g$ is the inertia matrix and $\boldsymbol{r}_g^b$ is the vector from the body fixed-frame origin to the center of gravity. $\boldsymbol{S}$ is the cross product operator matrix. Before the final vehicle was constructed, m, $\boldsymbol{I}_g$, and $\boldsymbol{r}_g^b$ were taken from the 3D model. Those estimates are shown in equation 4.3.

$$\boldsymbol{M}_{RB}^{CO} = \begin{bmatrix} 9.9000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9.9000 & 0 & 0 & 0 & 1.4850 \\ 0 & 0 & 9.9000 & 0 & -1.4850 & 0 \\ 0 & 0 & 0 & 0.3420 & 0.0010 & -0.0001 \\ 0 & 0 & -1.4850 & 0.0010 & 0.5577 & -0.0015 \\ 0 & 1.4850 & 0 & -0.0001 & -0.0015 & 0.3167 \end{bmatrix} \tag{4.3}$$

With a completed vehicle these numbers can be verified and varied if needed. For example, with no ballast, the uDrone is positively buoyant. Therefore, to maintain its depth without using power to stay underwater ballast will need to be added. By placing the ballast closer to the center of the vehicle, the moments of inertia will decrease. Conversely, by placing it further from the center the moments can be increased. Additionally, the ballast can be used to change the center of gravity relative to the center of buoyancy. This will create a moment in the vehicle that can help it stay oriented or level in specific ways. This relationship is discussed further in the section 4.4.

The Coriolis matrix is a function of the angular velocity of the vehicle and can be derived as shown in equation 4.4 from Fossen 2011c.

$$\boldsymbol{C}_{RB}^{CO}(\boldsymbol{\nu}) = \begin{bmatrix} m\boldsymbol{S}\left(\boldsymbol{\omega}^b\right) & -m\boldsymbol{S}\left(\boldsymbol{\omega}^b\right)\boldsymbol{S}\left(\boldsymbol{r}_g^b\right) \\ m\boldsymbol{S}\left(\boldsymbol{r}_g^b\right)\boldsymbol{S}\left(\boldsymbol{\omega}^b\right) & -\boldsymbol{S}\left(\left(\boldsymbol{I}_g - m\boldsymbol{S}^2\left(\boldsymbol{r}_g^b\right)\right)\boldsymbol{\omega}^b\right) \end{bmatrix} \tag{4.4}$$

Where $\boldsymbol{\omega}^b$ is the body-fixed frame angular velocity vector. Since this matrix is a function of the vehicle state, it will change with motion. The actual calculations of this matrix are handled by a MatLab program written as a companion to Thor Forson's book (Perez and Fossen 2009). The values based on the system inertia matrix in equation 4.3 along with a velocity state vector $\boldsymbol{\nu} = [1, 0, 0, 0, 0, 0]^T$, which represents only forward motion, is shown in equation 4.5.

$$\boldsymbol{C}_{RB}^{CO}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9.9000 \\ 0 & 0 & 0 & 0 & -9.9000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.9000 & 0 & 0 & 0 \\ 0 & -9.9000 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.5}$$

4.4 Hydrostatic Forces

The hydrostatic term $\boldsymbol{g}(\boldsymbol{\eta})$ accounts for the forces on the uDrone caused by gravity and buoyancy. The force of gravity, or weight (W) is equal to the mass of the uDrone times the acceleration due to gravity. The buoyant force (B) is the weight of the water displaced by the uDrone. This is calculated by multiplying the volume of the uDrone, the density of water, and the acceleration due to gravity. If the location of the center of gravity and center of buoyancy (or center of volume) of the uDrone are

not in the same spot then there will be a moment on the whole vehicle. Similarly, if the two forces are not equal then there will also be a net linear force on it. This value is a function of the orientation of the vehicle and can be seen in equation 4.6 (Fossen 2011c).

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} & (W-B)\sin(\theta) & & \\ - & (W-B)\cos(\theta)\sin(\phi) & & \\ - & (W-B)\cos(\theta)\cos(\phi) & & \\ - & (y_g W - y_b B)\cos(\theta)\cos(\phi) & + & (z_g W - z_b B)\cos(\theta)\sin(\phi) \\ & (z_g W - z_b B)\sin(\theta) & + & (x_g W - x_b B)\cos(\theta)\cos(\phi) \\ - & (x_g W - x_b B)\cos(\theta)\sin(\phi) & - & (y_g W - y_b B)\sin(\theta) \end{bmatrix} \tag{4.6}$$

Ballasting can be used to adjust the weight and center of mass on the uDrone. To simplify this equation the ballast is set so that the weight and buoyancy are equal and the center of mass and buoyancy are the same. This means the $\boldsymbol{g}$ vector will be zeros for all $\boldsymbol{\eta}$. These forces can be seen interacting on the uDrone in figure 10. Maintaining overlapped centers of buoyancy and gravity also makes the vehicle more nimble as there are less moments to overcome when rotating. This also has the benefit of making the uDrone more maneuverable. Equal weight and buoyancy reduces energy used as the uDrone does not need to overcome a force making it sink or float as it cruises through the water column.

There might be reasons to adjust the ballasting of the uDrone to impart other properties on it. For example, if the vehicle is slightly positively buoyant than it will float in the event of a power failure, allowing for it to be recovered more easily. Also, if the center of gravity is placed below the center of buoyancy, but still in line with it

in the z body direction, then the vehicle will be self righting in roll. This could make control easier as roll, which should be at zero most of the time, can be ignored.

4.5 Hydrodynamic Forces

The added mass system inertia matrix and added mass Coriolis matrix can only be determined through experimentation or complex fluid dynamic simulation and are not calculated in the scope of this thesis. The damping matrix, however, can be estimated using drag calculations.

The vehicle is modeled as a flat-faced cylinder moving through water in a direction along its axis. This estimation neglects two factors: 1) drag from the motors, and 2) rotational damping. Both of these forces are small in comparison to the drag of the flat-faced cylinder, so for the purposes of an approximate model, they can be ignored.

The force of drag in a single direction can be calculated using the drag equation:

$$F_D = C_D A \frac{\rho V^2}{2} \tag{4.7}$$

F_D = drag Force

C_D = coefficient of drag

A = cross-sectional area normal to velocity

ρ = fluid density

V = velocity

Based on the ratio of the diameter to the length of the main body of the uDrone, the Coefficient of drag in the direction of motion is estimated at 0.8. The density of

salt water is 1024kg/m^3 and the surface area is calculated from the diameter of the eight inch cylinder, which is 0.037m^2. Putting this all together, damping force can be calculated as a function of velocity. This is:

$$\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} -15.0u^2 \\ \sim 0 \\ \sim 0 \\ D_{roll} \\ D_{pitch} \\ D_{yaw} \end{bmatrix} \tag{4.8}$$

u is the velocity in the x body direction. The last three values which refer to the rotational damping. For the purposes of calculations in the next section these values were assumed to be zero. The second and third terms, however, will always be zero, or nearly zero, as the vehicle cannot independently move in the Y or Z direction.

4.6 Control Inputs

The control inputs are the forces put on the uDrone from the thrusters. The four motor of the uDrone are arranged on a plane, all facing the same direction. The center point between all the thrusters along that plane is the Center of Origin C_O for the vehicle. Thrusters one and three spin in a counterclockwise direction while thrusters two and four spin clockwise for forward thrust. This ensures that the vehicle has no roll moment when moving forward. Figure 10 shows the uDrone with the C_O and rotation direction of the thrusters.

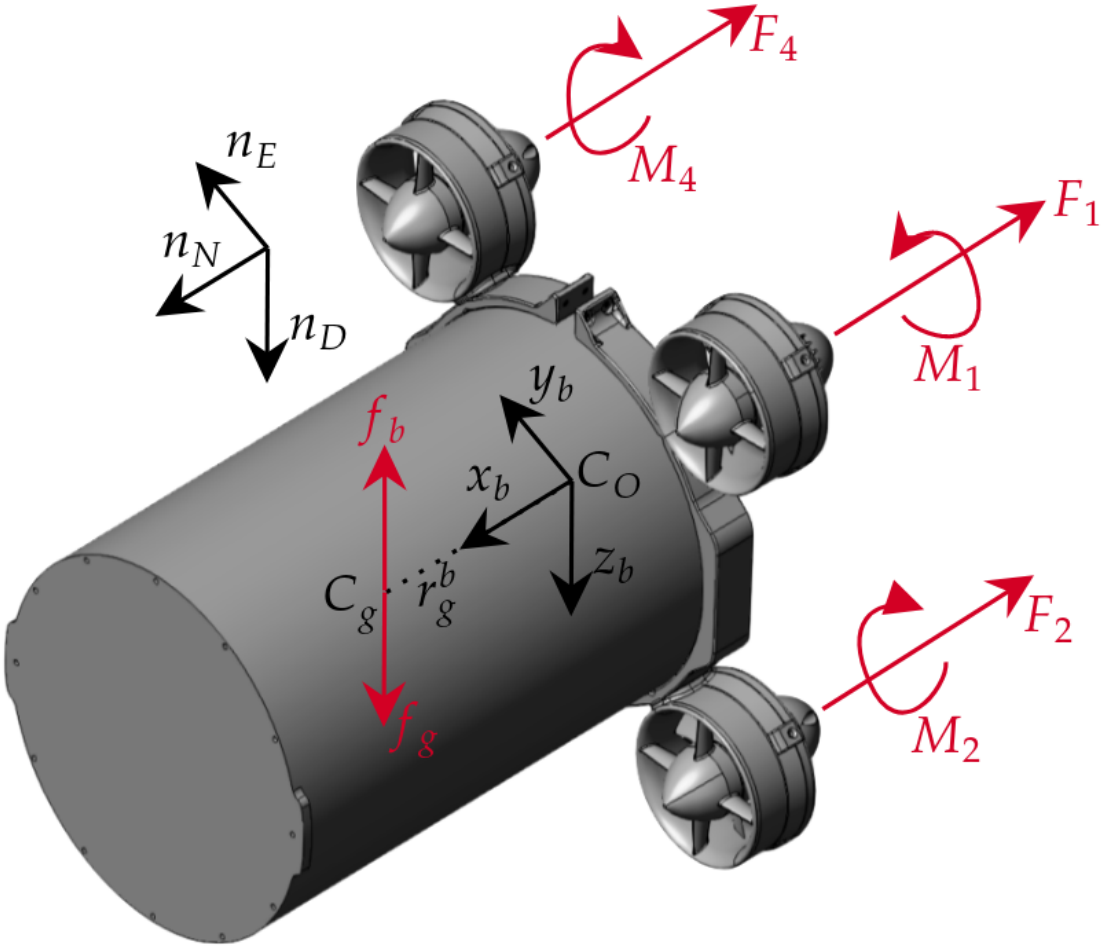

Figure 10. uDrone Coordinate Frame and Motor Directions

Using this convention it is possible to determine the actuation matrix, tau, as a function of thruster inputs.

$$\boldsymbol{\tau} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ 0 \\ 0 \\ M_1 - M_2 + M_3 - M_4 \\ (F_2 + F_3 - F_1 - F_4)\frac{L}{\sqrt{2}} \\ (F_1 + F_2 - F_3 - F_4)\frac{L}{\sqrt{2}} \end{bmatrix} \tag{4.9}$$

In equation 4.9 F_x indicates the linear force produced by the x^{th} motor and M_x indicates the angular momentum produced by the x^{th}. L is 0.115m, the length from C_O to the center of the motor, which is the same for each motor. Unfortunately, there is no data from Blue Robotics about the moment of the thrusters, so the relationship between control input to the roll moment will need to be determined experimentally. Data is available, however, correlating motor power and thrust force.

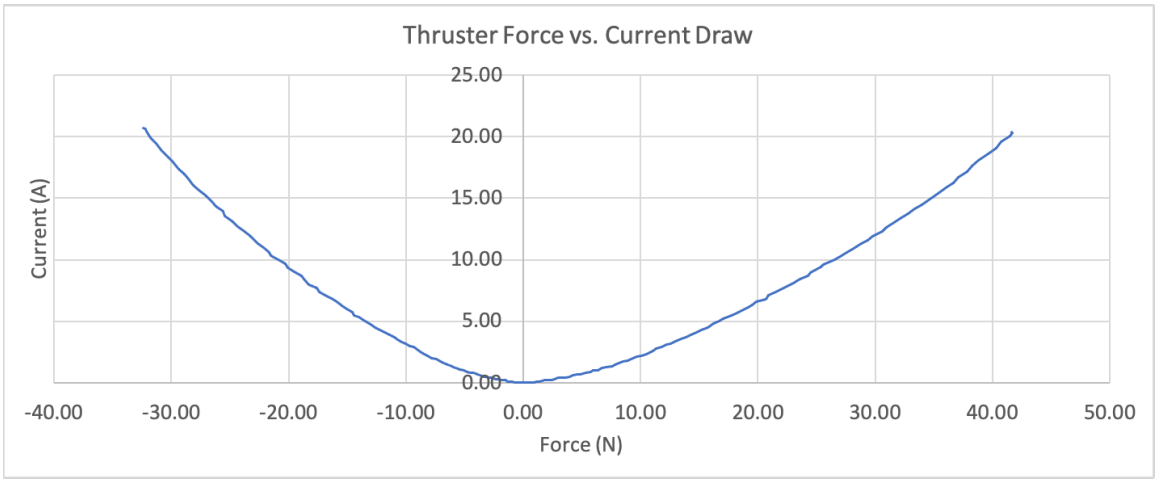

Figure 11. Force Output vs. Current Draw for T200 Thruster at 14 Volts

Source: Blue Robotics Inc. (2020b)

Using the figure 11, along with data about the uDrone battery, it is possible to calculate the maximum velocity, running time at this velocity, and running time at the

operational velocity of 1m/s. While the uDrone can carry two 14V, 18Ah batteries, the intent is to use one for thrusters and the other to control all electronics. Therefore, all longevity calculations are done assuming 18Ah are available for thruster actuation.

In order to calculate maximum velocity, the drag equation is set equal to the maximum thruster force. The T200 thrusters produce a maximum thrust of 41.68N each. This is 166.7N of total thrust. Setting this equal to $15u^2$ and solving yields a maximum speed of 3.33m/s. Maintaining this velocity takes 20.29A, for a total of 81.16A. The uDrone could only maintain this speed for approximately 13.3 minute with an 18Ah battery.

Typically, however, the uDrone will be cruising at a speed of 1m/s. At this speed, the drag force is approximately 15N. To maintain this thrust each motor must produce about 3.75N of force. To maintain this force each motor will use approximately 0.5A, for a total of 2A. In order to stay conservative, a 50% safety factor will be added to this value to account for uncalculated drag forces and actuation that affect orientations, not just forward motion. With this addition, the vehicle will use 3A for actuation during normal operation. With a single 18Ah battery used for thrusters, this will yield a dive time of approximately 6 hours.

Chapter 5

OPTIMAL CONTROL IN TWO-DIMENSIONAL SPACE

In this project [1], the principles of optimal control were applied to the uDrone. In order to simplify the process, the vehicle and world were modeled as a two-dimensional system with one control input. Dynamic programming was used with a cost function that satisfies the mapping goals of the vehicle while minimizing control input. This section lays out the methods, implementation, and results of this particular application.

5.1 Methods

5.1.1 Simplification & Assumptions

In order to reduce the complexity of this problem, the vehicle and world are modeled in two dimensions. Since the goal of the vehicle is to move in a straight line along the reef in the X-Z plane (where Z is down and X is forward), this is a reasonable assumptions. To further simplify the problem, the forward thrust is assumed to be at a constant value and the only control input is pitch rate. More assumptions were made to simplify the dynamic model, including constant forward drag, no rotational drag, no Coriolis forces, neutral buoyancy, corresponding centers of buoyancy and mass, and discrete state transitions.

[1] This particular application was completed as part of a group project undertaken by the author along with Evan Lamb and Jason Newton. Parts of this section are re-purposed from the project report.

5.1.2 Model

The first step of this project involved generating a model for the uDrone, defining a cost function, and determining constraints.

5.1.2.1 State Space Equations

In order to generate state space equations, equations of motion for an underwater vehicle were used (Fossen 2011a).

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad u = \begin{bmatrix} \dot{\theta} \end{bmatrix} \tag{5.1}$$

The set of equations 5.1 describe the state space, where X is the state vector and u is the control vector. x is the horizontal position, y is the vertical position, θ is the pitch, and $\dot{\theta}$ is the pitch rate of the vehicle.

$$\begin{aligned} \dot{x} &= \nu \cos(\theta) \\ \dot{y} &= \nu \sin(\theta) \\ \dot{\theta} &= u \end{aligned} \tag{5.2}$$

The set of equations 5.2 are the state space equation where $\dot{x}, \dot{y}$, and $\dot{\theta}$ are the derivatives of x, y, and θ respectively and v is the constant forward velocity.

$$\begin{aligned} x_{k+1} &= v \cos\theta_{k+1} * \Delta t + x_k \\ y_{k+1} &= v \sin\theta_{k+1} * \Delta t + y_k \\ \theta_{k+1} &= \dot{\theta}_k * \Delta t + \theta_k \end{aligned} \tag{5.3}$$

The set of equations 5.3 are the discretized state transition equations to go from time-step k to time-step $k+1$. Δt is the length of the time-step. This equation is derived from the state space equations 5.2.

5.1.2.2 Cost Function

$$
\begin{aligned}
J &= \int_{t_0}^{t_f} \left\| f\left(x\left(\tau\right)\right) - y\left(\tau\right) \right\|^2 + \left\| g\left(u\left(\tau\right)\right) \right\|_R^2 d\tau \\
f\left(x\left(t\right)\right) &\triangleq \text{desired } y \text{ at given } x\left(t\right) \\
g\left(u\left(t\right)\right) &\triangleq \text{motor control cost}
\end{aligned}
\tag{5.4}
$$

The cost function shown in equation 5.4 is a combination of two costs. First, there is the tracking cost which is defined as a vertical deviation from a desired path at any given x value. Its weight is fixed at $Q = 1$. Second, there is the control cost which is defined as the required power output by the uDrone's motors to attain a specific pitch rate. Its weight is varied through three R values: 0.01, 0.05, and 0.1. The function $f\left(\cdot\right)$ is defined by the desired trajectory. The cost of the motor control is based on the parameters of the motors used in the uDrone (Blue Robotics Inc. 2020b). The continuous cost function shown above was discretized according to the temporal time step prior to simulation in MATLAB.

5.1.2.3 Constraints

$$\begin{aligned} 0 \leq & x \leq 10 \\ 0 \leq & y \leq 4 \\ \frac{-11\pi}{24} \leq & \theta \leq \frac{11\pi}{24} \\ -70 \leq & \ddot{\theta} \leq 70 \end{aligned} \tag{5.5}$$

The set of equations 5.5 show the constraints for the system. The constraints around x and y are based on the problem definition. The constraint on θ is in place to ensure forward movement of the uDrone and prevent circular references. The constraint on $\ddot{\theta}$ is based on the motor parameters and model of the uDrone (Blue Robotics Inc. 2020b). This value is multiplied by Δt to determine the constraint on u. The constraints were discretized prior to simulation in MATLAB.

5.1.3 Dynamic Programming

To generate an optimal control policy, dynamic programming is used in MATLAB, using the methodology from (Kirk 2012). First, the program initializes all variables and the U*, J* matrices. Using the defined cost function, the cost (J) is computed for each of the final states (i.e. states having ($x = 10$)). The code then takes a step backwards in x according to its spatial discretization. The program then computes the "tracking cost" for each state. A higher cost is implemented for going too far beneath the desired path–this is abstracted as hitting the reef. Next, the code checks each of the 15 permissible control inputs at the states and determines 1) what the next state will be based on the state equations, 2) the "control cost" for that control

input based on the motor parameters, and 3) the total cost (J) for that choice by adding control cost and tracking cost. Comparing the costs for each of these states, the program chooses the state-action pair with the minimum cost and denotes it as the optimal control. This action is populated into the optimal control (U*) table. The cost associated with this action is populated in the optimal cost (J*) table. This process repeats until it has worked backwards to the initial state ($x = 0$). For each state along x, the values for y, θ, and u are discretized as well.

In a traditional dynamic programming example, the program will use interpolation to determine the J* and U* values at a given state when the state values fall in between the discretized values. Given its high computational cost, interpolation was replaced with a nearest neighbor approach. In this approach, the state being evaluated is simply assigned the J* value of the state which is nearest. Given that we have a 3D grid, this can be likened to landing inside a box with eight corners. These corners represent points at which no interpolation is necessary. Instead of determining the state's exact J*, the program just accepts the J* of the nearest corner and assigns it to the state.

5.2 Implementation & Simulation

This section details the specifics of the trials and simulations. Initial trials were conducted at courser discretization resolutions. The trails were made iterative finer until the step size described below was reached.

5.2.1 Path

The path set for the uDrone to follow is based on a piece-wise function shown in equation 5.6 and show in figure 12.

$$y = \begin{cases} 1 & 0 \leq x < 1 \\ 2 - \cos\left(\frac{\pi(x-1)}{4}\right) & 1 \leq x < 9 \\ 1 & 9 \leq x \leq 10 \end{cases} \tag{5.6}$$

Additionally, in order to model hitting the reef, any location more than one meter below the curve was given a very high cost.

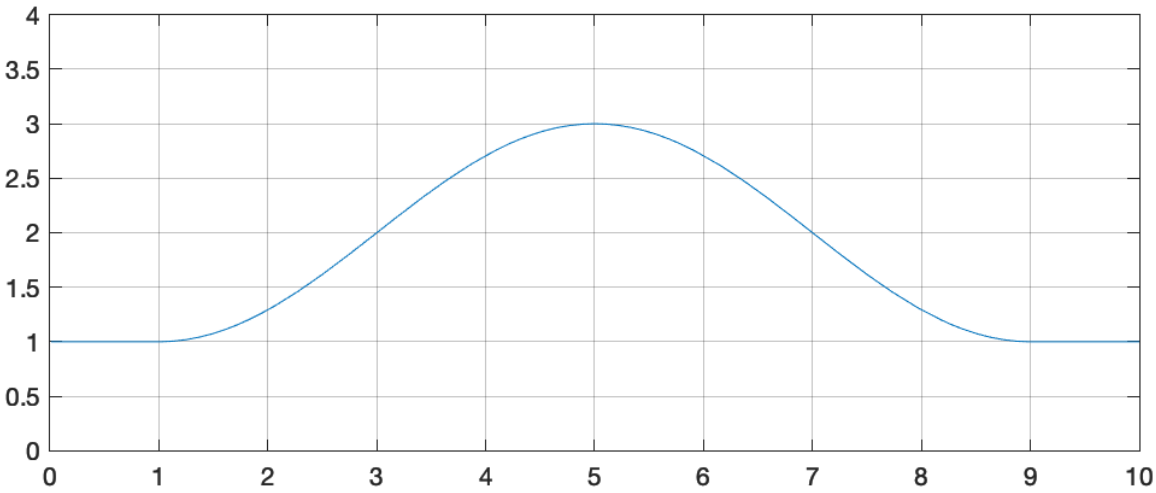

Figure 12. Desired uDrone path in defined state space

5.2.2 Cost Function Parameters

The cost function was determined directly from the motor parameters for the uDrone (Blue Robotics Inc. 2020b). Specifically, the force needed to pitch the vehicle

was used to determine the angular velocity. The power necessary to create this force value is the control cost. Two methods were used for determining the cost, which are called "True" and "Zeroed" in this paper. The true method uses the exact motor values, including the constant force to move the vehicle forward. Therefore, this method takes into account drag. The zeroed approach normalizes the control cost to zero when there is no control input. This method will minimize control action, but not necessarily penalize the vehicle for following a path more closely if it also causes the vehicles to have a lower velocity in the X direction. The control cost curves can be seen in figure 13.

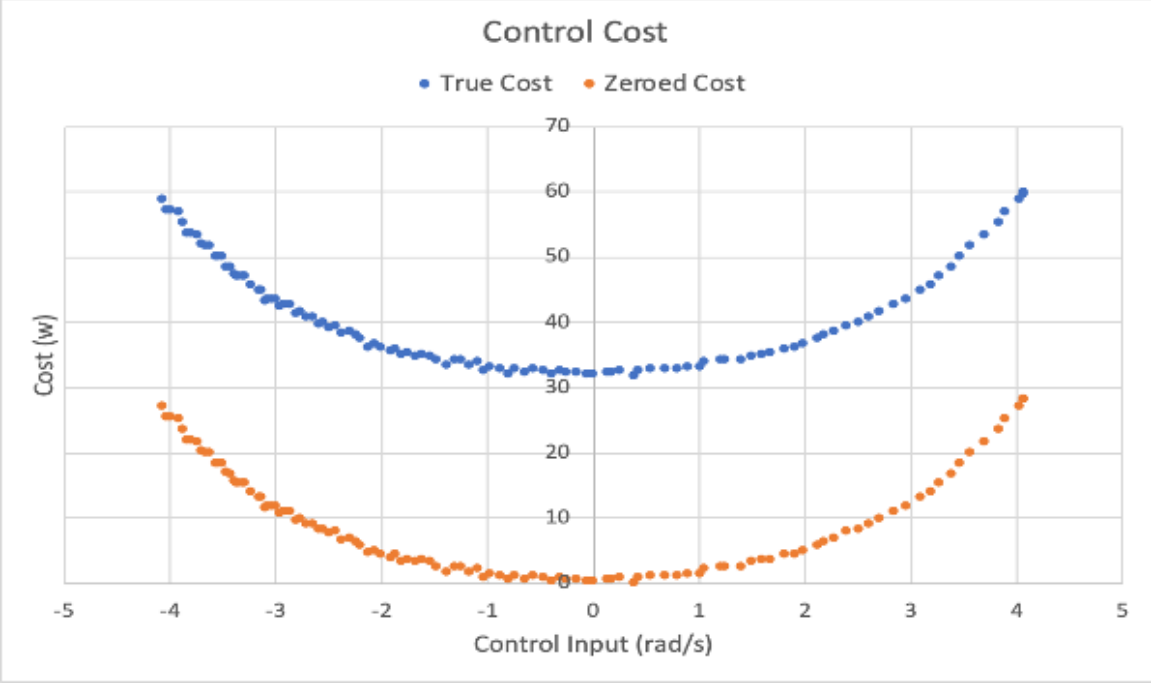

Figure 13. Control cost for true (blue) and zeroed (orange) methods

The ratio between control cost and tracking cost weight (R) was tested at three values, 0.01, 0.05, and 0.1. Since the values for control cost are in the double digit order of magnitudes and the value for tracking cost are in the single digit order of

magnitudes, an R value of 0.1 will treat the two costs as roughly equal. An R value of 0.01 will make the tracking cost 10 times more important than control cost.

5.2.3 Discretization

When discretizing, first a coarse trial was run at 10 Hz. After this was successful, more fine trials were run, up to 20 Hz. This timestep drove the step size for other variables via the constraint on moving forward in equation 5.7.

$$\Delta x \leq v \cos\theta_{max} * \Delta t \tag{5.7}$$

Additionally, from the timestep the limits on control input are established, as seen in equation 5.8.

$$\begin{aligned} \dot{\theta} &= \Delta t * \ddot{\theta} \\ -3.5 \leq \dot{\theta} &\leq 3.5 \end{aligned} \tag{5.8}$$

This, along with a desire to maintain fine discretization for grater accuracy, led to the final decision for discrete steps, as shown in table

Value	Step Size	Step Count
x	0.005 m	2001
y	0.01 m	401
θ	$\frac{\pi}{24}$ rad	23
U	0.05 $\frac{rad}{s}$	15

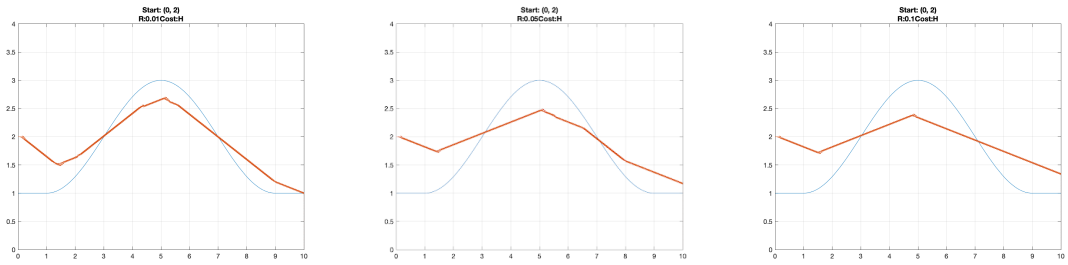

Figure 14. Optimal path using true cost method for various R values (Left) R = 0.01 (Center) R = 0.05 (Right) R = 0.1

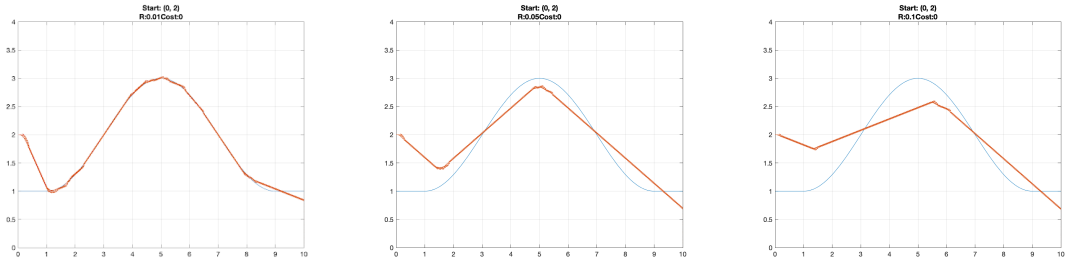

Figure 15. Optimal path using zero cost method for various R values (Left) R = 0.01 (Center) R = 0.05 (Right) R = 0.1

5.3 Results

Six total trials were run, each with a different cost function method and R value. In each trial 3D optimal cost and optimal control tables are generated. Based on the size of discretization, each table had over 18 million entries and the full trial took approximately 5 hours to complete. Using this data, two graphs are generated: the optimal path and the optimal control law.

5.3.1 Optimal Path

The first graph demonstrates the optimal path of a vehicle at a given starting point. This is useful for visualizing the vehicle motion and how much it follows the path, as

shown in figures 14 and 15. There are several conclusions that can be drawn from these figures: 1) As the R value increases, the vehicle deviates from the path more. This can be seen by comparing the graphs in figures 14 and 15, where the R value increases from left to right. This makes sense as a lower R value increases the weight of the path following relative to the control cost. 2) The true cost method causes the vehicle to deviate from the path more than the zeroed method. The true cost method also maintains the vehicle at a more horizontal orientation, thus increasing the component velocity in the X direction. This is shown by comparing figure 14 to figure 15. The result of the cost method comparison is intuitive since the cost to move forward will cause the optimal path to take the vehicle through the simulation in the least number of steps.

5.3.2 Control Law

The second graph is a visualization of the control law computed by the run. A control law is a table containing the optimal control choice at each state. This table would be loaded on to a vehicle and used to determine control, as running any dynamic programming problem in real time would be to computationally intensive and slow. Since it is not practical to display the full three-dimensional graph in this format, a slice is presented here. Figure 16 shows a single slice of the control law. Specifically, this figure shows the control law for a given x and y when θ is equal to zero. In this figure the red color represents areas of positive (counterclockwise) optimal control, the blue color areas of negative (clockwise) optimal control, and the white areas of zero optimal control.

Clearly visible in the figure is the dark red lower parabola which is the very high

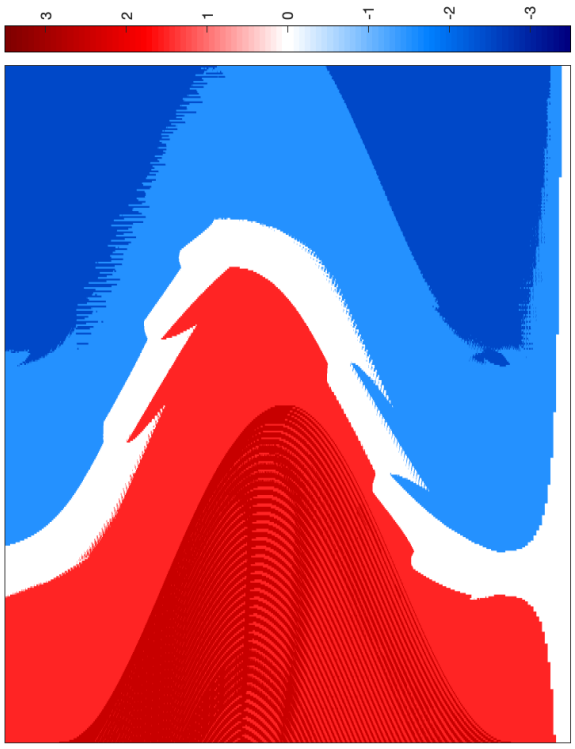

Figure 16. Optimal Control Law

cost area of hitting the reef that the control law tries to avoid. The lighter red areas are where the vehicle is behind or below the path and needs to pitch up to get on track. The light and dark blue areas are where the vehicle is above or in front of the path and needs to pitch down. The white band is when the vehicle is directly next to the path and should move straight in the body fixed axes.

5.4 Conclusions

This project showed that dynamic programming is a valid method for creating an optimal control law for an underwater vehicle. Since this vehicle's mission is to travel in a straight plane, the simplifications from three to two dimensions would still allow for a usable control law. Additionally, the project showed the effects of the cost

function on the optimal control policy. By varying the control cost calculation method (true versus zeroed) and changing the control weights (path following vs. control cost) the optimal trajectory changes. These trade offs need to be weighted based on the specific needs of the vehicle's mission. For example, if mapping fidelity and accuracy is paramount, a low R value and zeroed cost would be best. But, if coverage is more important, a higher R value and the true cost method are best.

The largest barrier to practical implementation is the compute time and static environment needed for dynamic programming. The compute time issue can be overcome by pre-computing a control law, such as mentioned above. Dynamic programming is useful when the path is known in advance. This occurs when there are existing maps of survey regions or the vehicle is returning to a site it previously visited. However, given the variable topography of reefs, it would be hard to pre-compute a control policy for exploration. In these cases, other optimal methods could be used such as model predictive control or optimal path generation with PID control.

Chapter 6

PID CONTROL IN SIMULATION ENVIRONMENT

The primary goal of this thesis project is to create a vehicle that satisfies the needs of the GDCS coral monitoring projects. Specifically, the uDrone needs to be able to follow the contour of the reef at a set distance. In order to show that this is possible, a simulation environment was developed that shows this capability. The details of how this simulation environment was created and the results are detailed in this section.[2]

6.1 Methods

This trial features a simulated uDrone following the terrain contours of a known reef profile. In order to do this, a simulation environment was built, a controller was developed and a contour profile was measured.

6.1.1 Simulation Environment Setup

A world was created in Gazebo that contained a reef model. In this case, the reef model obtained by me using 3D motion capture while scuba diving was used. The model was scaled to a near realistic size. The world not only has the reef topography but also as a photographic overlay of the reef. UUV sim was also used in the simulated

[2]The software for this application was written jointly by the author and A.L.G. Prasad, another member of the DREAMS Lab who is working on uDrone autonomy. Help and code snippets were also provided by Harish Anand, another DREAMS Lab member.

world. This allowed for mimicking some of the characteristics of water, such as buoyant forces and hydrodynamic damping.

The uDrone model was added to the word file. This particular uDrone model was outfit with a downward-facing, broad-beam, sonar. The particular sonar used was the hector gazebo sonar plugin (Kohlbrecher and Meyer 2016). The specifications of the simulated sonar are set up to match the Blue Robotics Ping Echosounder used in the actual uDrone. This meant a beam-width of 30 degrees was used.

An emulated instance of the PX4 flight stack was used for the direct motor control of the uDrone. This was done using the PX4 Software in the Loop (SITL) for simulation where the PX4 software is run on a software emulated flight controller. The air-frame configuration of the HippoCampus µAUV was used as it very closely mimics the motor configuration of the uDrone. The MAVROS plugin for ROS was used to communicate with the PX4 flight stack over MAVLink.

6.1.2 ROS Setup

The program was broken into nodes following the ROS development model. The diagram of the node layout can be seen in figure 17.

A special ROS node was created, called the uDrone Topic Modulator, in order to launch all of these components together and communication with the uDrone through MAVROS. This startup node's launch file loaded the reef and uDrone into the Gazebo simulation and initiated PX4 SITL. A C++ program written as part of this node connected to the vehicle set the flight mode to "offboard," and armed the vehicle. This process was necessary for the controller to begin sending messages over MAVLink to

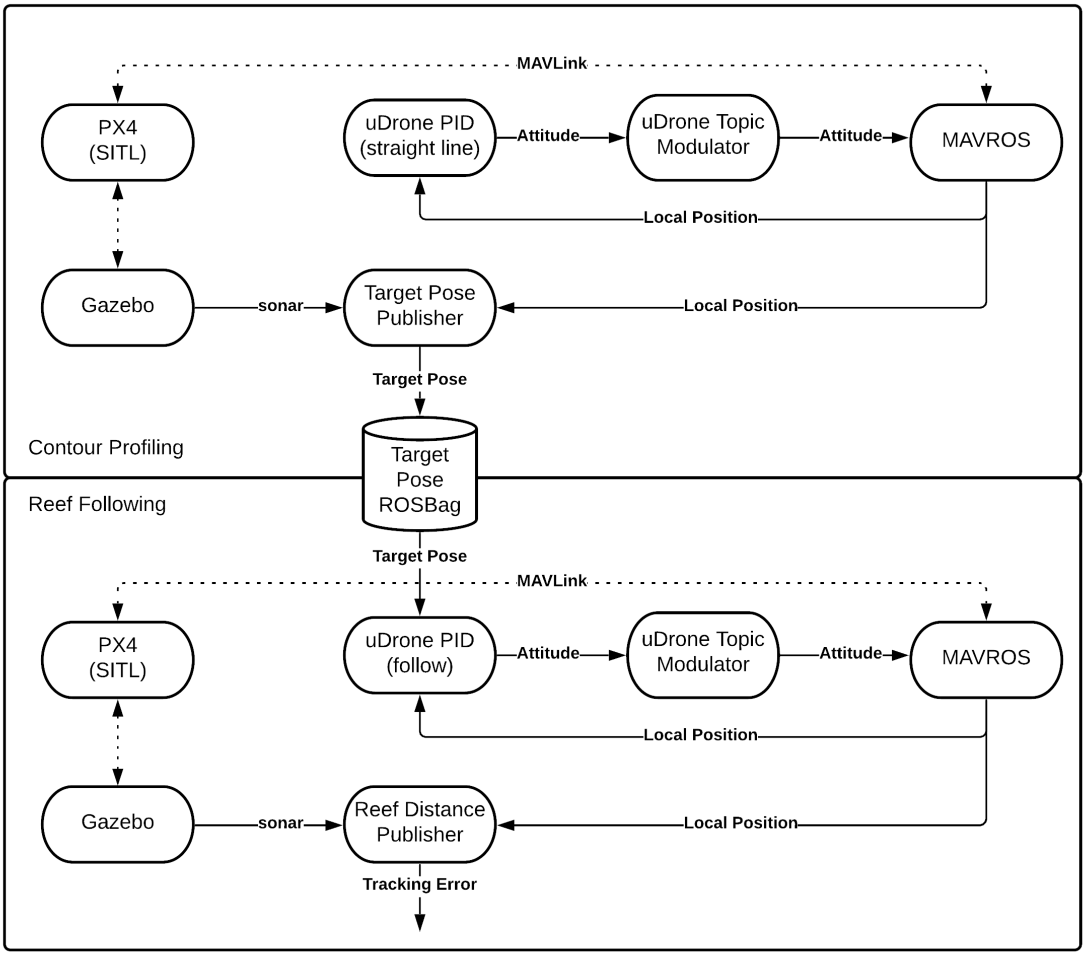

Figure 17. ROS Nodes for Contour Profiling and Reef Following in Simulation

the uDrone for control. It also listened for commands from other nodes and sent them along to the uDrone via MAVROS.

The PID controller was a separate node that took in the uDrone's local position via MAVROS along with a target position and pose. This node had two modes. In follow mode it listened for a target pose and then followed it. In contour profiling mode it followed a path set in the C++ code. The path used for this application was a straight line. The PID controller itself will be discussed in section 6.1.3.

A final node was developed that served two purposes, depending on the need. This is called the Target Pose Publisher or Reef Distance Publisher. This node took in the sonar data output from the simulated echosounder in Gazebo along with the local position from MAVROS. In the contour profiling cases it output the target pose, as described in section 6.2.In the reef following case it output the tracking error, or distance to the reef, used to calculate the results in section 6.3.

The element that connects the contour profiling with the reef following is a ROSBag. The ROSBag packages is a built in feature of ROS that allows for the recording of data while ROS is running. In this case it was used to record the target pose from the contour profiling run. Then it was played back during the reef following run in order to send the uDrone the desired pose and position.

6.1.3 Controller Development

For this application, a proportional–integral–derivative (PID) controller was developed to control the movement of the uDrone. While the PID is developed as a single controller, it controls the position in all three directions independently. The proportional term was found by comparing the desired position to the local position, as reported by the internal position estimator in PX4. This local position is relative to the vehicles starting point and determined using an extended Kalman filter that fuses data from the PixHawks internal IMU, Gyroscope, and compass. The proportional gain is set to 0.5. The integral term is found by summing past proportional errors. In this case, only the past two errors are summed and the gain is set to 0.005. The derivative term is estimated by calculating the change in error from the last time-step

to this time-step and dividing by the step size. The gain for the derivative term is 0.05.

The PID controller outputs a vector with a direction and magnitude, essentially pointing towards where the vehicle should go and how fast it should go there. In order to convert this into a usable control the method from Mohta et al. 2017 is used.

6.2 Contour profiling

A trajectory for reef following needed to be developed in order to accomplish this task. To do this, the simulation environment was started in the same way as for the final trial. However, instead of following a reef relative path, the uDrone was given the desired path of a straight line across the reef. As the uDrone traversed the reef, it took samples of the distance to the reef using the sonar. The final reef following path would follow the same trajectory on the X-Y plane as this straight line uDrone run. At each of these X and Y points, a new depth (Z) was recorded for the desired location. This point was calculated by subtracting the sonar depth value from the local depth value and then adding one. In the end, a trajectory was calculated and recorded that followed the reef at a distance of one meter.

6.2.1 Reef Following

Another instance of the simulation was initiated, with the uDrone starting in the same spot relative to the reef as in the previous, contour profiling, step. The PID controller node, in this case, listened for the trajectory on a ROS topic. As desired set-points were received the controller instructed the uDrone on how to move through

MAVROS. Meanwhile, the sonar readings were recorded in order to analyze the reef following the performance.

Figure 18 shows the uDrone moving over the reef in simulation. The vehicle can be observed pitched up to follow the reef. The blue cone below it is a visual representation of the sonar sensor which outputs the distance to the reef.

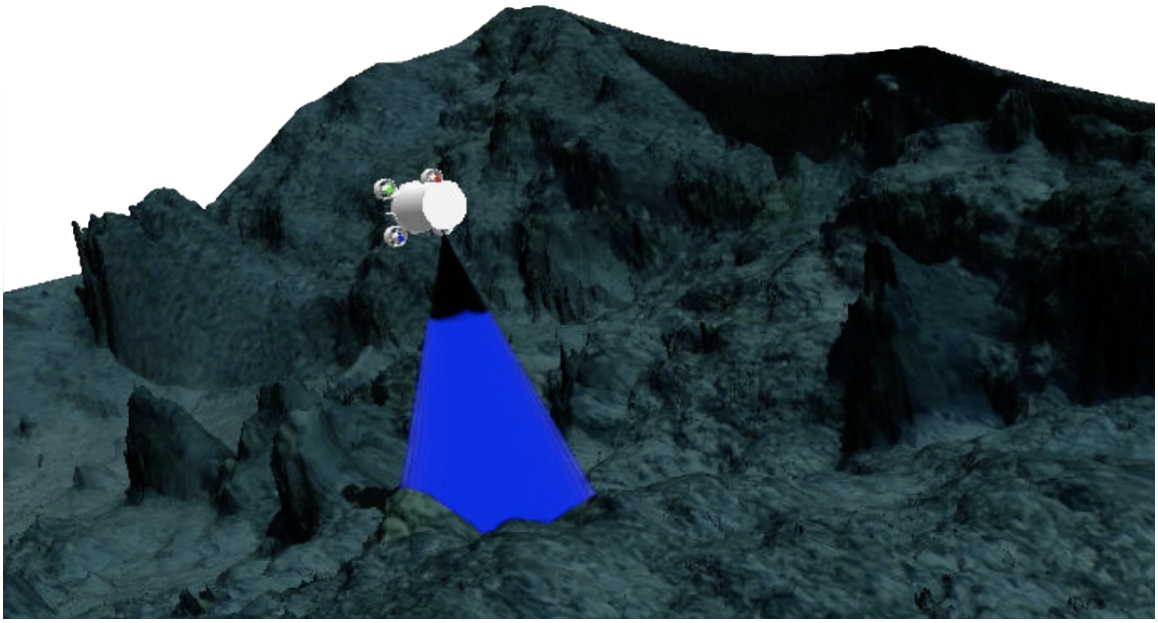

Figure 18. uDrone During Reef Following in Simulation

6.3 Results

The uDrone was able to follow the profile of the reef as measured by the initial reef mapping. Figure 19 shows the results of this trial. The blue line is the path of the uDrone relative to it initial location. The gray line is the sonar reading which is the distance from the uDrone to the reef. Using these two together, the orange line is

obtained, which is the position of the reef relative to the uDrone. The data starts at the 1m point, because it takes the uDrone some time and distance to reach a cruising altitude relative to the reef.

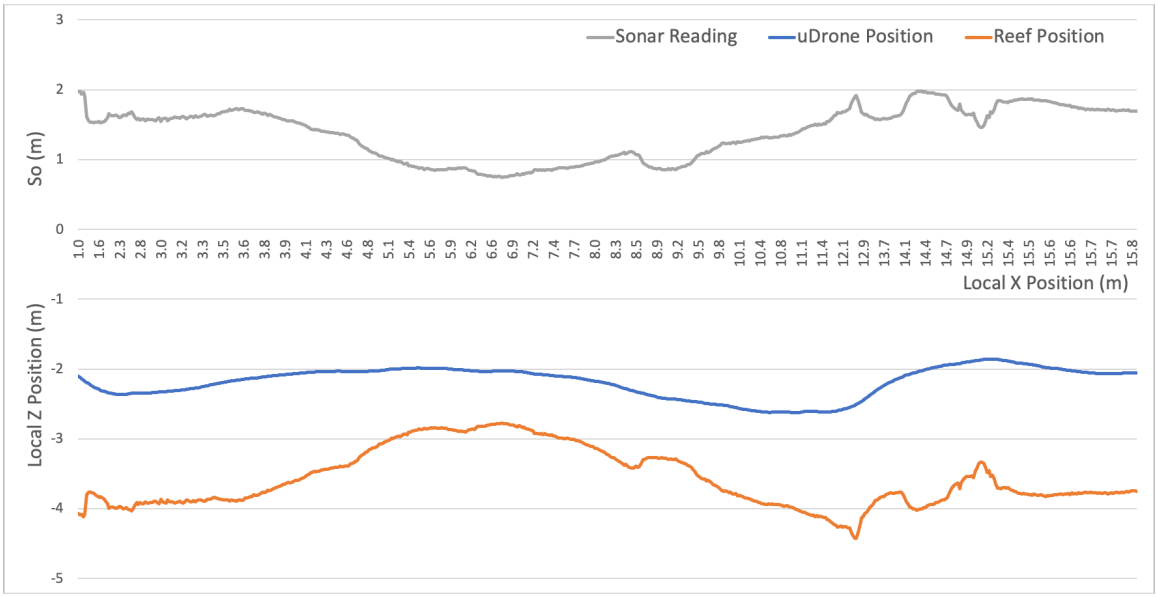

Figure 19. uDrone Reef Following Path and Sonar Reading

As is shown in figure 20, the mean squared error is typically below 0.4. It gets as low as 0.15 as the vehicle spends more time at the desired height between the five and ten meter mark.

One major source of error in this method is the sonar. The sonar projects a cone and then averages the distance over that cone projection. This means the initial pass, which was roughly one meters above the reef, took a scan of a larger section of the reef than the follow pass. This means the desired location was based on the average distance to the reef of a bigger segment of the reef than that recorded when only one meter above the reef.

Another source of error is the PID controller itself. The tuning of this controller

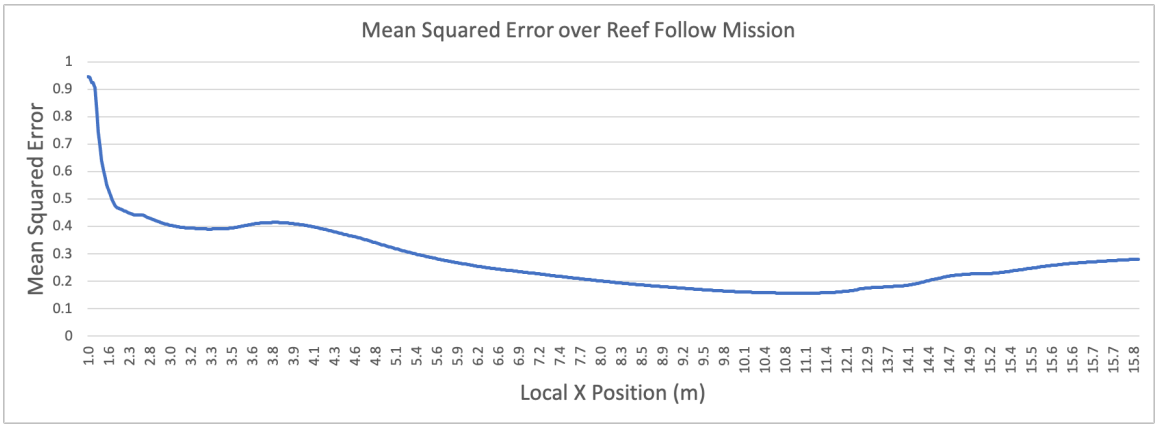

Figure 20. Mean Squared Error over Reef Follow Mission

will effect how closely the vehicle stays to its desired path. While the uDrone PID was tuned so that it would generally work, it might not be the optimal tuning parameters.

6.4 Conclusion

This experiment demonstrated that it is possible for the uDrone to practice reef following maneuvers with a PID controller and a known reef segment. As reef morphology data is collected for sections of reef, this method becomes more and more practical. Better resolution data for depth at various points will allow for even better results.

There are, however, drawbacks that would limit this approaches usefulness in a real world setting. First, localizing underwater is a difficult problem due to the lack of GPS. Therefore, it will be hard to start the uDrone from a known location relative to the reef from one trial to the next. This could be mitigated by using buoy to set start points, but that may requires adding man-made equipment to the reef. Second, this method relies on dead reckoning with the internal IMU to determine location. The error in the IMUs selected for the uDrone to not allow for dead reckoning over long

distances, so some other method for localization must be used. And third, the reef is comprised of living organisms which may between trial times. The ideal situation is for the vehicle to use vision in real time to determine its path.

Chapter 7

CONCLUSION

A novel underwater, open source, and configurable vehicle that can support experiments with both scientific and engineering goals was designed, constructed, and tested in simulation. Propulsion and sensor tests were carried out in a pool, in preparation for autonomy field tests. This final chapter will review the contributions of this thesis and discuss the ongoing and future work.

7.1 Contribution

A need for the uDrone was demonstrated. It will allow for greater volumes of coral reef imaging data to be obtained in a safe and reliable manner. Additionally, the uDrone will continue to produce advances in underwater autonomous navigation research from its joint simulation and real world testbed.

This thesis detailed the chosen hardware and software components and explained how they fit together. This system configuration could be used to develop underwater vehicles with similar technical capabilities.

The mathematical model of the vehicle dynamics will allow for future controller development. Specifically, this work can be used to create a model predictive controller which could greatly improve controllability of the uDrone.

The two control methods discussed in this document demonstrate how the uDrone navigates through a reef environment. In particular, the code written for the PID controller can be used for future research and development of autonomous controllers.

7.2 Ongoing Work

Work on the uDrone has continued as this thesis drew to a conclusion. This section discusses two areas where work is currently ongoing.

7.2.1 Water Testing

An initial set of tests has been conducted in a pool. This can be seen in Figure 21 and the video can be viewed at the URL associated with DREAMS Laboratory 2020. The first goal of these tests was to validate the waterproofing of the vehicles. The enclosure and seals are able to seal out water and the electronics were kept dry. The second goal is to test ballasting and buoyancy. The vehicle was very slightly positively buoyant, which is desirable. The center of gravity was slightly below the center of buoyancy, but in the same line. This caused the uDrone to sit horizontally in the water and have a slight moment to keep itself upright in roll, which is ideal. Finally, the vehicle dynamics were tested. This involved manually piloting it around the pool and verifying that it was agile and controllable. While there is no concrete measurement for this, it qualitative performed very well in the water.

7.2.2 Diver-Following

The testbed has allowed for proof of concept tests of vision based navigation. The first of these tests is being run by a DREAMS Lab member and uDrone contributor, A.L.G. Prasad. Using the same PID controller developed as part of this thesis, he was able to write code to enable the uDrone to follow a scuba diver in simulation. This is

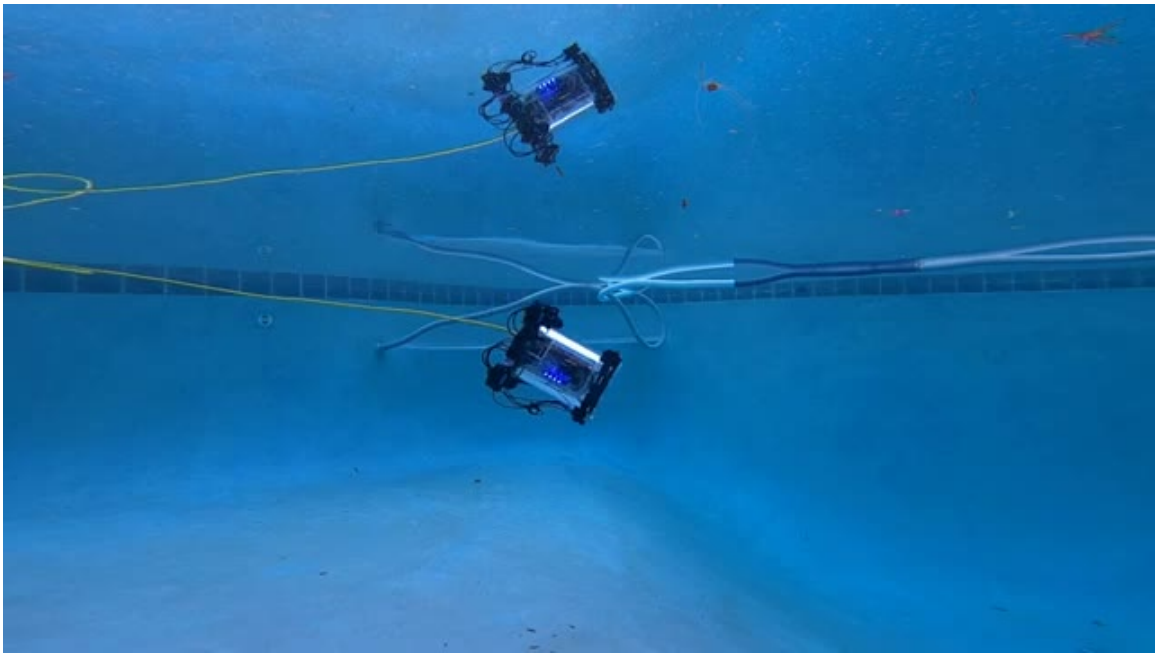

Figure 21. uDrone in the pool during a propulsion test
Source: DREAMS Laboratory 2020

only the first step in the Vision Based Navigation track, but it is a meaningful step towards autonomy.

7.3 Future Work

Despite the achievements laid out here, there is still substantial work that need to be performed for the uDrone to operate in a fully autonomous mode. The two most pressing areas of immediate future work are water tests and vision based navigation.

7.3.1 Vision Based Navigation

The end goal of the uDrone project is for it to be able to navigate unknown environments using vision. This will include aspects of visual-inertial odometry (VIO) and simultaneous localization and mapping (SLAM). Both of these fields are well developed on land but are still emerging in the underwater realm. For example, Ferrera et al. 2018 developed a method for visual odometer in turbid water using an inexpensive camera. Or, as previously mentioned, Manderson et al. 2020 shows a working application of underwater visual based navigation. Tools and datasets, like AQUALOC, are being created which can aid in the training of vision-based machine-learning algorithms for navigation (Ferrera et al. 2019). Underwater visual based navigation is a prime area of research right now and with the uDrone, DREAMS Lab and ASU are well positioned to become contributors and leaders in this area.

7.3.2 Real-World Water Testing

The next set of pool tests will verify that the uDrone can be autonomously controlled via ROS. Then, once it has been thoroughly tested in a controlled and confined environment, the uDrone will be taken to Hawaii where it will be tested in a real coral environment. The first few tests will include divers following the vehicle to watch for any issues. Eventually, it can complete its first solo autonomous missions.

7.3.3 Reinforcement Learning

With enough trails and data from the real world and an improved simulation it will be possible to create a controller for the uDrone based on reinforcement learning. This has the potential to be both more efficient and more robust than a PID or MPC. This new controller would be trained in simulation using collected data from real world experiments. It may employ other, pre-trained, vision algorithms or be trained as a full end-to-end algorithm.

7.3.4 Hardware Configuration

Several improvements can already be suggested for the uDrone hardware based on insights yielded from this thesis. Primarily, it was determined that switching the flat front plate of the enclosure to a domed plate would reduce the hydrodynamic drag on the uDrone and allow for the camera to be positioned at different angles. Additionally, more cameras can be added to the body at different angles in order to improve the vision based navigation that will come in the next phase.

In order to continue improving the vehicle configuration, several hardware experiments can be run. The first and most straightforward of these experiments is to test the impact of adjusting the vehicle's ballast. This would allow for different behaviors of the vehicle, which may be desirable for different types of missions. Furthermore, alternative thruster configurations can be tested. For example, if all the thrusters are pitched outward, then the uDrone would have some minimal control authority to move in the Y or Z directions independently of rotating. This would have the drawbacks of

being less efficient for motion in the X direction and greatly complicating any models or model based controllers that have been developed.

REFERENCES

AAUS, American Academy of Underwater Sciences. 2019. "Standards for Scientific Diving Manual." Accessed September 12, 2020. https://www.aaus.org/AAUS/About/Diving_Standards/AAUS/Diving_Standards.aspx?hkey=25acfc9a-aea5-4e7f-86c6-9c514c1e764c.

Blue Robotics Inc. 2020a. "Blue Robotics Lithium-ion Battery (14.8V, 18Ah)." Accessed August 22, 2020. https://bluerobotics.com/store/comm-control-power/powersupplies-batteries/battery-li-4s-18ah-r3/.

———. 2020b. "Blue Robotics T200 Thruster." Accessed August 22, 2020. https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster/.

———. 2020c. "BlueROV2 - Affordable and Capable Underwater ROV." Accessed October 2, 2020. https://bluerobotics.com/store/rov/bluerov2/.

———. 2020d. "Ping Sonar Echosounder for Underwater Distance Measurement." Accessed August 22, 2020. https://bluerobotics.com/product-category/sensors-sonars-cameras/sonar/.

———. 2020e. "Underwater Subsea Pressure Vessel Enclosure for ROVs and AUVs." Accessed August 22, 2020. https://bluerobotics.com/store/watertight-enclosures/8-series/wte8-asm-r1/.

Buzzacott, Peter, ed. 2016. *DAN Annual Diving Report, 2016 Edition.* https://www.diversalertnetwork.org/medical/report/.

Connect Tech Inc. 2020. "Orbitty Carrier for NVIDIA Jetson TX2/TX1." Accessed June 29, 2020. https://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/.

DREAMS Laboratory. 2020. "uDrone EVO 3 propulsion test." Accessed November 4. https://www.youtube.com/watch?v=sM6XzXgjIlo.

Dronecode Project Inc. 2019. "QGC - QGroundControl - Drone Control." Accessed July 7, 2020. http://qgroundcontrol.com/.

Dronecode Project, Inc. 2020. "Open Source Autopilot for Drones - PX4 Autopilot." Accessed July 7, 2020. https://px4.io.

Duecker, D. A., A. Hackbarth, T. Johannink, E. Kreuzer, and E. Solowjow. 2018. "Micro Underwater Vehicle Hydrobatics: A Submerged Furuta Pendulum." In

2018 IEEE International Conference on Robotics and Automation (ICRA), 7498–7503. doi:10.1109/ICRA.2018.8461091.

Ermakov, Vladimir. 2018. "mavros-ROS Wiki." Accessed July 7, 2020. http://wiki.ros.org/mavros.

Ferrera, Maxime, Vincent Creuze, Julien Moras, and Pauline Trouvé-Peloux. 2019. "AQUALOC: An underwater dataset for visual–inertial–pressure localization." *The International Journal of Robotics Research* 38, no. 14 (October): 1549–1559. doi:10.1177/0278364919883346.

Ferrera, Maxime, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. 2018. "Real-time Monocular Visual Odometry for Turbid and Dynamic Underwater Environments." *CoRR* abs/1806.05842. arXiv: 1806.05842. http://arxiv.org/abs/1806.05842.

Fossen, Thor I. 2011a. "Kinematics." Chap. 2 in *Handbook of Marine Craft Hydrodynamics and Motion Control,* 15–44. John Wiley & Sons, Ltd. doi:10.1002/9781119994138.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119994138.ch2.

———. 2011b. "Models for Ships, Offshore Structures and Underwater Vehicles." Chap. 7 in *Handbook of Marine Craft Hydrodynamics and Motion Control,* 133–186. John Wiley & Sons, Ltd. doi:10.1002/9781119994138.ch7. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119994138.ch7.

———. 2011c. "Rigid-Body Kinetics." Chap. 3 in *Handbook of Marine Craft Hydrodynamics and Motion Control,* 45–58. John Wiley & Sons, Ltd. doi:10.1002/9781119994138.ch3. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119994138.ch3.

Fukunaga, Atsuko, John Burns, Brianna Craig, and Randall Kosaki. 2019. "Integrating Three-Dimensional Benthic Habitat Characterization Techniques into Ecological Monitoring of Coral Reefs." *Journal of Marine Science and Engineering* 7, no. 2 (January): 27. doi:10.3390/jmse7020027.

Georgiades, C., A. German, Andrew Hogue, Hui Liu, Chris Prahacs, A. Ripsman, R. Sim, et al. 2004. "AQUA: an aquatic walking robot," 4:3525–3531. January. doi:10.1109/IROS.2004.1389962.

Hackbarth, A., E. Kreuzer, and E. Solowjow. 2015. "HippoCampus: A micro underwater vehicle for swarm applications." In *2015 IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS), 2258–2263. doi:10.1109/IROS.2015.7353680.

Hall, Jennifer. 2014. "The risks of scuba diving: a focus on Decompression Illness." *Hawaii J Med Public Health* 73, no. 11 Suppl 2 (November): 13–16.

Hydromea. 2020. "LUMA - Hydromea." Accessed July 7, 2020. https://www.hydromea.com/underwater-wireless-communication/.

Kirk, D.E. 2012. *Optimal Control Theory: An Introduction.* Dover Books on Electrical Engineering. Dover Publications. https://books.google.com/books?id=onuH0PnZwV4C.

Kohlbrecher, Stefan, and Johannes Meyer. 2016. "hector$_g azebo_p lugins - ROSWiki$." Accessed October 2, 2020. http://wiki.ros.org/hector_gazebo_plugins.

Li, Jiwei, Nicholas S. Fabina, David E. Knapp, and Gregory P. Asner. 2020. "The sensitivity of multi-spectral satellite sensors to benthic habitat change" [in English (US)]. *Remote Sensing* 12, no. 3 (February). doi:10.3390/rs12030532.

Li, Jiwei, David E. Knapp, Steven R. Schill, Chris Roelfsema, Stuart Phinn, Miles Silman, Joseph Mascaro, and Gregory P. Asner. 2019. "Adaptive bathymetry estimation for shallow coastal waters using Planet Dove satellites." *Remote Sensing of Environment* 232:111302. doi:https://doi.org/10.1016/j.rse.2019.111302.

Manderson, Travis, and Gregory Dudek. 2018. "GPU-Assisted Learning on an Autonomous Marine Robot for Vision-Based Navigation and Image Understanding." In *OCEANS 2018 MTS/IEEE Charleston,* 1–6. IEEE, October. doi:10.1109/OCEANS.2018.8604645.

Manderson, Travis, Juan Camilo Gamboa Higuera, Stefan Wapnick, Jean-François Tremblay, Florian Shkurti, David Meger, and Gregory Dudek. 2020. *Vision-Based Goal-Conditioned Policies for Underwater Navigation in the Presence of Obstacles.* arXiv: 2006.16235 [cs.RO].

Manhães, Musa Morena Marcusso, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. 2016. "UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation." In *OCEANS 2016 MTS/IEEE Monterey.* IEEE, September. doi:10.1109/oceans.2016.7761080.

Manzanilla, A., S. Reyes, M. Garcia, D. Mercado, and R. Lozano. 2019. "Autonomous Navigation for Unmanned Underwater Vehicles: Real-Time Experiments Using

Computer Vision." *IEEE Robotics and Automation Letters* 4 (2): 1351–1356. doi:10.1109/LRA.2019.2895272.

Mohta, Kartik, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, et al. 2017. "Fast, autonomous flight in GPS-denied and cluttered environments." *Journal of Field Robotics* 35, no. 1 (December): 101–120. doi:10.1002/rob.21774.

NTNU, Norwegian University of Science and Technology. 2020. "ROV Minerva." Accessed October 2, 2020. https://www.ntnu.edu/aur-lab/rov-minerva.

NVIDIA Corporation. 2020a. "JetPack SDK | NVIDIA Developer." Accessed June 29, 2020. https://developer.nvidia.com/embedded/jetpack.

———. 2020b. "Jetson TX2 Module | NVIDIA Developer." Accessed June 29, 2020. https://developer.nvidia.com/embedded/jetson-tx2.

Open Robotics. 2020. "ROS.org | Powering the World's Robots." Accessed July 7, 2020. https://www.ros.org/.

Open Source Robotics Foundation. 2020a. "Documentation - ROS Wiki." Accessed July 7, 2020. https://wiki.ros.org/.

———. 2020b. "Gazebo." Accessed July 7, 2020. http://gazebosim.org/.

Perez, T., and T. I. Fossen. 2009. "A Matlab Tool for Parametric Identification of Radiation-Force Models of Ships and Offshore Structures. Modelling, Identification and Control," MIC-30(1):1–15. https://github.com/cybergalactic/MSS.

PX4 Dev Team. 2020. "Hex Cube Black (FMUv2) PX4 User Guide." Accessed June 29, 2020. https://docs.px4.io/master/en/flight_controller/pixhawk-2.html.

SofarOcean. 2020. "Explore the underwater world with Trident." Accessed October 2, 2020. https://www.sofarocean.com/products/trident.

Sousa, A., Luis Madureira, J. Coelho, Jose Pinto, J. Pereira, J.B. Sousa, and Paulo Dias. 2012. "LAUV: The man-portable autonomous underwater vehicle." 3 (January): 268–274.

Stereolabs Inc. 2020. "ZED 2 -AI Stereo Camera | Stereolabs." Accessed June 29, 2020. https://www.stereolabs.com/zed-2/.

Thompson, David R., Eric J. Hochberg, Gregory P. Asner, Robert O. Green, David E. Knapp, Bo-Cai Gao, Rodrigo Garcia, et al. 2017. "Airborne mapping of

benthic reflectance spectra with Bayesian linear mixtures." *Remote Sensing of Environment* 200:18–30. doi:https://doi.org/10.1016/j.rse.2017.07.030.

BIOGRAPHICAL SKETCH

Alex Goldman is a Master's of Science student in the Robotics and Autonomous Systems program with a concentration in Artificial Intelligence at Arizona State University and a member of the Distributed Robotic Exploration and Mapping Systems Laboratory. Before attending ASU for his Master's, Alex started several entrepreneurial ventures and worked as a project lead on large projects involving both hardware and software. This led Alex to be an ideal candidate to take on the uDrone project, which leveraged his leadership experience while furthering his engineering education. Alex has had a lifelong interest in the aquatic environment and reef conservation. He started scuba diving at the age of 11 and at 19 he became a certified Scuba instructor. During his undergrad at the University of Michigan, he was one of the leaders of the Human Powered Submarine Team. After school, he volunteered as a scuba diver at the Shedd Aquarium in Chicago. He even proposed to his wife underwater. This is all to say: Alex has been personally interested in scuba diving and reef conservation for a very long time.