UNIVERSITY OF CALIFORNIA
SANTA CRUZ

AUTOMATING OCEANOGRAPHY: A ROBOTIC SURFACE SENSOR PLATFORM COMBINING FLEXIBILITY AND LOW-COST

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Bryant W Mairs

June 2015

The Dissertation of Bryant W Mairs
is approved:

Professor Gabriel Elkaim, Chair

Professor Ricardo Sanfelice

Professor Renwick Curry

Dr. Raphael Kudela

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Automating Oceanography: A Robotic Surface Sensor Platform Combining Flexibility and Low-cost

by

Bryant W Mairs

This work details the design, development, and testing of the autonomous surface vessel, the SeaSlug, including both its ground control station and simulation environments. This project differs from existing commercial and research platforms as it has been specifically designed to: **(i)** provide sufficient payload capacity and modularity for easy modification of the system's hardware and firmware, **(ii)** allow for extensive out-of-water testing within a robust simulation environment, and **(iii)** operate in the open ocean for a variety of medium-duration missions.

The system is built around a central communications network using the CAN protocol, with components operating as modular parts of well-defined subsystems. This facilitates either the modification or addition of hardware to support a wide variety of mission types. At 6.7m in length, sufficient payload capacity is available for additional sensors in a large internal cabin and two external payload bays that run vertically through the hull. An extensive simulation environment allows for testing of all subsystems before mission deployment.

Using the L_2^+ guidance algorithm adapted from unmanned aerial vehicles, the SeaSlug is capable of following a desired trajectory to within 2.0m across a range of weather conditions and sea states. This has supported its use for missions that include testing solar panel integration for extended mission endurance; evaluating additional guidance, navigation, and control algorithms; and collecting oceanographic data on thermal fronts using a scientific sensor adapted to the rear payload bay.

To my grandfather, George A. Mairs III, without whom I would not have had the drive and opportunity to pursue learning as far as I have.

Acknowledgments

I would first like to thank my parents, Todd and Candyce Mairs, who always pushed me to succeed and further my education. They made sure my room was stocked with science books even though I stayed up way too late most nights when reading them. They continued to push me to learn and succeed and their support was essential for making it this far.

To my advisers, Gabriel Elkaim and Renwick Curry, for their patience in explaining both robotics and controls theory. The Autonomous Systems Lab exists primarily because of their passion and willingness to teach.

To all of my friends and colleagues who have helped me directly or indirectly with debugging, testing, and launching the SeaSlug (including Pavlo Manovi, Jonathan Bruce, Jesse Harkin, Bar Smith, Max Dunne, Sharon Rabinovich, Caio Porto, Marcello Guarro, and Max Lichtenstein). Without them the SeaSlug would never have made it into the water, let alone be autonomous.

And to all of the people I met in Santa Cruz and at the University of California there. Santa Cruz has been a very special place to me and I will never forget the adventures we shared during breaks from work.

Chapter 1

Introduction

1.1 Overview

This thesis details the design of the autonomous surface vessel, the SeaSlug. This project has its origins as the *wg1* project started by Willow Garage [92] in 2006. A hull was designed and fabricated and the necessary internal components were integrated to provide for basic remote control and seaworthiness tests. Further development necessitated a system redesign and it was during this stage that the project was canceled. The system was then donated to the University of California in a non-operational state. The system internals were removed and replaced with components that complimented the other robotic platforms in use by the Autonomous Systems Lab. It is the development and integration of these electrical and software components (and testing thereof) that comprise the majority of the work of this thesis. This work has been an exercise in solving systemic issues for a large complicated robot including (but not limited to) integration, reliability, operator interface, and simulation.

The design impetus for the SeaSlug was to reduce the cost of data collection for oceanography. The state of the art for oceanographic data collection is currently defined by manned vessels. They are capable of a wide range of missions but their use is restricted by their high operating cost ($15k/day) and difficulty in scheduling (experimental lead times are often on the order of several months). A lack of access to manned vessels has become the limiting factor for oceanographers to perform their research.

Since this project began other autonomous systems have become viable for data collection applications and several commercial products now exist. Some systems provide short-range support to manned missions, expanding their effective sensor range. Other systems are capable of long solo missions, collecting data for months or years at a time while being monitored remotely. These projects have all been developed in response to the high cost of oceanographic data collection and demonstrate the viability of automation in lowering the cost of scientific data collection.

While these systems are capable of effective data collection missions, none provide sufficient flexibility in modifying the core hardware and software of the system. The difficulty of integrating additional components hinders research outside of the vehicle's original design. In some cases it can necessitate purchasing new hardware as the existing hardware cannot be adapted. This limits the system's usefulness and increases its cost of operation.

It has been an additional goal of the SeaSlug project to support general-use research in addition to scientific data collection. The size of the system supports the addition of significant hardware in a variety of internal and external locations. Additionally, the internal system is split between subsystems with well-defined interfaces. This modularity supports experimentation with alternate components and/or the addition of new subsystems. This modularity extends to the low-level control algorithms, which are easy to modify and test in simulation using both simulation (within the Simulink software environment) and hardware-in-the-loop (HIL) testing.

The SeaSlug has proven itself to be reliable and usable across a variety of missions. The L_2^+ guidance algorithm originally developed for UAV trajectory following was adapted to the SeaSlug and tested in littoral waters, following a planned trajectory to within ± 2.0m. This performance was demonstrated during scientific data collection missions that include the integration of conventional oceanographic sensors. Additionally, an external solar panel was also integrated and evaluated for supporting extended-duration missions.

1.2 Motivation

The top one meter of the ocean, known as the *sea-surface microlayer*, has been found to contain the key drivers for a number of environmental processes. For example, this layer carries anthropogenic pollution from rain and river runoff, which stays at the surface due to its lower salinity [41] [94]. This pollution is thought to contribute to harmful algal blooms (HABs), which are potentially dangerous to both human and aquatic life [83]. This top microlayer is home to a large variety of sunlight-dependent organisms, like algae and phytoplankton, that form the base of the marine food chain. Many studies have shown that these organisms are extremely sensitive to even minor changes in their environment with the health of these populations affecting the entire marine ecosystem [19]. Furthermore the sea surface is an active location for physical and chemical processes that affect global climate change [57]. Ocean acidity is also an important field of study and causing concern about the long-term stability of the marine environment [39]. The sea surface has been found to play a large role in acidification because of its ability to sequester vast amounts of carbon, a major output of industrialized nations [32].

To further their understanding of this important microlayer, oceanographers have to date employed a variety of data collection methods. Some of these are unmanned systems that include moored buoys, satellites, and shore-based observation platforms. Buoys are effective at continuously collecting data in a small area. Satellites provide large-scale observations of phenomena that pass within the sensor view of their orbit, but are not always available to survey a given area, or at the resolution desired. Finally shore-based platforms offer the same advantages as buoys, but are easier to maintain and can serve as test sites for new sensors and equipment.

Even with the ubiquitous availability of these tools, only manned surface vessels have the necessary range and payload flexibility required for many missions. The larger research vessels are capable of carrying scientists, their equipment, and supplies for missions lasting months at a time, albeit at very high cost. Manned vessels can also extend their effective sensor range with the use of remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs). Manned vessels are capable of deploying and monitoring several of these vehicles simultaneously, compounding their benefits.

However, the use of manned vessels imposes significant scheduling and cost issues. Vessels are rarely available for spontaneous deployments to study ephemeral events like algal blooms, as even at the largest research institutes they are reserved several months in advance. Their operating expenses can be prohibitive for all but the best-funded research institutes. In 2013 the National Science Foundation's (NSF) Academic Research Fleet, comprising a total of 21 vessels, required \$82M in operation and maintenance costs [68], accounting for 12.5% of the total facilities budget. Even the renting of a single vessel can be prohibitive. For example, the listed rates for the rental of the smallest research vessel in the fleet of the Monterey Bay Aquarium Research Institute (MBARI) costs upwards of $17,000USD per day [61].

For these reasons the SeaSlug was designed as a low-cost flexible research platform capable of short-term missions. Its design was guided by the following requirements:

Mission agnostic

The platform must be generic and support a variety of mission types. It will serve as a usable platform to test algorithms or hardware as well as perform a variety of data collection missions. This implies support for a wide array of hardware through a large payload capacity and multitude of internal and external mounting locations.

Extensible

In support of a variety of different missions, the system must be easy to modify. This extends beyond the mechanical and electrical systems to the operator interface and the internal guidance algorithms. Custom components are only to be designed and built when the necessary functionality cannot be found in a common off-the-shelf (COTS) product. The selection of COTS components prioritizes the use of common well-defined interfaces as much as component performance. This simplifies integration and supports on-the-fly replacement (thus reducing system downtime).

Low cost

As data collection costs are one of the largest consumer of funds for oceanographers, the system will target low cost operations. This is especially important for smaller institutions, which have both smaller budgets and reduced engineering and support staff. Additionally, these low operating costs can support experiments not previously possible. In support of low operational costs the system aims for simplicity to reduce the long-term maintenance cost. Therefore easily-replaceable COTS parts are used where possible, as they come with technical support and can be easily serviced or replaced. Another key feature of a low-cost platform is to allow for the reuse of existing hardware.

All-day longevity

Many data collection missions are short-term, on the order of a few days. The system must be capable of running an entire 24-hour day at a useful speed. The system must be tuneable for greater endurance at the cost of reduced range as the mission requires, and flexible enough to change these parameters on-the-fly.

Robust

To be applicable to general use the system must be capable of open-ocean deployments across a variety of water and weather conditions (from a Beaufort sea state of 0 to 7). This must also extend to the shallow waters of near-shore coastal areas and inland waterways. The marine environment is harsh on both mechanical and electrical components; the system must be tolerant of individual failures with the ability to recover and continue operation.

1.3 Existing Autonomous Surface Vessels

Research on autonomous surface vessels has been ongoing since the 1950s and many distinct vehicles have been developed for data collection, defense, and engineering experiments. For additional information a broad overview of surface vessel research is provided by [16]. A history of defense applications of ASVs is detailed by [12], while both [77] and [30] expound on the history of wind-powered vehicles. A complete history of ASVs, while interesting, is not relevant to this dissertation and only recent autonomous surface platforms are discussed.

At the time the SeaSlug project started in 2009, only the OASIS [44] and the Wave Glider [89] existed as functioning autonomous surface platforms. Both systems were being actively developed and experimented with, but neither were capable of meeting the specifications outlined above or meet the specific integration tasks desired for this project.

1.3.1 Wave Glider

The Wave Glider SV2 [89] is a commercial product developed by Liquid Robotics; their first open water deployment testing began in 2009. It targets extended-duration missions that last months or years by utilizing both solar and wave power and has successfully completed transits from San Francisco to Sydney, Australia. It is propelled by a passive mechanical system that harnesses wave power and is capable of speeds up to 1.5 m/s, with missions averaging 1.2m/s in the open ocean. Solar panels power the onboard systems with 5W allocated to any sensors stored in internal payload bays. Communication of status updates and telemetry to a home base is provided by an Iridium satellite modem. This communications channel also supports a remote interface for mission retasking, making this system highly suited to long-duration deep-ocean studying. Remote communications are centrally controlled by Liquid Robotics and they handle monitoring and retasking as desired by the mission operators.

This system has been successfully used to study a great variety of oceanographic features. The vehicle is almost silent because of its wave propulsion and

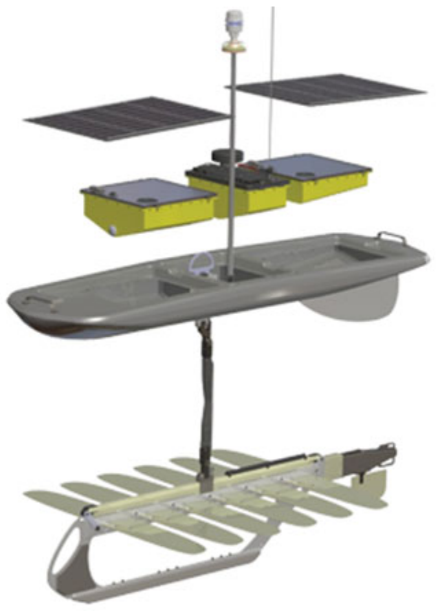


Figure 1.1: The Wave Glider SV2 architecture. From top the bottom is shown the solar panels, internal payload and sensors, surface float, tether, and glider propulsion.

has been used to track and study whales [91]. They have also analyzed the sensors commonly used by marine scientists and how they can be adapted to the WaveGlider in Table 1 in [26].

While this vehicle is incredibly effective at long-duration missions and deep-ocean studies, it has some limitations due to its physical design. Its underwater glider prevents it from use in harbors or shallow inland waters. It speed is also weather dependent, though the newest SV3 model has active propulsion. While even small waves are common on the open ocean, this propulsion is less effective in inland waters. Studying inland waterways is further complicated by the depth of the glider. Additionally its onboard payload capacity is limited and retrofitting existing sensors can be both expensive and problematic. The internal systems are tightly integrated and the feasibility of changing its internal electronics or software components are unknown as the entire system is proprietary.

1.3.2 OASIS

The Ocean Atmosphere Sensor Integration System (OASIS) was created in 2005 to be low-cost, reusable, and reconfigurable; enabling ocean missions up to several months in length [44]. The resultant vessel is 6m in length, with a custom fiberglass mono-hull containing three large internal payload bays. Scientific sensors can be mounted internally with a sea water intake. A mast supports additional fixed navigation sensors and communication electronics.

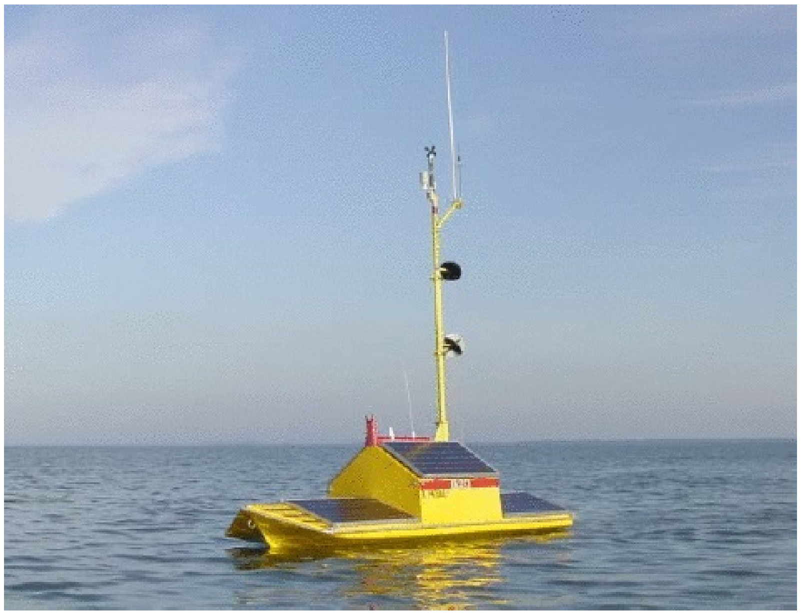


Figure 1.2: The OASIS during testing on the open ocean. The mast and raised central cabin contain the sensors and payload.

Published information on the OASIS provides no specifics of the guidance, navigation, and control systems or of a simulation environment for the OASIS. The literature only discusses the sensor capabilities with a 6-degree-of-freedom inertial sensor and inclinometers capable of tracking the vehicle's attitude. Remote communications are supported over cellular, RF, or Iridium modems.

Power is provided by six 170W solar panels backed by twelve 12-volt marine batteries. The batteries power both the onboard electronics and the electrical propulsion. [70] describes a mission where power draw exceeded power gain during the entirety of the mission, even at peak sunlight, although no numbers were provided.

Of the vehicles discussed here, the OASIS is most comparable to the SeaSlug. It uses active electrical propulsion and has a similar size and hull design. Its suitability for further experimentation was unknown as there is both little literature on the details of the system and no missions have described its guidance and navigation software or integration capabilities. The OASIS project has been discontinued as of 2012 due to a lack of funding.

1.3.3 Newest Platforms

Since 2009, two additional platforms were developed that target the very-long-duration missions and compete with the Wave Glider. Both of these systems also share the primary deficiency of the Wave Glider as well: high integration costs.

The AutoNaut [9], developed by MOST (Autonomous Vessels) Ltd., is another wave-powered ASV. It measures 3.5m in length and is capable of sustained speeds of 1.5m/s for months at a time. Development started in 2012 with initial testing in 2013. They are still being tested in the open ocean around England, though they have limited exposure in the scientific literature at this time.

The AutoNaut is a more conventional vessel than the Wave Glider, consisting of a single fiberglass hull. Wave power is provided by flexible fins mounted directly to the hull, instead of on an attached glider. The AutoNaut is therefore more flexible, as there is no underwater component preventing use in shallow waters. But like the Wave Glider, its size and custom, tightly-integrated internal systems limit experimentation.

The Saildrone [80] is a commercial product by Saildrone, Inc. that was started

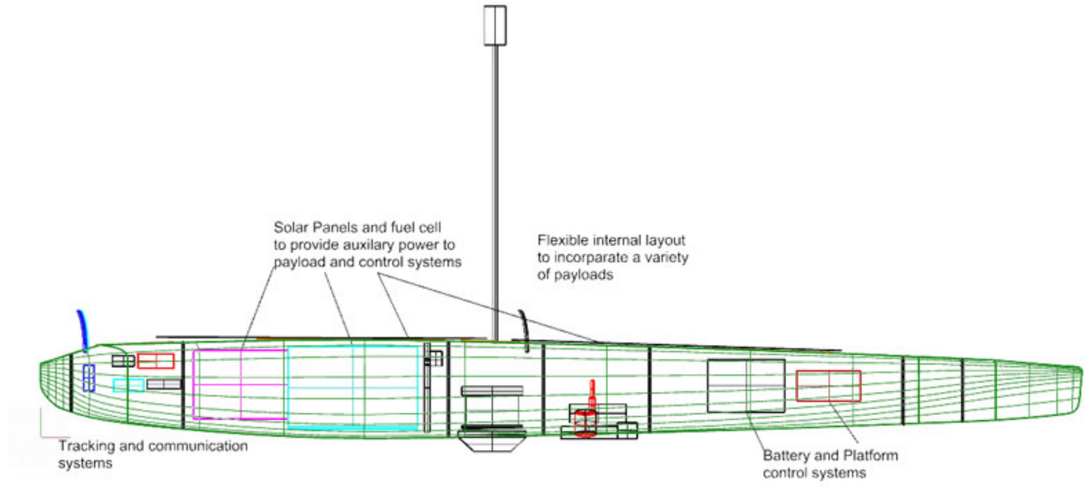

Solar Panels and fuel cell
to provide auxilary power to
payload and control systems
Flexible internal layout
to incorparate a variety
of payloads
Tracking and communication
systems
Battery and Platform
control systems

Figure 1.3: The Autonaut architecture. Shown is the base 3.5m version.

in 2013 and targets shorter-duration missions of under a year. It is wind-powered, with a 7.6m vertical wing enabling it to travel at up to 7.7m/s. The hull is a narrow, trimaran hull with a 2m keel and totals 5.8m in length. 10W of power is continuously available to the 100kg of scientific payload that can be integrated with the system.

The vessel has been deployed on missions lasting longer than three months, where ground speeds averaged 1.2m/s. Further scientific missions are planned including tracking of tagged sharks and their habitats as well as monitoring the aftermath of the 2010 Deep Water Horizon oil spill in the Gulf of Mexico.

Although these system are compelling because of endurance and mission capabilities, there are too few technical details or mission results available for an effective evaluation.

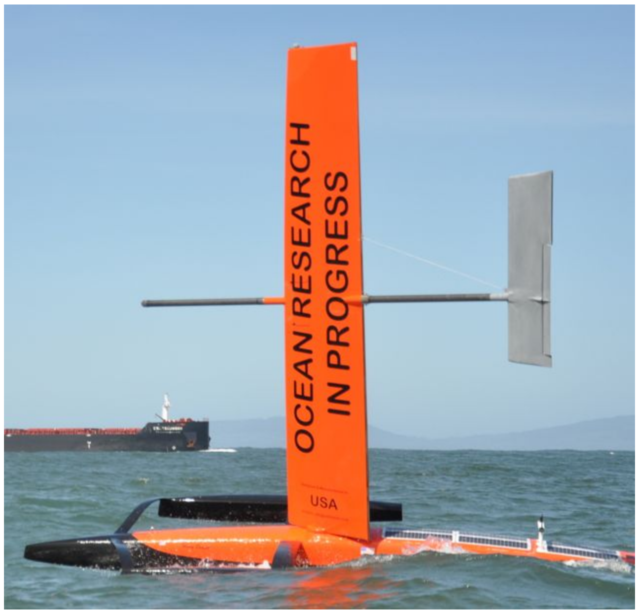


Figure 1.4: The Saildrone during a test run in the San Francisco Bay. Its trimaran hull and wing are easily visible. Solar panels and sensors are exposed on the deck at the rear.

1.3.4 Other Systems

The above platforms are all capable of long-term data collection missions and are the most directly comparable to the SeaSlug system described in this work. However, several other surface vessels target specific application niches that are worth mentioning here due to their unique features.

ASV Ltd. produces the C-Enduro [14], a 4m-long catamaran capable of long-duration missions. The system is unique in drawing power from three separate sources: solar panels, wind turbine, and a diesel generator. These can provide power for data collection missions lasting up to three months. The system also supports vertical profiling with an automated winch. Little has been published on this vehicle, with the exception of some work on simulated collision avoidance [81]. This system has a significant feature set and targets the same short-duration missions as the SeaSlug, but a lack of information regarding the integration of additional hardware and software prevents detailed comparisons.

The Jetyak [50] bears special mention here as it targets dangerous deployments where the vehicle may not be recoverable. It is a modified jet-propelled ocean kayak capable of single-day (6-8 hour) deployments. At 3.4m the system is small enough to be shipped to remote locations in standard shipping containers. Its estimated build cost is $15,000, making replacement of the vessel affordable in the case that it becomes too damaged to recover during a mission. Though its mission endurance is low, refueling is simple due to its gasoline engine. All of these features result in a vehicle that is easy to deploy in remote locations as a support vehicle for single-day data collection.

The largest ASV mentioned here is the SCOAP [22] that totals 11m in length. Both its large size and catamaran hull design facilitates the use of existing scientific payloads with minimal modification. At its nominal operating speed of 2.5m/s it is capable of month-long deployments because of its electric thrusters powered by both a large onboard battery store and diesel generator. This vessel was specifically designed for data collecting missions in shallow estuarine environments. Initial testing in a sheltered water environment shows promise, but additional open-ocean tests are forthcoming.

Medium-term deployments in rough weather states are supported by the SailBuoy [79]. This system is a 2m-long wind-propelled vehicle suitable for data collections of up to a year. It has been successfully tested over a 62-day mission where it collected data on surface temperature, salinity, and oxygen concentration [37]. Additional work discusses the performance of the system in harvesting wind energy for forward propulsion [33]. Though this system has proven itself capable of long ocean deployments, its size limits its payload to smaller existing hardware sensors and makes it difficult to use as a platform for vertical profiling.

1.4 Contributions

Through the development of this thesis, the following contributions were made to the state-of-the-art:

1. The design and implementation of a modular system architecture suitable for an autonomous surface vessel that is both highly-extensible and applicable to other robotic platforms.

2. The design, development, and validation of an open-source autonomous surface vessel that is capable of both general engineering experimentation and scientific data collection.

3. Analysis of power consumption of the subsystems of an electrically-propelled autonomous surface vessel and its potential for solar power generation to allow for extended duration missions.

4. The adaption and analysis of the L_2^+ control algorithm to an autonomous surface vessel.

5. The design and development of a robust simulation environment for an autonomous surface vessel that is suitable for algorithm and system testing on land.

1.5 Dissertation Organization

The following chapters detail the system architecture of the SeaSlug and the evaluation of the complete system. These chapters aim to provide substantial technical detail as this dissertation is meant to serve both as a discussion of the details of this project and as a reference for future electric autonomous surface platforms.

Chapter 2 describes the system architecture of the SeaSlug, including the motivation behind individual design decisions. Chapter 3 highlights the extensive simulation environment used for testing the system, including the mathematical models developed for it. These models are further expounded on in Chapter 4, which describes the sensor processing and controls algorithms used by the SeaSlug. The SeaSlug was evaluated over a period of weeks undergoing thorough testing and mission trials, which are discussed in Chapter 5. A detailed power analysis of the system, including experiments with solar energy scavenging, is described in Chapter 6. Chapter 7 provides the concluding summary and details of future work that can build on the SeaSlug platform and extend its capabilities. Additional appendices provide background on theoretical and technical aspects of this project.

Chapter 2

System Architecture

2.1 Introduction

The SeaSlug was originally designed by Willow Garage and gifted to the University of California at Santa Cruz in 2009. Since then it has undergone considerable work to become an operable system as part of the Autonomous Systems Lab's (ASL) research effort to automatically collect high-quality data using autonomous robotic systems. Specifically this work builds from knowledge and software components developed as part of the SLUGS autopilot project [58], both to aide in the development of the system and to ensure future maintainability.

The core tenants of the design of the SeaSlug are modularity and openness. One of the biggest limitations with existing platforms is the difficulty in modifying the system and integrating additional hardware. Those platforms are generally tightly-coupled and have limited physical, electrical, or software capacity for new components. It is therefore the goal of the SeaSlug to provide a system which is both capable of modification and amenable to it. Individual components therefore utilize open and common standards where possible. Additionally open source components are used wherever possible so that they can be modified as needed.

Beyond modularity, the SeaSlug also aims to be low cost and was designed for a low total cost of ownership. Existing systems are tightly-integrated and difficult to maintain without extensive, and often expensive, support from the manufacturer. Smaller research groups lack both the engineering ability to work on these tightly-integrated systems and the funds to pay for manufacturer support. The SeaSlug's design has been optimized to reduce both its upfront and long-term maintenance cost.

The resultant architecture of the SeaSlug is described in this chapter. As there were several competing goals influencing the design, commentary has been provided in defense of the current state and discussing possible alternative designs. Additional factors, such as time and funding constraints, has resulted in some components and subsystems that should be replaced or modified as budget and time permits. This chapter therefore seeks to provide not only a formal description of the system architecture but also a context for how the system was designed given the various constraints.

2.2 Mechanical Architecture

The SeaSlug has a single fiberglass hull. The deck of the SeaSlug is 2.0m wide in order to provide sufficient surface area for solar panels. It is tapered at each end, coming to a point in the front at the hull with a broad base at the rear where the mast is mounted. The mast serves as a mounting point for electronics while also providing an improved radar profile by means of a large reflector, visible as the large white cylinder in the upper-left of Fig. 2.1. With the low height of the vessel at 1.0m in the water, a radar reflector improves the SeaSlug's visibility for both human pilots and radar systems.

A keel-mounted propeller provides propulsion for the vessel. It is directly

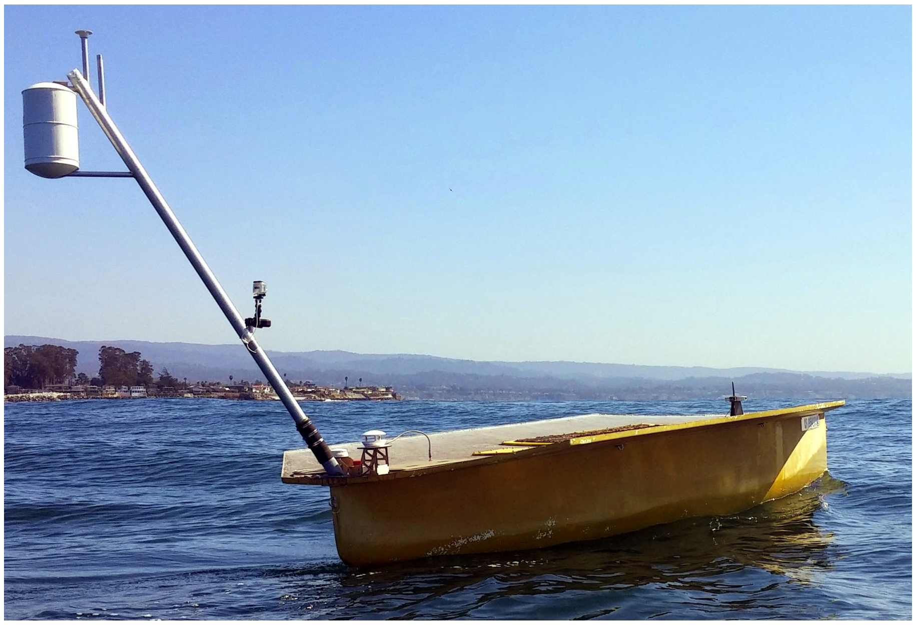


Figure 2.1: The SeaSlug driving autonomously in the Monterey Bay. External sensors are mounted at the rear, by the base of the mast. The mast provides visibility and a radar profile. At the front are running lights and the telemetry antenna.

driven by a brushless DC (BLDC) electric motor. A custom, 0.45m-wide propeller was designed to provide optimal torque at the nominal vessel speed of 1.6m/s. This speed is considerably slower than most manned vessels, but is sufficient for scientific data collection and other missions. Other ASV platforms target this same speed range, including the Wave Glider that averages 1.2m/s and has been very successful over a range of long-term missions. This low speed is also sufficient due to the typically low speed of open ocean currents. Around Monterey Bay, the test area of the SeaSlug, the surface current is usually 20cm/s or less [71].

The rudder applies the steering force at the rear of the vessel. It is driven with a 12V stepper motor through a geared chain drive and has a range of -45° to 45°. With the rudder at the rear and the weight distribution of the vessel, the center of rotation is towards the front of the vessel (directly above the bilge pump shown in Fig. 2.2).

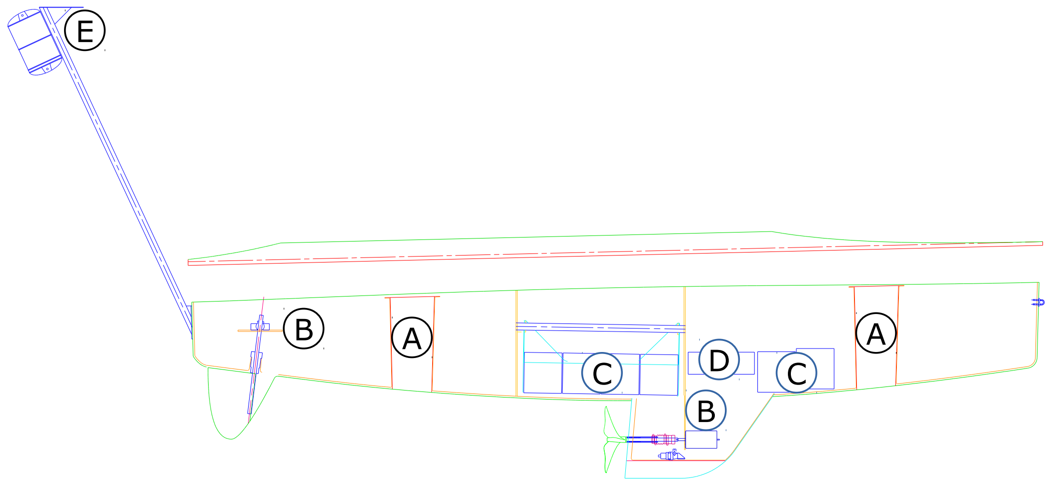


Figure 2.2: The mechanical layout of the SeaSlug. A) housing for primary control electronics, B) propeller & rudder actuators, C) 12V electronics batteries, D) 24V actuator batteries & ballast, E) radar reflector & self-righting buoyancy.

2.2.1 Self-righting

Though mono-hull designs are inherently stable, they can have two points of stability: one when right-side up and one when inverted. To remove the inverted stability point, a mast with buoyancy is mounted off the center axis of the vessel. With the inverted stability point removed, the vessel should self-right when capsized, which is a possibility in rougher sea states.

This process can be accelerated by using the 91kg of internal ballast, which can rotate up to 90° in either direction around the longitudinal axis. The combination of the internal ballast with the buoyancy of the mast should allow for rapid self-righting maneuvers, though this functionality has been neither implemented nor tested.

2.2.2 Payload Capabilities

The SeaSlug is designed to support a wide variety of scientific payloads. Accordingly, a significant amount of space has been reserved for mounting additional scientific sensors. While it could be argued that a flat, open deck allowing arbitrary mounting of components would have been ideal, this space has instead been dedicated to core guidance, navigation, and control (GNC) sensors and solar panels. In addition to deck space, there is both dedicated external space for payloads as well as substantial internal capacity.

For commonly replaced scientific sensors, two payload bays (A in Fig. 2.2) provide externally-accessible mounting locations. Each bay runs vertically through the entirety of the hull, providing these sensors with access to both the air and the sea surface. The payload bays are both identical, roughly a 0.3m square through the hull, providing a total of 0.1m^3 of internal space.

Each payload bay has enough space for mounting most common oceanographic sensors with few modifications. As a demonstration of this capability, a Sea-Bird Electronics 19plus V2 SeaCAT Profiler CTD has been modified to fit in the payload bay for running scientific missions. This sensor is commonly used for logging vertical profiles with sensors for depth, temperature, and salinity. While the depth sensors rely on pressure, and therefore do not provide any useful information at the surface, the salinity and temperature data collected is presented in Section 5.3. The sensor and its mounting inside the SeaSlug is shown in Fig. 2.3.

2.3 Modular Subsystems

A fundamental aspect of the SeaSlug is its modularity. This stems from the central communications network common to all subsystems. These subsystems

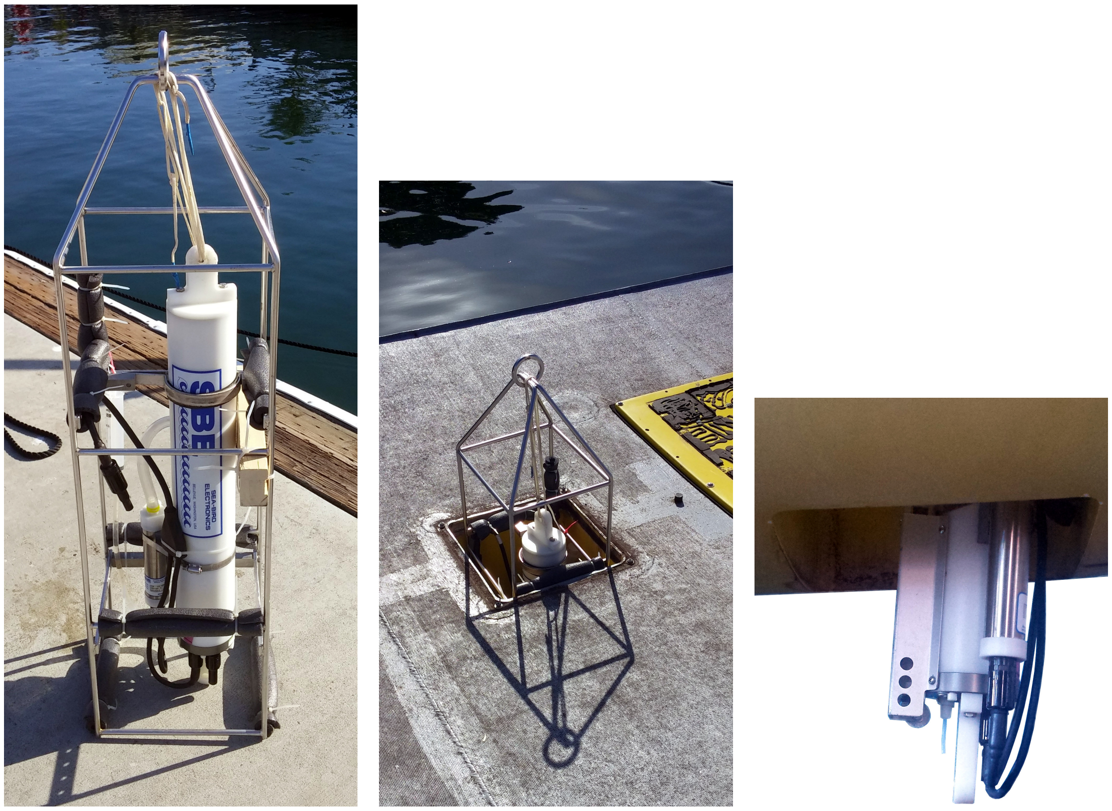

(a) The CTD sensor mounted inside its safety cage with all necessary mounting hardware. **(b)** The top-view of the CTD sensor as installed in the rear payload bay of the SeaSlug. **(c)** The bottom-view of the CTD sensor as installed in the rear payload bay of the SeaSlug.

Figure 2.3: The CTD sensor as used for scientific survey missions.

also have well-defined interfaces over the network: custom components and off-the-shelf components alike were designed or selected to use common interfaces. This provides flexibility in developing and changing the system as well as simplifying maintenance since components can be easily replaced with compatible equivalents.

In order to maximize the modularity of the design, two different power rails were implemented: a 24V and a 12V rail. The 24V battery bank powers the rudder and propeller motors and is powered by four 220Ah 6V sealed lead-acid batteries. The 12V power rail services the remainder of the onboard electronics including the sensors, control electronics, communications, and bilge pump with power provided by a single 98Ah marine gel battery. Additionally both the 12V and 24V voltages are typical of marine electronics, simplifying integration of off-the-shelf components.

Though lead-acid batteries are low cost, they have a relatively low power-to-weight ratio. An explicit goal when designing the SeaSlug was to minimize cost and be easy to maintain. Using common lead-acid batteries facilitates both of those goals. Additionally the weight of the 24V rail's batteries is used as ballast for self-righting maneuvers.

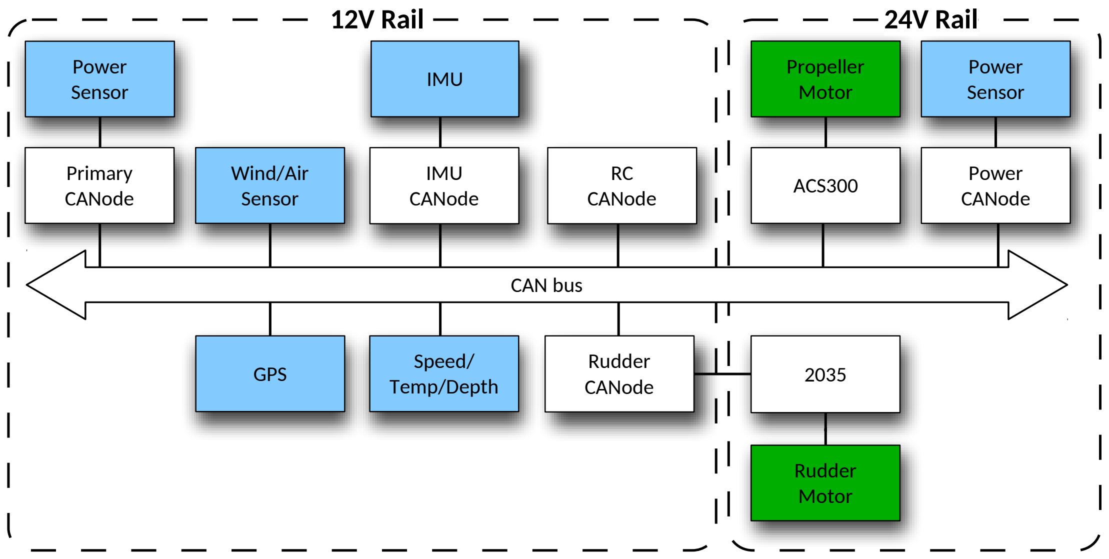


Figure 2.4: Diagram of onboard electrical systems including actuators (dark green) and sensors (light blue) along with their power source.

The central communications network is a Controller Area Network (CAN) bus. This bus provides a robust and high-bandwidth communications layer over a two-wire interface. Additionally it supports any node on the bus serving as the master (transmitting) or slave (receiving) role. This supports a distributed computation network where nodes can perform sensor input, processing, and coordination with other nodes without changing the network architecture.

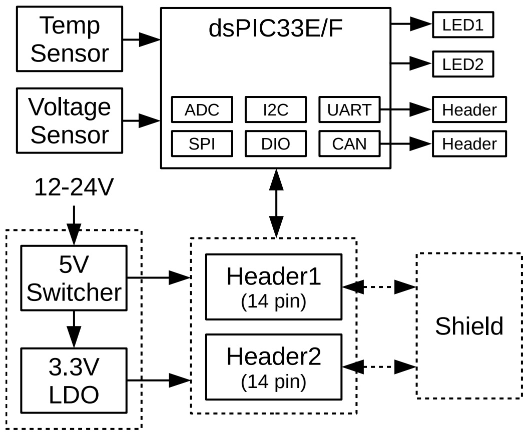
The utility of the CAN bus has been proven by its widespread use in a multitude of common vehicle protocols, including the marine-vessel-oriented NMEA2000 [59]. The NMEA2000 network specifies a number of requirements beyond the base CAN specification including a 250kbit baud rate, unregulated 12V power supply, and a standard set of messages. A more thorough discussion of the CAN and NMEA2000 standards can be found in Appendices C and D, respectively.

The CAN bus on the SeaSlug is compatible with the NMEA2000 protocol and uses the physical connectors and cabling required by the specification. As the NMEA2000 protocol is a superset of the CAN bus, this allows both standard CAN and NMEA2000 devices to be easily integrated into the system. Support for the NMEA2000 specification is highly desirable as supporting devices are rated for harsh marine environments. The internal network cabling includes power to all nodes such that hardware integration of new NMEA2000 subsystems with the SeaSlug is simple, requiring only the standard NMEA2000 connector.

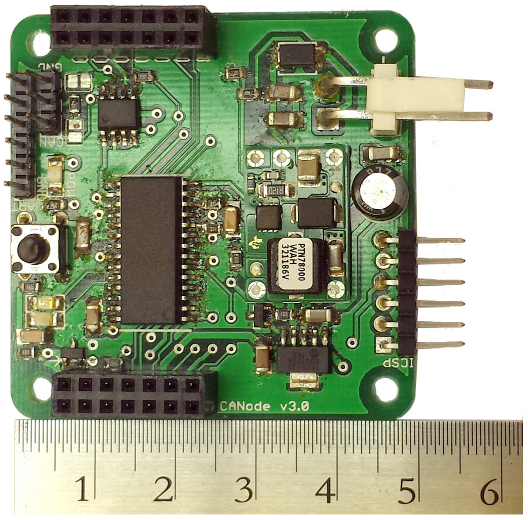
Unfortunately, the NMEA2000 protocol is closed, which requires reverse-engineering for integration of these systems. This is not a significant barrier as the protocol has been sufficiently reverse-engineered, and is merely an inconvenience. The canboat project [18] has developed a partial listing of the details of the entire messageset of the NMEA2000 standard.

2.3.1 The CANode Interface Board

While the onboard CAN bus provides the aforementioned benefits, this interface is not available on all electronics. Some sensors come provided with a bare-CAN or NMEA2000 interface, however many use incompatible interfaces that require translation. Development of a small, single-board computer with the following properties was required: powered from either the 12V or 24V power rail; had sufficient computational capability to run sophisticated control algorithms; and was power efficient so that a large number could be used in the vessel. Additionally, it required native CAN support, and a way to inexpensively extend its capabilities with additional circuitry or connectors.



(a) Block diagram of the architecture of the CANode.



(b) Photo of the topside of Version 3.0 of the CANode.

Figure 2.5: Overview of the Primary CANode.

The CANode was created to address this need, shown in Fig. 2.5. Its physical size is small, measuring only 50mm a side, allowing it to be mounted in a variety of locations. Additionally onboard power circuitry supports both the 12V and 24V voltages used on the SeaSlug. The board supports a variety of 28-pin dsPIC33-family processors that are produced by Microchip. They are low-cost and provide up to 70MIPS of processing power, sufficient for running any necessary translation or control algorithms. Additionally they provide a large number of hardware peripherals that provide important interfaces to external electronics. The interface peripherals used onboard the SeaSlug include:

Serial Peripheral Interface (SPI)

SPI is a high-bandwidth serial interface for short-range communication between integrated circuits, usually on the same board. It is also one of the standards for interfacing with SD cards.

Input Capture (IC)

IC supports decoding input signals where the uptime, downtime, or periodicity of the signal encodes the data. This is common with hobbyist electronics that use pulse-width modulation (PWM), such as incoming signals from an RC receiver.

Output Compare (OC)

The complement to Input Capture, and supports the output of many different types of uptime- and downtime-based signals, such as PWM or pulse-trains. For example, this is used to drive RC servos.

Analog-Digital Converter (ADC)

Interfacing with analog sensors uses the ADC, which decodes the analog signal into a 12-bit number representing a percentage of the possible range of that analog value.

Universal Asynchronous Receiver/Transmitter (UART)

A 2-wire serial interface equivalent to RS232, though at TTL-logic levels (0-3.3V). It can communicate with either UART or RS232/RS485 devices, though the latter require an additional voltage-conversion circuit. Commonly used for high-bandwidth datastreams between separate components over larger distances than SPI can support.

Enhanced Controller Area Network (ECAN)

This peripheral implements all Controller-Area Network (CAN) functionality in hardware, including message filtering and the low-level protocol. The output voltages from this peripheral are not compatible with the CAN bus and additional voltage-conversion circuitry is required. This is commonly done with an integrated CAN transceiver IC (such as Texas Instrument's SN65HVD233).

Digital Input/Output (DIO)

General-purpose digital input or output pins, capable of outputting or reading of either a digital-high (> 2.8V) or digital-low (< 0.8V) signal, useful for transmitting simple true/false values or interfacing with LEDs. All pins in the dsPIC33 family are capable of this.

These peripherals are exposed through two female headers on the board that support daughterboards containing additional circuitry or physical connectors. This follows from the "shield" expansion board pattern popularized by the Arduino platform [6]; the headers expose the processor pins (which support a variety of the above peripherals) and also the 3.3V and 5V regulated power rails generated by the onboard power circuitry. These provide up to 1.0A and 1.5A respectively, which has been more than adequate for all uses aboard the SeaSlug. This generic daughterboard architecture simplifies integration with additional hardware, be it additional circuitry or physical connectors, and has allowed the CANode to be used in most subsystems.

The current version of the CANode is Version 3.0, though Version 2.0 is still used onboard the SeaSlug. They are electrically-equivalent, enabling code sharing between the two versions, but their mechanical layout and support for shields differ. Some nodes use the newer dsPIC33E processors while others are required to use the older dsPIC33Fs. This difference is further explained in Section 2.4. Due to time and funding constraints, some systems aboard the SeaSlug, such as the Primary Node and the Rudder Node, were never updated to the newest version.

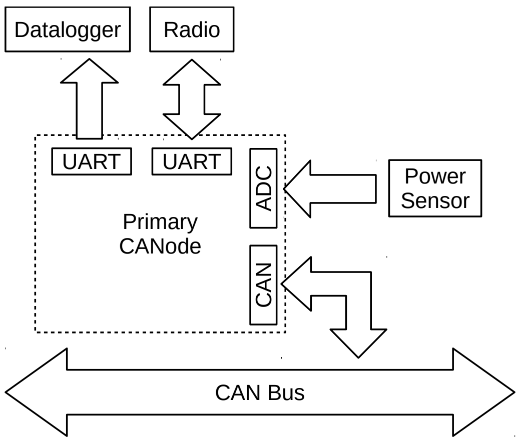
2.3.2 Primary Node

The Primary Node runs the main control algorithm and is the central control computer on the vessel. It receives sensor data over the central CAN bus, processes them in the main autonomous control algorithm, and outputs the resultant actuator commands back onto the bus. Communication with the ground control system (GCS) is also handled by this node over a bidirectional UART communications channel. The Primary Node receives operator commands from and outputs basic telemetry to the PC-based ground control station over this radio link. A separate UART channel outputs a more detailed telemetry datastream that is stored by a datalogger (see Appendix A). This interface is much faster and more reliable than the wireless connection to the GCS. A simple daughterboard is used that provides two additional status LEDs for debugging and additional connectors.

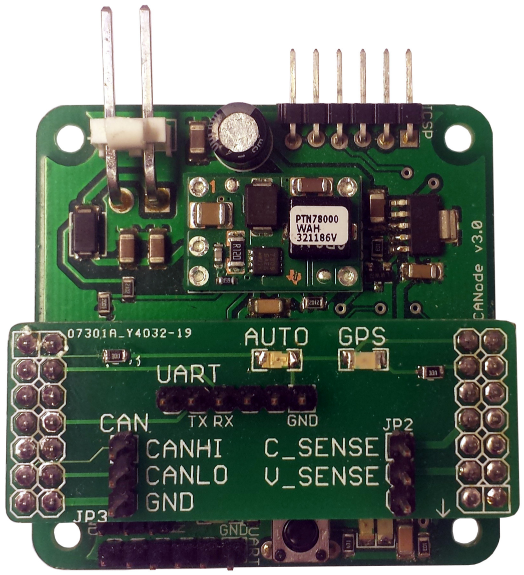
Additionally the Primary Node monitors the power use of the 12V battery banks. The Attopilot Voltage and Current sensor provides TTL-level analog outputs indicating the voltage level (from 0 to 51.8V) and current draw (from 0 to 90A) of the batteries. These analog outputs are directly measured by the Primary Node at 100Hz.

2.3.3 Control Sensors

The most important sensors on the SeaSlug are those providing direct input to the autonomous control algorithm. While some of these sensors are designed for marine use and provide a native NMEA2000 interface, others do not communicate over CAN directly and require a CANode to interface with the bus.



(a) The Primary CANode interfaces with both the telemetry radio and the datalogger over separate UART interfaces. It also reads the power consumption data for the control electronics battery over ADC.



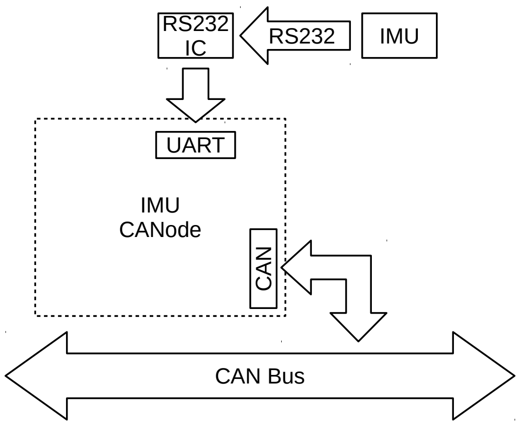
(b) The Primary CANode uses a custom shield so that the external connections can be clearly labeled.

Figure 2.6: Overview of the Primary CANode.

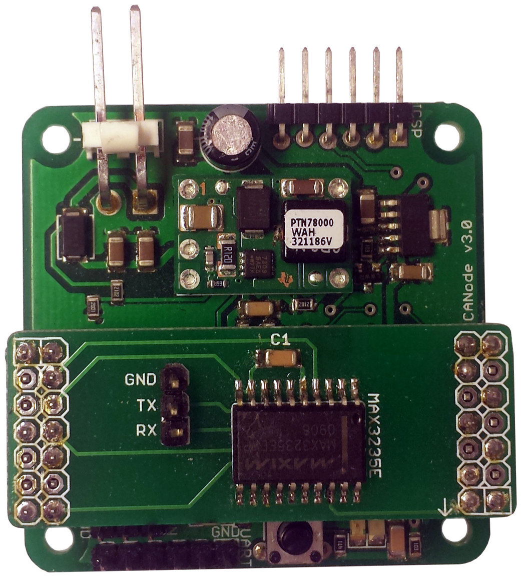
Attitude

The SeaSlug uses the Tokimec VSAS-2GM as its inertial measurement unit (IMU). The Tokimec IMU outputs the vehicles position as 3-2-1 Euler angles in radians and rotation rates as rad/s in the conventional p,q,r body frame coordinates. These representations are further explained in Appendix B. The IMU also provides a bearing to true north, providing an absolute reference for the orientation. The sensor data provided is output in a variety of different coordinate frames, requiring additional processing that is detailed in Section 4.3.

This sensor sits at the very front of the vessel, mounted inside the vessel on the front side of the forward payload bay. This was done to reduce the impact of magnetic noise on its internal magnetometers. It has an RS232 interface, requir-



(a) The IMU CANode converts the RS232 output of the IMU to equivalent CAN messages.



(b) The IMU CANode uses a generic RS232 adapter shield for communication with the Tokimec VSAS-12GM IMU.

Figure 2.7: Overview of the IMU CANode.

ing the use of a CANode for converting its data packets to CAN messages and outputting status notifications.

Global positioning

The position and velocity of the vessel are provided by a global positioning system (GPS) sensor. GPS sensors provide measurements relative to the fixed reference frame of the Earth's surface. The GPS sensor output requires additional processing to detect anomalous data and correct for the sensor offset from the vehicle center, which is described in Section 4.2.

The Maretron GPS200 was used for the SeaSlug's GPS sensor. It is rugged and suitable for a marine environment, which is important since a GPS requires an antenna mounted outside of the vehicle. The GPS200 is a complete unit with an integrated antenna, simplifying installation. It also has an NMEA2000 connection, allowing for easy integration into the system with a single cable.

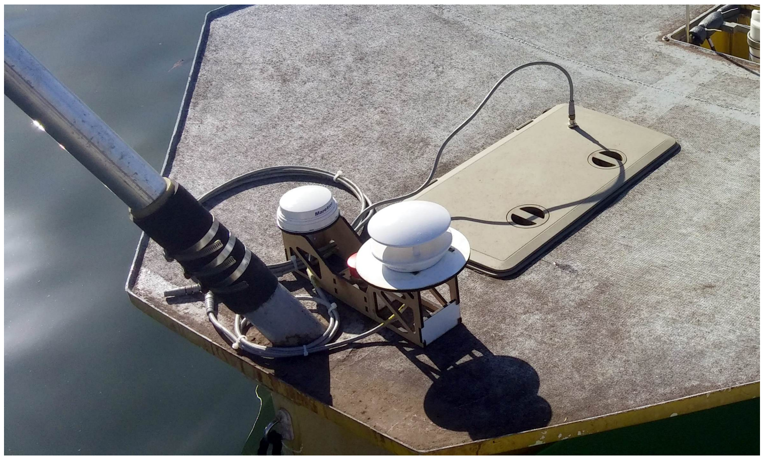


Figure 2.8: The rear of the SeaSlug. The GPS200 is the white sensor on the left, mounted to a scaffold around the emergency-stop button.

Hull speed

The GPS provides vehicle velocity relative to the ground. To determine the speed of the current, it is necessary to determine the actual speed of the vessel through the water. The current is then the difference between this water velocity and the ground velocity. The Airmar DST800 provides the forward hull speed of the vessel, along with some additional data. This sensor is again marine-rated and communicates directly over NMEA2000, simplifying integration.

2.3.4 Actuators

The SeaSlug relies on two actuators for control and movement: a rudder provides steering and a propeller provides propulsion. These subsystems are integrated with the CAN bus over which they receive commands and output their status. Each actuator also has their own interface electronics that maintain the last received command.

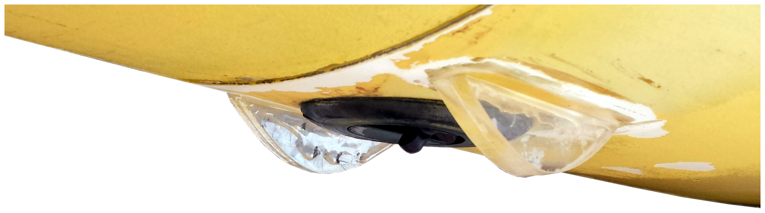
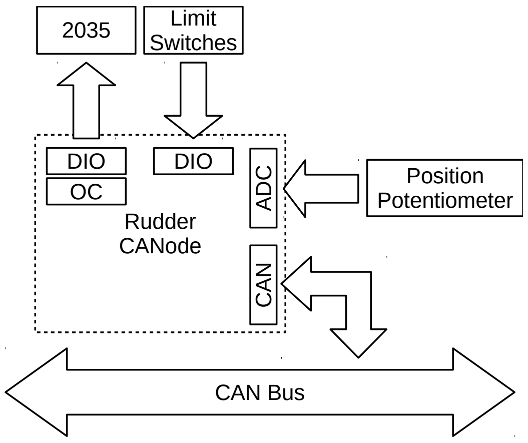


Figure 2.9: The DST800 hull speed sensor as mounted underneath the hull at the front of the SeaSlug. The black line in front of the sensor is the outline of the forward sensor bay. The acrylic pieces next to it protect the sensor when loading and unloading the SeaSlug from its trailer.

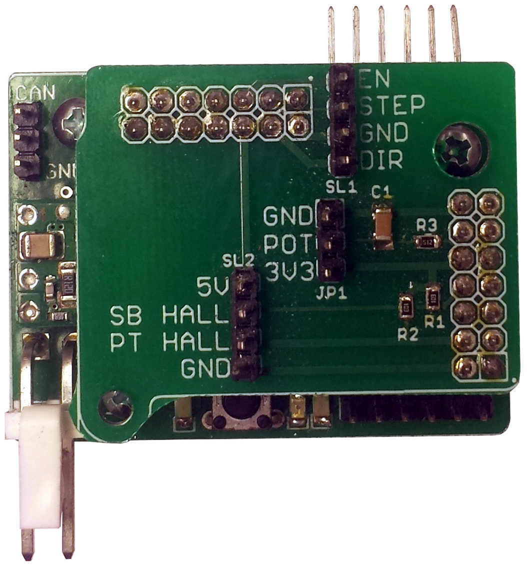
Rudder

The rudder controls the vessel's rotation, or yaw rate. It is driven indirectly by a stepper motor through a gear-and-chain system and can command a range of ±45°. A stepper motor is used because it can electronically hold its position and is also easy to drive use standard off-the-shelf driver boards. A servo motor provides similar benefits, but with the stepper motor it is easier control the exact rudder angle as it corresponds to an exact number of steps. By using a motor to hold the rudder position, the system has some compliance, yielding if it receives a significant impact. This prevents damage to the rudder and the hull in the case of a collision, although it comes at the cost of increased power usage to hold the rudder in position.

The rudder is driven at 24V by an optically-isolated motor driver board. This board is set to drive the motor at 2A while dropping down to a holding-current of 1A after 1.0s without any new control inputs. This current is sufficient to hold the rudder in place while also halving power use.



(a) The Rudder CANode reads the rudder's limit switches read as digital inputs with the position potentiometer read by the ADC peripheral. Motor commands are output with both general digital output pins and the Output Compare (OC) peripheral.



(b) The Rudder CANode uses a custom shield so that the external connections can be clearly labeled.

Figure 2.10: Overview of the Rudder CANode.

A CANode, powered by the 12V electronics power rail, connects to this motor driver board through opto-isolated inputs. It provides the CAN interface to this motor driver board such that the desired rudder angle can be commanded. It translates from the received command into the necessary driver board inputs, controlling the rudder to maintain the commanded angle. The node updates at 100Hz, with the rudder position detected by an analog potentiometer and two Hall-effect limit switches at each end of its range. While using an analog potentiometer is noisy and requires calibration, it is simple and low-cost. Calibration is done by sweeping between the two limit switches, recording the range of the potentiometer values. Over this range, the results are reasonably linear and provide an accurate reading of the rudder deflection angle to $\pm 0.09°$.

The CANode drives the rudder at 333Hz, at the high end of the possible range for both the motor and the driver board. With the rudder control loop update rate of 100Hz and the motor half-step angle of 0.0602°, this results in an actual controllable rudder angle resolution of 0.2°. A small deadband of ±1.72° was found to prevent oscillations around the desired angle, although it further limits the rudder's effective resolution. This loss of resolution can result in a constant offset in the rudder angle which must be handled by the autonomous controller. A properly-tuned control algorithm is robust to bias offsets, however, and small angles are handled with minimal error.

Propeller

Propulsion is provided by the propeller, which is directly driven by a brushless permanent-magnet synchronous motor (PMSM). The ACS300 motor driver board is connected to the 24V actuator power rail, and can push 15A to drive the motor in either direction. It has a native CAN interface for receiving commands and transmitting both status and motor speed.

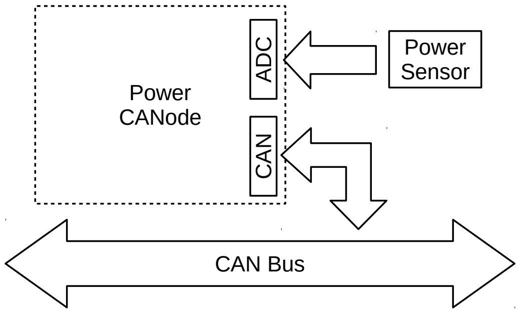
The propeller is commanded by setting the maximum current that the ACS300 should supply to the motor. At maximum throttle, the motor rotates at about 200rpm, which matches the designed optimal speed for the propeller.

2.3.5 Miscellaneous

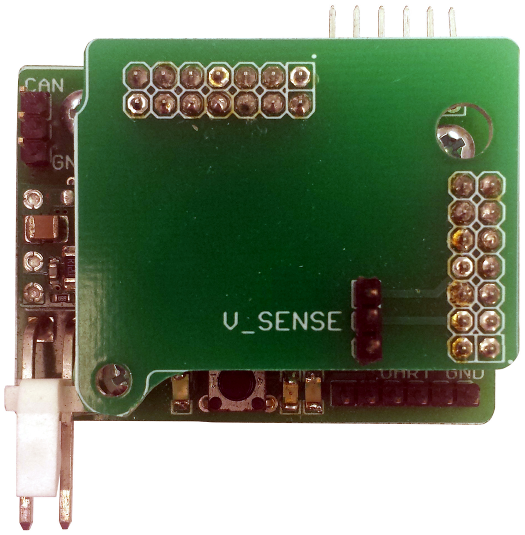
There are additional onboard subsystems that publish data to the CAN bus, but are not essential for autonomous control of the vessel. Their details are described below.

Power

As for the 12V battery bank, the power draw from the 24V battery bank is also monitored. This relies on a dedicated CANode that reads the output of an additional Attopilot Voltage and Current sensor at 100Hz and then outputs these values onto the CAN bus at 10Hz. The product of these two values provides an estimate of the total power use of the rudder and propeller motor.



(a) The Power CANode interfaces with the power sensor using the analog-to-digital conversion peripheral.

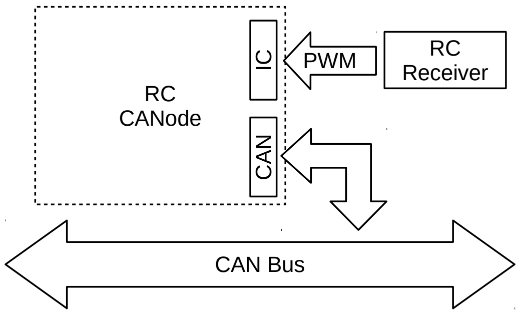


(b) The Power CANode only uses digital input pins, but uses a custom shield so that the connection can be clearly labeled.

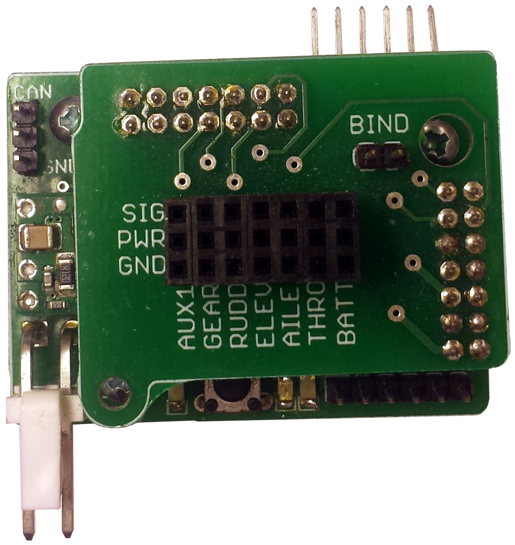
Figure 2.11: Overview of the Power CANode.

Backup Control

The RC Node is capable of controlling the SeaSlug's actuators directly, similar to the Primary Node. It provides a backup control interface for the human operator in case of a hardware or software failure in the system. The operator uses a dedicated hardware radio transmitter to communicate with the RC Node through a radio receiver. The receiver outputs PWM signals corresponding to the controller inputs. These signals are then interpreted by the CANode and converted to the appropriate rudder and throttle commands. This is further described in Section 2.5.2.



(a) The RC node uses the Input Capture peripheral to decode the pulse-width-modulated signals received from the RC receiver.



(b) The RC node uses a custom shield that provides a female interface for connecting the RC receiver directly.

Figure 2.12: Overview of the RC CANode.

Wind & Air

A Maretron WSO100 sensor provides wind and air readings. This is not a calibrated scientific sensor, and while not directly used by the onboard controller, it is valuable for analyzing the weather during a mission. This sensor uses ultrasonic sound waves to detect wind motion and has no moving parts thus requiring less maintenance than mechanical anemometers. This sensor is temporary mounted at the rear of the boat to the right of the GPS unit, as shown in Fig. 2.8. The permanent mounting location will be the top of the mast, above the radar reflector.

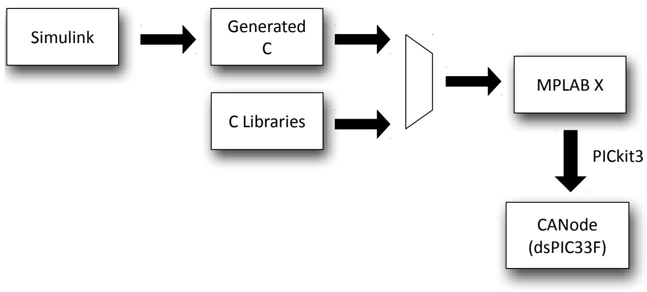
2.4 Embedded Firmware

All software for the SeaSlug has been optimized for ease-of-use and modularity. This required a priority on using open-source software and open standards where possible in additional to a development environment that would allow for rapid experimentation. Most of the onboard software is written in C. Some additional code is written in Simulink, which then generates C code for the final compilation and programming step. The development process changes slightly depending on whether the core firmware is written in C or Simulink with the differences highlighted in Fig. 2.13.

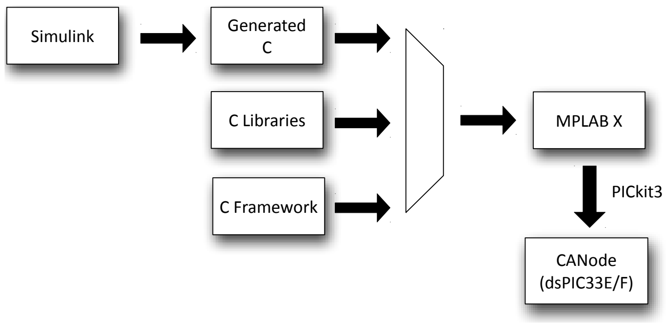
While C provides the lowest-level access to the hardware functionality of the dsPIC33 microcontroller, Simulink was used to speed up development of complex algorithms. Simulink was designed to simplify development, simulation & testing, and deployment of data flow algorithms, such as the control algorithms. Additionally its large standard library includes high-level time- and frequency-domain tools. The advantages of this development toolchain has been noted previously in [56], [66], and [58].

Two separate compilation toolchains are used for developing the firmware for the different CANodes on the SeaSlug. The primary toolchain uses custom C code for everything but the control algorithms. Those algorithms are written in Simulink and then transpiled automatically to C with the Simulink Coder tool. The generated C code is then called from a handwritten framework that includes the processor initialization code and main event loop.

The other toolchain relies on a Simulink library, the dsPIC Blockset by Lubin Kerhuel [49], to generate the processor initialization code and the main event loop as well as all control algorithms and peripheral driver code. This can be augmented with custom C code for additional functionality. The blockset was originally used



(a) Using the dsPIC Blockset for the CANode programming procedure. The blockset handles CANode initialization and core program functionality and can call into external C libraries.



(b) Programming procedure for using Simulink's code generation tool. CANode initialization and core program functionality such as the main event loop is handwritten in the C Framework.

Figure 2.13: Code compilation and CANode programming using mixed Simulink and C code.

for rapid prototyping as it possessed a useful feature in its ability to target a variety of processors. This was a compelling property prior to the development of the CANode. Unfortunately, limitations in the hardware peripheral drivers provided by the blockset resulted in more development moving to custom C code. While these drivers were good for prototyping or debugging purposes, they lacked adequate features or acceptable performance for a near-production system.

Over the course of this project several drivers were rewritten in C. In the end, the functionality provided by the blockset was reduced to the processor initialization code and the main event loop. This was not of sufficient benefit to warrant the extra complexity of the blockset and some nodes were converted to using custom C code for their initialization and the main event loop. This conversion was done for all nodes with the exception of the Rudder and RC Nodes, which are still reliant on the blockset for some of their peripheral drivers. This is an eventual goal for these subsystems, but was not completed due to time constraints.

2.5 Remote interface

There are two independent remote interfaces for the SeaSlug, as shown in Fig. 2.14. The main control interface is the ground control system (GCS), a tablet computer running the QGroundControl software [95]. A wireless radio link connects it to the Primary Node on the SeaSlug.

The second interface provides only manual control using a Remote Control (RC) transmitter. This interfaces with the SeaSlug through a separate wireless radio connection with the RC Node. The RC controller provides redundancy in case the primary fails. The details of this redundancy are further explained in Section 2.7.

2.5.1 Ground Control System

The GCS is comprised of both a hardware system and the control software running on it. The computer hardware is a Microsoft Surface Pro 2 tablet PC running Windows 8.1. Its low power use enables it to be used for several hours of sustained operation. Additionally, it generates little heat and can therefore be safely sealed in a waterproof enclosure.

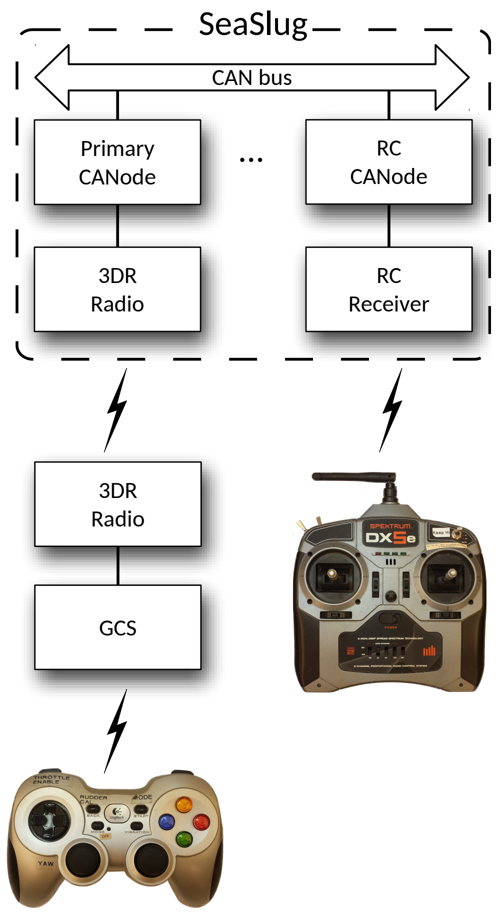


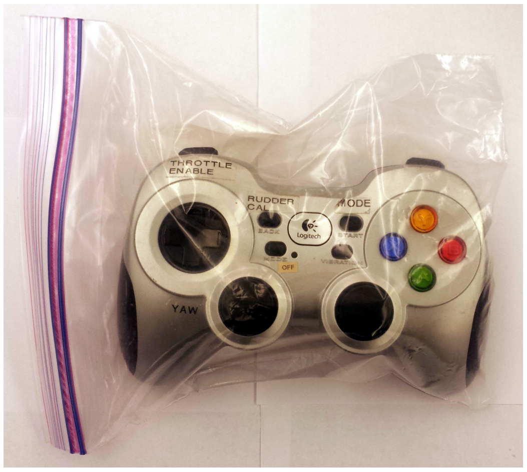
Figure 2.14: The remote interfaces to the SeaSlug. A groundstation provides a wireless interface, including manual control, to the vessel. A secondary controller, completely independent of the primary controller, provides an emergency backup. Lightning bolts indicate wireless connections.

User input occurs through a pen interface, allowing for continued use of the device even when its enclosure is wet, assuming that the touchscreen has been disabled. The 27.5cm screen is both high-resolution and bright enough (400nits) to be usable in direct sunlight.

A wireless connection to the SeaSlug uses the 915MHz 3DR Radio Set. This long-range radio can provide up to a 1.6km range if line-of-sight is maintained. Its low price and open-source firmware make it both easy to replace and modify as necessary. Interfacing with the radio is either through a USB serial port or over UART directly, making integration with both ground control software and embedded microcontrollers simple.

A Logitech F710 wireless gamepad, which has its own radio connection to the tablet through a proprietary radio receiver, provides a manual control interface to the human operator. The decision to use a wireless controller based on the observation that it can be easily weatherproofed, as shown in Fig. 2.15b.

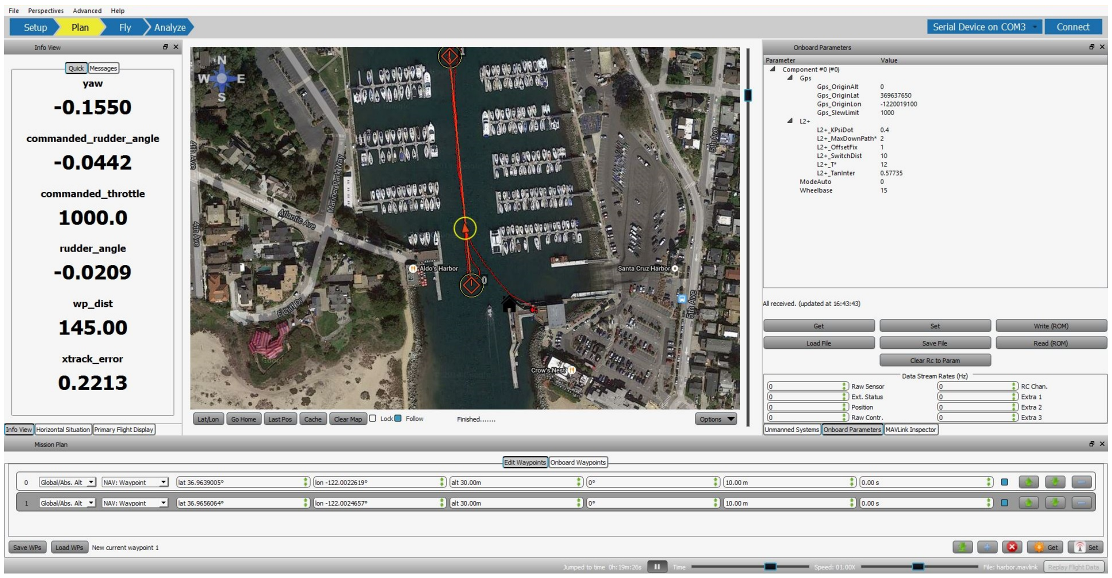
(a) The water-proofed GCS hardware. The tablet is enclosed in a thick zipper-locked plastic bag and protected by a custom foam enclosure that also provides buoyancy. The pen allows for user input and is secured to the tablet.



(b) The Logitech F710 gamepad used as the primary controller. A 1-quart resealable zipper storage bag is used to provide weatherproofing.

Figure 2.15: The Ground Control System hardware provides the remote interface to the SeaSlug.

The software interface to the SeaSlug uses the QGroundControl (QGC) ground station software [95]. QGC is open-source, cross-platform, and supports many different types of autonomous systems simultaneously. It has been implemented in C++ and is reliant on the Qt library, facilitating custom development. It is supported by its own development community, however, which continually develops new features, fixes bugs, and provides suppport.



yaw
-0.1550
commanded_rudder_angle
-0.0442
commanded_throttle
1000.0
rudder_angle
-0.0209
wp_dist
145.00
xtrack_error
0.2213

Parameter	Value
Component #0 (#0)	
Gps	
Gps_OriginAlt	0
Gps_OriginLat	369637650
Gps_OriginLon	-1220019100
Gps_SlewLimit	1000
L2+	
L2+_KPsiDot	0.4
L2+_MaxDownPath*	2
L2+_OffsetFix	1
L2+_SwitchDist	10
L2+_T*	12
L2+_TanInter	0.57735
ModeAuto	0
Wheelbase	15

All received. (updated at 16:43:43)

Data Stream Rates (Hz)

0	Raw Sensor	0	RC Chan.
0	Ext. Status	0	Extra 1
0	Position	0	Extra 2
0	Raw Contr.	0	Extra 3

0	Global/Abs. Alt	NAV: Waypoint	lat 36.9639005°	lon -122.0022619°	alt 30.00m	0°	10.00 m	0.00 s
1	Global/Abs. Alt	NAV: Waypoint	lat 36.9656064°	lon -122.0024657°	alt 30.00m	0°	10.00 m	0.00 s

New current waypoint 1

Jumped to time 0h:19m:26s Time Speed: 01.00X File: harbor.mavlink Replay Flight Data

Figure 2.16: QGroundControl running during an autonomous test in the Santa Cruz harbor. Shown are status variables (left), a map with the vehicle's current and past position and mission waypoints (center), and onboard parameter settings (right). The current mission details are shown at the bottom.

QGroundControl's feature list is extensive including the following:

- Crash-tolerant data logging
- Common avionics-based visualization aids
- Audio output of notable events and status changes
- Parameter editing
- Multi-vehicle control
- Offline map interface with mission planning
- Direct manual control through attached controllers

Of this list, the text-to-speech interface bears special mention. Using this interface QGC can rely information to the operator over audio. It has built-in support for several audio cues, such as when a waypoint has been reached and when the system state changes. Additional audio cues can be triggered by the unmanned system and read aloud as well. The SeaSlug uses this capability to alert the operator of its distance to the next waypoint and crosstrack error every 30s.

When monitoring the vessel during autonomous missions, the audio interface is sufficient for the operator to rely on it exclusively. This significantly reduces the ground station's power consumption and heat generation, both of which can limit mission duration. It also allows the operator to maintain visual observation of the vehicle at all times.

QGroundControl uses the MAVLink [62] library for its communication protocol. MAVLink is a general-purpose, message-based protocol, but specifically targets communications with unmanned systems. In addition to defining a general encoder and decoder as a core library, MAVLink also defines a common message set for remote vehicle communication. This common message set is used by QGC, making it operable with any vehicle that implements the same message set.

MAVLink defines this common message set with a standard XML syntax and is easily extended with custom messages creating separate dialects. A SeaSlug dialect is used that adds extra debugging and sensor data messages. This dialect supports the common message set and remains compatible with QGC. These message sets are automatically translated into a header-only C library that is used by both QGC and the autopilot itself. All messages can be both transmitted and received as long as QGC and the autopilot are compiled with the same MAVLink dialect.

2.5.2 Radio Control Transmitter

An additional interface has been implemented as a backup to the SeaSlug's primary GCS interface. The backup interface is minimal, supporting only rudder calibration, rudder control, and propulsion control. This is substantially more limited than the GCS interface, but in an emergency it is all that is required to safely operate the SeaSlug. The transmitter, an off-the-shelf Spektrum Dx5e that is commonly used by hobbyist RC airplane pilots, is paired with a Spektrum 6100e receiver. They have a range of about 200m and consume very little power. The hobbyist RC transmitter and receiver market is somewhat standardized and these components can be easily swapped with equivalent components from a variety of manufacturers without necessitating any changes to the RC Node.

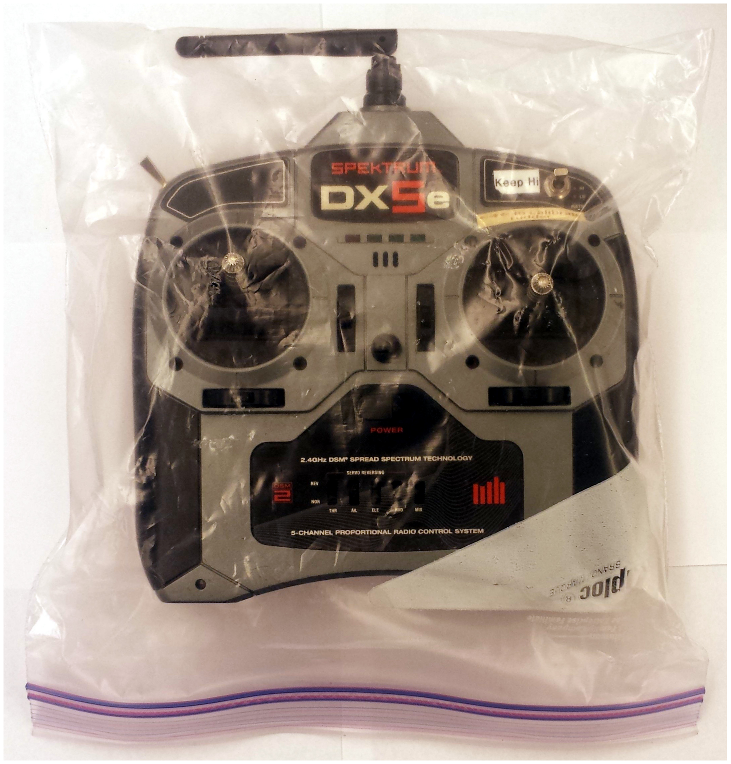

SPEKTRUM
DX5e
Keep Hi
POWER
2.4GHz DSM SPREAD SPECTRUM TECHNOLOGY
SERVO REVERSING
5-CHANNEL PROPORTIONAL RADIO CONTROL SYSTEM

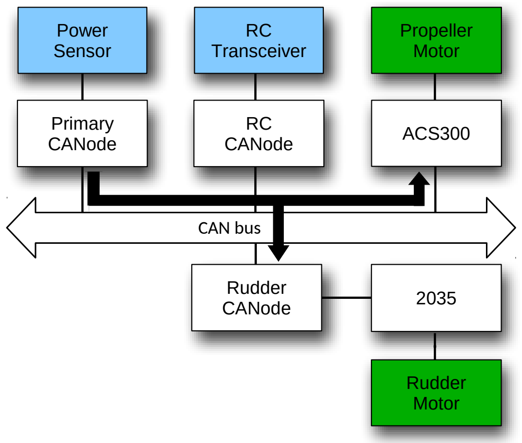
Figure 2.17: The Spektrum DX5e Radio Control transmitter used as the backup controller. A 1-gallon resealable zipper storage bag is used to provide weather-proofing.

The electrical diagram in Fig. 2.4 shows the RC Node that translates from the PWM input of the RC receiver into the CAN messages that control the propeller and rudder. This method bypasses the primary controller entirely so that it relies on the minimum amount of hardware and software, thereby reducing the chance of failure. This backup interface has proven reliable (and necessary) over the course of both testing and live missions. Many of these situations arose through operator error, such as the GCS running out of power or being accidentally disabled.

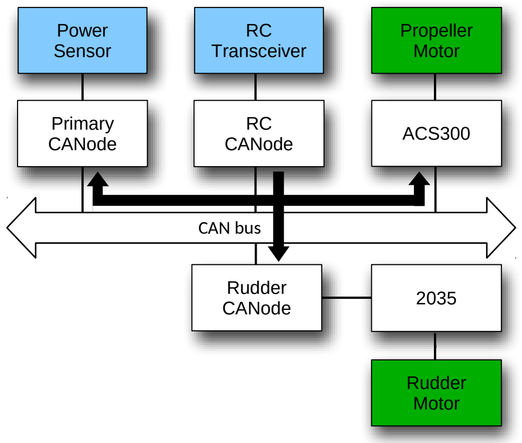
Under normal operation, the vehicle is guided by the primary controller, using either its own internally-generated actuator commands in autonomous mode or by forwarding the manual commands received from QGC. During manual override, when the transmitter is on and enabled, the RC Node broadcasts actuator commands directly. When the Primary Node detects these messages on the CAN bus, it enters a reset state and no longer transmits commands. In this way, the manual override is able to take control in all situations independent of actuator or bus failures. Fig. 2.18 shows the data flow in both the normal and manual override situations.

2.6 Mission Capabilities

Configuring missions for the autonomous controller is through the *waypoint* mission element. It dictates both a coordinate frame and three position coordinates in that frame: X, Y, and Z. In the global coordinate frame, waypoints are defined by latitude, longitude, and altitude above mean sea level (MSL). In the local frame, which is fixed relative to a position on the Earth's surface, the position coordinates are instead north, east, and down (relative to MSL). An additional local frame supports specifying position coordinates relative to the current vehicle position, though they are in the same local frame as above. This relative waypoint



(a) During normal autonomous or manual control modes the Primary Node commands the rudder and propeller.



(b) Under emergency manual control, the RC Node commands the rudder and propeller. The Primary Node detects this automatically and disables itself.

Figure 2.18: Flow of CAN actuator commands. Note that while all devices on the bus receive every message, most are ignored. The arrows indicate the broadcasting and receiving nodes.

mission type has proven useful in specifying a fixed waypoint pattern that can be run anywhere, as done for several of the missions run by the SeaSlug.

2.7 Safety, Fault Tolerance, and Error Recovery

Since the SeaSlug is comprised of a myriad of different systems, there is always the possibility of any one system experiencing some type of failure. The probability of this happening is increased by the harsh environment the SeaSlug experiences, which includes large temperature changes, high humidity, and salt water. While the backup controller has already been described, additional features were implemented to facilitate detecting and recovering from system errors.

Each CANode on the system has a unique identifier and transmits a *Node Status* message (Appendix E). These provide basic error and status reporting to both the RC and Primary Nodes. This information is supplemented by each controller keeping track of the last messages received from each node in order to determine if that node is enabled or active. *Enabled nodes* are powered and broadcasting CAN messages and *active nodes* are in a normal operating state (no errors/faults). This information is relayed to the operator through QGC to facilitate debugging. Additionally, several nodes use both the amber and red LEDs of the CANode to indicate their status and error states as well (though this is really only useful when testing the system in simulation as they are not visible during live deployments).

While many subsystems exist on the CAN bus, a core set is required for autonomous control and the loss of any of these is fatal for the autonomous controller. When an error occurs in one of these systems during autonomous control, the system enters a *Fault* mode, which means that the rudder is centered and then disabled and the propelled is immediately disabled, therefore drawing no power and asserting no force. Though the rudder centering after a fault could be problematic in certain scenarios, this behavior is highly desirable as it makes the motion of the now-adrift vessel much more predictable, which simplifies any necessary recovery operations. Recovery from this mode can only be accomplished by the operator switching to manual mode and then back to autonomous mode. A safeguard prevents enabling the autonomous mode if any core subsystems are not enabled and active. This prevents the system from ever being in an uncontrollable state, further reducing the chance of operator error. The GCS also announces when the *Fault* mode has been triggered and cleared.

This *Fault* mode can also be manually triggered by pressing the emergency-stop button located at the rear of the vessel. Its location near the mast (visible in Fig. 2.8) makes it easy to trigger with a boat hook or by driving alongside with the chase boat and pressing it directly.

The aforementioned *Fault* mode also triggers if the vessel is disconnected from the GCS for more than thirty seconds. This prevents the vehicle from operating in an unknown mode, as the operator would be blind to its internal state without the GCS. Thirty seconds was determined to be a long enough time that this functionality does not accidentally trigger during a normal operations, but is also short enough that the system can not go very far without direct supervision.

During development, some missions experienced actuator failure that was not handled appropriately by the system. The system could not recognize that the actuators were stalled and manual intervention was required to disable the system. To address this issue the propeller and rudder are now constantly monitored for appropriate behavior during system operation. The ACS300 motor driver board faults if the propeller stops rotating unexpectedly or other electrical faults are detected in the motor. The Rudder Node continually evaluates the rudder position, faulting if the rudder is not moving in the expected direction at at least half of the expected speed (this accounts for possible missed steps due to heavy wave action). In both of these situations the actuators are disabled and the *Fault* mode is triggered, warning the operator.

In order to prevent errors while under autonomous control, certain operator commands are disabled when in this mode. This includes initiating calibration procedures, altering of mission waypoints, and changing certain system parameters. In manual mode all system capabilities remain available to the operator.

2.8 Conclusion

The architecture of the SeaSlug is unique compared to other electric ASV projects. Its keeled fiberglass monohull with large deck space requires little power for propulsion while still providing ample room for solar panels. Additionally, the hull design provides for a large payload capacity to support a multitude of different scientific sensors for data collection missions.

The use of a central CAN bus for all onboard communications forced a clear segregation of the many onboard components into well-defined modules. The development of the CANode supported these subsystems as self-contained black-boxes that only exist as a small interface to the rest of the system. While the detailed architecture of the SeaSlug is complicated in detail, reasoning about individual subsystems or the interaction between subsystems becomes simple. It is this system-wide modularity that has allowed this complex system to be easily extensible, maintainable, and reliable during operations.

Chapter 3

Simulation

3.1 Introduction

The development of the SeaSlug relied extensively on both manual and automated testing to speed development. This was partially to solve logistical issues, as even a short boat launch involved several people and took several hours. Testing also serves to reduce developer error and to provide system verification and validation at every level of the system. The testing process involved first the individual algorithms, then individual subsystems, and finally the system as a whole. This form of *test-driven development* is necessary for any large and complex system as there are a myriad of potential points of failure.

A common problem with many existing autopilots is their simulation environment. They can be difficult to set up initially, unreliable during simulation, and problematic when extending the simulation to other aspects of the system. For unreliable simulators, a failure necessitates the conservative action of grounding the vehicle for safety even though it might be the simulator itself at fault and not the vehicle. Therefore simulator reliability can have a large impact on system uptime.

The SeaSlug simulation environment is designed to be both robust and easy to use such that simulation failures most often correspond directly to actual system failures. This allows the developer to spend their time correcting system problems and not debugging the simulation environment itself. To facilitate test-driven development, the developer must *want* to run tests; the easiest way to instill this desire is to make the simulator both reliable and effective.

Realistic mathematical models of the environment and vessel behavior in that environment were developed in support of an effective simulation environment. These models are neither novel nor controversial, but reproduce the real-world system's response in sufficient detail to enable testing and debugging of the system's control algorithms.

The mathematical models supporting software simulation and the overall simulation architecture are described in this chapter. Design decisions were made to make simulation testing easy and effective, and these are discussed and justified where appropriate.

3.2 Simulation Model

The simulation model incorporates three major systems: the vessel's kinematics, the actuators' dynamics, and the external environment. This section describes their derivation and implementation in the simulator. The goal of the simulator is to replicate the environment and the vehicle's response in enough detail for testing control algorithms. The mathematical models that underlie the simulator are therefore the simplest models that can still provide a reasonable approximation of reality, and are both robust and effective in testing.

3.2.1 Vehicle Kinematics

The kinematics of a vehicle describe its motion given its current state. This explicitly excludes any forces and moments acting on the vessel. The kinematics of the SeaSlug are known as the *bicycle model.* This simple model is derived from the geometry of circular motion, as shown in Fig. 3.1.

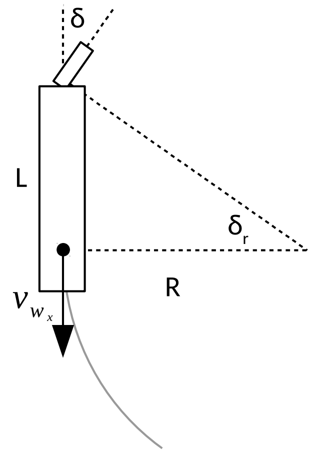


Figure 3.1: The bicycle model defines the vehicle's motion around a circle of radius R as a function of its wheelbase, L, and the rudder angle, δ_r. v_{w_x} is the forward water speed of the vessel.

The bicycle model describes a vehicle that can control its forward speed with steering capabilities at one end, as described in [4]. The distance from the steering actuator to the center of rotation of the vehicle is the governing parameter for this model, referred to as the *wheelbase*, L. From the geometry, the relationship between the rudder angle (δ_r), wheelbase (L), and resultant turn radius (R) are:

$$\tan \delta_r = \frac{L}{R} \tag{3.1}$$

A vehicle undergoing uniform circular motion, with a tangential speed v_{w_x} has a turn rate $\dot{\psi}$:

$$\dot{\psi} = \frac{v_{w_x}}{R} \tag{3.2}$$

Eq. 3.1 and Eq. 3.2 can be combined by using the curvature of the circular path induced by the vehicle's turn (of radius R) as a common term. The resultant equation equates the turn rate of the vessel and the rudder angle:

$$\dot{\psi} = -\frac{v_{w_x} \tan \delta_r}{L} \tag{3.3}$$

The rudder angle is constrained on the SeaSlug to ±45° and therefore the singularity that the tangent function has at ±90° is irrelevant. The negative sign is necessary because of how the coordinate frames of the rudder angle and yaw rate are defined: a positive rudder value induces a negative yaw rate.

The water velocity of the boat is modeled as a linear gain on the commanded throttle value, δ_t:

$$v_{w_x} = 1.9\delta_t \tag{3.4}$$

This differs from the actual throttle-to-velocity mapping, shown in Fig. 3.2, but is sufficient for simulation. The only requirement for the throttle mapping is that it spans the same range of possible water speeds as the actual vessel (0 to 1.9m/s).

The final part of the vehicle kinematics model is the vehicle position as a function of the current vehicle state. The change in the vehicle's north and east position arises from its heading and forward speed:

$$\dot{\begin{bmatrix} N \\ E \end{bmatrix}} = \begin{bmatrix} v_{w_x} \cos \psi \\ v_{w_x} \sin \psi \end{bmatrix} \tag{3.5}$$

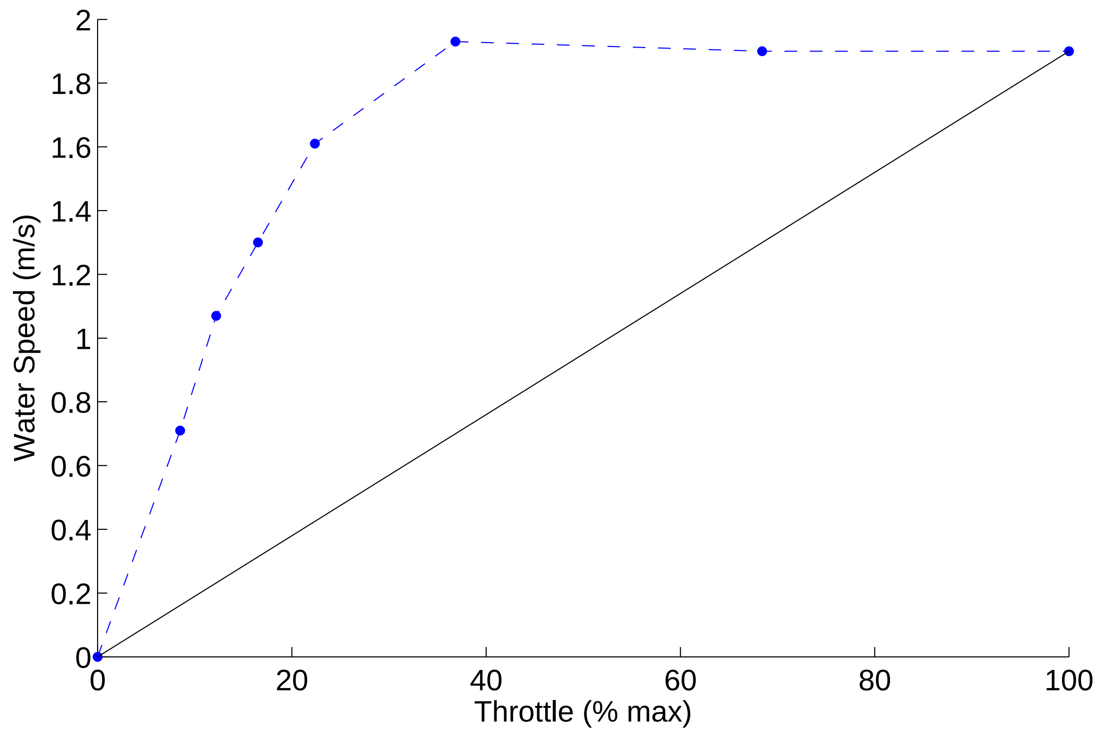


Figure 3.2: Vessel velocity through the water as a function of commanded throttle percentage from experimental data. The dashed-blue line shows real-world measurements and the solid-black line shows the mapping used in simulation.

where N and E are the north and east position coordinates of the vessel, v_{w_x} is the forward water speed, and ψ is the heading of the vessel.

This model assumes no side-slip or other hydrodynamics; v_{w_y} is assumed to be zero. This assumption fails in reality, though these unmodeled forces are relatively small and can be safely ignored for simulation purposes.

Combining equations 3.3, 3.4, and 3.5 together yields the complete equations of motion for the vessel, shown in Eq. 3.6. These equations have only a single vessel-dependent parameter, L, the distance between the center of rotation of the vessel and the center of effort of the rudder.

$$\overset{\bullet}{\begin{bmatrix} N \\ E \\ \psi \end{bmatrix}} = \begin{bmatrix} v_{w_x} \cos\psi \\ v_{w_x} \sin\psi \\ -\frac{v_{w_x} \tan\delta_r}{L} \end{bmatrix} \tag{3.6}$$

$$v_{w_x} = 1.9\delta_t$$

To determine the wheelbase value, a least-squares fit of recorded telemetry was used to determine L. The resultant wheelbase parameter was found to be 5.8789m. A plot of the expected yaw rate versus the measured yaw rate during harbor testing is shown in Fig. 3.3.

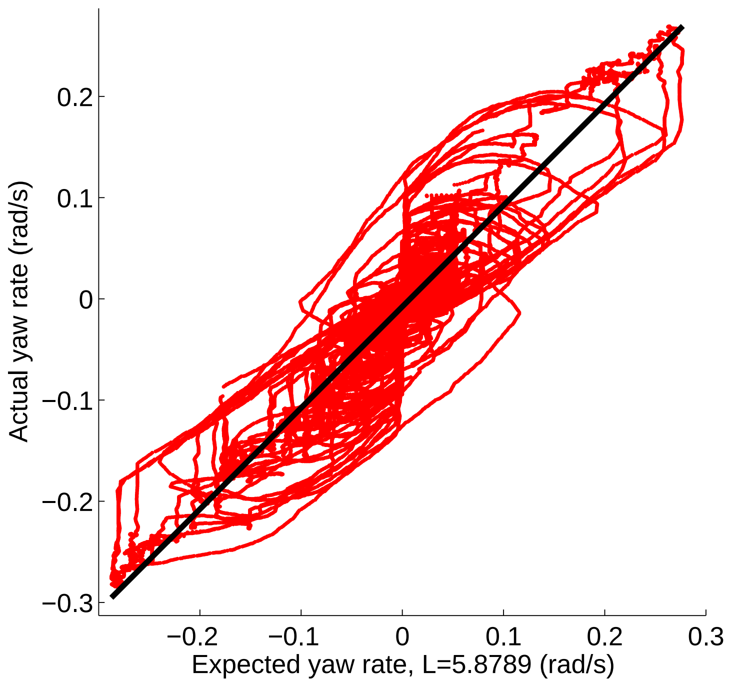


Figure 3.3: A plot of the turning rate as a function of both rudder angle and water velocity during harbor testing (red) against the modeled turning rate (black line). The black line is a least-squares fit of the data with slope 1.0005 and y-intercept -0.0079.

In Fig. 3.3 the actual and expected yaw rates should be identical if the model was exactly correct with the real world. The fit line (shown in black) should therefore have a slope of 1 and a y-intercept of 0. In reality the model does not exactly fit reality, though it tends to follow the expected model. This is shown by the fit line having a slope of 1.0005 and a y-intercept of -0.0079.

The wheelbase distance was measured on the SeaSlug and found to be 3.27m, smaller than the least square fit suggests it is. This difference is unsurprising as all unmodeled kinematics appear in the model within the wheelbase value. That it is larger than the measured value shows that the unmodeled forces are predominantly drag forces and reduce the turn rate.

3.2.2 Actuator Dynamics

This model assumes instantaneous action on the rudder, which is known to be false. Thus, the rudder dynamics were modeled and integrated into the simulator as they are important to the overall vessel dynamics. This actuator model closely matches the anticipated dynamics given the architecture of the rudder subsystem.

A stepper motor drives the rudder motor at a fixed step speed imposed by the hardware. The Rudder Node controls the motor using a bang-bang controller where the rudder is either moving at a constant rate or is held stationary. There is an additional delay in the system from when the controller transmits a command to when the rudder starts responding. The final model diagram is shown in Fig. 3.4.

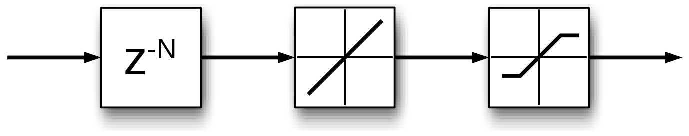


Figure 3.4: A block diagram of the rudder dynamics model. First is a discrete time delay of 0.08s, then a slew limit is imposed of 25.78°/s, and finally a saturation limit keeps the rudder within the range of ±45°

The saturation limit is set to the mechanical limit of the rudder, ±45°. Its turn rate has been measured as a consistent 25.78°/s. This turn rate is also independent of water speed, which is expected given the high power output of the rudder motor. The delay between transmitting a rudder command and rudder movement has also been measured and is consistently 0.08s.

A comparison of the rudder model with the actual rudder dynamics are shown in Fig. 3.5. This figure shows a 44s segment of a longer 1.23hr live test. During the entirety of the test the rudder model closely matched the actual rudder response with a mean error of 0.38° and a standard deviation of 0.64°.

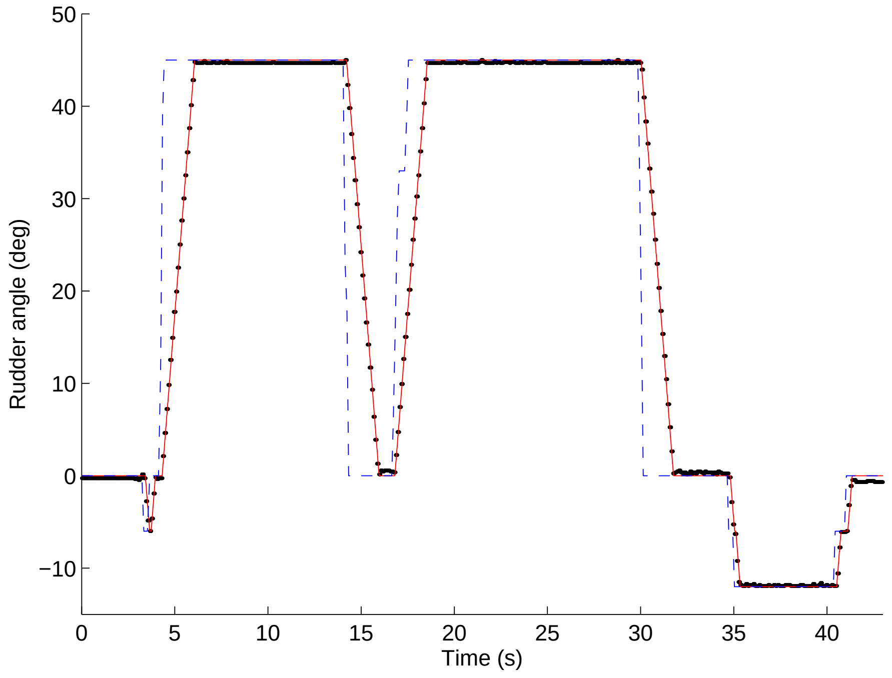


Figure 3.5: A comparison of the actual rudder dynamics (black dots) to the simulated dynamics (red line). The commanded rudder angle is shown as the blue dashed line.

No dynamics are used for the throttle in simulation. Changes to the throttle are therefore propagated immediately in the simulator and the vehicle's speed always matches the throttle value. Because the throttle values are not changed during an autonomous run, the controller has not needed to handle these changes and so these dynamics were not measured or implemented in the simulator.

3.2.3 Environmental Effects

The simulation model has been extended beyond the vessel kinematics to include very basic environmental effects. These effects are implemented as a single global velocity affecting the vehicle. This constant velocity term is referred to as the water current, v_c, though it can be used to approximate the cumulative effect of all external forces including the wind, waves, and currents. This single term cannot capture all of the dynamics of wave and wind motion, but it has proven effective for approximating external currents for testing. When added to the kinematic model of the vessel, the final equations of motion are shown in Eq. 3.7.

$$\overset{\bullet}{\begin{bmatrix} N \\ E \\ \psi \end{bmatrix}} = \begin{bmatrix} v_{w_x} \cos\psi \\ v_{w_x} \sin\psi \\ -\frac{v_{w_x} \tan\delta_r}{L} \end{bmatrix} + \begin{bmatrix} v_{c_n} \\ v_{c_e} \\ 0 \end{bmatrix} \tag{3.7}$$

$$v_{w_x} = 1.9\delta_t$$

3.3 Software Simulation

Simulink is used as the simulation environment for the SeaSlug. Both the vehicle's controller and the system plant are implemented as Simulink models, which forms the basis of the simulator. The overall development and testing procedure with the SeaSlug is shown in Fig. 3.6.

The first phase of testing the SeaSlug, shown on the left in Fig. 3.6, focuses solely on the control algorithms. These are tested in a software-only simulation environment on a PC with both the vehicle controller and the environment run in

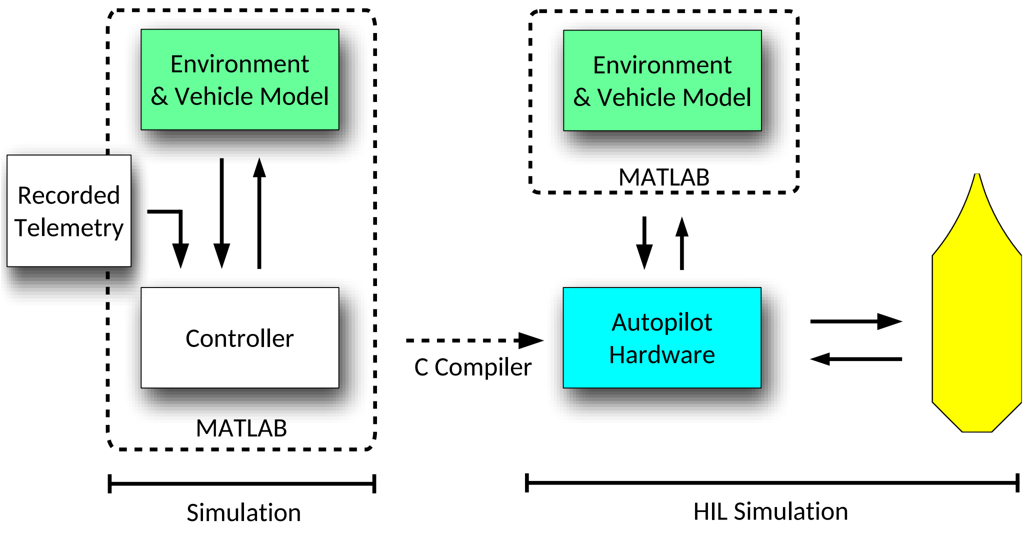


Figure 3.6: The three stages of the testing and development process: software simulation, hardware-in-the-loop with the controller running on the embedded hardware, and possibly connected to other subsystems of the SeaSlug.

Simulink. Only the initial conditions of the vessel and the environment are configurable, including its mission, starting position, orientation, and the global water current; there is no runtime input. This is by design as it reduces the complexity of the simulation and restricts testing exclusively to the vehicle controller.

The control algorithms on the Primary Node operate at 100Hz. The simulation environment updates the environment and sensors at 100Hz instead of a higher rate that better emulates the real world. During testing this higher rate was found to have no noticeable difference in either the simulator or controller output. As running the simulator at a smaller timestep would only serve to increase execution time of the simulation, the simulator has been set to a fixed time step of 0.01s.

At every simulation timestep the controller commands are processed by the plant and translated into sensor data, with the same sensor output as the onboard sensors. The interfaces between the different components of the simulator match those of the SeaSlug to provide a realistic simulation environment. Both the datatypes and sample rates of data are configured such that they are identical when run in simulation on the 64-bit Simulation PC or onboard the 16-bit dsPIC33 microcontrollers. This acts as the first *sanity check* during simulation, where even with the performance and numerical precision restriction of the microcontroller the system operates as expected.

3.3.1 Replay Simulation

An additional simulation environment has been created for analyzing system performance during missions after they have been run. An onboard datalogger (see Appendix A) records all of the inputs, outputs, and some intermediate calculations of the controller at every execution timestep of the controller. This data can then be played back by the *Replay Simulator* through the system's control algorithms to generate new rudder commands. This has proven valuable for testing how different parameters affect the control outputs and for analyzing sensor irregularities. While this simulator can only be run open-loop, it has provided valuable insight into controller behavior.

3.4 Hardware-in-the-loop Simulation

The software simulation environment is limited in its testing capabilities because the actual hardware of the system is not used. Additional problems can arise when the control algorithm is compiled onto the Primary Node and the Primary Node is integrated with the other subsystems on the CAN bus. Testing the system with simulated sensor data while using hardware components of the actual system is known as *hardware-in-the-loop simulation* (HIL).

While HIL simulation is an essential part of the development process for complicated systems, it is especially important for the SeaSlug. Substantial time and manpower is required for a single day of testing and losing a day to integration bugs is expensive. Therefore HIL simulation is a core component of both the SeaSlug's development and pre-launch test procedure, as shown in Fig. 3.6.

In HIL the Primary Node is interfaced with the Simulink simulation model, which provides a simulated environment with the same sensor outputs as the hardware sensors. In this way, all system functionality is identical to how the vessel is normally operated during a mission. The system architecture when running HIL simulations is shown in Fig. 3.7.

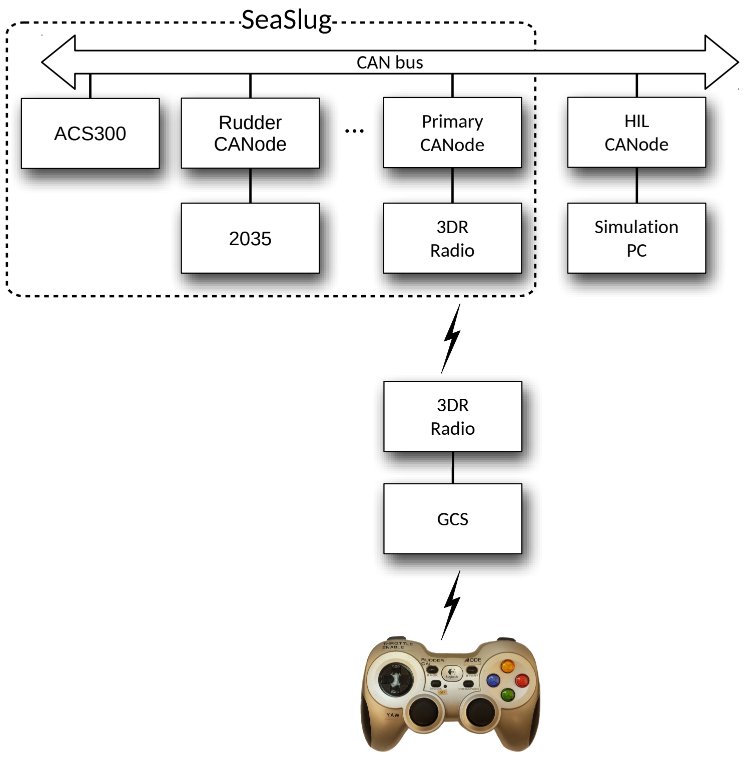


Figure 3.7: The system architecture during HIL simulation. Any combination of non-essential sensor nodes can be connected during testing, such as the rudder or propulsion subsystems.

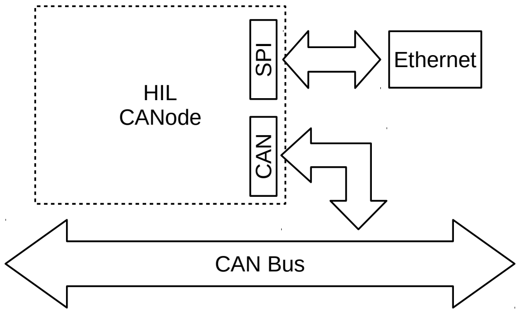
HIL testing is highly configurable and different subsystems can be connected and operating during simulation. HIL only requires that the Primary Node is connected to run the controller and to communicate with the GCS. The simulator emulates the necessary sensor inputs: the GPS, IMU, and water speed sensors. Non-essential subsystems can also be connected during simulation. Their output does not affect the simulation, but other nodes can process their data as if the system is operating on the water.

The actuator outputs do however affect the simulator. When the interface board detects the output from the actuators it forwards that data to the simulator. In this mode the real actuator outputs are fed back to the simulator as input to the simulation models, allowing the actuators, complete with their real dynamics, to be tested in drydock. When the actuators are missing from the CAN bus the simulator emulates their dynamics. It is therefore possible to test all subsystems except the navigation sensors in HIL simulation. The ability to test the entire system, including the actuators, has been especially important when testing the system's failure modes and implementing fault tolerant logic.

3.4.1 HIL CANode

Simulation flexibility is the result of an HIL interface board that communicates with both Simulink and the CAN bus of the SeaSlug. This board emulates the CAN messages output by the onboard subsystems, adapting to the testing configuration automatically based on which nodes it detects. In this way, no subsystems are configured differently for HIL simulation, nor are they even aware that such functionality exists. This allows for exact testing of the system as it operates with the real sensors during deployments. The architecture of this node is detailed in Fig. 3.8.

A general-purpose Ethernet shield has been designed for the CANode that provides a single physical Ethernet port connected through Microchip's ENC28J60 Ethernet IC. Onboard firmware relies on Microchip's DHCP, UDP, and IP li-



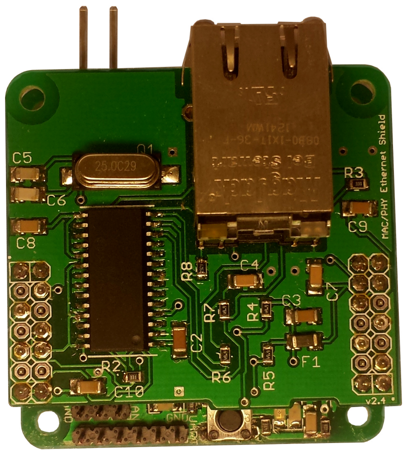


Figure 3.8: The HIL high-level architecture. This node converts between CAN and UDP interfaces with an Ethernet IC connected over SPI.

braries, which are already configured to work with the ENC28J60 IC. For convenience, the HIL Node is set up as a server, automatically configuring any connected computer so that no manual setup is required.

Communication with the simulator is done after every timestep (0.01s) and results in a total of 64kb/s of bandwidth. The Ethernet port is set to operate at its lowest speed of 10Mb/s, as this is sufficient to transmit all necessary data and reduces power use slightly. These messages consist of the data packed with a header, footer, and checksum. The exact format is shown in Fig. 3.9. While UDP supports checksumming packets directly, this is unsupported by Microchip's UDP library, and is instead implemented within the message instead. An additional checksum field provides this validation as a 1-byte XOR of all bytes in the Data field. Messages that fail to pass checksum validation are ignored.

%	&	Data (big-endian)	^	&	Checksum (1-byte)

Figure 3.9: The wire-format of the UDP packets used for communicating with Simulink. The data field consists of smaller integers, floating-point numbers, etc. packed in big-endian format with no padding.

The HIL Node utilizes a split architecture to process both incoming messages and to schedule outgoing messages. Both the UDP packets from Simulink and the CAN messages from the Primary Node are received and processed in real-time. A 100Hz event loop is used for scheduling both the CAN and UDP output messages. The output message rates use a global message scheduler to transmit at specific rates, which match the exact transmission rates of the subsystem being emulated. Details on the messages and their transmission rates are in Appendix E.

3.5 Conclusion

The primary goals of the simulation environment were to provide a sufficient vehicle and environmental model to support software simulations and full-system testing during development and before launches. The kinematic and dynamic models developed in Simulink are close enough to reality to support debugging and testing of the control algorithms, as shown by the data in this chapter.

Though pure-software simulation is effective for testing the control and navigation algorithms, the complexity of the SeaSlug necesitates HIL testing as well. In HIL testing the control algorithms are running on the embedded hardware and in effect entirely unaware that they are not controlling the physical vessel. Further the HIL environment was created such that any individual hardware subsystem can be connected to the CAN network and will also respond as if on the physical vessel. This is facilitated by the HIL CANode which provides the connection between the CAN bus and the Simulink simulation environment. It supports testing all aspects of the system, including manual control, before launching the vessel.

With the robust simulation environment presented in this chapter all subsystems of the SeaSlug can be evaluated in support of continued development.

Chapter 4

Control Architecture

4.1 Introduction

The SeaSlug is capable of a variety of missions in littoral waters. This environment can be difficult to navigate because of large external forces such as wind, waves, and current. Controlling most sea vessels is further complicated by their non-holonomic nature (they cannot move in an arbitrary direction). Most boats cannot move sideways, but can move forwards/backwards and rotate. The onboard control algorithms command the system's actuators to compensate for these disturbance forces and track a desired path, where this path is usually specified as a series of waypoints.

The data collection systems that oceanographers use vary widely in their trajectory-following capabilities. Manned vessels are commonly off course by tens to hundreds of meters, even when relying on an autopilot. Underwater gliders can be off by hundreds of meters because they lack a global reference system and rely primarily on the less-accurate inertial navigation while underwater.

The SeaSlug aims for better performance than either of these systems, targeting a path following accuracy of a few meters. While this performance is unnecessary for most current oceanographic data collection missions—since there is high tolerance for course-tracking errors—this level of performance opens up new possibilities in oceanographic research including navigating narrow waterways such as shoals, harbors, or inland waters.

This chapter describes the onboard guidance, navigation, and control software of the SeaSlug as two major processes: the processing of sensor data and the generation of rudder angle commands. The throttle command is open-loop and remains constant during an autonomous mission with speed control a focus of future work (see Section 7.2).

4.2 Position Filtering

The primary sensor for the SeaSlug is its GPS, which outputs a global position as latitude, longitude, and altitude (though the altitude portion is discarded as the SeaSlug does not leave the ocean's surface). Using these spherical coordinates directly for control is both unnecessary and computationally expensive. Instead, these global coordinates are converted to a local coordinate frame through the multi-step process illustrated in Fig. 4.1.

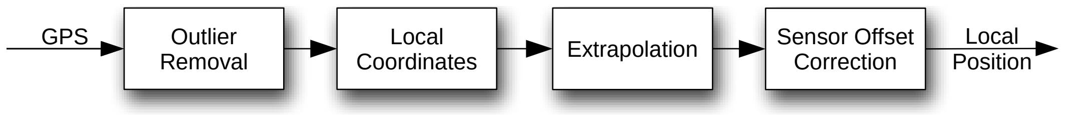


Figure 4.1: The filtering process for position data. Input is directly from the GPS with the output a position in the local coordinate frame.

4.2.1 Outlier removal

The first step in processing the GPS data is to remove any obviously incorrect data from the input data stream. This pre-filter prevents errors in further stages from operating on incorrect data. Most of this incorrect data is from GPS readings where there is not enough satellite coverage to give an accurate reading.

On open waterways, a clear view of the sky can be assumed to always be available, so invalid positions are rare with the exception of GPS start up. During this start up process the precision of the readings can change as different satellites are used in the solution. At startup all GPS readings are discarded until five consecutive GPS readings are of high enough accuracy for position sensing. This is indicated by the GPS unit as a “3D” fix.

After this initial startup very few GPS errors occur. These rare, transient errors are sometimes simple to detect as the reported position jumps back to the origin (0,0), which is off the coast of Nigeria. However, there are other instances where the sensor values suddenly jump to another position that is not the origin. Anomalies in position occur for a variety of reasons, including a change in the satellite constellation used to produce a fix. The loss of a key satellite can result in sub-optimal geometry for the multilateration algorithm that determines the vessel’s position.

To filter out these errors a slew limit is imposed on position updates after startup. While it would be possible to use a multiple of the vehicle’s water or ground speed, using a fixed value has proven more robust. This limit is not used to smooth between position readings that may be incorrect, but to eliminate clearly erroneous ones. As such it merely needs to be set higher than the vehicle could ever possibly move in its environment and lower than what the erroneous readings are.

A fixed limit of 11m/s was used. This was set sufficiently high so that towing the vessel or possible wave action would not accidentally trigger it. This filtering is done directly on the spherical coordinates so it is set as $0.0001°/\text{s}$. Again, instead of being a pure slew limit on position changes, it is actually a cutoff for discarding the GPS data point entirely. This data is discarded because it is very likely that position updates that violate the slew limit are invalid readings.

4.2.2 Converting to local position

The next stage in position filtering is to convert the global coordinates of latitude/longitude to a simpler coordinate frame, one defined in meters North and East. By orienting this frame along the latitude and longitude lines, north and east respectively, the system's position becomes more intuitive and easier to use in calculations.

Most systems affix a two-dimensional Cartesian plane tangent to a reference point on the Earth's surface for their local coordinate frames, known as the local tangent plane (LTP), such as the plane in Fig. 4.2. For missions covering small regions of the Earth's surface, using a single reference position with the LTP is certainly accurate enough. For the SeaSlug, however, a curvilinear surface approximating the curvature of the Earth at a reference point is used.

Reference Ellipsoid

The first step of mapping a GPS reading into the local coordinate frame is to define the shape of the Earth. The best models use a flattened ellipsoid, as shown in Fig. 4.3 This shape has only two parameters, that of the semi-major axis length a and the semi-minor axis b. For use as models of the Earth's surface, the origin of the ellipsoid is usually geocentric, defined as the center of Earth's mass.

The most common reference ellipsoid is the WGS84 standard [27]. It defines the Earth's ellipsoid with parameters shown in Table 4.1. While there are more accurate models, such as the North American Datum (NAD) and Ordnance Survey

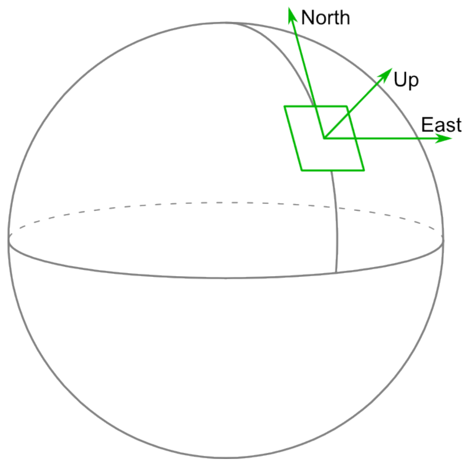


Figure 4.2: The local tangent plane shown on the Earth's surface (represented as a sphere). On this plane, coordinates are defined in East-North-Up (ENU) coordinates. North-East-Down (NED) is another coordinate system commonly used with the LTP. Only an ENU frame is shown here for simplicity.

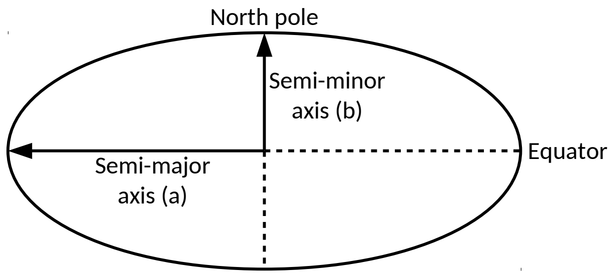


Figure 4.3: An ellipsoid, such as that defined by the WGS84 parameters. An ellipsoid is entirely defined by its semi-major axis a and semi-minor axis b.

Great Britain (OSGB), they are all region-specific, with NAD is specific to North America and OSGB for Great Britain. The difference in accuracy between these models is often smaller than the application requires and the convenience of a global model outweighs these inaccuracies; the SeaSlug GNC system uses the WGS84 model.

It should be noted here that this ellipsoid only defines a very simple approximation of the Earth's surface. The ellipsoid is corrected by the geoid, the Earth's surface as defined by equal gravitational potential. This is necessary because Earth's gravity is not constant across its surface. This difference between the

a	semi-major axis	6378137
b	semi-minor axis	6356752.3142
e^2	first eccentricity squared	6.69437999014e-3

Table 4.1: The parameters for the generalized Earth ellipsoid as defined by the WGS84 standard. Only two of these parameters are necessary for defining the WGS84 ellipsoid using Eq. 4.5, but all are shown here for completeness.

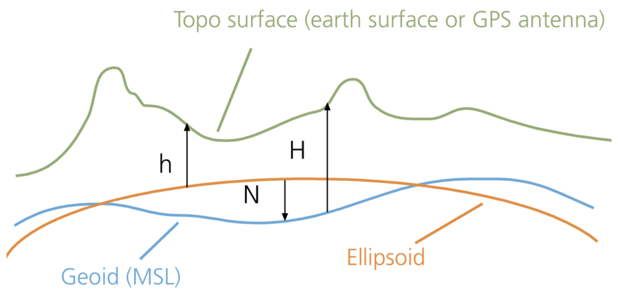


Figure 4.4: Reproduction of part of Figure 2 from [35]. Shows the difference between the geoidal and ellipsoidal models of the Earth's surface and its actual topography.

ellipsoid and the geoid is the *geoidal separation* as shown in Fig. 4.4. WGS84 also defines a high-resolution model of this geoid so that the GPS-calculated altitude can be converted to the more useful and intuitive mean sea level (MSL). This can have a significant difference in reported altitude as the range of geoidal separations defined by WGS84 are from -105m to +80m.

Converting to Local Coordinates

With the reference ellipsoid defined by the WGS84, The GPS-reported latitude and longitude can be converted to a position on the local curvilinear surface on that ellipsoid as shown in Fig. 4.5. This coordinate frame still uses a reference point as the origin, much as the local tangent plane does, with the position being in meters relative to this origin.

This curvilinear surface was chosen because it is more accurate than the local tangent plane and is still computationally efficient; the final implementation re-

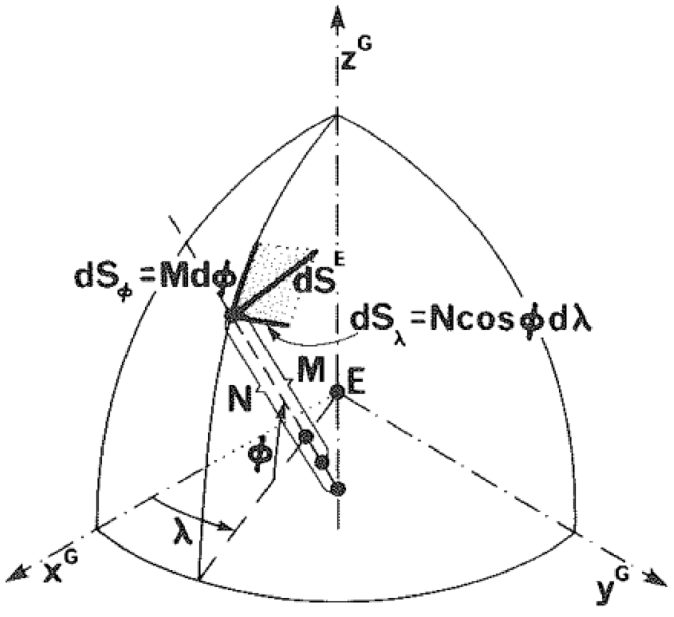


Figure 4.5: Figure 16.17 reprinted from [87]. Shows the local position, denoted dS, of the vehicle from a reference location in the local curvilinear space. This is split into the north-component dS_ϕ and the east-component dS_λ.

quires three multiplications, a cosine term, and a subtraction, all of which is done at the full fixed-point precision of the GPS data.

Because the curved surface of the ellipsoid is used as the local coordinate frame, determining the relative position of the vessel from a reference location is straight-forward: the local position of the vessel is the arclength of the ellipsoid along both the north and east directions, referred to as S_ϕ and S_λ respectively (where ϕ refers to latitude and λ refers to longitude). These distances are integrals of the curvature at every position along the direction as shown in Eq. 4.1. $M(\phi)$ and N are curvatures, known as the *meridian radius of curvature* and *prime vertical radius of curvature* respectively. They are defined in Eq. 4.2 and Eq. 4.3, where ϕ is latitude and λ is longitude and e and a are the parameters of the reference ellipsoid in Table 4.1.

$$\begin{aligned} S_\phi &= \int_{\phi_i}^{\phi_j} M(\phi) d\phi \\ S_\lambda &= \int_{\lambda_i}^{\lambda_j} N \cos(\phi) d\lambda \end{aligned} \tag{4.1}$$

$$M(\phi) = \frac{a(1-e^2)}{\sqrt[\frac{3}{2}]{1 - e^2 \sin^2 \phi}} \tag{4.2}$$

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi_{ref}}} \tag{4.3}$$

Both of these equations for curvature are derived from the geometry of an ellipse. The *prime vertical radius of curvature* is derived in Section 15.4 of [87], though there it is defined differently, as shown in Eq. 4.4. The geometric relationship between these radii can be seen in Fig. 4.5.

$$N = \frac{a}{\sqrt{a^2 \cos^2 \phi_{ref} + b^2 \sin^2 \phi_{ref}}} \tag{4.4}$$

This is actually equivalent to Eq. 4.3 using the relationship between the defining parameters of an ellipse from Eq. 4.5.

$$e^2 = \frac{a^2 - b^2}{a^2} \tag{4.5}$$

In the longitudinal direction, there are only constant terms inside the integral, and it simplifies as shown in Eq. 4.6. This is because at a fixed latitude, the curvature of the reference ellipsoid is constant.

$$\begin{aligned} \int_{\lambda_i}^{\lambda_j} N \cos(\phi) d\lambda &= N \cos(\phi) \int_{\lambda_i}^{\lambda_j} d\lambda \\ &= N \cos(\phi)(\lambda_j - \lambda_i) \end{aligned} \tag{4.6}$$

Rewriting this for clarity, the east/west axis of the curvilinear space can be calculated from Eq. 4.7.

$$S_\lambda = N\cos(\phi)\Delta\lambda \tag{4.7}$$

The latitudinal axis integral, on the other hand, is more complicated because the ellipsoid's curvature changes along this axis. Therefore a simplifying assumption is made that the curvature is a constant, fixed to the value at coordinate frame's reference position. Given the endurance of the SeaSlug, the territory covered in the latitudinal direction for a single mission is under 1000km, making this a reasonable assumption.

By fixing the prime radius of curvature as the curvature at the reference point, it becomes a constant, and calculating the arclength across latitudes simplifies to:

$$S_\phi = M(\phi_{ref})\Delta\phi \tag{4.8}$$

The final equations implemented on the SeaSlug are shown in Eq. 4.9, where $M(\phi_{ref})$, N, ϕ_{ref}, and λ_{ref} are all constants, with only (ϕ, λ) varying as the vehicle moves.

$$\begin{aligned} north &= M(\phi_{ref})(\phi - \phi_{ref}) \\ east &= N\cos(\phi)(\lambda - \lambda_{ref}) \end{aligned} \tag{4.9}$$

Therefore the calculation of $M(\phi_{ref})$ and N can be done offline at compile-time. The run-time calculations necessary are therefore very inexpensive, requiring only two subtractions and three multiplications.

4.2.3 Position extrapolation

The GPS sensor only provides position and velocity updates at 4Hz, though the onboard controller operates at 100Hz. Instead of limiting the command rate of the vessel to 4Hz, the position is extrapolated between GPS updates. This extrapolation is done in the tangent curvilinear frame. This simplifies the calculations as the vehicle position is in the same units as the reported GPS velocity.

This extrapolation assumes that the GPS-reported velocity of the vehicle is constant between updates. This assumption holds because of the slow acceleration of the vessel, even during turns, combined with the short time between GPS updates. The final position of the vehicle is the sum of the last GPS position update and the current integral of the ground velocity. When a new update is received from the GPS, the position is updated and the integral reset to 0. These new values are not blended or smoothed into previous position estimates which results in discontinuities in the vehicle position estimates. The onboard control algorithms have proven robust to these discontinuities and further improvements have been deferred to future work.

Fig. 4.6 shows a comparison of the GPS's reported latitude and the corresponding north position as calculated in the local curvilinear coordinate frame. The latitude signal is low-frequency, only 4Hz, and that updates the position used by the extrapolation code, which provides a vehicle position estimate at the controller frequency, 100Hz. Sometimes these GPS updates are not always received (seen between 12s and 14s in Fig. 4.6); velocity integration continues through several missed updates. Even in that case, however, the extrapolation is quite accurate, matching up well with new updates when they are finally received.

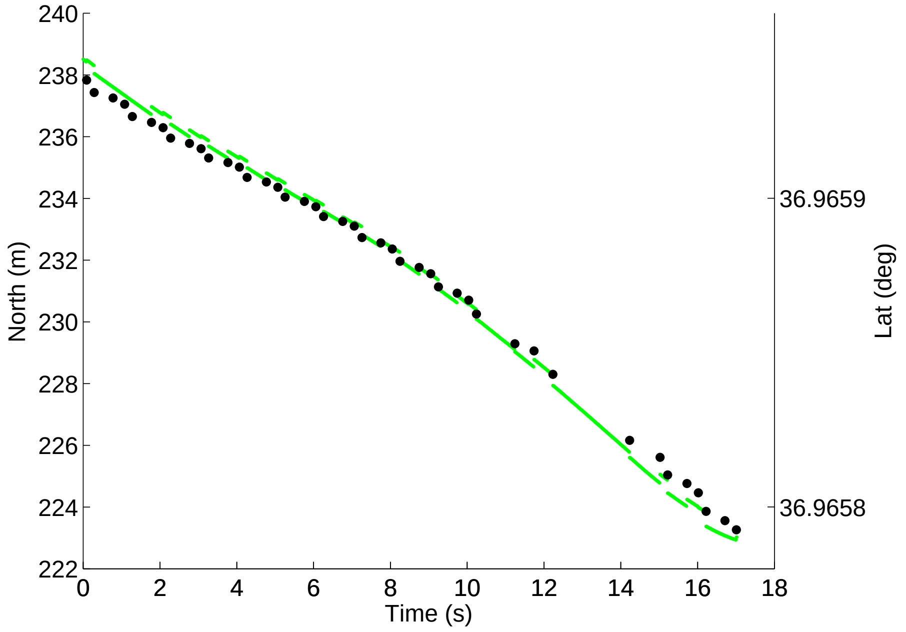


Figure 4.6: A plot of the calculated local position showing how velocity integration increases the temporal resolution of the received sensor data. The black dots indicate GPS position updates and the green segments are the extrapolated local position.

4.2.4 GPS Offset Correction

The GPS sensor used on the SeaSlug is mounted at the rear of the vessel, roughly three meters from the center of rotation of the vessel, which has been calculated as being slightly forward of the vessel's physical center. This results in a reported position that is offset by three meters. The reported velocity is also incorrect as it includes the velocity induced by the circular motion of the GPS about the vehicle's center of rotation. This relationship between the vehicle center and the GPS measurements are:

$$\begin{aligned} P_{gps} &= P_{center} + P_{offset} \\ V_{gps} &= V_{center} + V_{induced} \end{aligned} \tag{4.10}$$

Since the desired output from the sensor is the position and velocity of the center of rotation of the vessel (hereafter referred to as the center) the above equations can be reordered as shown in Eq. 4.11. The n and b scripts have been added to denote the reference frame as either the global *navigation frame* or the local *body frame.*

$$\begin{aligned} \overset{n}{P}_{center} &= \overset{n}{P}_{gps} - \overset{n}{P}_{offset} \\ \overset{n}{V}_{center} &= \overset{n}{V}_{gps} - \overset{n}{V}_{induced} \end{aligned} \tag{4.11}$$

It should be noted that while the equations provided in this section operate in three-dimensions, they are constrained to two-dimensions for the SeaSlug. Because the altitude of the SeaSlug is uncontrollable, sensing it is irrelevant and onboard calculations ignore the altitude, either setting it to zero or omitting it completely.

Correcting the position from the GPS requires calculating $\overset{n}{P}_{offset}$ from Eq. 4.11. In the body frame the GPS antenna has been measured as 2.709m behind and

0.155m to the left of the vessel center. While the GPS antenna is situated above the plane of rotation for yaw, the extra velocity this offset induces is small enough to ignore.

Since the GPS offset is measured in the body frame, it is converted into the navigation frame so that it can be used in Eq. 4.11. This conversion uses a rotation matrix that maps a three-dimensional vector from the body frame to the navigation frame:

$$\begin{aligned} \overset{n}{P}_{offset} &= \overset{b \longrightarrow n}{R} \cdot \overset{b}{P}_{offset} \\ &= \overset{b \longrightarrow n}{R} \cdot \begin{bmatrix} -2.709 \\ -0.155 \\ 0 \end{bmatrix} \end{aligned} \tag{4.12}$$

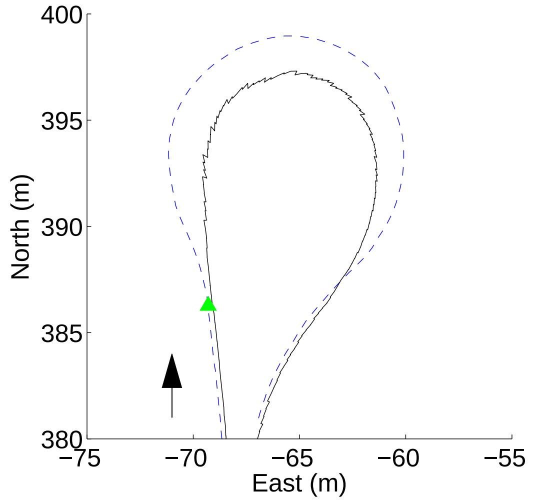
Correcting for the linear velocity induced by the boat's rotation is dependent on the sensor's distance from the center and its angular velocity, as follows from uniform circular motion. This induced velocity is:

$$\overset{n}{V}_{induced} = \overset{n}{\Omega} \times \overset{n}{P}_{offset} \tag{4.13}$$

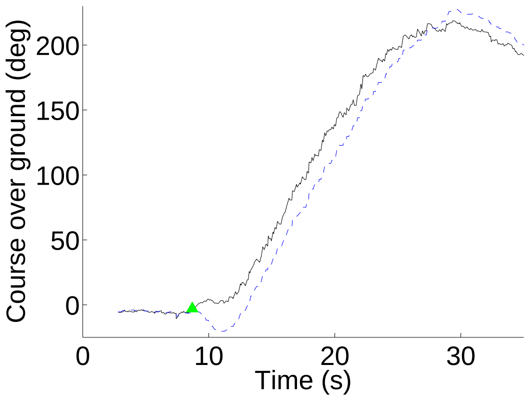
While $\overset{n}{P}_{offset}$ has been calculated in Eq. 4.12, the angular rates of the vessel in the navigation frame now need to be defined. Rate gyroscopes in the IMU measure angular rate directly in the body frame. They can be converted by using the rotation matrix $\overset{n \longrightarrow b}{R}$ such that

$$\overset{n}{\Omega} = \overset{b \longrightarrow n}{R} \cdot \overset{b}{\Omega} \tag{4.14}$$

$\overset{n}{P}_{offset}$ follows from Eq. 4.12, and thus the final equation for the induced velocity in the navigation frame is:



(a) Vehicle position in the local coordinate frame.



(b) Vehicle course.

Figure 4.7: The GPS offset from the vehicle position shown through vehicle position and course-over-ground during a 180° right turn. The green triangle indicates where the turn starts in both plots. The dashed-blue line is the uncorrected GPS output and the solid black lines are the corrected values.

$$\begin{aligned} \overset{n}{V}_{induced} &= (\overset{b \longrightarrow n}{R} \cdot \overset{b}{\Omega}) \times (\overset{b \longrightarrow n}{R} \cdot \overset{b}{P}_{offset}) \\ &= \overset{b \longrightarrow n}{R} \cdot (\overset{b}{\Omega} \times \overset{b}{P}_{offset}) \end{aligned} \tag{4.15}$$

The final equations for calculating the position and velocity of the vessel center are shown in Eq. 4.16. These are derived by substituting the derived forms for the extra velocity from Eq. 4.15 and position from Eq. 4.12 into the original GPS correction equation, Eq. 4.11.

$$\begin{aligned} P_{center} &= P_{gps} - \overset{b \longrightarrow n}{R} \cdot \overset{b}{P}_{offset} \\ V_{center} &= V_{gps} - \overset{b \longrightarrow n}{R} \cdot (\overset{b}{\Omega} \times \overset{b}{P}_{offset}) \end{aligned} \tag{4.16}$$

Fig. 4.7 shows that positive feedback can occur without correcting the GPS position and velocity for the sensor offset. These plots show the GPS position as reported by the sensor and again after being corrected for its offset during a

180° right-handed turn. At the start of the turn, indicated by the green triangle, the sensor position initially moves opposite the desired direction. Eventually the GPS sensor starts turning right, and the system is responding as expected. This is more visible in the course-over-ground data in Fig. 4.7b. It is this initial left-turning action that poses a problem for the controller, as it is positive feedback, which can result in overall controller instability.

Offset measurement delay

To implement the offset correction for the GPS sensor, the IMU must be used to provide both the current heading and the yaw rate. These sensors have different measurement delays in their sensor readings. For GPS units, delays are commonly on the order of 0.5s to 1.5s. For IMU sensors like the gyroscopes, the latency is on the order of milliseconds.

For this vessel, however, there is no estimation nor correction of the GPS delay. In simulation, the GPS readings are delayed 1.5s, using a conservative estimate of the actual delay. This delay, however, is never used in the controller as it is likely not correct. Without factoring in the delay, correction of the GPS data with IMU data is mathematically incorrect. However, since the vehicle experiences low acceleration, differences in gyro readings across the timespan of the GPS delay are small.

Using the current gyroscope readings results in overcompensating for the offset, which can be seen in Fig. 4.8. There the vehicle was heading south, and then turned right to head north-west. These hard turns make the GPS offset correction most noticeable and the overcompensation here is obvious. The angle of the groupings of positions show the direction of the corrected velocity vector. The corrected velocity vector ends up pointing further into the turn, leading to the

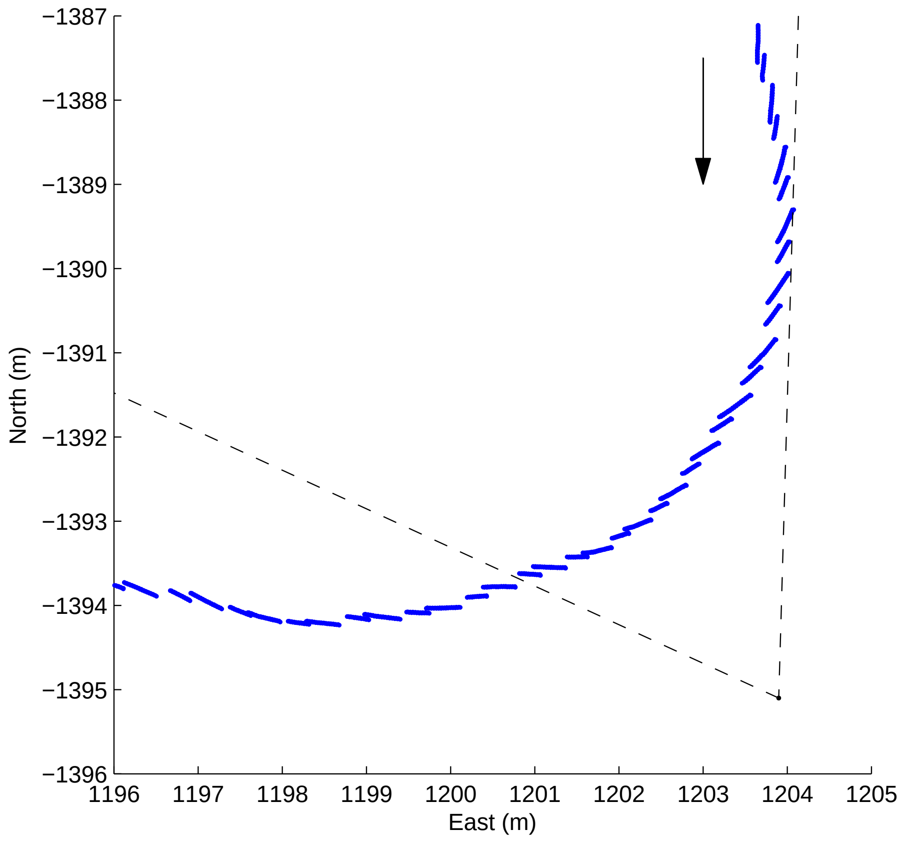


Figure 4.8: A right-hand turn from a southerly trajectory to a north-west one. The overcorrection is apparent in the segments that point into the turning circle.

apparent feathering of the recorded position track.

While this is technically incorrect, it ends up being a useful feature. This overcorrection effectively provides some phase-lead to the position estimate through turns. By overestimating the vehicle's course, it reduces the overshoot when turning back onto the line. This could also be accomplished with better system tuning or a more accurate vehicle model, but this behavior is left as-is, both because it is beneficial and the fix is non-trivial.

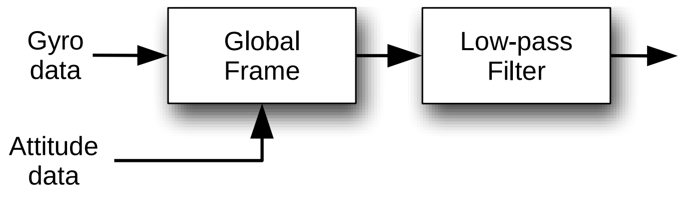


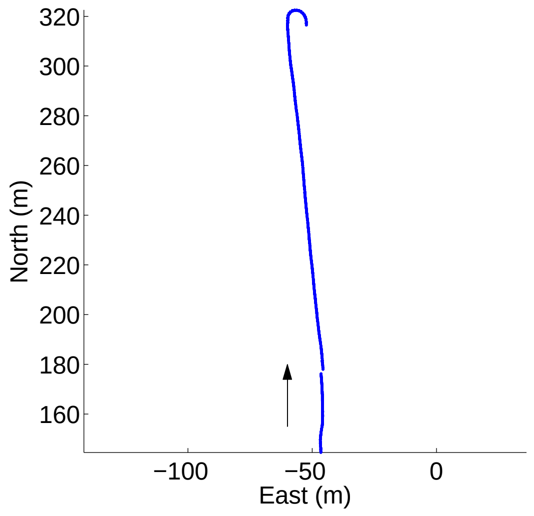
Figure 4.9: The chain of filtering steps for processing the gyroscope data output by the IMU.

4.3 IMU filtering

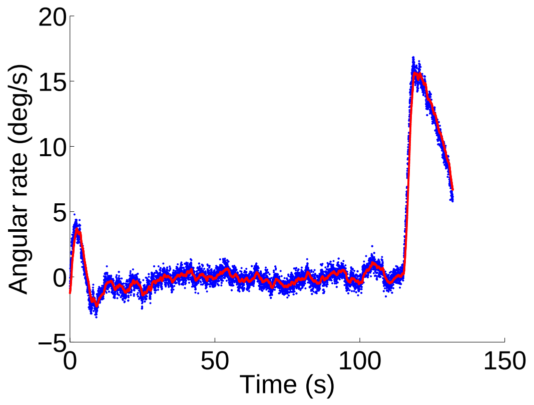
The IMU reports its gyroscope data as a vector of body-frame rotation rates about the x-, y-, and z-axes. The main use of the gyro data is for the yaw-rate of the vessel: how fast it is rotating around the global Z-axis. If the vehicle is not perfectly level, then the body-referenced z-axis rotation rate reported by the gyro is different than the actual vehicle's yaw rate. Therefore, the yaw rate of the vehicle body relative to the inertial frame must be computed. This is a two-step process as shown in Fig. 4.9.

The first step, converting body-frame gyro data to navigation-frame Euler angle rates, uses Eq. B.4, which is a single matrix multiplication. The derivation of this matrix is shown in Appendix B.4.

The second step is a low-pass filter as gyroscope data is notoriously noisy. Because the SeaSlug is so large, it experiences low angular velocities in the real world. Therefore an exponentially-weighted moving average filter with a 2Hz cutoff frequency was used. While this cutoff frequency might appear quite low, it is well above the system bandwidth as the SeaSlug never exceeds more than $20°/s$ about any axis.



(a) The vehicle's position in the local coordinate frame.



(b) The yaw rate as reported by the gyroscope for the body-frame z-axis and after filtering and conversion to the navigation frame.

Figure 4.10: Position and corresponding yaw-rate plots of an autonomous run. There is an initial right-turn at the start with a much larger 180° turn at the end.

4.4 L2+ Control

4.4.1 Introduction

For data collection missions, scientists are accustomed to generating a vehicle trajectory from a sequence of waypoints, defined in global latitude and longitude. The L_2^+ algorithm [25] has been flight tested on the SLUGS autopilot controlling unmanned aerial vehicles (UAVs). It is capable of following arbitrary trajectories, including ones comprised of connected line segments. It is also vehicle agnostic, as it does not account for vehicle dynamics, and can be readily adapted to surface vessels. For these reasons it was chosen for mission guidance on the SeaSlug.

This algorithm is a pure-pursuit guidance law, where an aim point is chosen and the vehicle is directed towards that point. It builds on the L_1 control law originally developed for UAVs [72]. The L_1 control law used a fixed look-ahead

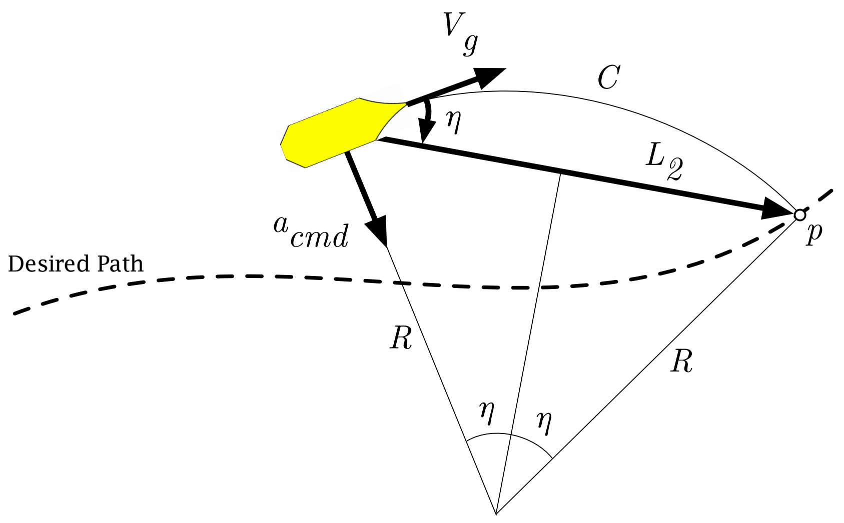


Figure 4.11: The geometry behind the L_2^+ control law. p is the aim point dictated by the L_2 look-ahead vector, which is defined as a time and not a distance. a_{cmd} is the resultant acceleration necessary to intercept p on a circular path.

distance to determine an aim point, p, along the desired path as shown in Fig. 4.11. The desired vehicle trajectory to intersect that point is then specified as the arc, C, described by the angle, η, which is dependent on the vehicle's velocity vector and the L_1 vector. Tracking this arc is done by commanding a lateral acceleration. This is then mapped into a commanded roll angle for UAVs by using a kinematic model.

The original L_1 control law used a fixed length look-ahead vector, which could cause instability at high speeds, as the effective gain is proportional to the vehicle's ground speed. Because the L_1 vector was fixed, an increase in velocity causes the look-ahead time to shorten, which equates to a higher gain. If the look-ahead vector is instead defined as a fixed time, T^*, instead of a fixed distance, the system is stable over a much wider range of speeds. The look-ahead vector length, $|L_2|$, is then calculated by $V_{g_x}T^*$. This modification was found to be necessary for operation in environments where the vehicle's ground speed could change drastically, such as from wind in the case of UAVs.

Additionally, the L_2^+ control law makes several modifications to the domain in which this controller can operate. The L_1 algorithm only worked when the vehicle was within L_1 of the desired path. Outside of this, the controller had undefined behavior. The L_2^+ algorithm expands the operational domain of the controller to include regions further than the look-ahead distance from the desired path. This makes it possible for the vehicle to intercept and follow the desired path starting from any position and orientation.

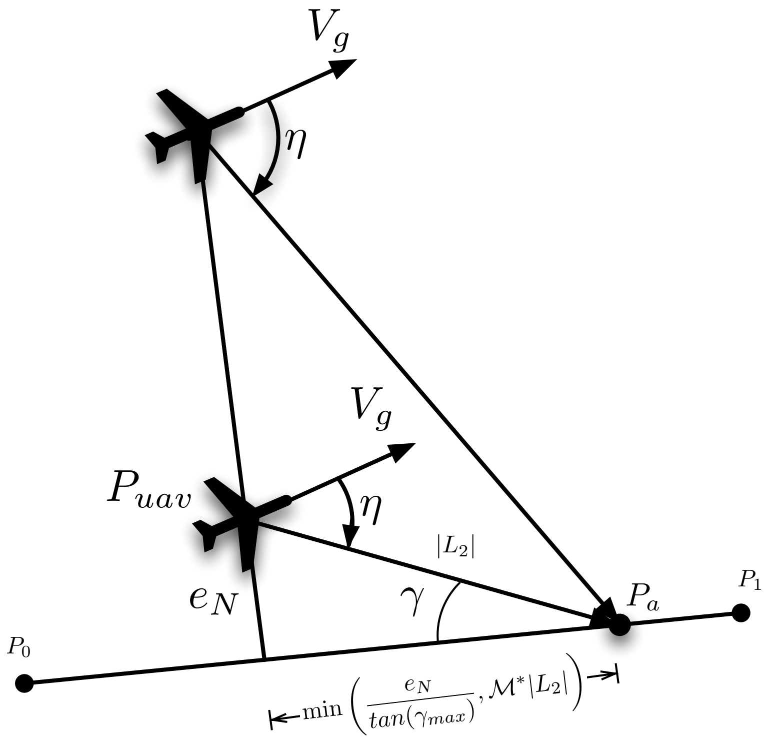

V_g
η
V_g
P_{uav}
η
$|L_2|$
e_N
γ
P_a
P_1
P_0
$\min\left(\frac{e_N}{tan(\gamma_{max})}, \mathcal{M}^*|L_2|\right)$

Figure 4.12: Reproduction of Figure 5 from [25]. Shows the restrictions on the aim point, P_a, when the vehicle is far away from the desired path.

When the vehicle is far from the path, different parameters are used to place the aim point. Being far away from the path is defined as having a crosstrack error of more than $T^*|V_g|$. In this domain, the intercept angle γ_{max} and the along track distance gain $\mathcal{M}^*$ define the aim point (see [25] for a full description of the L_2^+ control logic). The resultant aim point is also restricted so that it cannot be past the next waypoint. This geometry is shown in Fig. 4.12.

On top of this waypoint navigation behavior, there is additional initial-point and return-to-base (RTB) functionality. The initial point behavior can be configured to specify an initial target point to reach before proceeding along the first path segment. This is defined as a distance ahead of the first waypoint but aligned with the first line segment between the first waypoint and the second. This overrides the standard behavior of attempting to intercept the straight-line path between the first and second waypoint. The return-to-base functionality follows similar behavior, where a specific location is set as an aimpoint with the L_2 vector is pointing towards it. This mode is generally triggered in an emergency situation when the vehicle has lost its connection to the human operator or there has been some other system failure.

4.4.2 L2+ for Surface Vessels

In the original formulation of the L_2^+ algorithm the output is commanded lateral acceleration. UAVs can command this directly by controlling roll angle. Most surface vehicles, however, cannot command an equivalent lateral acceleration, though they can command a turn rate. To support such vehicles the radius of the turn is instead used to calculate a desired yaw rate. Then the necessary control surface command can be calculated using a model of the vehicle kinematics. In the case of the SeaSlug, L_2^+ commands the rudder angle.

Using the inverse bicycle model introduced in Section 3.2.1, with a properly tuned wheelbase parameter L, it is now possible to convert the lateral acceleration from the L_2^+ algorithm into a rudder angle, which the SeaSlug can command. The yaw rate of the SeaSlug is defined in Eq. 3.7, reproduced here in Eq. 4.17. Note that v_{w_x} is the hull velocity through the water and δ_r is the rudder angle:

$$\dot{\psi} = -\frac{v_{w_x} \tan(\delta_r)}{L} \tag{4.17}$$

From the physics of circular motion, the linear ground velocity is related to the turn rate of the vessel. The ground velocity is used here because the control error is defined as relative to the ground-fixed waypoints.

$$\omega = \frac{|V_g|}{R} = \dot{\psi} \tag{4.18}$$

Additionally, the lateral acceleration necessary to induce circular motion of radius R is:

$$a_y = \frac{|V_g|^2}{R} \tag{4.19}$$

Again, this equation uses ground velocity because the controller operates with ground-relative distances. From Eq. 4.18 and Eq. 4.19, the necessary turn rate for a given lateral acceleration can be calculated:

$$a_y = |V_g|\dot{\psi} = \frac{-|V_g| V_{w_x} \tan \delta}{L} \tag{4.20}$$

Now that an equation involving both the lateral acceleration and rudder angle has been found, the rudder angle can be resolved:

$$\delta = -\arctan \frac{a_{cmd} L}{|V_g| V_{w_x}} \tag{4.21}$$

Eq. 4.21 provides the necessary mapping from the commanded acceleration output of the L_2^+ controller to a commanded rudder angle for the SeaSlug surface vessel.

4.4.3 L2+ for Slow Surface Vessels

It should be noted that the L_2^+ algorithm assumes that there are no vehicle dynamics, so that $a_{cmd} = a_{actual}$. With the planes that have been flight tested with L_2^+, this has been a reasonable assumption because of how quick their response time is.

This assumption, however, breaks down for the SeaSlug, which has a much slower yaw-rate response. To account for this a feedback term was added for the yaw-rate. This compensates for the response delay and has the effect of adding phase back to the controller. Therefore by reducing the commanded turn rate slightly as the system turns faster it improves the yaw-rate response time.

The final commanded rudder angle algorithm is shown in Fig. 4.13. In addition to the L_2^+ algorithm, there is both the yaw rate feedback term as well as the translation from the lateral acceleration command generated by L_2^+ into a rudder angle command.

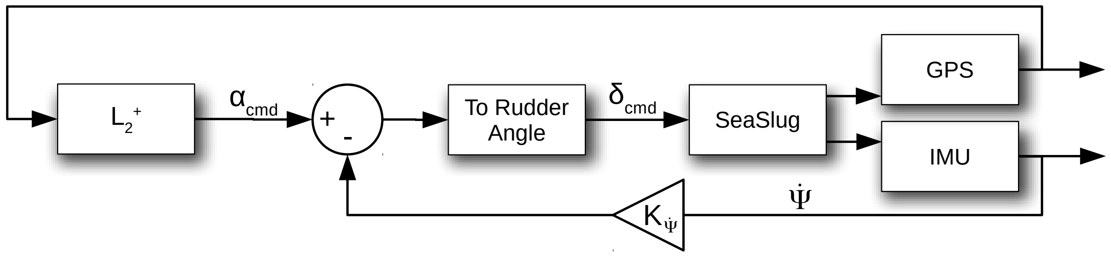


Figure 4.13: The complete controls algorithm for rudder control. Inputs are filtered sensor data and the final output is the desired rudder angle. Yaw rate is fed back to account for the system's slow response time.

4.4.4 Parameter Tuning

Parameter tuning for the L_2^+ controller is through the main parameter that controls the system response: T^*, the look-ahead time. Smaller values for this parameter increase the effective controller gain, larger values effectively decrease the controller gain.

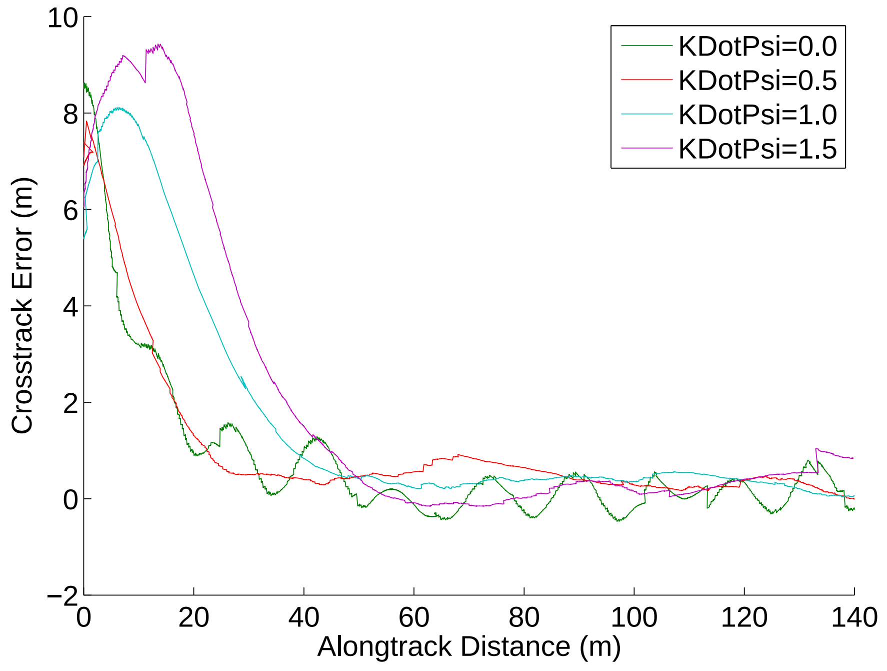
Given the slow response of the SeaSlug, it was estimated that T^* would be large, between 5 and 15s. Stability analysis of the L_2^+ algorithm done in [25] shows that for the controller to be stable, T^* must at least 3 times the roll response lag of the system. For the surface vessels, this translates to the yaw rate delay. Testing various values led to the final chosen parameter of 12s. All results shown in Chapter 5 use this value unless otherwise indicated.

Yaw Rate Feedback

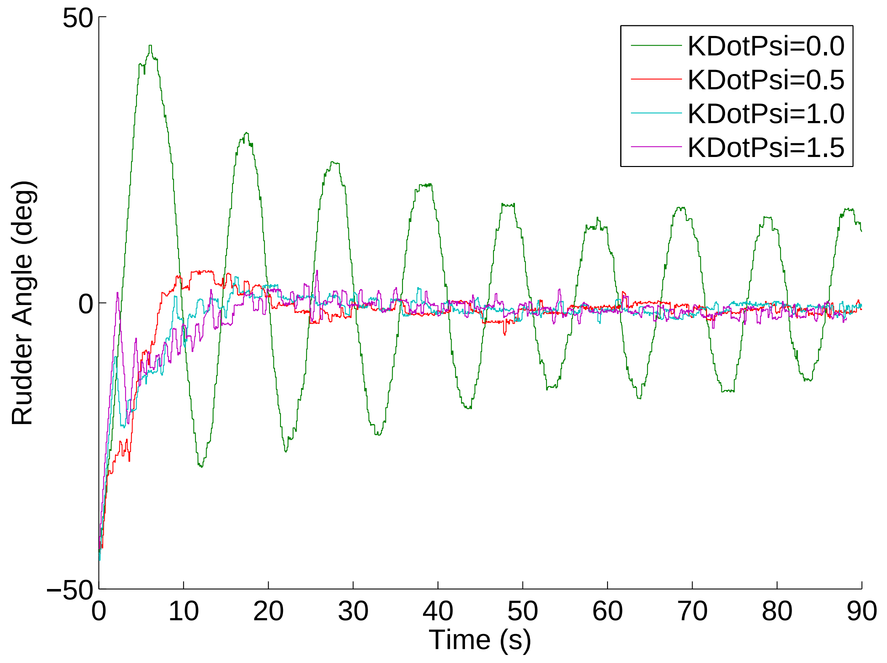
The yaw rate feedback gain, $K_{\dot{\Psi}}$, is the other parameter important for system stability. This term compensates for the system's slow turn rate response by accounting for its current turn rate. Tuning this parameter is independent of testing the L_2^+ parameters and was done after suitable values for T^* were found.

Some of this testing is shown in Fig. 4.14, which shows both the crosstrack error of the vessel and its rudder angle over four different values of $K_{\dot{\Psi}}$. This testing was done by inducing a 180° turn by using a two-waypoint line. The turn initially results in the rudder saturating and then slowly reducing its angle until it hits zero. These figures are all normalized to the time at which the rudder first leaves saturation; until that point the response is essentially the same.

From the figures it is clear that the optimal value is around 0.5. This value provides nearly the same rise time as without the feedback but lacks the undesirable oscillations that occur with no $K_{\dot{\Psi}}$ term. The higher values result in much longer rise times and at a value of 1.5 the oscillations return. Further testing showed that a value of 0.4 was the ideal balance between rise time and settling time. This value was used for all further testing.



(a) Crosstrack error.



(b) Rudder angle.

Figure 4.14: System performance when reacquiring the desired line after a 180° turn. Both figures start when the rudder first exits saturation.

Intercept Behavior

The controller's behavior when too far away from the path is specified by the along track distance $\mathcal{M}^*$ and the intercept angle γ_{max} parameters. They define vehicle behavior when far away from the line as shown in Fig. 4.12.

The $\mathcal{M}^*$ parameter is a gain on the look-ahead distance. During testing of the L_2^+ algorithm with the SLUGS autopilot it was found that 2.0 was an acceptable value for most use-cases. This value is used for all experiments done with the SeaSlug.

The intercept angle γ_{max} is more mission-dependent, as it provides a trade-off between reaching the line as fast as possible and going to the next waypoint as fast as possible. Fig. 4.15 shows how the intercept angle affects initial line acquisition. With higher intercept angles, the vehicle prioritizes following the desired path. With lower ones, the priority is to approach the next waypoint. For most testing γ_{max} was set to 30°, as that provided a reasonable balance between line following and mission time.

4.5 Conclusion

The SeaSlug's path-following capabilities are dependent on both effective sensor data and a capable algorithm. The position filtering was developed gradually as new sensors were added and deficiencies were found in existing sensors. For this reason the sensor filtering could be described as crude, where simplicity won out over more complicated sensor fusion algorithms. From the results presented in Chapter 5, this work is adequate for control, though further work could improve the sensor data.

The L_2^+ was chosen as the primary controls algorithm for the SeaSlug as it was

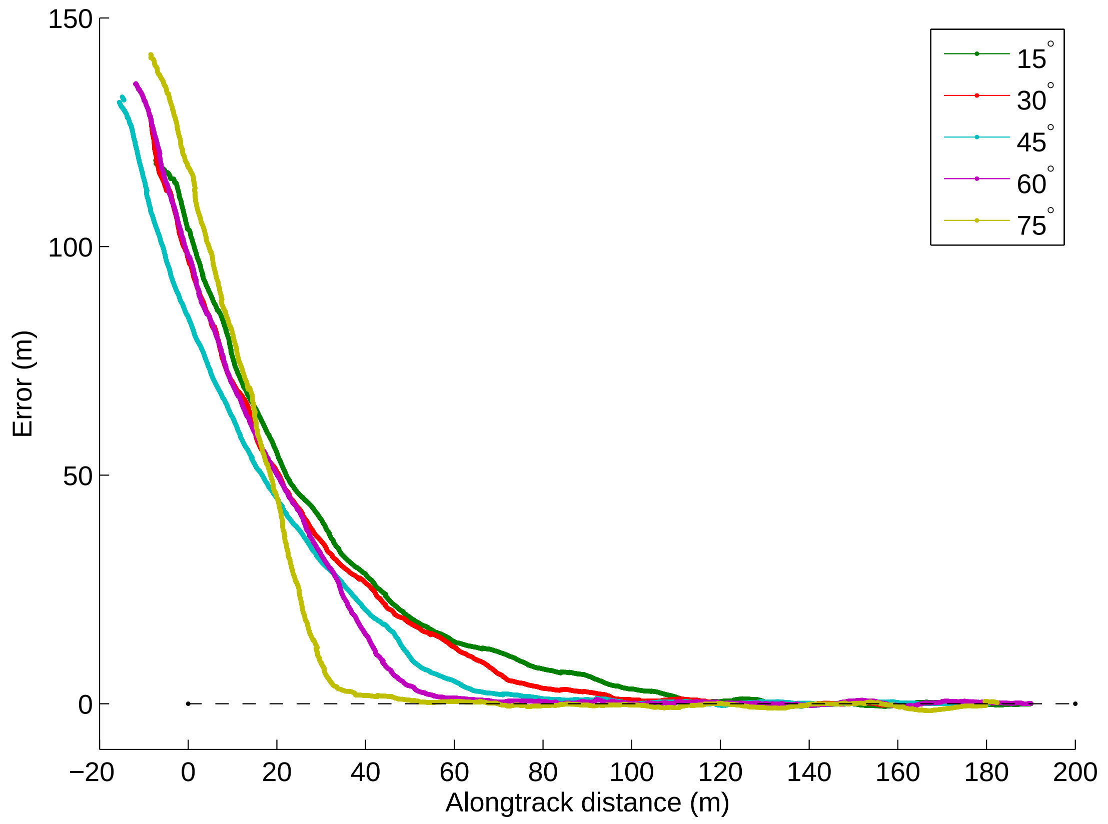


Figure 4.15: Plot of the crosstrack error for an initial line intercept for varying values of the intercept angle, γ_{max}.

previously developed in the Autonomous Systems Lab and had the potential to run successfully on an unmanned surface vessel with relatively minor adaptations. This serves as a validation of the algorithm itself both to its robustness and its adaptability.

The final result of the work discussed in this chapter is a comprehensive sensor fusion scheme, correcting for sensor offset from the center of rotation of the vessel and interpolating between GPS position updates. Using these results, a control algorithm framework was developed that successfully demonstrated line acquisition and path following in littoral waters. These results are presented more completely in Chapter 5.

Chapter 5

Experimental Results

5.1 Introduction

The final goal of the SeaSlug project is to support short-term data collection missions in the open ocean. This relies on a multitude of factors including vessel logistics, system reliability, path following performance, sensor integration, and remote interface usability. The system has been extensively tested in the Santa Cruz Harbor and proven to be reliable. Parameter tuning was done in this environment and system performance was sufficient to warrant continued testing in the unprotected waters of Monterey Bay.

The logistics of deploying and operating the SeaSlug in open water is more complicated than inside the harbor. The exact details of these logistics and the process of launching and operating the SeaSlug are not presented as they vary depending on both the mission and the testing area. Instead this chapter presents the vessel's performance and the collected data from several missions in the littoral waters of Monterey Bay.

5.2 Waypoint Navigation

The most important feature of the SeaSlug for scientific applications is its autonomous waypoint navigation capabilities. Scientists can specify a sequence of GPS coordinates to define the path the vehicle should follow. This is similar to how manned scientific missions are conducted, but the SeaSlug is much more accurate at following the desired path as qualified later.

For these waypoint tests, the L_2^+ controller was used with the parameters described in Section 4.4.4.

5.2.1 Basic Waypoint Test

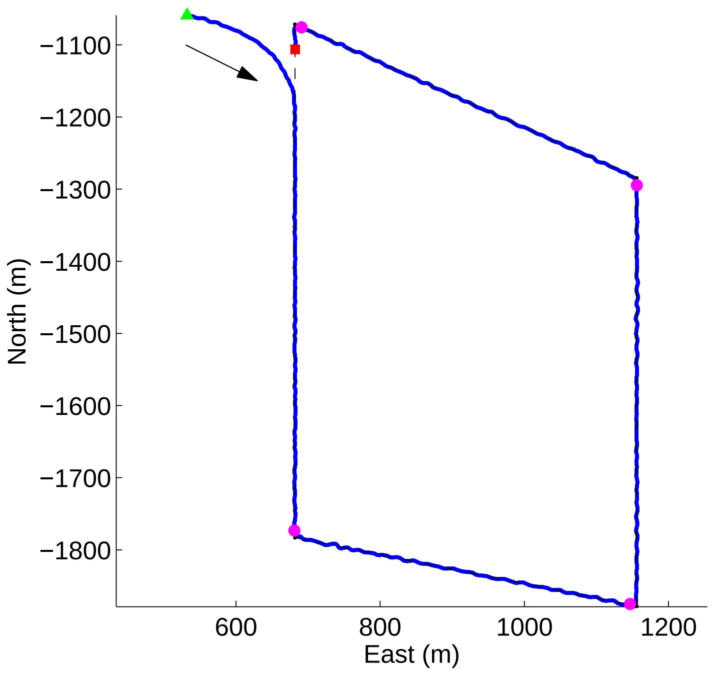


Figure 5.1: Vehicle track following a four-waypoint trapezoid. Coordinates are in the local tangent plane. The green triangle and red square indicate the start and end locations respectively. The magenta dots indicate when the vessel switched to the next waypoint. The dashed black line indicates the desired path.

Initial ocean tests examined the performance of the L_2^+ controller while following a simple trapezoidal path. This trajectory totaled 2.3km in length, sufficient for an initial performance evaluation in littoral waters. The trapezoidal shape was chosen to avoid the buoys and kelp that are near the shore of Santa Cruz and still demonstrate both the waypoint transition and line-following capabilities of the L_2^+ algorithm.

The mission began with the SeaSlug located 150m west of the first waypoint and headed east under full throttle. The total mission length was 200m longer than planned due to this initial starting position. The vessel averaged a ground speed of 1.6m/s yielding a total mission time of 26 minutes and 37 seconds. The weather was calm with light wind and small 1.0m swells (Beaufort scale 2).

Fig. 5.1 shows the position of the vehicle during the mission in the vehicle's local coordinate frame. When the vessel is first switched into autonomous mode there is an initial oscillation in the vehicle's position as it settles onto the curved trajectory generated by the L_2^+ algorithm (visible near the green triangle). Once it has acquired this intercept trajectory, the position stabilized substantially.

The crosstrack error for all four line segments are overlaid on top of each other and shown in Fig. 5.2. The mean error was 0.13m with a standard deviation of 1.0m. These statistics were calculated by ignoring the initial line acquisition, defined as the portion of the mission before the vehicle crosstrack error first falls below 2.0m since the start of the mission. After this initial intercept the vessel almost never deviates from the desired course by more than 2.0m except during waypoint transitions.

During waypoint transitions the crosstrack error jumps to several meters, as the system switches when is distance-to-go is less than 10m. These initial errors at the start of every line segment are visible in Fig. 5.2. This error is positive for left-hand turns, as the system is then left of the desired path and for right-hand

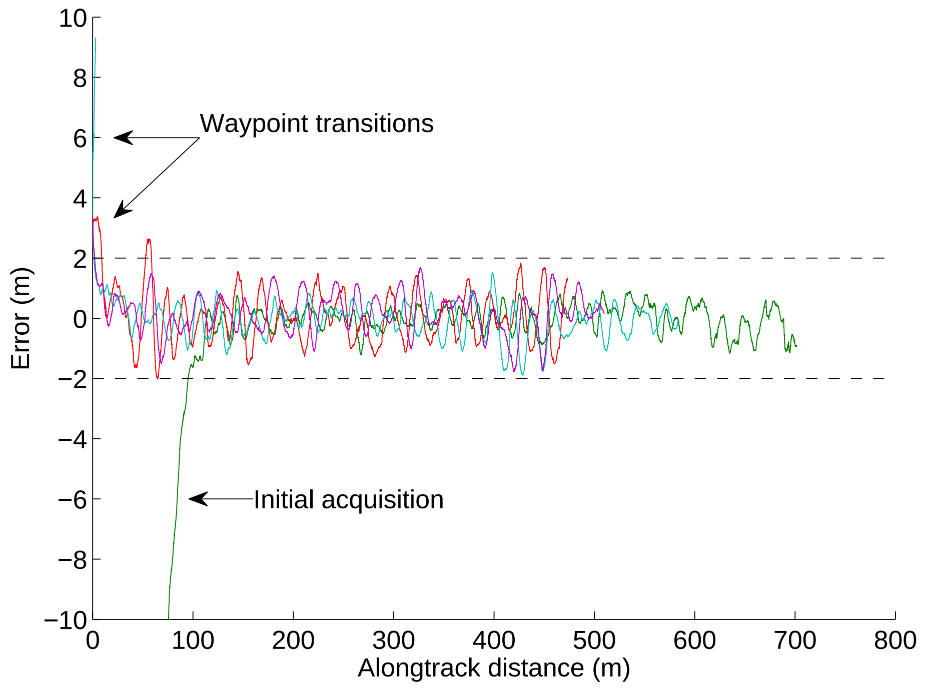


Figure 5.2: Overlaid crosstrack errors for trapezoid mission path.

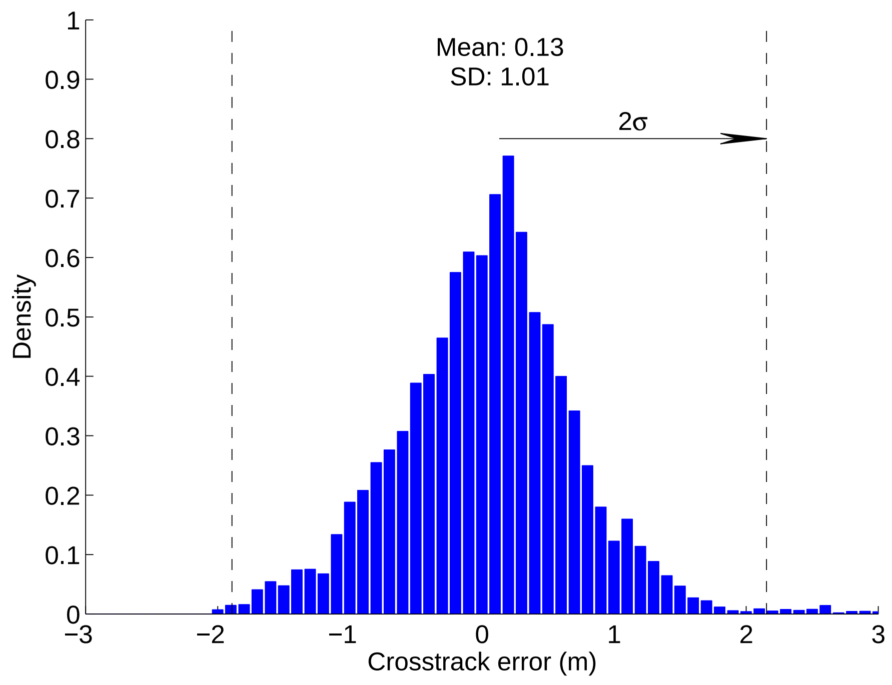


Figure 5.3: Histogram of the crosstrack error for trapezoid mission path. The crosstrack error from before the path is initially acquired is ignored here.

turns it is negative. Since the trapezoid pattern resulted in only left-handed turns the distribution of crosstrack errors is slightly weighted towards positive errors, as shown in Fig. 5.3. Though the main set of errors are clustered tightly around zero error, there is a "tail" off to the right of positive errors from the waypoint transitions.

5.2.2 Complex Waypoint Test

Previous harbor testing has shown the SeaSlug capable of complex waypoint navigation and the trapezoid test described above demonstrated basic waypoint navigation in ocean waters. A more complex waypoint test was then planned to evaluate waypoint navigation in the open ocean. The path for this test was still four waypoints, but this time in the shape of a bowtie, where the longer transits cross over each other. Additionally, this waypoint sequence was chosen to demonstrate how the L_2^+ controller follows highly-acute angles during waypoint transitions. This was a shorter test, only 1.3km in length, to also test how the system handled shorter distances between waypoints.

The vessel started the test approximately 120m west and 60m north of the initial waypoint, facing south-south-east and under full throttle. Because of this initial starting location, the total traveled distance was 300m longer than the waypoints specified. Completing this mission took 16 minutes and 35 seconds with the vessel traveling at an average ground speed of 1.6m/s.

Fig. 5.4 shows the desired waypoint sequence and vessel path during the mission. The 360° turn at the start was a glitch due to additional code in the controller that has since been removed; this anomaly has not been observed since. Due to logistical reasons this test was not repeated after the code fix.

The effect of swells coming in pairs from the south-west is evident in both the

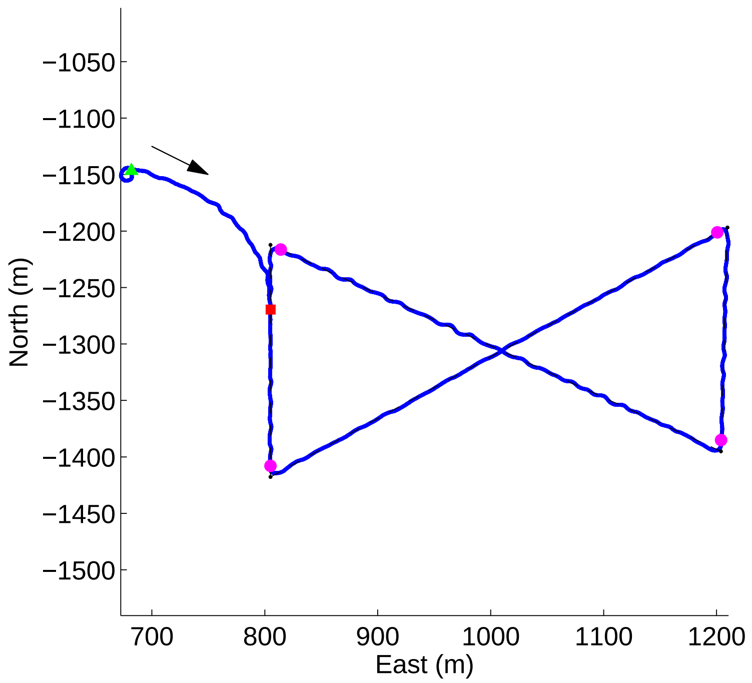


Figure 5.4: Vehicle track following a 4-waypoint bowtie. Coordinates are in the local tangent plane. The green triangle and red square indicate the start and end locations respectively. The magenta dots indicate when the vessel switched to the next waypoint. The dashed black line indicates the desired path.

position plot and the overlaid crosstrack error plot in Fig. 5.5. The pink trace is the final north-west leg of the mission and shows much higher errors than the other legs. The effect of the swells is most pronounced here as they are beam seas (nearly perpendicular to the vessel). The red trace shows the error during north-east transit of the bowtie and these errors are much reduced. This is because the swells are following seas (in the direction of motion of the vessel).

The mean crosstrack error is -0.02m with a standard deviation of 1.1m, excluding initial line acquisition. Fig. 5.6 shows the exact distribution of crosstrack errors during the mission. The crosstrack errors again stay below 2.0m very consistently after initial line acquisition. Their mean is also very close to zero due to the equal number of left and right turns during the mission, so that the error during waypoint transitions effectively cancel out.

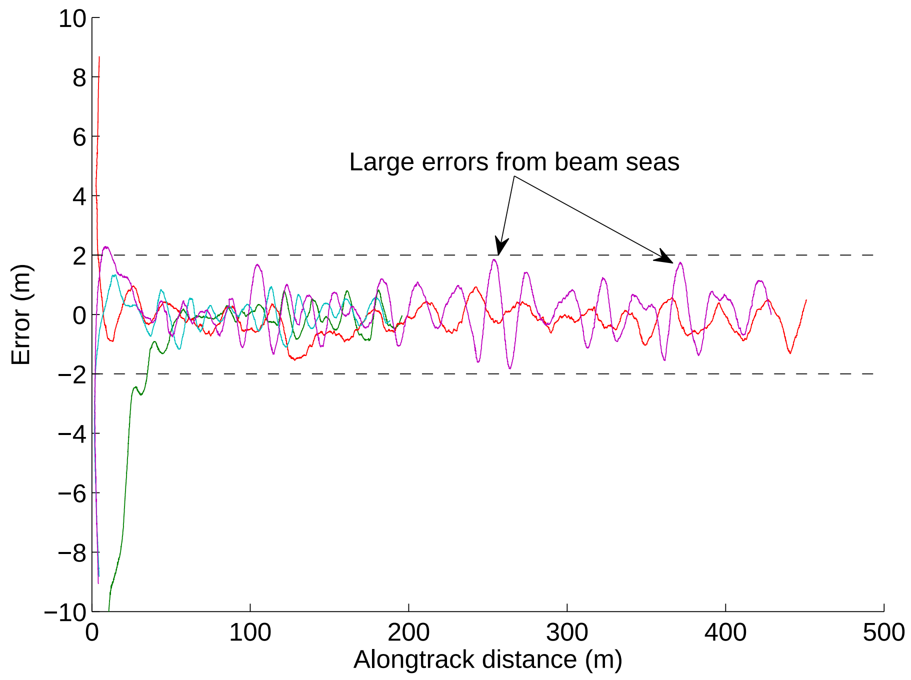


Figure 5.5: Overlaid crosstrack errors for bowtie mission path.

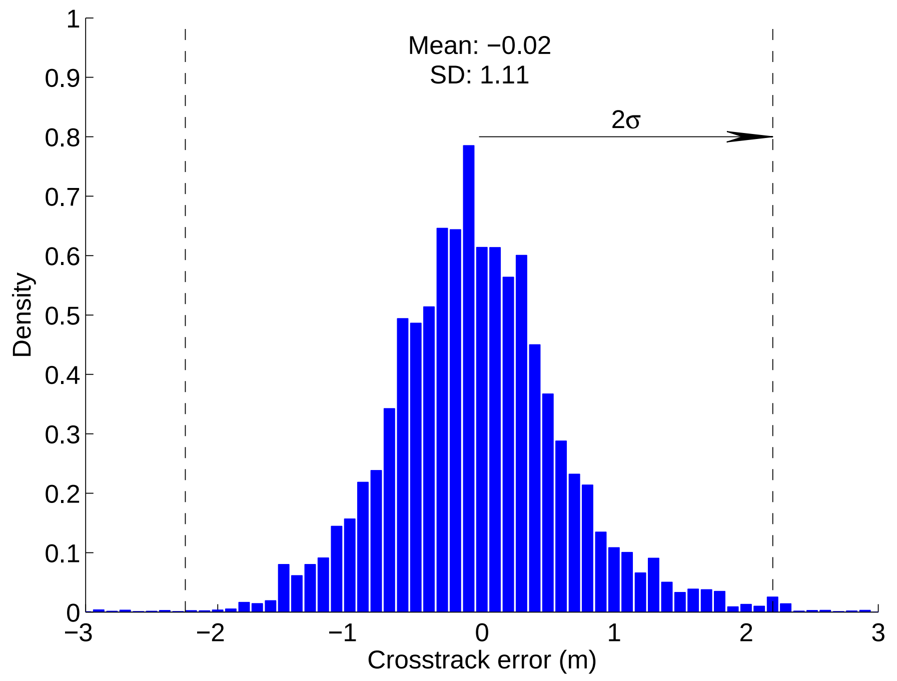


Figure 5.6: Histogram of the crosstrack error for bowtie mission path. The crosstrack error from before the path is initially acquired is ignored here.

5.2.3 Repeatibility

Both the trapezoid and bowtie missions were run twice in 2015, first on March 27th and then again on April 3rd. The duplicate runs were performed to demonstrate that the performance of the SeaSlug is repeatable and consistent across different sea and weather conditions. While March 27th was mostly calm (with the exception of large swells towards the end of testing), there was only light wind on June 3rd along with some larger waves (Beaufort state 3). The location of both missions are shown in Fig. 5.7.

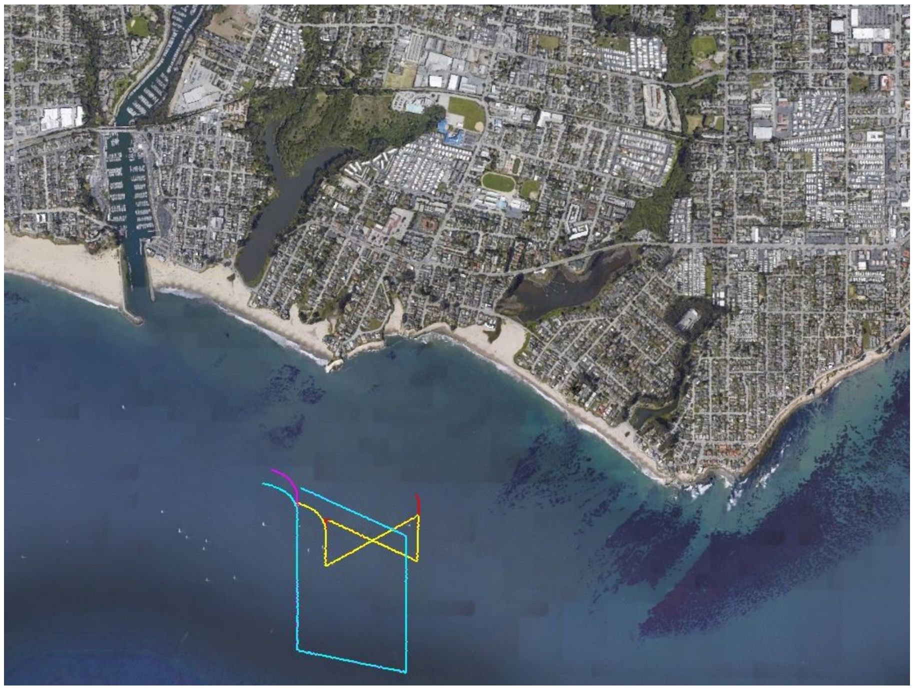


Figure 5.7: The locations of the trapezoid and bowtie missions as executed off the coast of Santa Cruz. The blue and purple traces are the first and second trapezoid mission runs respectively. The yellow and red traces are the first and second bowtie runs respectively. For reference the Santa Cruz Harbor is visible in the upper-left corner. Map data ©2015 Google, CSUMB SFML, CA OPC.

Table 5.1 shows the mean and standard deviation of the crosstrack error for both the runs of the trapezoid and bowtie missions. The first day of missions show higher standard deviations due to the swells out of the southwest, although they

	Mean (m)	Standard Deviation (m)
Trapezoid, Run 1	0.12	0.96
Trapezoid, Run 2	0.05	0.82
Bowtie, Run 1	-0.06	1.02
Bowtie, Run 2	0.05	0.88

Table 5.1: Crosstrack error distributions for the four repeated runs.

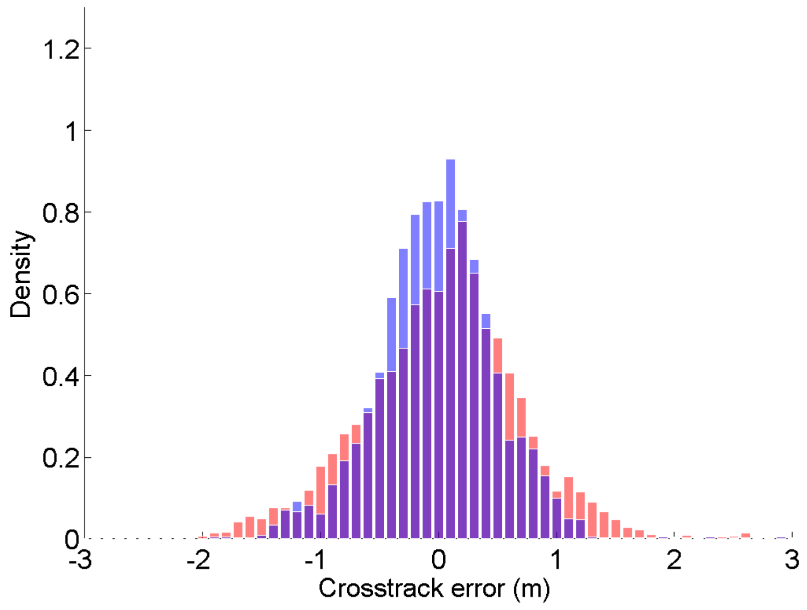
have a similar mean. While the wave and wind conditions were slightly worse on the second day their effect on the vessel was less pronounced than the large swells of the first day.

The distributions of the crosstrack errors for both repeated missions are shown as histograms in Figure 5.8. Fig. 5.8b especially demonstrates the better performance during the second bowtie mission. Here the errors are much closer to zero and very rarely exceed 1.0m.

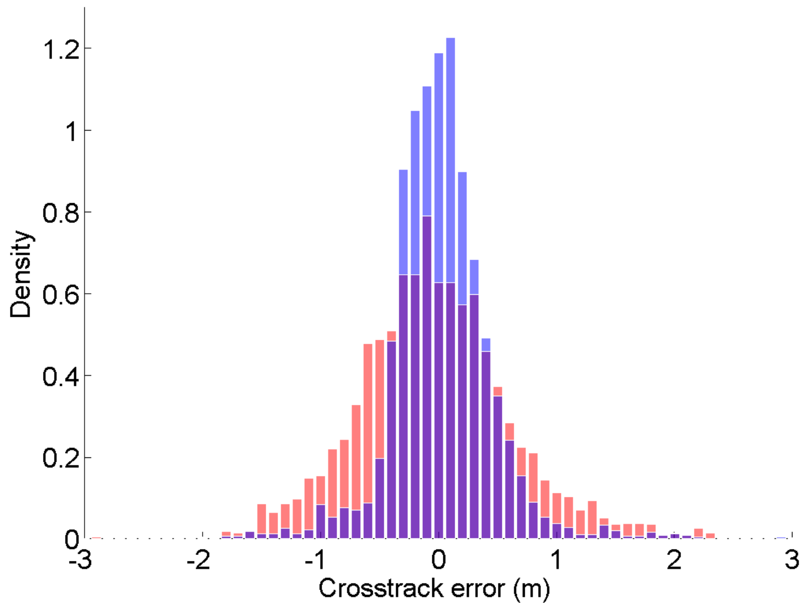
Figure 5.9 shows the vehicle position during both runs of the trapezoid and bowtie missions. At this scale, the 2.0m crosstrack errors are barely noticeable, except for the first bowtie run, where the large double-swells are visible as pairs of oscillations during the north-west transit. Additionally, the initial intercept was significantly different for the second bowtie run, where the mission was started at Waypoint 2 instead of Waypoint 0. This was done merely for convenience, with the waypoint sequence identical for both runs.

5.3 Scientific Deployments

For oceanographic data collection missions, marine scientists have specific sampling patterns that are used depending on the mission objectives and whether they are taking surface or depth measurements. The current capabilities of the SeaSlug limit it to surface measurements, and as such only those capabilities are discussed. Chapter 7 describes plans for integrating vertical profiling for depth sampling.

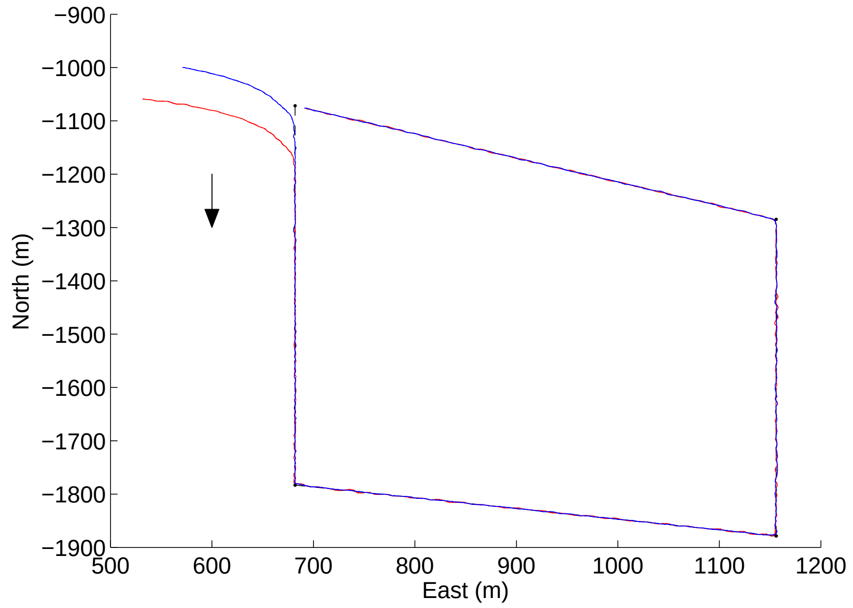


(a) Trapezoid missions.

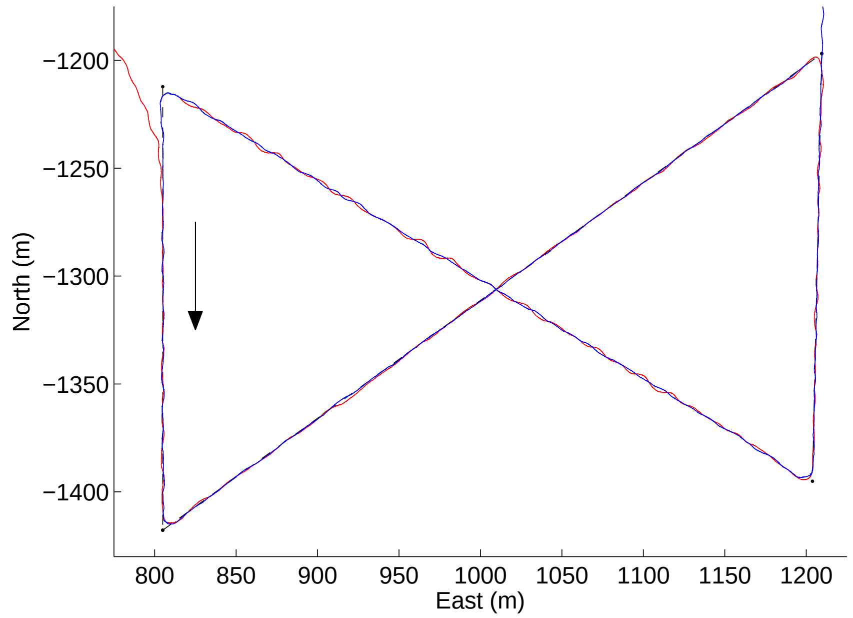


(b) Bowtie missions.

Figure 5.8: Overlaid crosstrack error histograms for repeated runs of the trapezoid and bowtie missions. The first run is shown in red and the second is in blue.



(a) Trapezoid missions.



(b) Bowtie missions.

Figure 5.9: Vehicle position during two separate mission runs performed on 03/27/2015 (shown in red) and on 04/03/2015 (shown in blue)

It should be noted that in practice there is significant flexibility in the mission trajectories for manned vessels. Depending on how rough the ocean is, and what the swell is like, pilots sometimes adapt their trajectories to minimize discomfort while still covering the targeted area. This is generally not done for unmanned vessels as operator comfort is not a factor.

The primary sampling pattern for data collection is the *boustrophedon pattern*, a back-and-forth path where each transit over the area is slightly offset (also commonly referred to as a *lawnmower pattern*). This pattern is commonly adjusted depending on whether the goal is spatial coverage or sampling resolution, referred to as sparse or dense sampling respectively. Generally autonomous vessels perform one type of sampling and manned vessels the other, such that there is both a dense sampling of small region of interest and a larger sampling of the surrounding area as part of the control. This sampling behavior is shown in Figure 1 from [84], reproduced in Fig. 5.10. For this mission two sampling patterns are visible, the dense sampling performed by the manned vessel and the sparse sampling by the autonomous underwater vehicle.

This lawnmower pattern is the most common survey trajectory, but it is sometimes modified depending on the situation. For studying dispersion of substances it is altered to sample a triangular region with one of the points near the center of dispersion, commonly just up-current of it. Coarse sampling is then performed in the direction of the dispersal. This path can also double-back on itself to give multiple samples at the same location at different times. Figure 1 from [34], reproduced in Fig. 5.11, shows the triangular sampling used for analyzing the outflow of the Elkhorn Slough as it enters the Monterey Bay.

For the first scientific applications of the SeaSlug, a region of interest was selected for a variety of sampling missions. With the ocean currents flowing south

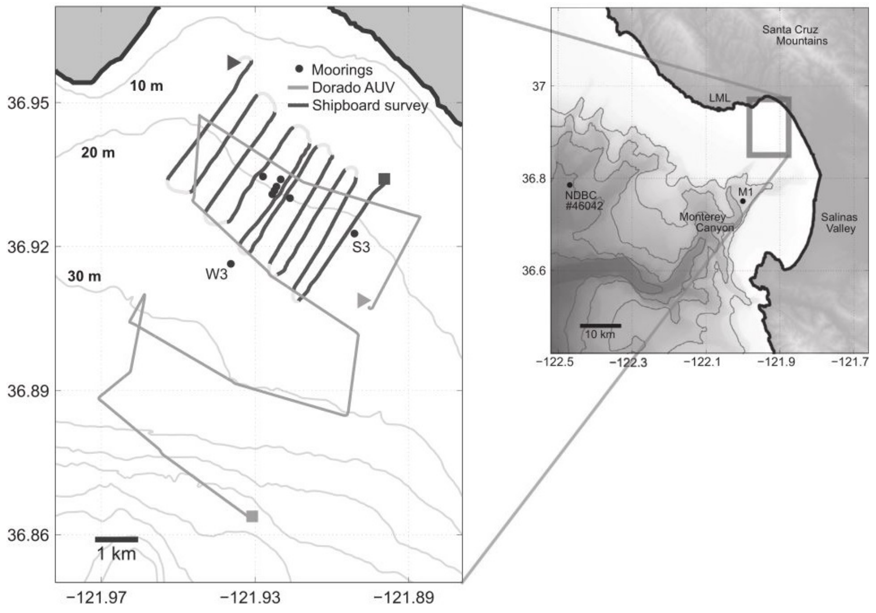


Figure 5.10: Partial reproduction of Figure 1 from [84] which shows a fine lawnmower sampling pattern by a manned vessel (dark grey) and a coarse lawnmower sampling by an AUV. Triangles and boxes represent start and end locations respectively. Additional data was recorded by moorings, shown as dark grey dots.

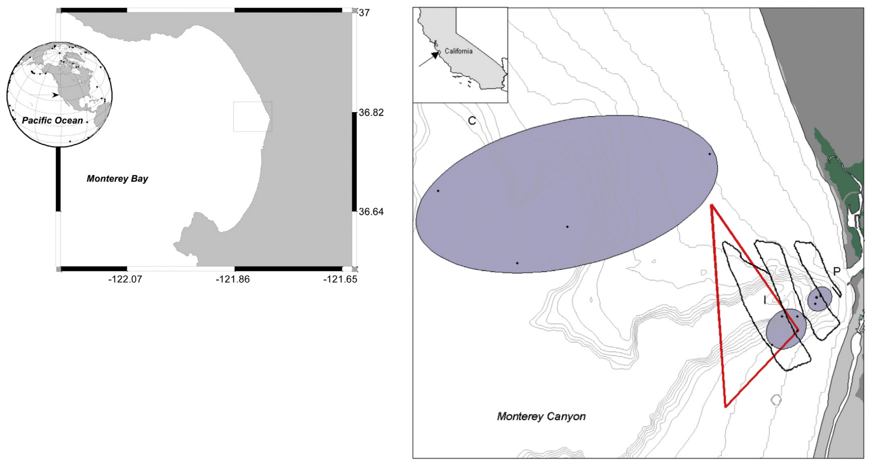
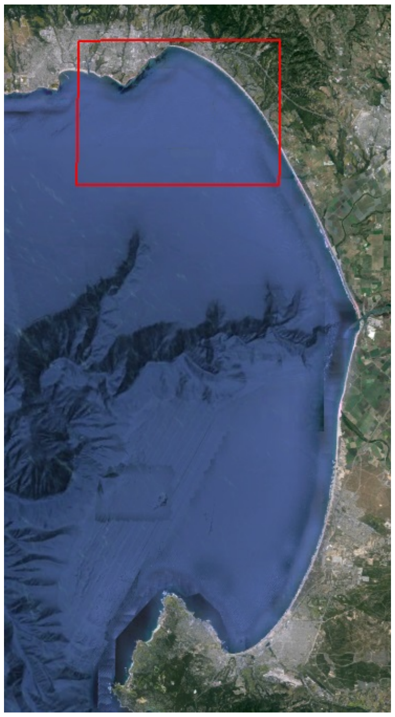
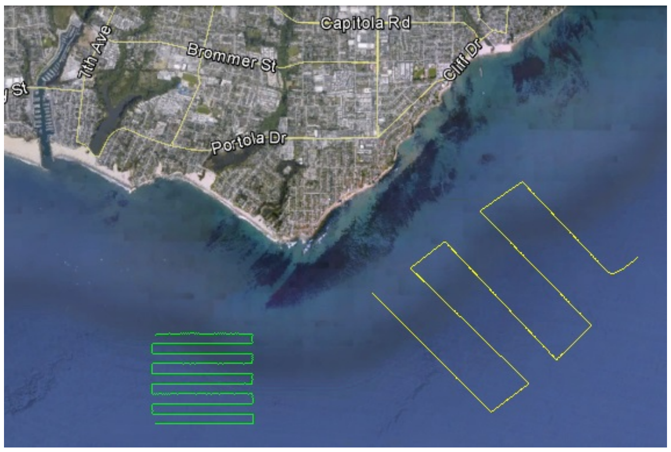


Figure 5.11: Fig.1 from [34] which shows a triangular sampling mission examining the outflow of the Elkhorn Slough into the Monterey Bay (black transits). The red lines and black dots are sampling done by other vehicles.

along the California coast past Monterey Bay, it induces a counter-clockwise current flow within the bay. As a result the northern region, highlighted in Fig. 5.12a, is home to several interesting environmental and biological events and has been an active area of study as seen in [84] and [85]. This area is a common location for both algal blooms as well as large temperature or salinity gradients, events that marine scientists are interested in studying.



(a) A region of scientific interest in Monterey Bay. Algal blooms and large temperature/salinity gradients have been commonly found here. Map data ©2015 Google, SIO, NOAA, U.S. Navy, NGA, MBARI, CSUMB SFML, CA OPC.



(b) A close-up of the area around Soquel Point, in Santa Cruz, CA. Two missions were run here, one studying algal blooms (left, green) and another to detect salinity and temperature fronts (right, yellow) missions. Map data ©2015 Google, CSUMB SFML, CA OPC.

Figure 5.12: Satellite imagery of the areas investigated by the SeaSlug.

For these missions, a marine conductivity-temperature-depth (CTD) sensor was added as the scientific payload. It measures both the water salinity and temperature. This is a common sensor used for both vertical profiling and surface data. Demonstrating the SeaSlug's capabilities with this sensor gives a reasonable baseline of usability for the scientific capabilities of the SeaSlug. This sensor is described in more detail in Section 2.2.2.

5.3.1 Algal Bloom

For the first scientific mission with the SeaSlug, it was planned to simulate a harmful algal bloom (HAB). This environmental phenomenon occurs when algae grows dense enough in an area to damage the ecosystem. These are short-term events and are correspondingly difficult to study as they grow and disperse over the course of a few weeks. Most research institutions have their missions planned and resources allocated several months in advance, and so find it difficult to study these events.

No active HAB was ongoing during the desired mission timeframe, and so one was simulated. In this scenario it was assumed that an active HAB existed at the mission location, shown as the smaller green trace Fig. 5.12. To simulate coverage of this algal bloom a 1.0km patch of water was densely sampled with a boustrophedon pattern with the CTD sensor.

The mission totaled 10.9km in length and took a total of 2 hours and 12 minutes to complete. The mission waypoints and vehicle track are shown in Fig. 5.13. During the mission, the vessel averaged a ground speed of 1.4m/s and traveled a total distance of 11.2km. The extra 300m traveled by the vessel is due to environmental disturbances pushing the vessel off of the desired track and the resultant corrections. This mission was run in the morning of April 17th, from approximately 11:00 to 13:00 local time. Initially there was little wind and small

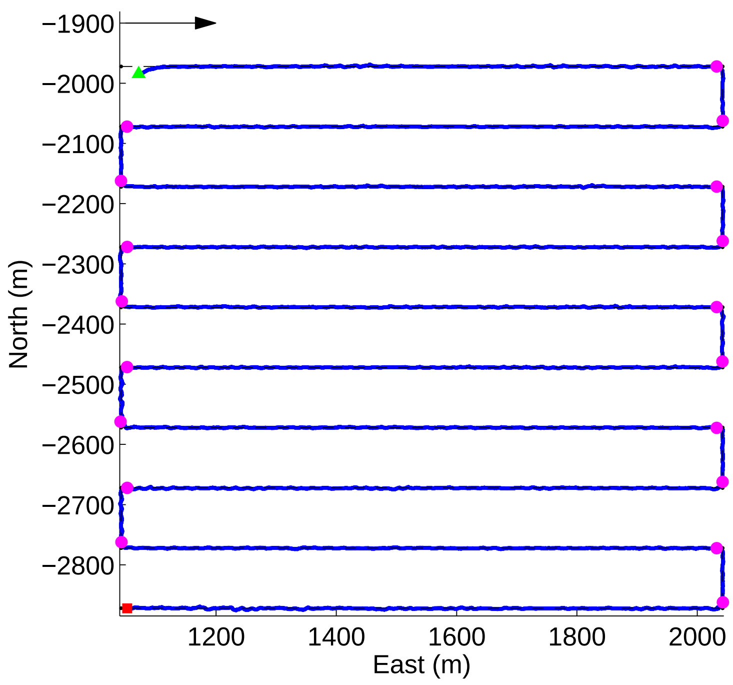


Figure 5.13: Position plot of the vessel in local coordinates. The green triangle and red square indicate the start and end of the mission. The larger magenta points are where the vehicle switched waypoints.

0.5-1.0m swells from the north-west-west (Beaufort scale 2). By the time the mission finished, the wind had picked up to 6-8m/s and the swells had increased in frequency and height to 1-2m (Beaufort scale 4).

Here it should be noted that the vehicle was run at maximum throttle, which resulted in an average water speed of only 1.4m/s. While wave action and environmental factors can affect water speed, the 0.5m/s reduction from the peak observed waterspeed is the result of the CTD sensor mounted in the rear payload bay. During initial testing a dummy sensor payload was mounted in the sensor bay and it has a smooth bottom that matches the shape of the hull exactly. Replacing that with the CTD sensor therefore adds significant drag. Further work in constructing hydrodynamic mountings for the CTD, and any additional sensors, could significantly reduce their impact on water speed.

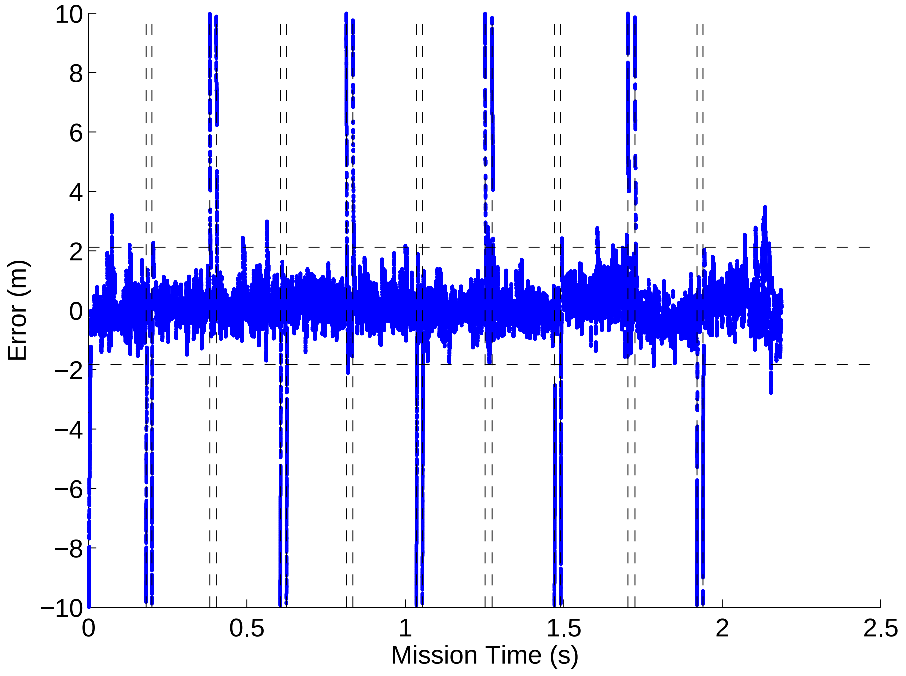


Figure 5.14: Crosstrack error for entirety of the Algal Bloom mission.

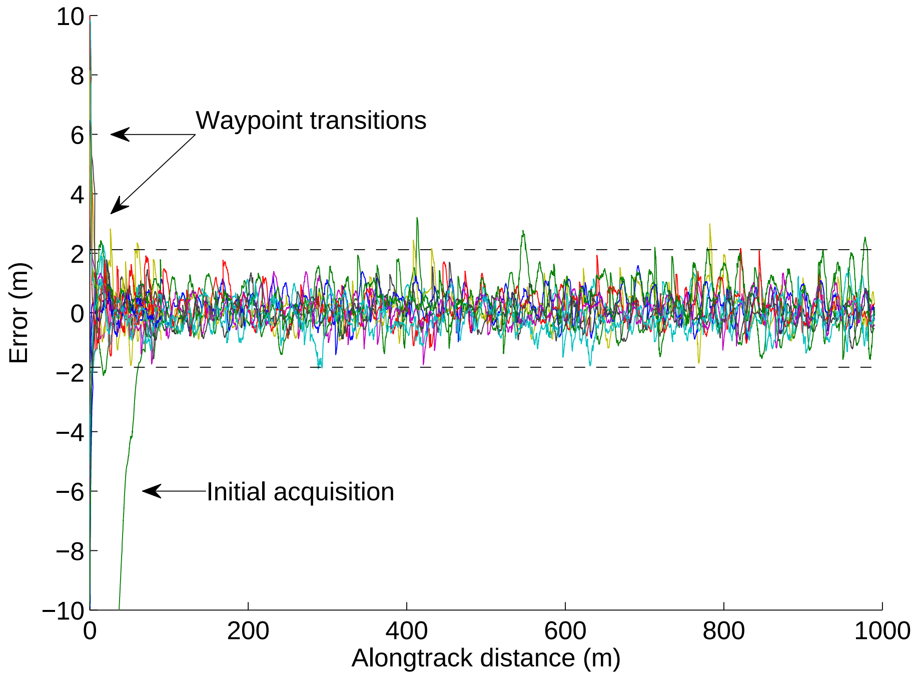


Figure 5.15: Crosstrack error for each path segment overlaid on top of each other during the Algal Bloom mission.

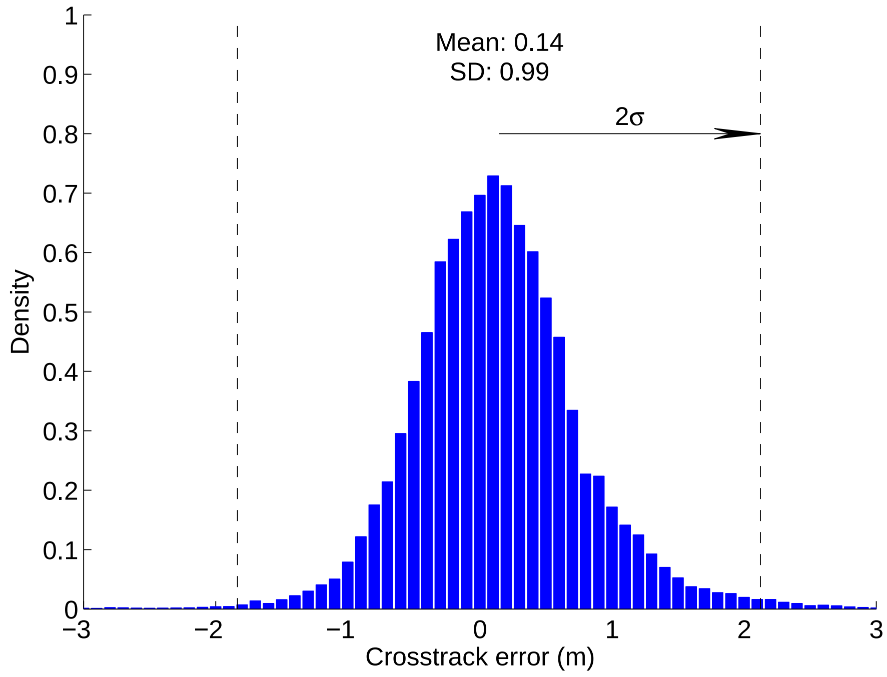


Figure 5.16: Histogram of the crosstrack error throughout the Algal Bloom mission.

System performance as measured by the crosstrack error is similar to the waypoint navigation tests. The weather was initially comparable to the first waypoint tests but had worsened significantly by the end of the mission. This is noticeable in the later portion of the total crosstrack-error plot shown in Fig. 5.14. The wind and waves were from the north-west and made the western transits much harder for the vessel. The last two western legs (during the 1.5hr to 1.75hr and 1.9hr to 2.24hr time segments) show the increased crosstrack error compared to the eastern segment between them, supporting this conclusion.

Fig. 5.15 shows all crosstrack errors overlaid on top of each other by line segment. While crosstrack errors occasionally exceeded 2.0m, they were generally within ±1.0m. Even with the poor performance during the last two western transits, the crosstrack error is rarely above 2.0m, with the waypoint transitions being a large cause of these. A complete error distribution histogram is shown in Fig. 5.16. It should be noted that this is still better precision than required for marine sensing missions and is more than adequate for complex navigation missions.

While the CTD sensor was installed and onboard, it did not operate correctly and no data was obtained from it. Instead data from the DST800 water speed sensor was used as it also reports the water depth and temperature at 1.0s intervals. While it is not a scientifically-calibrated sensor, it nonetheless provides data representative of what could have been collected on this mission. The water depth data should also be accurate enough to provide to NOAA for integration into their existing water depth maps. Fig. 5.17 shows a position plot with the collected data.

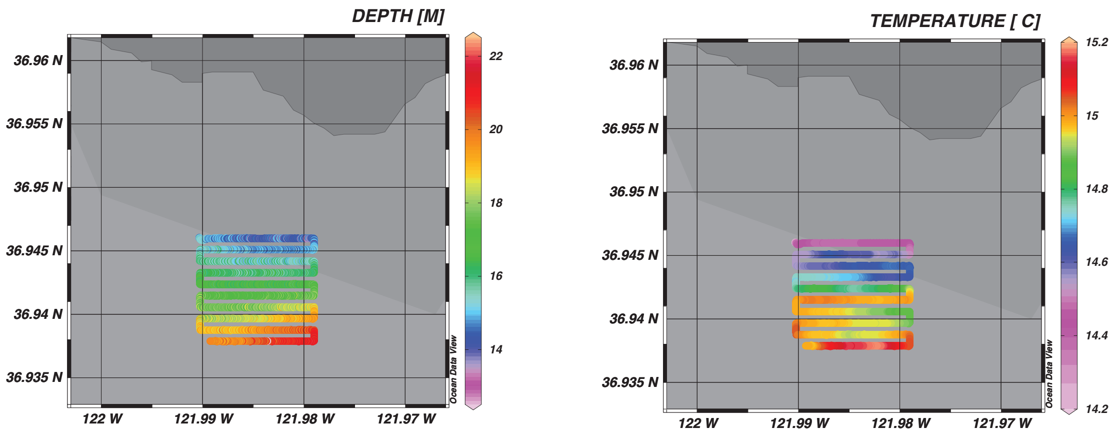


Figure 5.17: Water depth and temperature as reported by the onboard DST800 sensor during the Algal Bloom mission. The left plot shows the vehicle position, the upper-right plot shows water depth, and the lower-right plot shows the water temperature.

The recorded water depth readings are comparable to those as measured by the scientific community. They also increase as the sampling moved south, which is expected as the shoreline was roughly directly north of the test area. The bathymetry lines shown in the plot indicate that the sea floor gradient is towards the north-east, which is supported by the sensor data. Towards the south-east corner of the sampling area, however, it switches to a north-west gradient.

The water temperature data is a little more interesting, though exact conclusions are hard to draw as this sensor was not designed for scientific sensing. Before the mission it was reset to its initial factory calibration and the sensor is advertised as having an accuracy of 0.5°C. The sensor detected a gradual warming of the water by the end of the mission. Although there was full sun during the entire mission, it's unlikely that solar heating would cause such a significant rise in surface temperature over just a few hours. The small spots of slightly cooler water during the transit instead suggest that there were multiple small thermal fronts in the area, possibly slowly migrating during the mission.

5.3.2 Front Detection

An additional data collection mission analyzed the water temperature and salinity in the region of interest highlighted in Fig. 5.12a. The goal of this mission was to detect gradients in the temperature or salinity of the surface layer. This is referred to as *front detection*, following from the term for large gradients in atmospheric effects.

This mission was run in the morning of May 1st of 2015 from approximately 11:00 to 13:00 local time. Fig. 5.18 shows the waypoints and vehicle position during the mission in the vehicle's local coordinate frame. Fig. 5.12b shows the location of the Front Detection mission (the yellow trace on the right) off Soquel Point in Santa Cruz, CA.

The mission path consisted of roughly 1.5km-long transits spaced about 0.5km apart, totaling 10.0km. The actual mission length as executed was 10.5km and took 2 hours and 33 seconds to complete, with an average groundspeed of 1.1m/s.

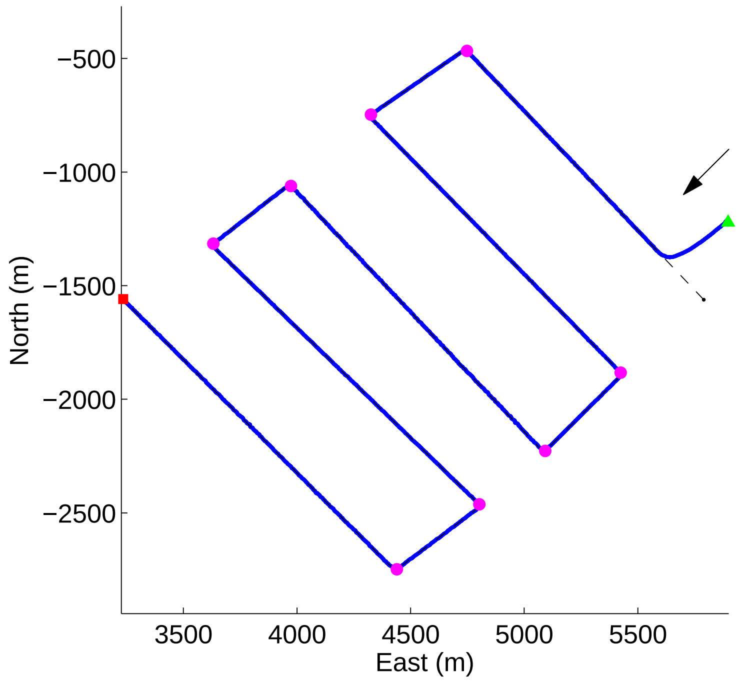


Figure 5.18: Position plot of the vessel in local coordinates. The green triangle and red square indicate the start and end of the mission. The larger magenta points are where the vehicle switched waypoints.

This ground speed is significantly slower than for previous missions because the CTD sensor significantly increased hull drag and there was more wave action. The weather during this mission was worse than during the Algal Bloom mission, with a consistent 1-2m swell with smaller waves of about 0.3m. The wind was also significant at approximately 2.5m/s out of the north-west (Beaufort scale 2).

The combination of smaller waves with the large swell resulted in larger crosstrack error during the last two north-west transits. This is noticeable in the crosstrack error plot in Fig. 5.19 in the two longer transit legs (1.0hr to 1.5hr and 2.1hr to 2.6hr). The south-west transits before and in between have noticeably smaller crosstrack errors.

The CTD sensor was operable on this mission and collected data on both water salinity and temperature, with both shown in Fig. 5.22. The temperature data is

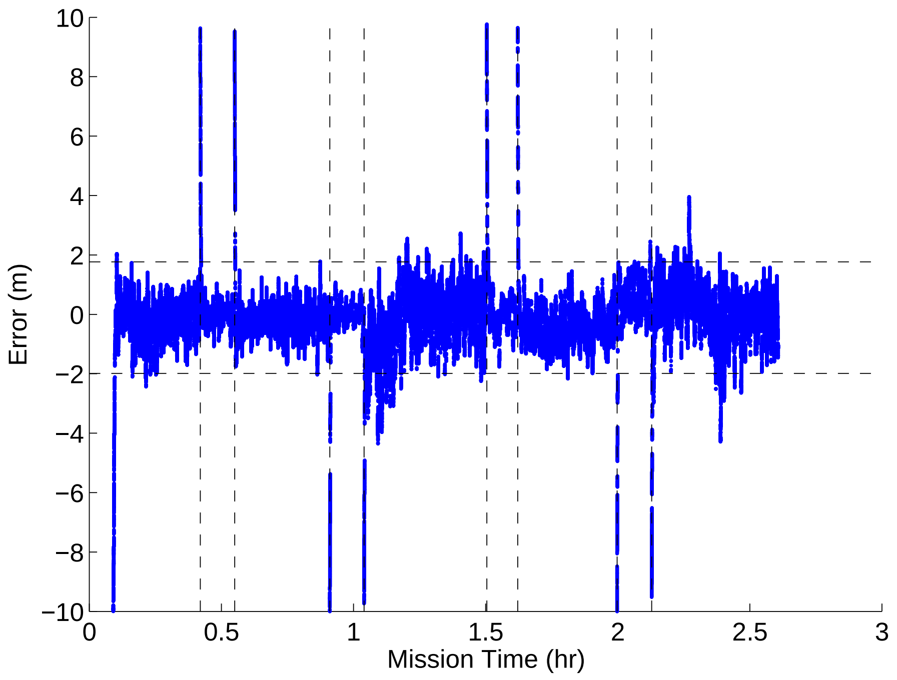


Figure 5.19: The overall crosstrack errors for the during of Front Detection mission.

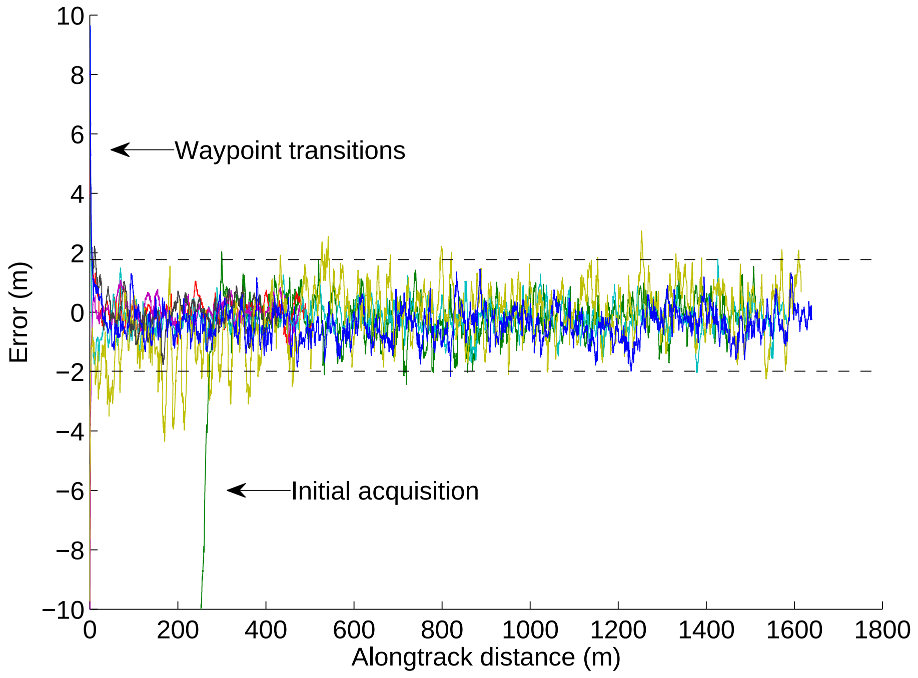


Figure 5.20: The crosstrack errors for the Front Detection mission all overlaid over each other for each line segment.

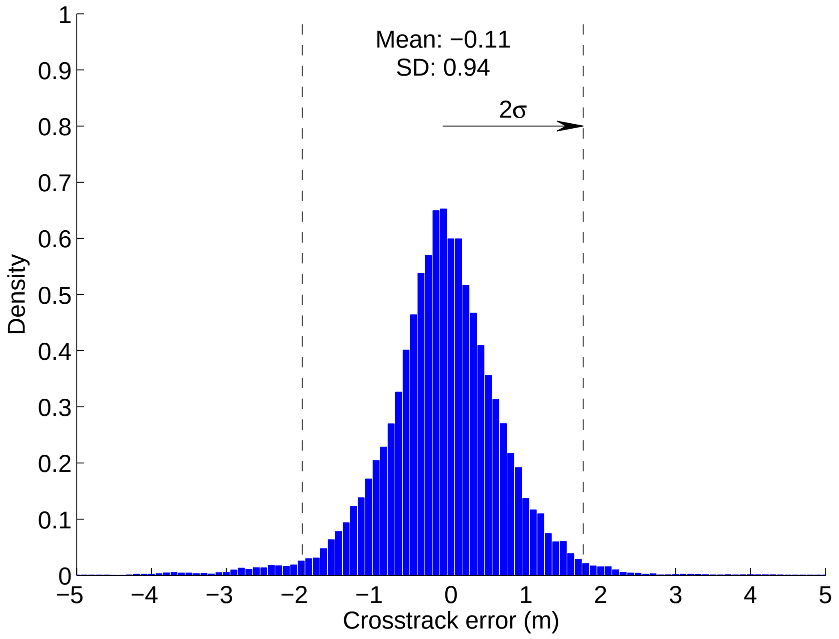
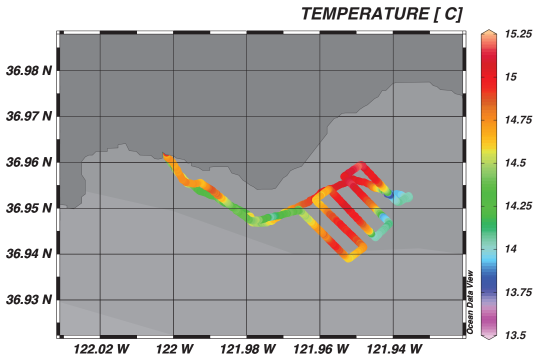


Figure 5.21: A histogram of the crosstrack errors during the Front Detection mission. The initial line acquisition, defined as the period from mission start until the crosstrack error is less than 2.0m, has been ignored.

especially interesting as it shows multiple fronts. There is a warmer segment of water near the harbor mouth, which is expected due to the runoff of the harbor. A thermal front also exists as a warm region that runs north-east through the main sampling area. On both sides it is surrounded by water approximately $0.5 - 1.0$°C cooler.

The salinity data does not reveal any fronts over the sampled area. However, near the harbor entrance salinity does decrease noticeably. Salinity is usually lower near the shore due to freshwater runoff, but it is especially pronounced near the harbor mouth. The use of freshwater for cleaning boats, possibly combined with a freshwater source that drains into the harbor, is the likely cause of this reduced salinity.



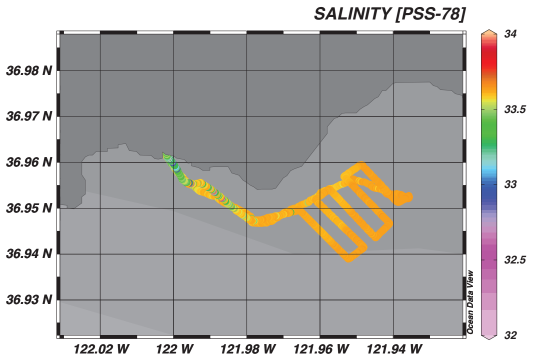


Figure 5.22: Water data as captured by the CTD sensor during the Front Detection mission. The left plot shows water temperature indicated by color. On the right is a position plot of salinity, again indicated by color.

5.4 Conclusion

The missions results presented here were obtained over multiple days in 2014 and 2015 and demonstrate reliable operation of the SeaSlug over many hours, and multiple missions. Both the system itself and its remote interfaces were robust to the harsh environment and allowed for a wide variety of missions to be undertaken while at sea. It was even possible to change mission parameters as necessary during testing.

Throughout these missions the performance of the L_2^+ algorithm was notable, as it had never been tested on a surface platform in littoral waters. It proved itself quite capable with the SeaSlug almost always capable of following complicated waypoint paths to within ± 2.0m in many weather and sea conditions. This far exceeds all manned and underwater systems used by oceanographers today and is comparable or better than other ASVs.

Using this capability a variety of missions were run that measured the salinity, temperature, depth, and other key oceanographic parameters in scientifically interesting areas and produce high-fidelity data in an automated fashion (though for safety purposes a chase boat was monitoring the SeaSlug at close range at all times).

The combination of the system's reliability across sea states, path-following performance, and payload integration capabilities tests demonstrate that the SeaSlug is ready for longer data-collection missions with a larger suite of oceanographic sensors.

Chapter 6

Power Analysis

6.1 Introduction

As the SeaSlug relies solely on electric propulsion, its endurance is directly related to onboard energy storage and the power use of the system. Its onboard power capacity was chosen to provide more than a day of operation at maximum speed. This provides an operational range of about 246km and supports a wide range of short-term data collection missions.

While other ASVs have an endurance of months or more, the SeaSlug has more modest endurance goals of weeks of continual operation. This was chosen as the right balance between system cost and system utility. The short mission durations therefore necessitate a base of operations for the SeaSlug where it receives maintenance, payload changes, and battery charging. The SeaSlug does not aim to replace either the manned vessels or the long-range unmanned vessels used in oceanography, but instead to compliment them as a rapidly-deployable, medium-duration mobile sensor platform.

Though most ASVs use electrical control systems there is a dearth of information on their power use and energy harvesting performance in the literature.

This chapter provides data on the power use of the SeaSlug and its potential for extending mission range with the addition of solar panels.

6.2 Power Use

Power usage data was recorded for both battery banks during the missions presented in Chapter 5. Measurements were done directly by digital multimeters that reported the battery voltage and the current draw. This section analyzes the different subsystems on board and how they contribute towards the total power budget of the SeaSlug.

6.2.1 Control Electronics

The control electronics on the SeaSlug operate directly off the 12V battery bank. There is some variability with their power use as some systems have multiple modes of operations. For example, the power use of the datalogger varies depending on whether it is receiving data and writing it to the SD card. But overall during an autonomous mission the power use will be nearly constant.

The total power use of the control electronics has been directly measured as 8.6W. This power use is broken down by component in Table 6.1. The power use for all components is the maximum power use found under all of its operating modes, and as such is a conservative estimate of total power use.

The sum of the power use per-component only totals 8.46W, slightly less than the 8.59W measured for the entire system. While part of this is due to measurement error, the CAN bus does contribute to system power use. During transmission of dominant bits (binary 0), the CAN bus is is held at a 5V differential. Assuming that dominant bits are 50% of the total transmission traffic, and the

Component	Power Use
3DR Radio	0.75W
Datalogger	0.43W
DST800	0.50W
GPS200	0.50W
IMU CANode	0.44W
Power CANode	0.62W
Primary CANode	0.62W
RC CANode (w/ RC receiver)	0.80W
Rudder CANode	0.77W
Tokimec VSAS-2GM	1.50W
WSO100	1.53W

Table 6.1: The power use of the control electronics that run off of the 12V battery bank.

network is properly terminated with 60Ω resistance, then its power use is approximately 0.2W.

6.2.2 Rudder

The rudder stepper motor is driven by the Applied Motion 2035 motor driver board. This board has three operating modes: Driving, Idle, and Disabled. In the Driving mode the board drives the motor at a 2A per phase, resulting in 23.90W of power use. When the board has not received a step input for 1.0s, the board switches to its Idle mode and holds the motor with 1A per phase. In this mode it draws 10.46W. If the motor driver board is disabled, it uses 2.49W.

The power use of the rudder was analyzed during the Algal Bloom mission. The L_2^+ algorithm has been tuned for crosstrack error performance such that the vehicle follows the desired trajectory quite closely. This can be problematic when there are large environmental disturbances such as the wind and waves. The rudder commands are frequently changing and the rudder controller is rarely idle long enough to enter the power-saving mode, entering it only 0.01% of the time. This rudder activity is quite constant and 88.6% of the time the rudder command is changed within 0.1 seconds of when it last stopped.

6.2.3 Propeller

Fig. 6.1 shows how the water speed correlates with power use and demonstrates how the propeller motor is the largest variable in the SeaSlug's power use. When idling, the ACS300 driver board consumes 3.04W of power. As the commanded current is increased the total power use reaches 115.2W.

This data was collected in the Santa Cruz Harbor by averaging the waterspeed and actuator battery power use during both an up-harbor and down-harbor transit. Making the assumption that there is little current across the harbor, this provides an accurate approximation of the true water speed. During this testing neither of the sensor payload bays were in use resulting in the higher 1.9m/s speed at maximum throttle.

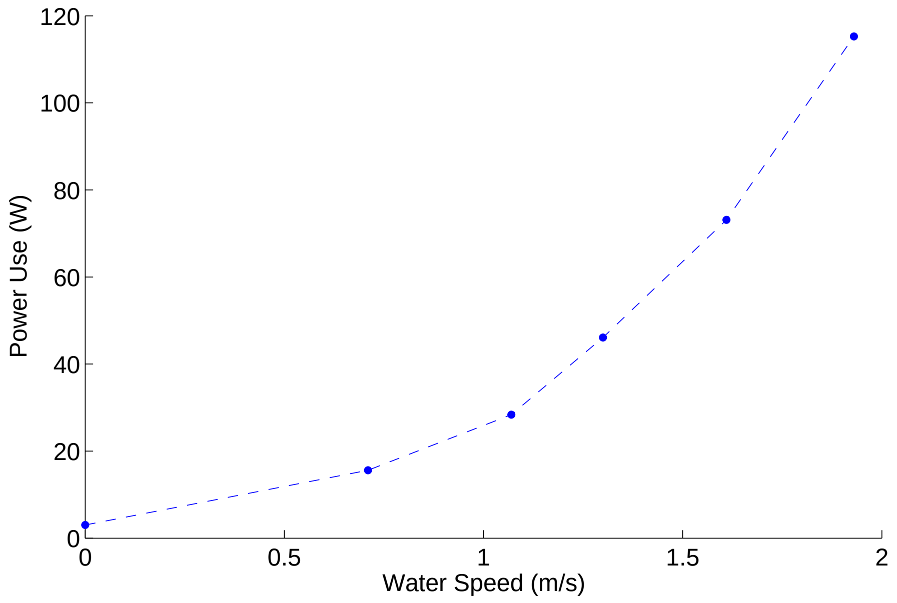


Figure 6.1: Propeller power use as a function of water speed.

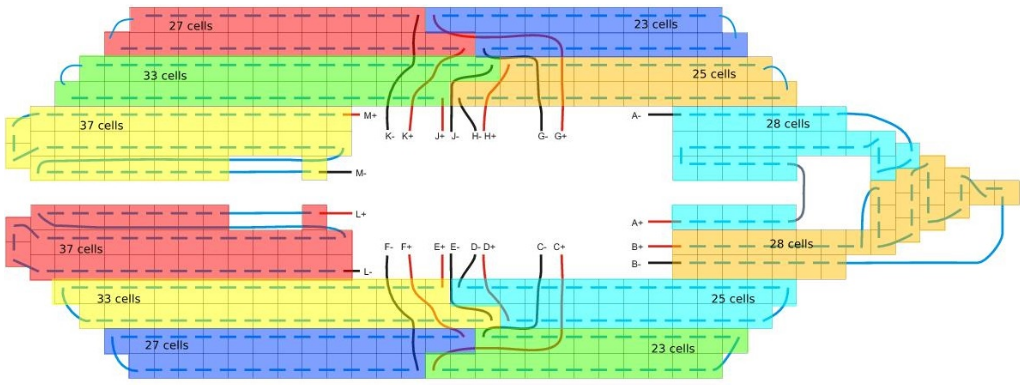
For scientific missions it is generally not imperative that the vessel move at high speeds. Many ocean events that oceanographers study evolve slowly and scientists are generally more concerned with covering surface area versus completing the mission within a specific time window. Existing ASVs have seen successfully used on long-duration ocean missions while averaging speeds as low as 1.2m/s (within the range of the SeaSlug's operating speeds). But even lower vehicle operating speeds are still capable of performing useful data collection missions, allowing for reduced power use and extended mission duration.

6.3 Energy Scavenging

The initial construction of the SeaSlug had individual solar cells mounted on the deck as shown in Fig. 6.2. A total of 412 solar cells were used, connected electrically into 15 cell groups. Each cell provides 3.1W of peak power from the 125mm square Sunpower A-300 mono crystalline silicon cell, with a total conversion efficiency advertised as 21.5%. At peak solar exposure this solar cell arrangement could potentially provide 1.23kW of power.

Unfortunately these solar cells did not prove robust to the elements or to continued maintenance of the vessel and they were removed. To evaluate replacement panels a low-cost 20W solar panel was added to the top of the main hatch. There is no part number or company information for the panel, but its low cost implies similarly low performance. It is therefore used here as a conservative estimate for how much power can be collected at sea.

The solar panel was connected to the Xantrex SW-MPPT60-150 charge controller which charged a 12V battery independent of the other two battery banks. This charge controller uses a maximum power point tracking (MPPT) algorithm to optimize the output of the solar panel. The Xantrex charger provides a Xanbus



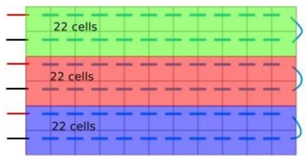


Figure 6.2: The original solar cell arrangement of the SeaSlug, with the vessel's bow on the right. The center hatch also has solar cells mounted on it as shown in the lower-left.

interface which is compatible with the main CAN bus of the SeaSlug.

This panel was tested as part of the Algal Bloom mission on the open ocean. This mission lasted several hours in the late morning on a bright, sunny day in Monterey Bay. The resulting power input data is shown in Fig. 6.3.

For the first 24 minutes the panel output averaged 11.9W, which is 60% of the possible performance of the panel. After this time the batteries were fully charged and the MPPT algorithm used by the charge controller adjusted the system load to reduce the power output of the panel to 7.6W. As this reduced output was not related to the solar input of the panel, a clear sunny day can then be inferred to provide 11.9W of power while mounted on the SeaSlug. The panel is therefore capable of outputting 62.5W/m^2.

The panel was also tested during a foggy day, during the Front Detection mission. That day there was a dense fog in the morning that the sun only started

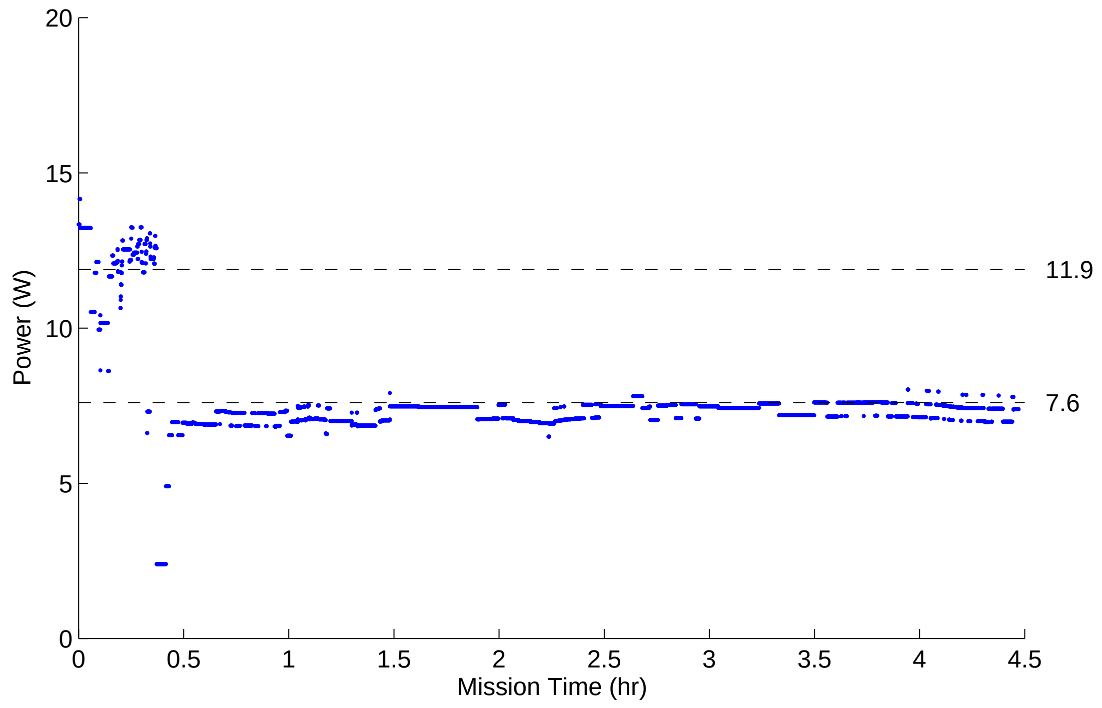


Figure 6.3: Solar power input during the Algal Bloom launch. It was a clear sunny day during this test. The average before the MPPT algorithm reduced panel output is 11.9W. The mean for the entire mission is 7.6W.

to break through about halfway through the test. The power generated by the panel is shown in Fig. 6.4. Even though there was intense fog, the panel averaged 9.5W during the entire mission.

6.4 System Endurance

The SeaSlug has separate battery banks for the controls electronics and the actuators. Their different capacities and loads result in different endurance estimates for each. For the electronics battery banks, 2.3kWh of energy is available. This is capable of powering the constant 8.59W load of the electronics for 274 hours (11.4 days).

The 24V battery bank provides substantially more capacity at 5.28kWh be-

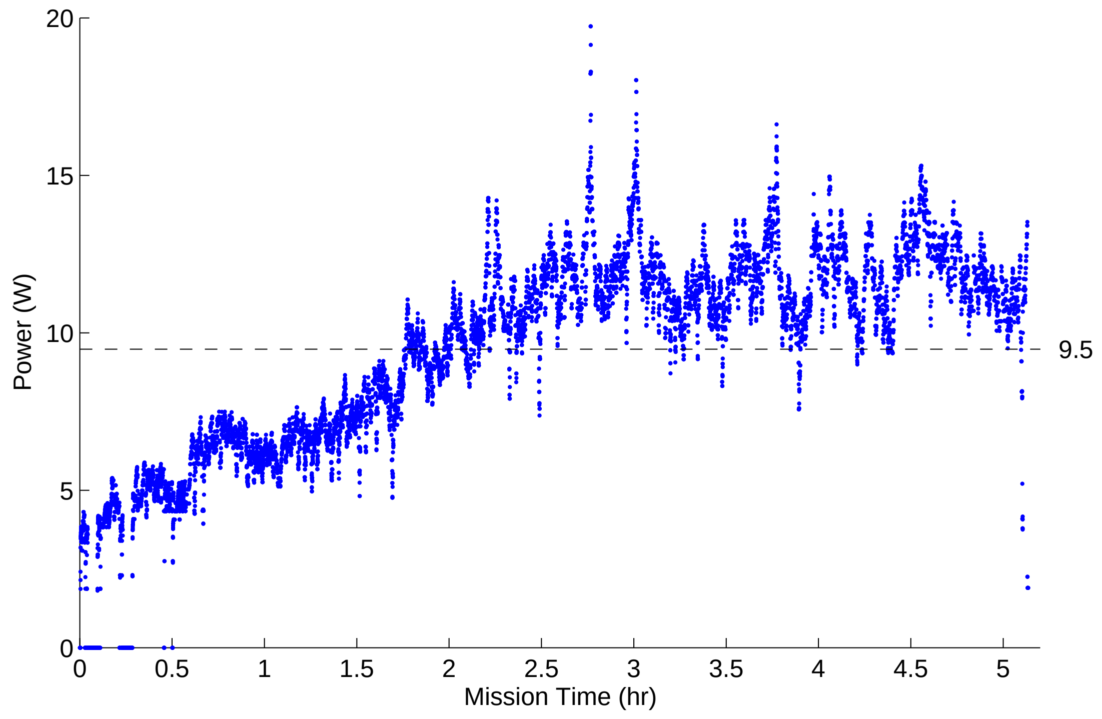


Figure 6.4: Solar power input during the Front Detection missions. It was a cloudy day, though the sun started to peak through the clouds about halfway through the mission.

cause of the larger power draw of the actuators. As previously shown, the power use of the actuators is dependent on the rudder and throttle commands. The actuator battery endurance is shown in Fig. 6.5 as a function of the commanded throttle, shown as the water speed value. The relationship between the throttle values and vessel water speed was previously established (see Fig. 3.2 in Chapter 3).

Table 6.2 shows the relationship between vessel speed, power use, and endurance. At maximum throttle, the SeaSlug is capable of 38 hours of operation. This increases to 134 hours when running at one-third throttle. And when the system is idling, the actuators have 391 hours of power. This idle value is representative of the system maintaining its position, where both the prop and the rudder are rarely active and nearly always in their low-power idle modes. Though

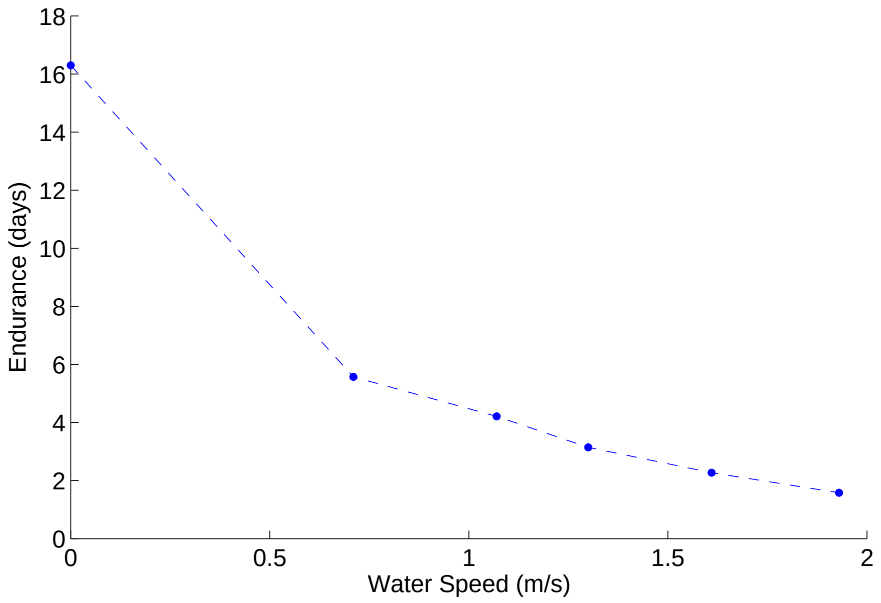


Figure 6.5: Actuator battery bank endurance as a function of waterspeed.

Water Speed (m/s)	Actuator Power Use (W)	Endurance (days)
0	13.5	16.3
0.71	39.5	5.6
1.07	52.3	4.2
1.30	70.0	3.1
1.61	97.0	2.3
1.93	139.1	1.6

Table 6.2: Data used in Fig. 6.5 showing the actuator power use, water speed, and endurance of the actuator battery banks across a range of throttle values.

the actuators are capable of idling for 16 days, the control electronics will run out of power in 11.4 days, limiting the total system endurance.

The vessel speed controls the trade-off between the distance traveled and mission longevity. Fig. 6.6 shows the maximum distance the SeaSlug can cover at different water speeds. This is determined by the energy use of the system as a function of distance, which is also shown. If optimizing for endurance, then the system should be kept idle, but if maximizing for distance covered, a water speed of 1.1m/s provides the maximum distance of 389km.

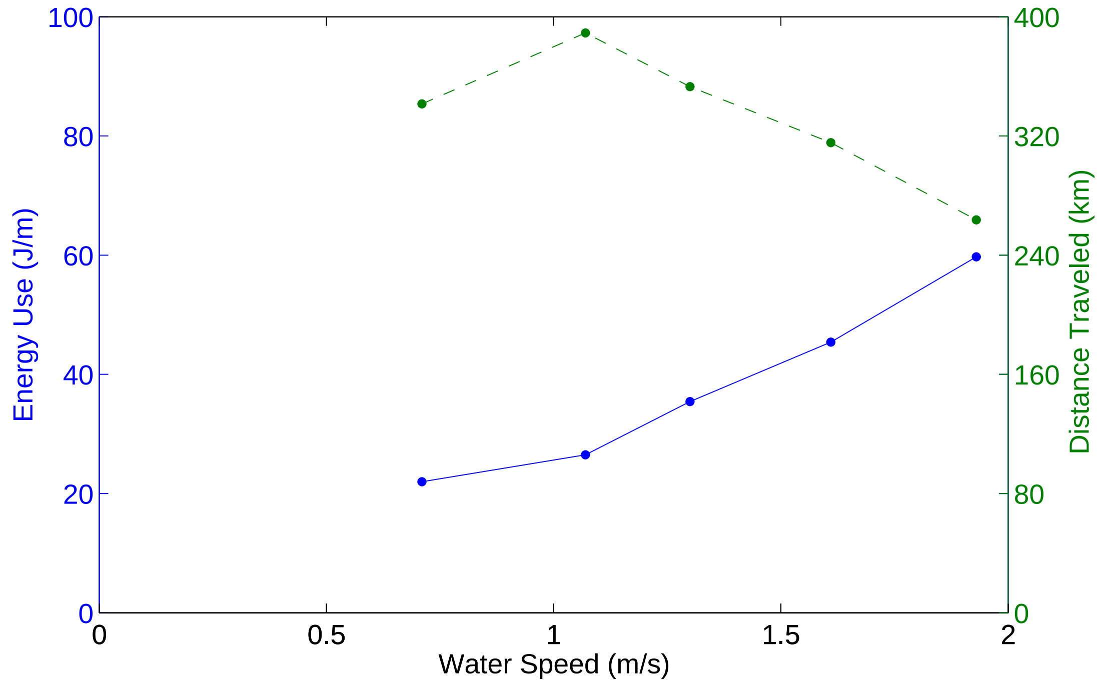


Figure 6.6: Energy use per meter (solid blue line) and total mission distance (dashed green line) for different water speeds.

6.4.1 With Solar Panels

With the measurements of solar panel performance and system energy usage, simulations were run to determine the possible mission endurance of the SeaSlug in various scenarios. For these simulations the biggest factors affecting system endurance are the number of usable sunlight hours in a day and the operating speed of the vessel. For simplicity these simulations ignored the separate power rails and assumed a single power supply for the entire vessel. Additionally, the water speeds of the SeaSlug assumes no additional drag from the sensor payload; the vessel speed is a function of payload hydrodynamics as well.

The tested solar panel averages 62.5W/m^2 output on a sunny day. The SeaSlug has a total of 9.33m^2 of deck space, therefore the potential power output is 583.1W (roughly half of the original power input from the original cells). Assuming that there are six hours of sunlight every day, the SeaSlug is capable of operating at full speed for 107 days, traveling a toal of 17,565km. Applying these same calculations to the foggy day data from the Front Detection mission, yields a total of 465.2W. Again assuming six-hour days, the SeaSlug is capable of 7.9 days of operation at full speed, traveling a total of 1,299km.

This is, however, an unrealistic scenario. In Santa Cruz, California and the surrounding Monterey Bay it only averages 226 sunny days a year and one-in-three days will be lacking in sunlight. If this scenario is tested in simulation, the results look like Fig. 6.7. Here the SeaSlug was able to last for more than 146 hours (6.1 days) before exhausting its battery reserves. During this time it traveled a total of 999km.

By reducing its run speed, the SeaSlug can operate for even longer periods without sunlight. Fig. 6.8 shows the a simulation of the vessel's power reserves where the weather is three no-sun days followed by a day with only three hours of usable sunlight. By running at 1.6m/s during the sunny hours and 1.1m/s otherwise, the SeaSlug can attain 231 hours of runtime (9.6 days) and travel a total of 941km.

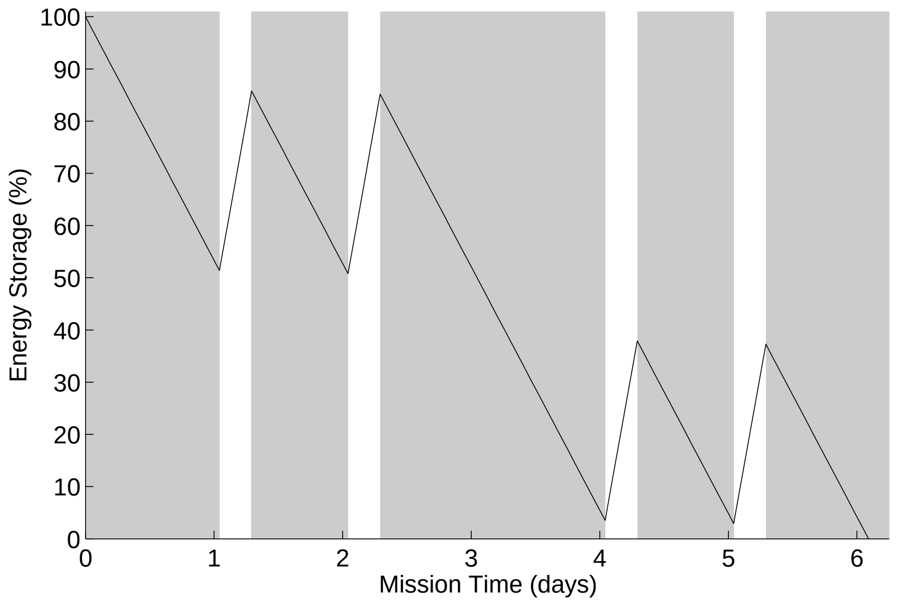


Figure 6.7: Energy reserves of the SeaSlug operating at full throttle during a simulated mission that starts at 8am where two out of every three days have six hours of usable sunlight. Unusable sunlight hours are indicated by the grey background.

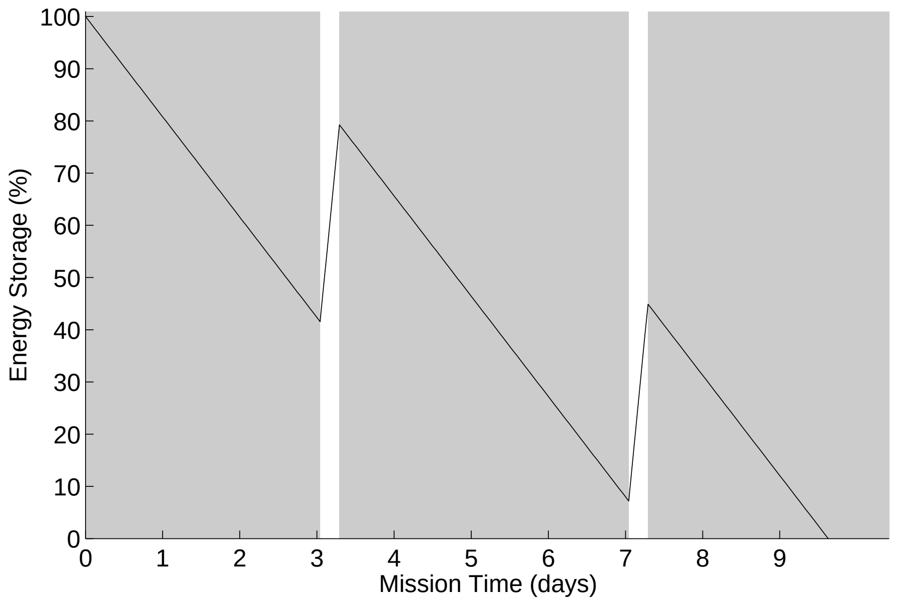


Figure 6.8: Energy reserves of the SeaSlug during a simulated mission that starts at 8am with three hours of usable sunlight every three days. Daytime water speed is 1.6m/s and nighttime water speed is reduced to 1.1m/s. Unusable sunlight hours are indicated by the grey background.

While these simulations assume that the SeaSlug is continuously under throttle, its endurance be lengthened even further by idling to wait for sufficient sunlight to recharge. Some data collection missions will not require the vessel to be constantly moving and can take advantage of these very low power modes.

For example, a common near-shore event is the discharge of anthropogenic runoff several days after a rainfall. As the exact time of this event is unknown, it can be expensive to study with manned vessels. This mission is difficult for most existing ASV's (e.g. the Wave Glider cannot operate in shallow waters). The SeaSlug, however, can comfortably idle in the shallows near runoff locations and collect data across a shoreline transit every few hours. In this way the precise time of the event does not need to be known but the data can still be collected.

Studying algal blooms is another situation where endurance is more important than operating speed. These sudden population explosions of algae occur suddenly and only last a few days or weeks. Both manned and longer-endurance vessels are usually already tasked with missions and cannot be reassigned, making this suitable to the shorter-endurance SeaSlug. It can repeatedly survey the bloom over its lifetime, idling in between surveys or when sunlight is unavailable and its energy reserves are low.

6.5 Conclusion

During missions the largest consumers of power for the SeaSlug are the actuators. Therefore adjustments to the throttle can extend the full-speed runtime of 38-hours to up to 17 days at slower speeds. This allows the SeaSlug to be immediately deployed as a support vessel to complement an existing fleet with additional sensor capabilities.

System endurance can be even further enhanced with the addition of even very low-cost solar panels to the deck. With the tested panel, 600W of power is available during bright sunlight, far more than the total power use of the system at full speed (148W). Even with a significant number of cloudy days the system can stay at sea for weeks at a time. If there are extended resting periods or a large number of sunny days, this duration could be even longer. For the two data-collection scenarios previously discussed, mission runtime could be infinite because the system remains idle the majority of the time. The addition of solar panels to the SeaSlug allows it to out-perform any existing all-electric ASV for mission endurance.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

In summary, this dissertation has presented the SeaSlug as an embodiment of the design goals laid out in Section 1.2. The specific contributions of this work are:

> *The design and implementation of a modular system architecture suitable for an autonomous surface vessel that is both highly-extensible and applicable to other robotic platforms.*

The SeaSlug architecture uses a CAN bus to provide a shared communications network between subsystems. Careful design of custom subsystems, along with the use of COTS components with well-defined interfaces, provides a highly-modular system that facilitates the modification, replacement, and addition of sensors, actuators, and other subsystems. Few assumptions about the hardware exist outside of the individual subsystems and this architecture is applicable to other surface vessels, or even ground vehicles.

The modularity of this architecture was evaluated both during development and with the testing of a solar power subsystem. This subsystem was installed for the evaluation of solar panels with an accompanying CAN-enabled solar charging unit. Necessary changes were mostly within the specific subsystem, providing trivial integration with few modifications to the rest of the system.

> *The design, development, and validation of an open-source ASV that is capable of both general engineering experimentation and scientific data collection.*

The internals of the SeaSlug were developed as part of this work, as the external hull design was previously commissioned by Willow Garage [92]. This development included the design and implementation of the modular architecture and the selection of components for providing sensor data and integration with the actuators. Most components were COTS, which facilitated their service or replacement. A few components were custom designed as suitable alternatives were not available at the desired price. This included a data logger for recording telemetry at high data rates from the primary computer. A single-board computer (SBC), called the CANode, was also developed to serve as a computation platform and to integrate components to the onboard CAN bus. While both components were designed for use aboard the SeaSlug, each are generally applicable to other robotic systems. This includes the operator interface, which uses a low-cost weatherproof tablet that runs generic robotic interface software that is open-source and can be readily adapted to any system.

The complete system was evaluated over a multitude of experiments in both a sheltered-water harbor environment and the littoral waters of Monterey Bay, California. This included both performance evaluations of the system's control algorithms and scientific data collection experiments. During the data collection missions valuable scientific data on water depth, salinity, and temperature was found.

> *Analysis of power consumption of the subsystems of an electrically-propelled ASV and its potential for solar power generation to allow for*

extended duration missions.

Power consumption of the individual components of the SeaSlug was measured over all operating modes. This includes the COTS components, custom hardware, and the actuators. While power use is usually provided by the documentation for COTS parts, these measurements are generally inaccurate. The custom hardware and actuators had a variety of operating modes that required specific measurements. The resultant measurements supported estimates of total system endurance while station keeping and at a variety of travel speeds.

The potential for extended-duration missions was explored through the addition of a solar panel. This low-performance panel was evaluated on two missions to estimate its true output on both sunny and cloudy days. Combined with the collected power use data, estimates can be made of system endurance if additional panels are added. The addition of even low-cost solar panels would extend mission endurance to at least several days, dependent on the vehicle speed and weather conditions. Some realistic scenarios support infinite deployment times.

The adaption and analysis of the L_2^+ control algorithm to an ASV.

The L_2^+ control algorithm previously developed by the Autonomous Systems Lab has been extensively tested in simulation and on an UAV. It is vehicle-agnostic and therefore applicable to other vehicle types, but had never been tested on any other system. This work demonstrated the necessary system and algorithm modifications required for operation onboard the SeaSlug, which has both different control surfaces and system dynamics than the original test vehicle. This algorithm was capable of following the desired mission trajectory with less than 2.0m of crosstrack error. This performance was reproducible during a variety of missions and across a range of sea and weather conditions.

> *The design and development of a robust simulation environment for an ASV that is suitable for algorithm and system testing on land.*

The logistics of live testing of the SeaSlug necessitated a robust simulation environment. Aditionally, this environment was necessary to develop and tune control algorithms prior to testing. This was facilitated by the use of Simulink for both control algorithm development and the simulation environment. By using Simulink, the central control algorithm operates onboard the SeaSlug exactly as it does in simulation. A software-only simulator allows for testing control algorithms independently of the rest of the system. This is supplemented by a replay simulator that allows recorded telemetry from missions to be played back in order to debug algorithm failures or to test potential modifications on real sensor data. An interface computer, built on the common CANode hardware, bridges the software simulator environment to the SeaSlug's CAN bus. The actuators can be added to this CAN bus to allow for their actual dynamics to be used in simulation. When the actuators are not on the bus, their dynamics are simulated instead by high-fidelity models. This testing environment allows for both testing of the system and algorithms before a launch as well as evaluating the system performance after live deployments.

Though all of the above goals were attained as part of this work, the combined contribution of all of them is more valuable than their individual summaries suggest. The SeaSlug has been shown to be capable of completing scientific missions in littoral waters lasting several hours. It has been designed to be extensible to other mission types through its flexible payload capacity. And significantly longer missions are possible with the addition of low-cost solar paneling. With the conclusion of this work oceanographers now have another tool at their disposal for studying the world's oceans.

7.2 Future Work

The SeaSlug platform has been proven to be robust, reliable, and capable of a variety of open-ocean missions, lending itself to further development and experimentation. Planned future work seeks to extend its capabilities as a mobile data-collection platform with additional improvements to its endurance, sensor capabilities, and level of autonomy.

1. The guidance, navigation, and control algorithms of the SeaSlug are currently limited to the rudder. Extending these algorithms to the propulsion subsystem can both improve mission endurance and add mission capabilities. The throttle command will be altered to be a fixed ground or water velocity instead of motor current. Missions can then specify this velocity, selecting ground- or water-relative velocities dependent on the mission objectives. Commanding water speed would be used when power efficiency is required, while commanding ground speed would allow the vessel to take advantage of environmental effects for power savings, such as the wind or water current.

2. Solar power has been shown to be a viable option to extend mission durations significantly, as presented in Chapter 6. Panels that are efficient, low cost, and robust to the marine environment will be selected and installed on the deck. A solar charge controller, which regulates the power input of the solar panels and the power output of the batteries, will be added that is robust to the marine environment and can interface with the CAN bus. The resultant solar power subsystem can then be monitored by the Primary Node and human operator.

3. Roll control of up to $\pm 60^\circ$ is possible with the 91kg of movable ballast

mounted in the SeaSlug. Most of the necessary hardware for this has already been acquired, but additional work is required to complete the addition of a roll control subsystem to the vessel. An additional CANode is required to interface between the Primary Node and the ballast motor controller so that a ballast angle or vessel roll angle can be commanded. The efficiency of solar panels is dependent on the cosine of the incidence angle of the sun, as described by Lambert's cosine law [53], and reducing that angle provides an appreciable increase in solar power generation. Optimal trajectory planners have already been developed for UAVs that accounts for power drain as a function of flight speed and power gain as a function of roll angle [51]. This algorithm could be adapted to the SeaSlug, as those inputs map exactly once roll control has been developed. Given that the SeaSlug can maintain a roll angle outside of turns, unlike a UAV, modifications may make the algorithm even more suitable to the SeaSlug than the UAVs it was originally designed for.

4. Marine scientists are often interested in collecting sensor data at a range of depths. When vertical profiling is done at different locations it provides a three-dimensional view of the ocean. Manned vessels and underwater gliders are both capable of vertical profiling, and commonly work in concert to collect sufficient data. Though the SeaSlug is currently unable to collect data at depth, it is technically feasible. The addition of this functionality would allow the SeaSlug to replace both manned vessels and underwater gliders. An automatic winch will be added to one or both of the payload bays. This allows for the payload to slot securely into the bay when not deployed but to also lower down to depth. Both the Lizhbeth ASV [46] and the C-Enduro [14] demonstrated the feasibility of vertical profiling with a

winch on an ASV.

5. Sustained autonomous deployments require reliable long-range communications for system monitoring and mission retasking. Most ASVs rely on the Iridium satellite communications network for two-way communications with home base. A modem will be integrated into the SeaSlug that can provide a 1200bps connection. As this connection has much less bandwidth than the 57600bps radio connection currently in use, MAVLink may not continue to be usable as it consumes significant bandwidth. Datastream compression or modifications to the MAVLink protocol will be investigated to support these lower baud rates so that QGroundControl can continue to be used.

6. Sensor payloads are currently limited to stand-alone packages that include a power source and data storage capabilities. These sensor packages are usually limited to less than 48 hours of operation. Longer deployments will require external power and data storage facilities, which can be provided by the SeaSlug through an external connector. Many sensors run at the same 12V that is already available from the electronics battery bank. Additionally, RS232 is the common interface for these sensors, which is supported by the datalogger. The addition of a dedicated datalogger for the sensors can provide many months of data storage. Integrating this datalogger with the remote communications interface will allow for preliminary analysis of the gathered scientific data. This connection will also be available to any winch-mounted sensors through a contact-less induction system.

7. Obstacle detection and collision avoidance are essential for the heavily-trafficked harbors and near-shore regions targeted by the SeaSlug. A variety of detection mechanisms have already been tested in existing ASVs including

radar [2], sonar [42], and vision [28] [47] [40]). Some marine vessels directly transmit their position and velocity with an Automatic Identification System (AIS) transmitter, which is mandatory for vessels over 22m [1]. An AIS receiver could extend any other sensors with additional sensing abilities. A range of sensors will be integrated and tested for obstacle detection. This will then feed into a high-level trajectory planner to provide collision avoidance capabilities. The COLREGs [23] provide standard rules for interactions with other manned vessels, and have been successfully integrated into a trajectory planner for an ASV [52]. This algorithm allows for adhering to the COLREGS while also accounting for additional unguided obstacles and the vehicle's mission plan.

8. With the short mission durations supported by the SeaSlug, docking and launching procedures will be common. These are delicate procedures, with little room for error, and currently require manual control. Extending the obstacle detection and path planning algorithms to support these procedures would remove some of the last vestiges of required human interaction.

Appendix A

Datalogger

A.1 Introduction

With the development of the SeaSlug, recording telemetry for later analysis was crucial to debugging some algorithm and system failures. This extra debugging information eventually became too large to transmit to the GCS for logging. Therefore a hardware datalogger was added to the SeaSlug that logged telemetry over a hardline connection to the Primary Node.

Several dataloggers were evaluated as part of the SLUGS project [58], but none were found to be reliable at the bandwidth required. The commercial and open-source dataloggers that are available are either very high cost or have high data loss rates. Therefore the SLogger (short for SLUGS Logger) was developed for the high-bandwidth data-logging needs of the SeaSlug, supporting logging UART data streams at up to 115200baud with data loss rates below 0.4%.

A.2 System Architecture

With the CANode (introduced in Section 2.3.1) already designed to serve as an extensible computation platform, it was used as the base for the SLogger. The dsPIC33E processor was used as it provides both the computation performance and the necessary UART and SPI interfaces. While this platform includes additional hardware for the SLogger, its compatibility with a 12V power rail made it an ideal initial development platform. The system has been designed to be sufficiently hardware-agnostic, with the only alteration necessary for operation on different hardware being the pin mapping. Work is already underway in the Autonomous Systems Lab to create a smaller and lower-cost version for use in testing of the SLUGS autopilot.

With the CANode already providing most hardware requirements, only a microSD card connector was required. A CANode shield, containing both additional connectors and the microSD card slot was developed. The shield is the red circuit board shown in Fig. A.1.

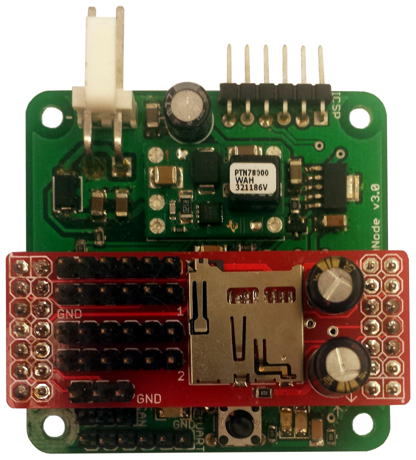


Figure A.1: The datalogger hardware. The base CANode is the green circuit board. Additional connectors and the miroSD card are on the red shield.

The software architecture is shown in Fig. A.2. Incoming UART data is handled by a hardware ping-pong buffer, commonly known as double-buffering. Once one of these buffers is full, the UART begins to fill up the other buffer.

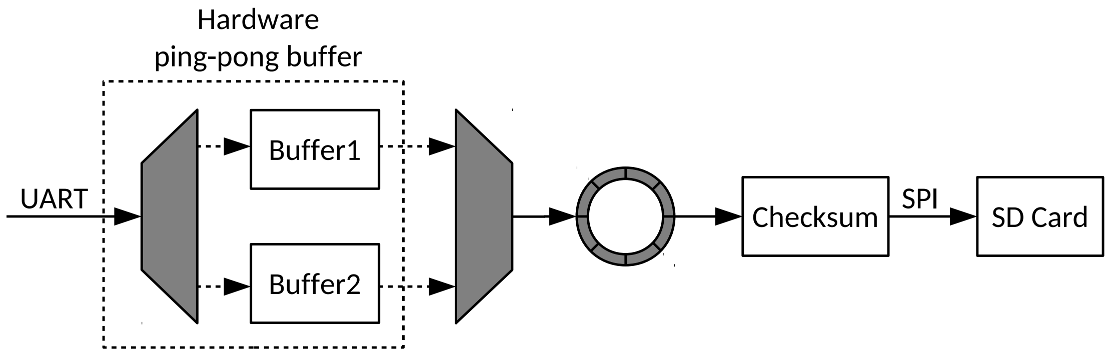


Figure A.2: The datalogger architecture overview. Data is received over UART into a circular buffer that then empties onto an SD card.

Once a UART buffer is full, it triggers an interrupt that copies the data into an internal software circular buffer. The circular buffer provides storage for up to 20 of these filled UART buffers, which uses all available processor RAM (the dsPIC33EP256MC502 used has 32KB of RAM). During testing this has proven sufficient to avoid buffer overruns.

Before copying the data from the circular buffer to the SD card, it is first encapsulated in a packet. This packet is the same size as a block on the SD card, 512 bytes, as that is the base unit of data transfer with the card. The encapsulation process uses 6-bytes to store a file identifier and checksum with the data, which is therefore 506 bytes. The file identifier is used to determine the blocks that make up a file, as overallocation is used to provide the necessary performance. The checksum is an XOR of all data bytes in the packet and provides basic integrity validation. The complete encapsulation format is shown in Fig. A.3.

SD cards are commonly formatted with either the FAT16 or FAT32 filesystems. These are widely supported by all operating systems and were used with the SLogger for this reason. Microchip provides a library that is capable of interacting

%	ˆ	File ID (1-byte)	Data (506-bytes, big-endian)	Checksum (1-byte)	%	&

Figure A.3: The data packet format used for storing 512-byte data blocks in the log files. Fields without an explicit size are 1 byte.

with either FAT filesystem and provides file- and directory-level access to the data on the SD card. In also encapsulates all SPI peripheral interactions as part of this.

A.3 Functionality

Configuration of the SLogger is done with a `config.txt` file stored in the root directory of the SD card. This file was designed to be extensible, to support logging through a variety of peripherals. Currently only the UART is supported, and the configuration file specifies which input pins are receiving the data and at what baudrate. Should the specified configuration of the SLogger fail for any reason, a red LED is illuminated and the system waits until an SD card with a proper configuration file is inserted. Once the system is configured an amber status LED blinks to indicate that the system is now operational.

When the system is logging data, it is robust to failure from both power and SD card issues. The card is always left in a state where already-written data is recoverable, though there may be junk data included at the end of the file from the way the system over-allocates files when they are created. Therefore if the card is accidentally removed, power is cut, or the SD card fails to accept new data, the data already written to it should still be usable.

When powered-on the datalogger scans through the SD card searching for log files that match its file naming scheme and starts a new log file with the subsequent number. In this way existing data is never overwritten and data is recorded as it is received. This process is also done on insertion of an SD card.

In addition to storing a log file, the SLogger also records a metadata file for storing status messages during logging. In this file messages are timestamped with the number of seconds since power on. The log is started with how the system started up, from either a card insertion or a processor brown-out. Additional informational messages are appended as well, such as when the circular buffer overflows, enabling debugging should there be a systemic issue during a long data logging process.

The data recorded on the SD card is encoded in the packet format described in Fig. A.3. To obtain the original logged data, an extraction script is provided with the logger that outputs the original binary log of the received data. It checks the file identifier and checksum as it scans the file. Once the file identifier changes or the end of the log file is reached, the file has finished processing. Any packets that fail to pass the checksum validation process are dropped from the final binary log.

Appendix B

Attitude

B.1 Introduction

In order for a vehicle to correctly control itself it is necessary that it understands its own orientation in 3-dimensional space. This is known as the vehicle's *attitude*, which defines the body frame relative to an inertial navigation frame that serves as a reference, which is generally the Earth.

There are many ways to represent the final vehicle's orientation, though there is usually a convention for the various fields of science and engineering. This extends to the method of converting coordinates from the body frame to the navigation frame or vice-versa. This is because the conversions are a sequence of rotations around intermediate axes, which can be defined in multiple ways.

This chapter focuses on one of the more fundamental attitude representation systems, *Euler angles*. These angles are likely the reader's first exposure to attitude representations and is common among the hobbyist and DIY community because of its simplicity.

B.2 Euler Angles

Euler angles are one of the most common and simplest ways to represent attitude. They are a triplet of angles that represent separate rotation angles to define a coordinate frame relative to the inertial frame. While the coordinates can be either right- or left-handed, right-handed coordinates are more commonly used and only they are discussed for simplicity.

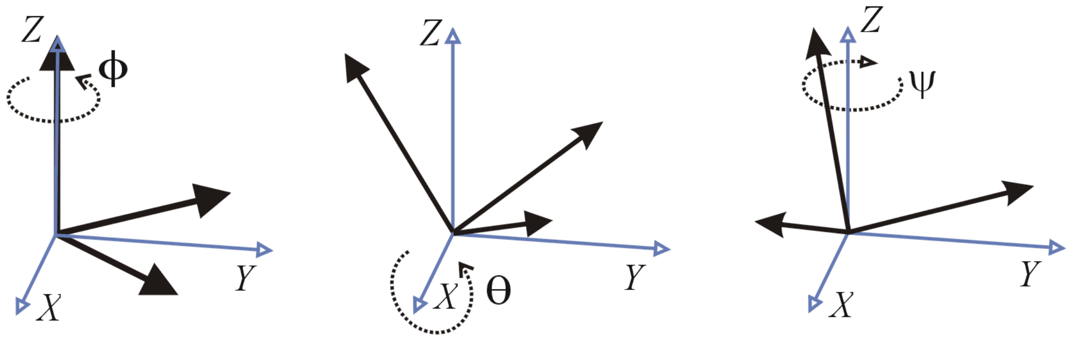


Figure B.1: Shows the extrinsic Euler angle rotation sequence Z-X-Z. The first rotation rotates around the reference Z-axis. The second is around the reference X-axis. And the third is around the reference Z-axis again. This can fully describe any orientation in 3-dimensional space.

There are twelve different ways to define the rotations given that the coordinate frames are three-dimensional and as such require three separate rotations. These twelve transformations are split into two groups: Euler angles and Tait-Bryan angles. Euler angles, also referred to as proper Euler angles, use the same axis for the first and last rotations: (Z-X-Z, X-Y-X, Y-Z-Y, Z-Y-Z, X-Z-X, Y-X-Y). Tait-Bryan angles, sometimes also referred to as Euler angles, use different axes for all three rotations: (X-Y-Z, Y-Z-X, Z-X-Y, X-Z-Y, Z-Y-X, Y-X-Z).

These rotations are further broken up depending on whether extrinsic or intrinsic rotations are used. Extrinsic rotations are more intuitive as they define the axes about which the rotations are performed as one of the three fixed axes of the reference frame. Fig. B.1 shows the common Z-X-Z rotation sequence using extrinsic angles.

Intrinsic angles on the other hand use axes from three different coordinate systems. These coordinate systems are defined as the position of the rotating coordinate system after each set of rotations. So the first rotation rotates the XYZ_0 coordinate frame to XYZ_1, the second from XYZ_1 to XYZ_2, and so on. Fig. B.2 shows the Z-X-Z rotation sequence using intrinsic angles.

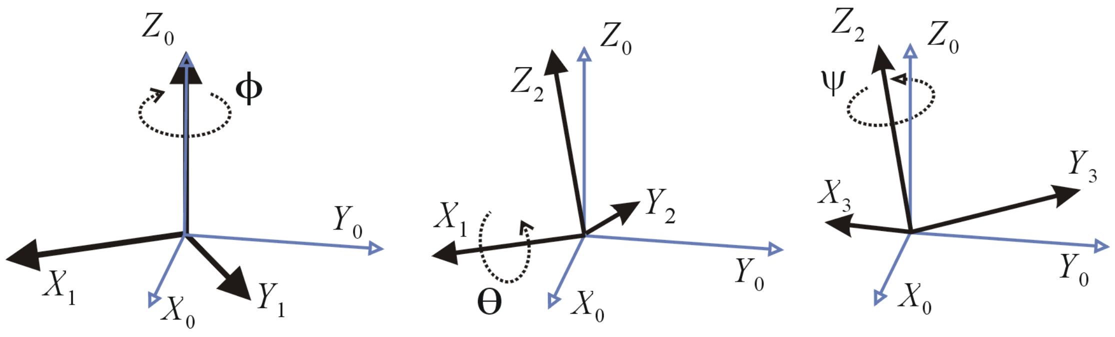


Figure B.2: Shows the intrinsic Euler angle rotation sequence Z-X-Z. The first rotation rotates around the reference Z-axis. The second is around the rotated X_1-axis. And the third around the now doubly-rotated Z_2-axis again.

B.3 Euler Angle Conventions

Tait-Bryan Z-Y-X angles using intrinsic rotations have become a de facto standard for both aerospace and nautical work. This is for two reasons: First, yaw is relative to the reference frame. This makes the heading a consistent reading for inertial observers and pilots alike. Secondly, the use of intrinsic angles make reasoning about the pitch and roll angles intuitive and easy to reason about for human pilots.

As shown in Fig. B.3, the transformation from the global reference frame to the body frame then follows as:

1. Rotate about the global Z axis by ψ, call this yaw.

2. Rotate about the transformed Y_1 axis by θ, call this pitch.

3. Rotate about the doubly-transformed X_2 axis by ϕ, call this roll.

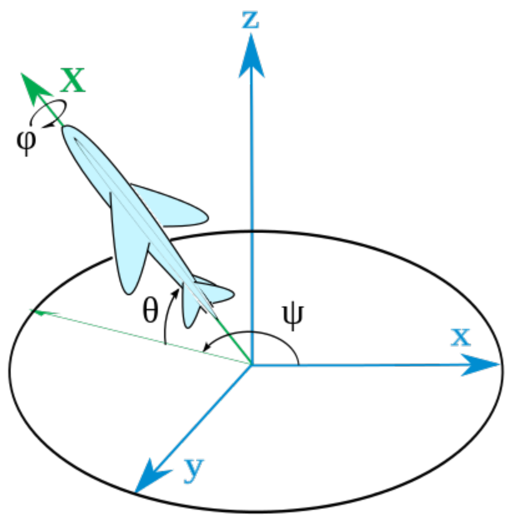


Figure B.3: The Tait-Bryan Z-Y-X intrinsic rotation sequence for describing attitude follows a similar rotation sequence to that shown in Fig. B.2. First a yaw rotation is applied to the Z_0 axis. Then a pitch rotation is applied to the Y_1 axis. And finally a roll rotation is done about the X_2 axis. “Plane.svg” by Juansempere is licensed under CC BY 3.0.

B.3.1 Gimbal lock

While Euler angles are intuitive to reason about, they do have a singularity and so not every position is uniquely identifiable. If the pitch of the vehicle is 90°, then the yaw and roll axes are aligned, which is referred to as *gimbal lock*. Since they are aligned, their sum is the only thing that can be determined and not either value individually. Therefore the system does not actually know how it’s oriented in space without additional inputs.

While this is a problem for systems like satellites, and airplanes in rare circumstances, this is not an issue for ground vehicles. During normal operation their pitch does not approach 90°. For the SeaSlug, the wave action might causes pitching as high as 30°, but for pitch angles near the singularity it is likely the vehicle is already lost and as such controlling it is unnecessary.

B.3.2 Using Euler angles

While the above description of Euler angles is sufficient for an intuitive understanding of them, the math supporting it deserves a more formal presentation. As an attitude representation is really a single rotation, it can be expressed as a single rotation matrix. With the yaw, pitch, roll representation intrinsic rotations are used, so subsequent rotations rotate about the transformed axes from the previous operation. What this means is that representing the final rotation can simply be done by left-multiplying the rotation matrices in the order of yaw, pitch, and roll:

$$\overset{n\longrightarrow b}{R} = \overset{n\longrightarrow b}{R}(\phi, \theta, \psi) = R(\phi)R(\theta)R(\psi) \tag{B.1}$$

,

These individual matrices can be easily calculated by using the standard 3x3 rotation matrices. The first rotation is for the yaw angle, represented by ψ, which is about the inertial z-axis:

$$R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The next rotation is for pitch, given the symbol θ, which rotates about the now-transformed Y_1 axis:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

The roll angle, ϕ, is applied in the final rotation about the Z_2 axis:

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

By multiplying the above matrices together as shown in Eq. B.1, the complete matrix for rotating from the navigation to the body frame is shown below. Note that the shorthand for the cos() and sin() functions, as $c()$ and $s()$ respectively, is used due to limited space.

$$\overset{n\to b}{R}(\phi,\theta,\psi) = \begin{bmatrix} c(\psi)c(\theta) & c(\theta)s(\psi) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\theta)s(\phi) \\ s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \tag{B.2}$$

B.4 Euler angle rates

Now while the IMU reports attitude as Euler angles, it reports rotation rate relative to the body frame, oriented as shown in Fig. B.4. It is important to note that these rates are not equivalent to the derivative of the yaw, pitch, and roll angles. To convert the gyro sensor data into yaw, pitch, and roll angular rates requires extra derivation that does not use the rotation matrix from Eq. B.2.

Converting the sensor angle rates to Euler angle rates relies on rotating the

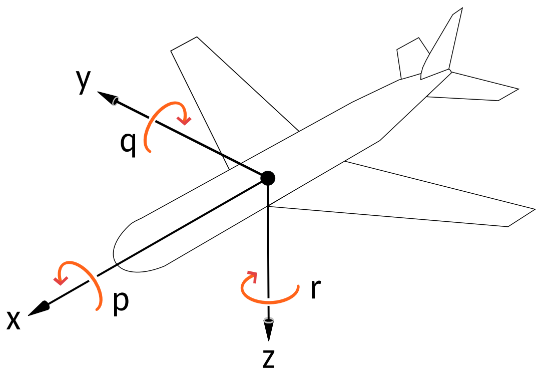


Figure B.4: The principal axes used for representing coordinates in the body frame are x, y, and z. The rate of rotation around these three axes are p, q, and r respectively. This work is a derivative of "Yaw Axis Corrected.svg" by Jrvz, used under CC BY. This work is also licensed as CC BY-SA 3.0 by Bryant Mairs.

individual readings back to the appropriate coordinate frame that they are represented in. The pitch rate is defined relative to the Y_2 axis. However, the y-axis gyro reading from the sensor is measured relative to the Y_3 axis, and therefore needs to be rotated. The same process applies to the yaw and roll rates.

The simplest derivation of the transformation matrix is to actually calculate its inverse, the mapping from the Euler angle rates to the sensor rates. The key idea here is that each of the sensor values needs to be rotated from its coordinate frame to the sensor frame. For the roll rate, there is no necessary conversion as it is defined in the body frame. For the pitch rate, that is defined in the XYZ_2 frame and needs to be rotated to the body frame. And for the yaw rate, it needs to be rotated from the XYZ_1 frame to the body frame.

For the following equations the gyro rates of p, q, and r relate to rotations being about the body-frame x, y, and z axis as illustrated in Fig. B.4.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = I_3 \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \overset{2\rightarrow b}{R} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \overset{1\rightarrow 3}{R} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$
$$= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi)R(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\theta) & \sin(\phi)\cos(\theta) \\ 0 & -sin(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{B.3}$$

To obtain the desired conversion matrix requires inverting the matrix from Eq. B.3. It is important to note here that while rotation matrices are orthogonal, the above is neither a rotation matrix nor orthogonal. Because of this the inverse has to be derived and the resultant matrix is shown in Eq. B.4.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$
$$= \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{B.4}$$

Appendix C

CAN bus

C.1 Introduction

The Controller Area Network (CAN) bus serves as the central communications network on the SeaSlug over which all onboard subsystems communicate with each other using a well-defined interface. Most embedded protocols are relatively simple and only define the first and second layers of the OSI model to provide a direct connection between two devices. The CAN bus implements the third layer, the network layer, as well and allows multiple devices to transmit and receive on the same bus. The protocol also defines additional details like arbitration and error handling due to its anticipated use in large networks operating in noisy environments.

Therefore the details of the CAN bus are relegated to this appendix. The CAN bus provides a number of useful features for a large robotic system like the SeaSlug, and these features are described in more details here. Not every part of the CAN standard is detailed here, as there is more detail available from [54] and the CAN standard itself [48].

C.2 Overview

CAN is a message-based, multi-master bus protocol originally developed for automotive applications by Bosch in 1983. It was designed as a communications network for many microcontrollers, without the need for a host computer. It was further defined as the ISO 11898 standard [48].

Each CAN message is called a *frame* and there are 4 different frame types:

1. **Data frame**: Contains a message identifier as well as a data payload
2. **Remote frame**: Requests a message be sent
3. **Error frame**: Transmit by any node when it detects an error
4. **Overload frame**: Injects a delay between the data and/or remote frame.

Messages that can be transmit on the CAN bus fall into four frame types: data, remote, error, and overload. This chapter only details the *data frame*, as that is the only one used in the SeaSlug project. The data frame exists in two different forms, one with an 11-bit identifier (CAN2.0A) and one with a 29-bit identifier (CAN2.0B). Both data frames can be transmit on the same bus and CAN controllers that support the extended data frame also support the standard data frame.

The identifier of a data frame message defines its contents. These definitions exist external to the network as part of a message set and are commonly part of a standard that builds on top of the CAN bus, like NMEA2000 [59].

There is no destination address for messages in the base CAN protocol, and nodes can read all messages written to the bus. Most messages contain the state of a node and are transmit regularly to keep other nodes in sync. The bus can therefore be modeled as a distributed system using a shared memory space where

	Arbitration Field (12 bits)		Control Field (6 bits)			Data field (0..64 bits)				
SOF	Identifier 11 bits	RTR	IDE	res	DLC 4 bits	Data 0..8 bytes	CRC 15 bits		ACK 2 bits	EOF 7 bits

Figure C.1: The components that make up a standard CAN frame. Each field is 1-bit unless otherwise specified. See TableC.2 for details on the individual fields.

each value has a fixed update rate. CANopen [21] expands on this idea and exposes this shared memory model to the developer directly.

C.3 Electrical Specifications

Electrically CAN is a two-wire differential signal, so it is half-duplex as only one node can write to the bus at a time. One of the lines is pulled high to VCC, and is referred to as CANH. The other line is pulled low to GND and is labeled as CANL.

The entire bus is impedance-terminated with 120Ω resistors across the differential pair, one at each end of the network. These resistors not only minimize signal reflection at high bit rates, but also provide voltage stabilization for the bus.

As a differential signal, bits are specified by a voltage difference between CANH and CANL. Nominally a CAN bus operates across the voltage range of 0 to 5V.

To represent a “1” value on the bus, both the CANH and CANL lines are undriven. Both lines therefore have similar values, which should be close to 2.5V, which they converge to when undriven. The voltage difference is specified as between -1.0V and +0.5V.

To represent a “0” value on the bus, the CANH line is pulled high to +5V and

Field Name	**Size (bits)**	**Description**
Start of frame (SOF)	1	Indicates the start of a new frame. Always 0.
Identifier	11	A value that uniquely identifies the information in this message.
Remote Transmit Request (RTR)	1	Dominant (0) for data frames and recessive (1) for remote frames
Identifier Extension (IDE)	1	Dominant (0) for standard data frames, recessive (1) for extended data frames.
Data Length Code (DLC)	4	The number of bytes of data payload for this message.
Data	0..64	Between 0 and 8 bytes of data.
Cyclic Redundancy Check (CRC)	15	The CRC value for this message.
CRC delimiter	1	Recessive (1)
Acknowledge (ACK)	1	Transmitter sends a recessive (1) bit and other node acknowledge this message by overwriting it with a dominant (0) bit.
ACK delimiter	1	Recessive (1)
End of frame (EOF)	7	All bits are recessive (1).

Figure C.2: Fields in a standard data frame message.

	Arbitration Field (32 bits)					Control Field (6 bits)			Data field (0..64 bits)			
SOF	Identifier 11 bits	SRR	IDE	Identifier 18 bits	RTR			DLC 4 bits	Data 0..8 bytes	CRC 16 bits	ACK 2 bits	EOF 7 bits

Figure C.3: The components that make up a standard CAN frame. Each field is 1-bit unless otherwise specified. Unnamed fields are reserved/unused bits and are dominant (0). See TableC.4 for details on the individual fields.

Field Name	Size (bits)	Description
Start of frame (SOF)	1	Indicates the start of a new frame. Always dominant (0).
Identifier B	18	Additional bits making up the complete identifier for the message.
Substitute Remote Request (SRR)	1	Always recessive (1).

Figure C.4: Fields in in an extended data frame that are not in a standard data frame. Note that these fields are not in order, see Fig. C.3 for their ordering.

the CANL line is pulled low to 0V. This is referred to as a dominant bit because it overwrites a recessive bit that is being written to the bus. The bus topology is therefore that of a wired-AND, where if any node attempts to write a “0” onto the bus, it succeeds.

C.4 Timing

Data and remote transmissions are not allowed to be transmit back-to-back, and an *interframe gap* is imposed in between. During this time the bus sits in the recessive state until the gap has elapsed. Overload and error frames do not adhere to this rule, which effectively gives bus errors priority over data and remote frames. This guarantees the network stays consistent, even when saturation or bus errors occur.

Due to the size of CAN buses, both in terms of physical length and number of nodes, care must be taken to account for clock skew between nodes. Each node has its own oscillator and slightly different timings, so phase shifts can occur between nodes. Therefore a spatial synchronization algorithm is required to coordinate timing.

The first part of synchronization is *hard synchronization*, where the internal bit clocks of every node is reset at the falling edge indicating the Start-Of-Frame.

Additional synchronization occurs during a transmission frame. This is done by segmenting a bit time into three parts: Sync, Time Segment 1 (TS1), and Time Segment 2 (TS2). An edge is expected during the Sync segment, while the bit value is actually sampled between TS1 and TS2. Now if an edge occurs outside of the Sync segment, for example during the TS1 segment, the timing for the TS1 segment is restarted lengthening this bit. And if an edge is detected during the TS2 segment, indicating that the next bit has started, so the next bit time is started immediately.

It should be noted that this *soft synchronization* is only done for the next bit time after synchronization was done, so this process can occur often in a network with large phase shifts.

C.5 Data Rates

As a reult of the metadata associated with each data message, the effective data rate of the bus is much lower than its set speed. For example, with a fully-loaded 1Mbit CAN bus with no collisions and the fastest bit timings, the effective data rate for extended messages is only 48.9%. The use of standard frames raises this to 57.7%, but it is still close to only half of the actual transmission speed.

For networks with many nodes transmitting a variety of different message types, congestion further reduces this data rate. For example, the SeaSlug has nine different nodes transmitting on the CAN bus and at least two nodes transmit messages 100 times a second.

C.6 Arbitration and Priority

Message arbitration takes advantage of the wired-AND topology and an implicit message priority to determine which node holds the bus and can transmit onto it. All nodes constantly monitor the bus and attempt to write to it a certain time after the last transmission finished, referred to as the interframe space. This time is equal to 7 bit values and the network is in the recessive state during this time.

Once that time has elapsed, all nodes with messages to transmit attempt to transmit their message on the bus. Since dominant bits overwrite recessive bits, if a node detects a dominant bit when it is writing a recessive bit, it stops trying to transmit and instead switches to receiving. Therefore the one that wins priority is the node that was able to successfully write all the arbitration field bits to the bus. Nodes that lost this round try again to transmit next round. In this sense message transmission on a CAN bus follows a priority queue model.

There are some additional details for arbitration between different frame types. Remote frames with the same identifier as a data frame lose precedence to the data message because of the dominant value for the RTR bit. Additionally, standard frame messages take priority over extended messages, because their IDE bit is dominant.

C.7 Fault Tolerance

Error handling in the CAN protocol has been designed to both detect invalid messages and to also preserve network integrity in the case of a malfunctioning node by removing it from the network.

The first way the network stays synchronized is by the use of the ACK field in each data and remote frame. During this time the network is in the recessive state and it is required that another node indicate that the message transmit correctly by driving the bus to a dominant value. It should be noted that the only thing this indicates is that there is one other node is on the network and it read the message correctly. It does not actually indicate that the message was processed by that node at all. The failure of a node to ACK is an error for the transmitting node.

A fifteen-bit Cyclic Redundancy Check (CRC) is calculated and transmit with every data and remote frame and is verified by receiving nodes. This CRC has a Hamming Distance of six, meaning it can detect up to six single-bit errors or up to fifteen sequential burst errors. This CRC value is not used for error correction, however. For receiving nodes, a message failing the CRC check triggers an error.

Bit stuffing is the process of inserting a single bit of the opposite value once 5 bits of the same value have been transmit. This extra bit is automatically removed by receiving nodes. This is partly done to maintain synchronization, as soft synchronization is only done during bit transitions. Another reason is to prevent large DC biases from entering the network. Outside of the EOF and interspace frame fields, six consecutive identical bit values is then a bit stuffing error.

Additional network errors occur when the bus does not have the expected value. During reception this can be in one of the fixed-form fields (delimiters, EOF, etc.) and is referred to as a *form error*. *Bit errors* on the other hand occur when a transmitting node detects a different bit on the bus than what it transmit.

The above five errors make up all of the errors that are handled by the CAN protocol. As soon as any node detects an error, it interrupts transmission to transmit an error frame. All nodes then discard the message that was being transmit and increment their internal error counters. The transmitting node then restart transmission.

Each node maintains two separate error counters: one for receiving and one for transmitting. There are several rules for incrementing the error counters, and there are three different error states that a node can exist in depending on the error counters. The most important detail is that transmitting nodes increase their error counters faster than receiving nodes. Once their error count is high enough they enter the *Bus Off* state, and stop participating in the network, including ACKing messages. The CAN hardware must be reset by the application.

More details of how errors are handled can be found in Chapter 4.1.4 of [54].

Appendix D

NMEA2000

D.1 Introduction

The NMEA2000 protocol [59] is a protocol that builds upon the CAN protocol, but provides additional specifications for almost all layers of the OSI model. It has become the de facto standard for marine electronics since its introduction, replacing the less-flexible NMEA0183 protocol. Most commercial marine electronics therefore communicate over NMEA2000, which defines both a physical connector and the message set for devices.

The NMEA2000 protocol is unfortunately a proprietary protocol and so is unavailable except to purchasers of the specifications. Though this is unfortunate, it has not lessened the protocol's usefulness in providing a common interface for commercial sensors. Additionally, the community has reverse engineered much of the protocol's message definitions and parsing algorithms.

This chapter describes the most important aspects that the NMEA2000 protocol adds over the ISO11898 standard that describes the CAN bus. It does not provide enough detail to completely integrate with all aspects of the NMEA2000 protocol, but has been sufficient for processing data output by NMEA2000-devices.

D.2 Overview

The NMEA2000 protocol specifies many details at the physical layer. This includes the supported voltage range of 9-16V, which can be provided directly by standard 12V lead-acid batteries. The connectors are required to be one of two different types, depending on the application. Smaller vessels, such as the SeaSlug, only need the smaller of the two cable sets, which support up to 3A of current and can run for distances of 20m. These DeviceNet connectors are also IP67 rated, which is necessary for harsh marine environments.

While the CAN protocol itself can support arbitrary baud rates, J1939 sets a fixed baud rate of 250kbit, which NMEA2000 inherits. This improves significantly on the 4.8kbaud of the NMEA0183 standard it replaced. At this bit rate, the maximum length of the bus is 250m.

Nodes operating on the same NMEA2000 bus support higher-level arbitration than that specified with the CAN standard. Nodes negotiate for themselves a Node Identifier (between 0 and 255). This is used for identifying nodes in node-to-node messages. This also places a hard limit of 256 NMEA2000 nodes on a single CAN bus.

D.3 Message Formats

Messages in the NMEA2000 protocol all correspond to the CAN2.0B standard, which uses the extended 29-bit header. The header is split into different fields depending on which message header type is being used. Protocol Data Unit 1 (PDU1) describes messages which have a destination address while PDU2 headers describes broadcast messages. This is done as shown by Fig. D.1, which splits the 29-bit extended header into the following sections:

- **Priority field** (3 bits): This field takes advantage of how the CAN protocol prioritize messages.

- **Extended data page** (1 bit): Always 0 for NMEA2000.

- **Data page** (1 bit): Always 1 for NMEA2000

- **PDU Format** (8 bits): If this value is 240 or greater than this message is a PDU2 message. Otherwise this is a PDU message.

- **PDU Specific** (8 bits): For PDU1 messages, this specifies the destination node. For PDU2 messages, this specifies an extension to the PGN.

29-bit CAN2.0B Identifier

P	P	P	E	D	F	F	F	F	F	F	F	F	S	S	S	S	S	S	S	S	A	A	A	A	A	A	A	A
Priority			0	1	PDU Format								PDU Specific								Source Address							

Figure D.1: The 29-bit CAN2.0B identifier is split into 6 fields within the J1939 spec: Priority, Extended Data Page (EDP), Data Page (DP), PDU Format (PF), PDU Specific (PS), Source Address. For the NMEA2000 standard, the EDP is always 0 and the DP is always 1.

These fields make up the Parameter Group Number (PGN), which identifies the specific message being sent. This is generally the function of the CAN identifier, but using the header this way allows for specifying the receiving node while still allowing broadcast packets. The PGN is constructed from the 29-bit CAN ID differently depending on the type of message as shown in Fig. D.2.

The PGNs identify messages that are part of a common message set fully-defined by the NMEA2000 standard. These messages are split into two sets: messages that manage the network and messages that contain data. Only the data messages have been reverse-engineered with details on all messages used in the SeaSlug (detailed in Section E.5).

24-bit Parameter Group Number (PGN)

							E	D	F	F	F	F	F	F	F	F	S	S	S	S	S	S	S	S
PDU1	0	0	0	0	0	0	0	1	PDU Format								PDU Specific							
PDU2	0	0	0	0	0	0	0	1	PDU Format								0							

Figure D.2: How the PGN is defined for the two different types of messages in SAE J1939. PDU1 is a message intended for a specific node. PDU2 is a general broadcast message. The PS field's meaning depends on the value of the PF field. When PF ≥ 240, the PS field represents Group Extension (GE), and is included in the Parameter Group Number. When PF < 240, the PS field contains the target address and is not included in the Parameter Group Number.

Some PGNs contain more than either bytes of data and therefore cannot be sent in a single CAN frame. NMEA2000 defines the Fast Packet message type that can transmit up to 223 bytes. Fig. D.3 shows the format of Fast Packets.

Data Payload (8 bytes)

	Byte[0]								Byte[1]	Bytes[2..7]
Message 1	S	S	S	0					Size	Data[0..5]
Message 2	S	S	S	1					Data[0]	Data[1..6]
⋮										
Message n	S	S	S	F	F	F	F	F	Data[0]	Data[1..6]

Figure D.3: Data packing format for NMEA2000's Fast Packet. All messages use the first payload byte to store the Sequence ID (3 bits) and the Frame Counter (5 bits). The Sequence ID is as described previously, but is only the highest-order bits. The Frame Counter is the frame number, starting at 0 for the first frame and incrementing by 1 for every subsequent frame. The first message also uses a 2nd payload byte to store the total payload size for the message, which can be up to 255 bytes.

D.4 Predefined Messages

The messages defined by NMEA2000 are varied in scope and grouped into two types: those for transmitting data and those for modifying the nodes on the network. Details on the NMEA2000 messages decoded or transmit by custom

CAN nodes, see Section E.5

For data messages, a common reoccurring field is the Sequence Identifier (SID). This is a number representing the timestep that the data was acquired during and counts upwards. Messages received from the same node with the same SID are samples from the same time. There is no fixed rate that the SID should increase at, so this is the only information that the SID provides.

Appendix E

CAN Messages

E.1 Introduction

This section summarizes all of the CAN messages that are used onboard the SeaSlug. Some of these messages are only decoded from proprietary hardware while others are transmit to communicate with other hardware. These messages are all transmit on the only CAN bus onboard, which is connected as described in Chapter 2.

E.2 Custom Messages

All custom messages used on the SeaSlug are standard frame messages, which makes it easier to differentiate them from the NMEA2000 messages and prevents possible collisions. Additionally these messages are little-endian and byte-aligned.

0x080 - Rudder Details

This message is broadcast by the Rudder Node at 4Hz and provides low-level details of the rudder actuator sensors including the raw sensor values, system state, and potentiometer limits of the rudder range.

0x081 - Set Rudder State

This message is received by the Rudder Node and controls its operating mode. The rudder can be enabled, calibrated, or reset through this message.

0x082 - Set Rudder TX Rate

This message is received by the Rudder Node and allows for configuring the transmission rate of the Rudder Details (0x080) and Rudder (PGN127245) message.

0x090 - Node Status

This message is transmit by every CANode on the bus at a frequency of 2Hz. It contains the node's identifier, temperature, processor utilization, status, and error state.

E.3 Tokimec Messages

The VSAS-2GM broadcasts its own set of messages for all of its data. What follows are descriptions for the messages that are used on the SeaSlug, even though more messages are supported and broadcast by this device. For a complete list, reference the VSAS-2GM Operation Manual. All messages described here are standard-frame CAN messages, with big-endian fields and unused bits set to zeros.

0x100 - Angular Velocity

Contains the three-axis filtered angular velocity rates of the sensor as measured by onboard gyroscopes. Output at 25Hz.

0x101 - Acceleration Data

Contains the filtered accelerometer data. Output at 25Hz.

0x102 - Attitude Angle

Contains the filtered attitude data as yaw/pitch/roll. Output at 25Hz.

0x107 - Status

Contains the sensor status and GPS fix. Output at 25Hz.

E.4 ACS300 Messages

The ACS300 motor driver board has its own set of messages. Their base addresses are configurable, with the defaults for received messages being 0x300 and transmitted messages being 0x400. This was left as-is on the SeaSlug as it does not cause any conflicts. All ACS300 messages use a big-endian data format and are standard frame messages.

0x301 - Write Parameter

This message is received by the ACS300 and updates the value for a parameter. See the ACS300 reference manual for parameter details. This is used for setting the motor current command.

0x402 - Heartbeat

When the ACS300 is powered-on and enabled this message is transmit at 100Hz. It contains four parameters that can be configuring by setting the CN.DA, CN.DB, CN.DC, and CN.DD parameters.

E.5 NMEA2000 Messages

These messages follow the conventions described in Appendix D. While the devices in the SeaSlug transmit and receive a vast number of NMEA2000 messages (see their documentation for details), only the messages transmit or processed by CANodes are described below for brevity.

PGN126992 - System Time

Broadcast by Maretron's GPS200 GPS at 1Hz. It contains an absolute time reference as a date & time since the Unix epoch.

PGN127173 - DC Source Status

Broadcast by the Xantrex XW-MPPT60-150 solar charge controller at 2Hz. Contains a configurable source ID, so this message corresponds to either the power input of the attached solar panel or the power output of the battery.

PGN127245 - Rudder

This message can contain either the commanded rudder angle and is transmit by the Primary and RC Node. It is also used to broadcast the actual rudder angle by the Rudder Node at 10Hz.

PGN128259 - Speed

Broadcast by the Airmar DST800 triducer at 1Hz, this message contains the forward water speed

PGN128267 - Depth

Also broadcast by the DST800 at 1Hz, this message contains the water depth.

PGN129025 - Position Rapid Update

The GPS200 broadcasts this message, which contains just the sensed latitude and longitude. It's transmit at 5Hz.

PGN129026 - CoG/SoG Rapid Update

The GPS200 broadcasts this message at 4Hz, which contains the course-over-ground and speed-over-ground.

PGN129029 - GNSS Position Data

Transmit by the GPS200 at 1Hz, this message is a variable length message transmit as a Fast Packet. It is useful for obtaining the number of satellites used in the solution as well as the altitude.

PGN129539 - GNSS DOPS

The dilution of precision data computed by the GPS200. Necessary to verify the status of the GPS fix. It is transmit at 10Hz.

PGN130306 - Wind Data

Transmit by the Maretron WSO100 air & wind sensor at 10Hz, it contains the speed and direction of the wind.

PGN130310 - Environmental Parameters

Transmit by the DST800 at 1Hz, contains the water temperature.

PGN130311 - Environmental Parameters 2

Transmit by the WSO100 at 2Hz, contains the air temperature, humidity, and pressure.

Appendix F

Bill of Materials

The SeaSlug has been designed for a low initial build cost and low maintenance and operational costs. Marine-grade components were used where available as part of reducing the maintenance costs at the expense of the initial build cost. Additionally common off-the-shelf components were used to facilitate servicing through first-party warranties or simply entire component replacement.

The complete estimated cost for producing a new SeaSlug is $28,551 USD. The following list provides a price breakdown of all components of the system, though some have been estimated as the actual component cost is unknown:

Hull

Hull	Custom fiberglass design	$20,000
Misc	Fasteners, wires, cabling, etc.	$1,000

Rudder

Motor	Anaheim Automation 34YSG207S-LW8-R5	$400
Motor driver	Applied Motion 2035	$181
Limit sensors	2x Cherry MP102103	$15
Position sensor	Vishay/Spectrol 157-11502	$25

Propulsion

Propeller	Custom	\$800
Motor	MCG IB46004	\$600
Motor driver	ACS300 w/ CAN interface	\$400

Sensors

GPS	Maretron GPS200	\$300
Water speed	Airmar DST800	\$300
Wind/air	Maretron WSO100	\$700
IMU	Tokimec VSAS-2GM [1]	\$2,000
Power	AttoPilot 50V/90A Voltage and Current Sensor (x2)	\$20

Controls Electronics

Controllers	6x CANodes with interface shields	\$400

Groundstation

Tablet computer	Microsoft Surface Pro 2 32GB	\$420
Primary controller	Logitech F710	\$40
RC transmitter	Spektrum DX5e	\$60
RC receiver	Spektrum AR6100e	\$50
Weatherproof enclosure	Custom enclosure	\$50
Radios	915MHz 3DR radio set	\$100
USB Hub	D-Link DUB-H4	\$20

[1] No longer available, replaceable with \$100 3Space IMU by YEI Technology

Misc

Software	MATLAB, Simulink, and necessary libraries	\$800
Batteries	2x 12V Deka Dominator 98Ah, 4x Centennial AGM CBGC2-AGM	\$600
Shore charger	DualPro PS4	\$500
Solar panel	ICO-SPC-20W	\$50
Solar charger	Xantrex XW-MPPT60-150	\$600
	Total	\$28,551

Appendix G

Electronic Resources

The following list provides URLs for accessing the various online resources used onboard the SeaSlug. This includes code repositories, electronic and mechanical CAD designs, and additional reference documentation.

`https://github.com/Susurrus/Autoboat`

This is the main code repository for the project. It contains all code for building the code for all onboard mircocontrollers. See README.md for more details.

`https://github.com/Susurrus/MicroSimulink-Library`

This is a Simulink library required for some of the Simulink models used for the SeaSlug.

`http://byron.soe.ucsc.edu/wiki/autoboat`

This is the primary webpage for the SeaSlug. This wiki contains much of the high-level reference data for the vessel. Mostly digital documents and details on the subsystems and components with the vessel.

`http://byron.soe.ucsc.edu/projects/SeaSlug`

This is the primary data repository for the SeaSlug. All data collected, along with portions of its analysis, is available through this website.

`git://byron.soe.ucsc.edu/ASL_eCAD.git`

This code repository contains the electronic-CAD commonly shared between projects in the Autonomous System Lab at UCSC. Most importantly it contains the CANode project (shown in Fig. 2.5), which is the base controller used in multiple places on the SeaSlug (see Fig. 2.4).

`https://github.com/mavlink/qgroundcontrol`

This is the central code repository for the QGroundControl project. This is the remote control interface to the vessel that runs aboard the GCS.

`https://github.com/mavlink/mavlink`

This is the central code repository for the MAVLink project. This is the communications protocol used by the SeaSlug for communication with QGroundControl.

`https://github.com/Susurrus/SLUGS-Logger`

The central code repository for the SLogger datalogger project, described in Appendix A. It contains the firmware for the datalogger board, which relies on a CANode with accompanying microSD shield, and supporting scripts for testing the hardware and extracting recorded data.

Bibliography

[1] AIS requirements. http://www.navcen.uscg.gov/?pageName=AISCarriageReqmts, 2003.

[2] Carlos Almeida, Tiago Franco, Hugo Ferreira, Alfredo Martins, Ricardo Santos, José Miguel Almeida, João Carvalho, and Eduardo Silva. Radar based collision detection developments on usv roaz ii. In *OCEANS 2009-EUROPE*, pages 1–6. IEEE, 2009.

[3] J. Alves, P. Oliveira, R. Oliveira, A. Pascoal, M. Rufino, L. Sebastiao, and C. Silvestre. Vehicle and mission control of the delfim autonomous surface craft. In *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pages 1–6. IEEE, 2006.

[4] Omead Amidi and Chuck E Thorpe. Integrated mobile robot control. In *Fibers' 91, Boston, MA*, pages 504–523. International Society for Optics and Photonics, 1991.

[5] H Alemi Ardakani and TJ Bridges. Review of the 3-2-1 euler angles: a yaw–pitch–roll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK*, 2010.

[6] The Arduino homepage. http://arduino.cc.

[7] The ArduPilot project. http://www.diydrones.com/notes/ArduPilot/.

[8] ASV Global civilian and research unmanned marine vehicles. http://www.asvglobal.com/commercial-unmanned-marine-vehicles.

[9] Autonaut USV. http://www.autonautusv.com. [Online; accessed April 6th, 2015].

[10] JG Bellingham, B Hobson, MA Godin, B Kieft, J Erikson, R McEwen, C Kecy, Y Zhang, T Hoover, and E Mellinger. A small, long-range auv with flexible speed and payload. In *Ocean Sciences Meeting, Abstract MT15A*, volume 14, 2010.

[11] Michael R Benjamin and Joseph A Curcio. Colregs-based navigation of autonomous marine vehicles. In *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pages 32–39. IEEE, 2004.

[12] Volker Bertram. Unmanned surface vehicles–a survey. *Skibsteknisk Selskab, Copenhagen, Denmark*, 2008.

[13] Gabriele Bruzzone, M Bibuli, and M Caccia. Autonomous mine hunting mission for the charlie usv. In *OCEANS, 2011 IEEE-Spain*, pages 1–6. IEEE, 2011.

[14] C-Enduro ASV. `http://www.asvglobal.com/science-and-survey/c-enduro`.

[15] M. Caccia, R. Bono, G. Bruzzone, G. Bruzzone, E. Spirandelli, G. Veruggio, and AM Stortini. Design and exploitation of an autonomous surface vessel for the study of sea-air interactions. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3582–3587. IEEE, 2005.

[16] Massimo Caccia. Autonomous surface craft: prototypes and basic research issues. In *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pages 1–6. IEEE, 2006.

[17] Massimo Caccia, Marco Bibuli, and Giorgio Bruzzone. Integration of acoustic devices on small usvs: the charlie experience. In *Control & Automation (MED), 2011 19th Mediterranean Conference on*, pages 424–429. IEEE, 2011.

[18] The CAN Boat project. `https://github.com/canboat/canboat`. [Online; accessed June 25th, 2015].

[19] David J Carlson. Phytoplankton in marine surface microlayers. *Canadian Journal of Microbiology*, 28(11):1226–1234, 1982.

[20] JW Choi, Renwick E Curry, and Gabriel Hugh Elkaim. Continuous curvature path generation based on bézier curves for autonomous vehicles. *IAENG International Journal of Applied Mathematics*, 40(2), 2010.

[21] CiA CiA. 301 v4. 2.0–canopen application layer and communication profile, can in automation e. *V., Feb*, 2011.

[22] Daniel L Codiga. A marine autonomous surface craft for long duration, spatially explicit, multi-disciplinary water column sampling in coastal and estuarine systems. *Journal of Atmospheric and Oceanic Technology*, 32:627–641, 2014.

[23] Navigation rules. `http://www.navcen.uscg.gov/?pageName=navRulesContent`, 2006.

[24] Joseph Curcio, John Leonard, and Andrew Patrikalakis. Scout-a low cost autonomous surface platform for research in cooperative autonomy. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 725–729. IEEE, 2005.

[25] Ren Curry, Mariano Lizarraga, Bryant Mairs, and Gabriel Elkaim. L_2^+, an improved line of sight guidance law for uavs. *American Control Conference*, June 2013.

[26] Tom Daniel, Justin Manley, and Neil Trenaman. The wave glider: enabling a new approach to persistent ocean observation and research. *Ocean Dynamics*, 61(10):1509–1520, 2011.

[27] NIMA USA Department of Defense. World geodetic system 1984–its definition and relationships with local geodetic systems. Technical Report TR8350.2, Department of Defense, NIMA USA, January 2000.

[28] Matthew Dunbabin and Alistair Grinham. Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas emission monitoring. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5268–5274. IEEE, 2010.

[29] G. Elkaim and CO Boyce. An energy scavenging autonomous surface vehicle for littoral surveillance. In *Proceedings of ION Global Navigation Satellite Systems Conference*, 2008.

[30] Gabriel Elkaim. *System Identification for Precision Control of a WingSailed GPS-Guided Catamaran.* PhD thesis, Stanford University, 2002.

[31] Veronika Eyring, Ivar SA Isaksen, Terje Berntsen, William J Collins, James J Corbett, Oyvind Endresen, Roy G Grainger, Jana Moldanova, Hans Schlager, and David S Stevenson. Transport impacts on atmosphere and climate: Shipping. *Atmospheric Environment*, 44(37):4735–4771, 2010.

[32] Richard A Feely, Christopher L Sabine, Taro Takahashi, and Rik Wanninkhof. Uptake and storage of carbon dioxide in the ocean. *Oceanography*, 14(4):18, 2001.

[33] I Fer and D Peddie. Navigation performance of the sailbuoy. *Bergen-Scotland mission*, 2012.

[34] Andrew M Fischer, John P Ryan, Christian Levesque, and Nicholas Welschmeyer. Characterizing estuarine plume discharge into the coastal ocean using fatty acid biomarkers and pigment analysis. *Marine environmental research*, 99:106–116, 2014.

[35] Witold Fraczek. Mean sea level, gps, and the geoid. *ArcUsers Online*, 2003.

[36] Christopher R German, Micheal V Jakuba, James C Kinsey, Jim Partan, Stefano Suman, Abhimanyu Belani, and Dana R Yoerger. A long term vision for long-range ship-free deep ocean operations: Persistent presence through coordination of autonomous surface vehicles and autonomous underwater vehicles. In *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES*, pages 1–7. IEEE, 2012.

[37] Mahmud Hasan Ghani, Lars R Hole, Ilker Fer, Vassiliki H Kourafalou, Nicolas Wienders, HeeSook Kang, Kyla Drushka, and David Peddie. The sailbuoy remotely-controlled unmanned vessel: Measurements of near surface temperature, salinity and oxygen concentration in the northern gulf of mexico. *Methods in Oceanography*, 10:104–121, 2014.

[38] Roberto Grena. An algorithm for the computation of the solar position. *Solar Energy*, 82(5):462–470, 2008.

[39] John M Guinotte and Victoria J Fabry. Ocean acidification and its potential effects on marine ecosystems. *Annals of the New York Academy of Sciences*, 1134(1):320–342, 2008.

[40] Yan Guo, Miguel Romero, Sio-Hoi Ieng, Frederic Plumet, Ryad Benosman, and Bruno Gas. Reactive path planning for autonomous sailboat using an omni-directional camera for obstacle detection. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 445–450. IEEE, 2011.

[41] John T Hardy. The sea surface microlayer: biology, chemistry and anthropogenic enrichment. *Progress in Oceanography*, 11(4):307–328, 1982.

[42] Hordur K Heidarsson and Gaurav S Sukhatme. Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 731–736. IEEE, 2011.

[43] John R Higinbotham, PG Kitchener, and John R Moisan. *Development of a New Long Duration Solar Powered Autonomous Surface Vehicle.* IEEE, 2006.

[44] J.R. Higinbotham, J.R. Moisan, C. Schirtzinger, M. Linkswiler, J. Yungel, and P. Orton. *Update on the development and testing of a new long duration solar powered autonomous surface vehicle.* IEEE, 2008.

[45] Roger Hine, Scott Willcox, Graham Hine, and Tim Richardson. The wave glider: A wave-powered autonomous marine vehicle. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–6. IEEE, 2009.

[46] Gregory Hitz, François Pomerleau, M-E Garneau, Cédric Pradalier, Thomas Posch, Jakob Pernthaler, and Roland Y Siegwart. Autonomous inland water monitoring: Design and application of a surface vessel. *Robotics & Automation Magazine, IEEE*, 19(1):62–72, 2012.

[47] Terry Huntsberger, Hrand Aghazarian, Andrew Howard, and David C Trotz. Stereo vision–based navigation for autonomous surface vessels. *Journal of Field Robotics*, 28(1):3–18, 2011.

[48] ISO 11898-1:2003 - Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling, 2003.

[49] Lubin Kerhuel. dsPIC blockset. `http://www.kerhuel.eu/wiki/Simulink_-_Embedded_Target_for_PIC`.

[50] Peter Kimball, John Bailey, Sarah Das, Rocky Geyer, Trevor Harrison, Clay Kunz, Kevin Manganini, Ken Mankoff, Katie Samuelson, Thomas Sayre-McCord, et al. The WHOI Jetyak: An autonomous surface vehicle for oceanographic research in shallow or dangerous waters. In *Autonomous Underwater Vehicles (AUV), 2014 IEEE/OES*, pages 1–7. IEEE, 2014.

[51] Andrew T Klesh and Pierre T Kabamba. Solar-powered aircraft: Energy-optimal path planning and perpetual endurance. *Journal of guidance, control, and dynamics*, 32(4):1320–1329, 2009.

[52] Yoshiaki Kuwata, Michael T Wolf, Dimitri Zarzhitsky, and Terrance L Huntsberger. Safe maritime navigation with colregs using velocity obstacles. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4728–4734. IEEE, 2011.

[53] JH Lambert. Photometria sive de mensura et gradibus luminis colorum et umbrae (augsburg, 1760). *German translation by E. Anding (Leipzig, Verlag von Wilhelm Engelmann, 1892)*, 1892.

[54] Wolfhard Lawrenz. *CAN System Engineering: From Theory to Practical Applications.* Springer, New York, 1997.

[55] K.E. Laws, C. Bazeghi, S.C. Petersen, and J.F. Vesecky. An autonomous sensor platform vessel for marine protected area monitoring. In *Geoscience and Remote Sensing Symposium,2009 IEEE International,IGARSS 2009*, volume 4, pages IV–991 –IV–994, july 2009.

[56] Alexander Leonessa, Jeremiah Mandello, Yannick Morel, and Miguel Vidal. Design of a small, multi-purpose, autonomous surface vessel. In *OCEANS 2003. Proceedings*, volume 1, pages 544–550. IEEE, 2003.

[57] Peter S Liss and Robert A Duce. *The sea surface and global change*. Cambridge University Press, 2005.

[58] Mariano Lizarraga. *Design, implementation and flight verification of a versatile and rapidly reconfigurable UAV GNC research platform*. PhD thesis, University of California, Santa Cruz, 2009.

[59] Lee A. Luft, Larry Anderson, and Frank Cassidy. Nmea 2000: A digital interface for the 21st century. Technical report, National Marine Electronics Association, Jan 2002.

[60] J. Manley and S. Willcox. The wave glider: A persistent platform for ocean science. In *OCEANS 2010 IEEE-Sydney*, pages 1–5. IEEE, 2010.

[61] MBARI vessel, vehicle, MARS, labor, test tank rates. `http://www.mbari.org/dmo/ship_rates.htm`, 2013.

[62] Lorenz Meier. Mavlink: Micro air vehicle communication protocol. http://qgroundcontrol.org/mavlink/start.

[63] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, pages 1–19, 2012. 10.1007/s10514-012-9281-4.

[64] Hossein Mousazadeh, Alireza Keyhani, Arzhang Javadi, Hossein Mobli, Karen Abrinia, and Ahmad Sharifi. A review of principle and sun-tracking methods for maximizing solar systems output. *Renewable and Sustainable Energy Reviews*, 13(8):1800–1818, 2009.

[65] Karl N Murphy. Analysis of robotic vehicle steering and controller delay. In *Fifth International Symposium on Robotics and Manufacturing (ISRAM)*, pages 631–636. Citeseer, 1994.

[66] W Naeem, T Xu, R Sutton, and A Tiano. The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 222(2):67–79, 2008.

[67] NOAA. Ocean. http://www.noaa.gov/ocean.html.

[68] NSF 2015 budget request to congress - major multi-use research facilities. `https://www.nsf.gov/about/budget/fy2015/pdf/32_fy2015.pdf`. [Online; accessed June 25th, 2015].

[69] IMO/FAO/UNESCO-IOC/WMO/WHO/IAEA/UN/UNEP Joint Group of Experts on the Scientific Aspects of Marine Environmental Protection (GESAMP). *The Sea-surface Microlayer and Its Role in Global Change.* Reports and studies. UN, 1995.

[70] Philip Orton and John Moisan. Coastal ocean air-sea co_2 flux measurements from an autonomous research vessel, 2007.

[71] Jeffrey D Paduan and Leslie K Rosenfeld. Remotely sensed surface currents in monterey bay from shore-based hf radar (coastal ocean dynamics application radar). *Journal of Geophysical Research: Oceans (1978–2012)*, 101(C9):20669–20686, 1996.

[72] S Park, J Deyst, and J How. A new nonlinear guidance logic for trajectory tracking. *AIAA Guidance, Navigation and Control Conference and Exhibit*, Jan 2004.

[73] António Pascoal, Carlos Silvestre, and Paulo Oliveira. Vehicle and mission control of single and multiple autonomous marine robots. In *ADVANCES IN UNMANNED MARINE VEHICLES EDITOR*. Citeseer, 2005.

[74] Thomas Pastore and Vladimir Djapic. Improving autonomy and control of autonomous surface vehicles in port protection and mine countermeasure scenarios. *Journal of Field Robotics*, 27(6):903–914, 2010.

[75] Gregg W Podnar, John M Dolan, Alberto Elfes, Stephen Stancliff, Ellie Lin, JC Hosier, Troy J Ames, John Moisan, Tiffany A Moisan, John Higinbotham, et al. Operation of robotic science boats using the telesupervised adaptive ocean sensor fleet system. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1061–1068. IEEE, 2008.

[76] Robin R. Steeves. Mathematical models for use in the readjustment of the north american geodetic networks. Technical Report 1, Energy, Mines, and Resources Canada, April 1984.

[77] Patrick F Rynne and Karl D von Ellenrieder. Unmanned autonomous sailing: Current status and future role in sustained ocean observations. *Marine Technology Society Journal*, 43(1):21–30, 2009.

[78] PF Rynne and KD von Ellenrieder. A wind and solar-powered autonomous surface vehicle for sea surface measurements. In *OCEANS 2008*, pages 1–6. IEEE, 2008.

[79] Offshore sensing - sailbuoy. `http://http://sailbuoy.no/`. [Online; accessed June 2nd, 2015].

[80] Saildrone - revolutionizing ocean science. http://www.saildrone.com. [Online; accessed April 6th, 2015].

[81] A Savvaris, H Niu H Oh, and A Tsourdos. Development of collision avoidance algorithms for the C-Enduro USV. In *Proceedings of the 19th IFAC World Congress, 2014*, pages 12174–12181. IFAC, 2014.

[82] E.W. Schlieben. SKAMP - an amazing unmanned sailboat! *Ocean Industry*, pages 38–43, 1969.

[83] Kevin G Sellner, Gregory J Doucette, and Gary J Kirkpatrick. Harmful algal blooms: causes, impacts and detection. *Journal of industrial microbiology & biotechnology*, 30(7):383–406, 2003.

[84] Jeff C Sevadjian, Margaret A McManus, J Ryan, Adam T Greer, Robert K Cowen, and Clifton B Woodson. Across-shore variability in plankton layering and abundance associated with physical forcing in monterey bay, california. *Continental Shelf Research*, 72:138–151, 2014.

[85] Amanda HV Timmerman, Margaret A McManus, OM Cheriton, Robert K Cowen, Adam T Greer, Raphael M Kudela, Kathleen Ruttenberg, and Jeff Sevadjian. Hidden thin layers of toxic diatoms in a coastal bay. *Deep Sea Research Part II: Topical Studies in Oceanography*, 101:129–140, 2014.

[86] Thomas W Vaneck, Claudia D Rodriguez-Ortiz, Mads C Schmidt, and Justin E Manley. Automated bathymetry using an autonomous surface craft. *Navigation*, 43(4):407–417, 1996.

[87] Petr Vanicek and E. J. Krakiwsky. *Geodesy, the concepts*. North Holland Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam New York New York, N.Y, second edition, 1986.

[88] Jianhua Wang, Wei Gu, Jianxin Zhu, and Jubiao Zhang. Energy consumption analysis of electric propulsion system used in autonomous surface vehicle. In *Computer and Automation Engineering, 2009. ICCAE'09. International Conference on*, pages 191–195. IEEE, 2009.

[89] Wave Glider SV3. http://liquidr.com/technology/waveglider/sv3.html.

[90] Douglas C Webb, Paul J Simonetti, and Clayton P Jones. Slocum: An underwater glider propelled by environmental energy. *Oceanic Engineering, IEEE Journal of*, 26(4):447–452, 2001.

[91] Sean Wiggins, Justin Manley, Eric Brager, and Brad Woolhiser. Monitoring marine mammal acoustics using wave glider. In *OCEANS 2010*, pages 1–4. IEEE, 2010.

[92] Willow garage. `http://www.willowgarage.com/`.

[93] S. Wood, M. Rees, and Z. Pfeiffer. An autonomous self-mooring vehicle for littoral & coastal observations. In *OCEANS 2007-Europe*, pages 1–6. IEEE, 2007.

[94] Oliver Wurl and Jeffrey Phillip Obbard. A review of pollutants in the sea-surface microlayer (sml): a unique habitat for marine organisms. *Marine Pollution Bulletin*, 48(11):1016–1030, 2004.

[95] ETH Zurich. Qgroundcontrol: Ground control station for small air land water autonomous unmanned systems. http://qgroundcontrol.org/.