Human-Autonomy Teaming for Improved Diver Navigation

by

Lieutenant Jesse Pelletier, USN
B.S., United States Naval Academy (2016)

Submitted to the Department of Mechanical Engineering and the Joint Program in Applied Ocean Science & Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
and the
WOODS HOLE OCEANOGRAPHIC INSTITUTION
February 2022

Author .....
Joint Program in Applied Ocean Science & Engineering
Massachusetts Institute of Technology
& Woods Hole Oceanographic Institution
January 10, 2022

Certified by .....
John Leonard
Samuel C. Collins Professor of Mechanical and Ocean Engineering
Massachusetts Institute of Technology
Thesis Supervisor

Certified by .....
Lee Freitag
Principal Engineer
Woods Hole Oceanographic Institution
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Chairman, Mechanical Engineering Committee for Graduate Students
Massachusetts Institute of Technology

Accepted by .....
David Ralston
Chairman, Joint Committee for Applied Ocean Science & Engineering
Woods Hole Oceanographic Institution

Human-Autonomy Teaming for Improved Diver Navigation

by

Lieutenant Jesse Pelletier, USN

Submitted to the Department of Mechanical Engineering and the Joint Program in
Applied Ocean Science & Engineering
on January 10, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Diving operations are inherently complex due to navigation and communication limitations. Until recently, fixed-beacon acoustic localization techniques have served as the primary means of improving diver navigation. However, modern artificial intelligence and acoustic modem technologies have enabled accurate relative navigation methods between a diver and an autonomous vehicle.

Human-robot collaboration takes advantage of each member's strengths to create the most effective team. This concept proves especially advantageous within the ocean domain, where humans are naturally deficient navigators. Yet humans serve as the team's creative spirit, offering the critical thinking and flexibility needed to succeed in an unpredictable and dynamic environment. Recent underwater human-robot cooperative navigation systems typically rely on autonomous surface vehicles (ASVs), specially designed underwater vehicles, or stereo cameras. This thesis proposes a diver navigation method exhibiting significantly improved accuracy over dead reckoning without relying on a surface presence, cameras, or fixed acoustic beacons. Specifically, we develop and evaluate the communication architecture and autonomous behaviors required to guide a diver to a target location using subsurface human-autonomous underwater vehicle (AUV) teaming with no requirement for ocean current data or exact diver speeds. By depending on acoustic communication and commercial AUV navigation capabilities, our method has increased accessibility, applicability, and robustness over former techniques.

We utilize the Woods Hole Oceanographic Institution (WHOI) Micromodem 2's two-way-travel-time (TWTT) capability to enable range-only single-beacon navigation between two kayaks serving as proxies for the diver and Remote Environmental Monitoring Units (REMUS) 100 AUV. During processing, a nonlinear least-squares (NLS) method, called incremental smoothing and mapping 2 (iSAM2), utilizes odometry and range measurements to provide real-time diver position estimates given unknown ocean currents. Field experiments demonstrate an average online endpoint error of 4.53 meters after transits four hundred meters long. Additionally, simulations test our method's performance in more challenging situations than those experienced in the field. Overall, this research progresses the interoperability of divers and AUVs.

Thesis Supervisor: John Leonard
Title: Samuel C. Collins Professor of Mechanical and Ocean Engineering
Massachusetts Institute of Technology

Thesis Supervisor: Lee Freitag
Title: Principal Engineer
Woods Hole Oceanographic Institution

Acknowledgments

Thank you to the United States Navy for giving me the incredible opportunity to attend the Massachusetts Institute of Technology and Woods Hole Oceanographic Institution Joint Program. Throughout the rest of my career, I hope to prove that their investment in me and my education was worthwhile.

Thank you to Dr. John Leonard, my research and academic advisor, for his constant support and guidance throughout my time in the Joint Program. It has been a privilege to be advised by such a pioneer in autonomous robots. I am eternally grateful for his leadership and belief in me as a student and young engineer. Additionally, I truly appreciate his support for the United States Navy and military students within the Joint Program. Dr. Leonard's humility is inspiring and has reinforced that learning is a lifelong venture.

Thank you to Lee Freitag, my co-advisor, for his generosity in time and teaching. Lee is a leader in acoustic communication, and I am fortunate to have been advised by him. Before starting the Joint Program, numerous people within the United States Navy acquisitions community recommended I work with him. I want to thank him for taking a chance on me as his first graduate student. Learning from him has been an incredible experience, and I hope I have set the stage for future Navy officers within the Joint Program to join his lab and benefit, as I have, from his expertise.

Thank you to Dr. Eric Gallimore for his guidance and instruction in acoustics and programming. I am grateful for the time he has taken out of his busy schedule to teach me `ros_acomms` and `ros_remus`. Dr. Gallimore is conducting cutting-edge research in underwater vehicle autonomy, and it has been a privilege to work with such a gifted engineer during my time in the Joint Program.

Thank you to my fellow members of MIT's Marine Robotics Group. A special thank you to Kevin Doherty, Alan Papalia, and Qiangqiang Huang for helping me understand iSAM2.

Thank you to The Acoustic Communications Group at WHOI for welcoming me into their lab and teaching me about the Micromodem 2. I am especially grateful for the software and engineering expertise of Sandipa Singh, Keenan Ball, and Dennis Giaya. Without them, our field experiments and the subsequent data analysis would not have been possible.

Thank you to the incredible professors and teaching assistants at MIT and WHOI. Their ability to seamlessly transition to online learning while maintaining the highest standard of instruction is a credit to their professionalism and speaks to why MIT and WHOI remain two of the most respected research institutions in the world.

Thank you to Ed O'Brien and the WHOI Diving Program. Ed is one of the most encouraging, generous, and supportive people you will ever meet. He is a huge asset to WHOI, and I feel very fortunate to have had him as my diving instructor. I have truly appreciated his friendship through my time in the Joint Program. He has selflessly dedicated so much of his time teaching me diving history and helping me with my research by providing valuable insight and feedback. His passion for diving is infectious and is something I seek to mirror. Ed is truly an expert in his field, and it was an honor to learn from him and progress my diving skills under his instruction.

Thank you to Sean Whalen for teaching me about the REMUS vehicles and the VIP software. Without his instruction, guidance, and support, I could not have conducted the simulations in my thesis research.

Thank you to Ensign Peter Ventola for being a great lab partner in class and for helping with my field experiments. Anyone that spends time around Peter will be humbled by his

intelligence, leadership, and generosity. I am excited to see where his Navy career takes him and hope to work together again in the future.

Thank you to Dr. Carl Kaiser for guiding me through my first year in the Joint Program and giving me a solid foundation of engineering skills that have benefited me throughout my thesis research. Early in my graduate studies, he encouraged me to pick a thesis topic that would allow me to conduct field experiments. I would echo that advice to any incoming Joint Program student since planning and executing field experiments has been one of the most fulfilling parts of my time in graduate school. I credit my motivation to perform these tests to Dr. Kaiser's guidance.

Thank you to Commander Brendan O'Neill, who has been my mentor and research partner throughout my time in the Joint Program. I can easily say that working alongside him to progress the diver-AUV team has been an absolute privilege and the highlight of my time in graduate school. We started with the common interest of researching something with the potential to give back to our Navy community. That interest has evolved over the past year into a desire to progress the diver-AUV team, and I believe this research took the first step in that direction. I am grateful for his patience as I struggled to succinctly explain how my software worked or what I planned to do for our field experiments. I am also eternally thankful for his mentorship and have appreciated his course corrections in leadership, planning, and writing when I got off track. His drive to improve himself and give back to the community is inspirational and something I hope to emulate throughout my career. I am excited for the future of the diver-AUV team as Commander O'Neill focuses his Ph.D. research on developing key capabilities that will contribute to its professionalism.

Most of all, thank you to my family. The next couple of sentences are insufficient to thank them all, but I will keep this as short as possible to save the reader. Thank you to my mother-in-law, father-in-law, and sister-in-law for welcoming me into their family as one of their own. Thank you for your unparalleled generosity, for the countless meals, for teaching me it is OK to go to a restaurant more than 15 minutes away from your house, and for bringing me into all your amazing family traditions. Thank you to my mom, dad, and brother for being the best family anyone could have. Thank you for teaching me the value of hard work. Thank you for instilling in me the drive to learn and improve myself constantly. Thank you for endlessly supporting me through all my adventures and endeavors. Thank you for setting an incredible example for me in every facet of life. Finally, thank you to my best friend and wife, Peri, who inspires me daily with her work ethic and dedication as a veterinary student at Tufts University. She is the best teammate anyone could have, and no words will adequately capture how much her love and support have impacted me. Any success I find in life would not be possible without my family, and I hope to live my life in a way that will make them proud.

Funding

The United States Navy funded my graduate education. The Office of Naval Research also partially supported this work under grant N00014-18-1-2832.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the motivation for this thesis and a brief overview of each chapter.

1.1 Motivation

In 1997, an IBM computer called Deep Blue beat Garry Kasparov in chess, marking the first time a computer won a chess match against a reigning world champion. In response, Kasparov invented Advanced Chess, where human-computer teams face off against each other. The interesting discovery from this experiment was that, although computers commonly beat humans, human-computer teams always defeated a computer playing alone [1]. The human-machine team was superior.

Human-robot collaboration relies on effective communication to reduce the human's cognitive load and physical demand, thereby improving the team's efficacy. Recent advances in this field have allowed robots to serve increased roles within care facilities, schools, hospitals, entertainment, industry, space exploration, and military operations [2]. Among other places, the confidence in human-robot teaming is visible in recent military research and documentation. By 2010, the United States Army Telemedicine and Advanced Technology Research Center had already invested in several projects to use robots to assist with casualty evacuation [3]. Since then, the Department of Defense specified its investment in the human-robot combat team with the Third Offset Strategy introduction in 2014 [4]. The 2018 National Defense Strategy continued this trend by acknowledging the role that autonomous systems and emerging technologies will have on the battlefield while reaffirming that the United States military's strength is its people [5].

A group where humans collaborate with autonomous robots is called a human-autonomy team (HAT) [6]. Moravec's Paradox describes one reason for the interest in this emerging relationship: "it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility" [7]. In other words, humans and robots have complementary skill sets. Another critical benefit of retaining the human in the loop is to ensure the ethical and legal use of force. The United States military is at the forefront of developing ethical principles for artificial intelligence and autonomous systems [8] [9]. Therefore, HATs have the potential to result in the most effective and ethical combination.

The navigation and communication challenges inherent to divers provide a unique opportunity for HATs to excel in the ocean domain. Humans are deficient underwater navigators without the help of a compass, especially in dark or murky waters. Divers commonly utilize magnetic compasses and kick counts to navigate from point to point. Many magnetic diving compasses are marked in five- to ten-degree intervals, making it difficult to hold an accurate bearing and decreasing the diver's navigation ability. For example, a consistent error of five degrees for one kilometer results in an endpoint error of nearly 90 meters.

Uncertain ocean current predictions increase inaccuracy. Assuming a constant ocean current of 0.1 knots throughout a one-kilometer dive can leave the diver over 140 meters from their intended target. The U.S. Navy Dive Manual defines the work limits of SCUBA divers as below a one-knot ocean current [10], so experiencing ocean currents above 0.1 knots is common. Often, divers find the ocean current value for their dive plan by utilizing ocean current readings from distant buoys. Thorough dive planning can mitigate the risk of failure by acknowledging the inherent error in this measurement and building in appropriate contingencies. Still, this inaccuracy significantly contributes to the uncertainty of success and extended time allotted for even simple underwater navigation operations.

The difficulty of underwater tasks in conjunction with diving's physiological dangers shows why diving in pairs is necessary. However, dive pairs are limited in their ability to communicate underwater, regularly relying on a short list of hand gestures to send simple messages to each other. Low visibility conditions inhibit this type of communication because it requires both divers to see each other. Additionally, the signals must be agreed upon before the dive, making it challenging to accurately convey complicated messages or dive

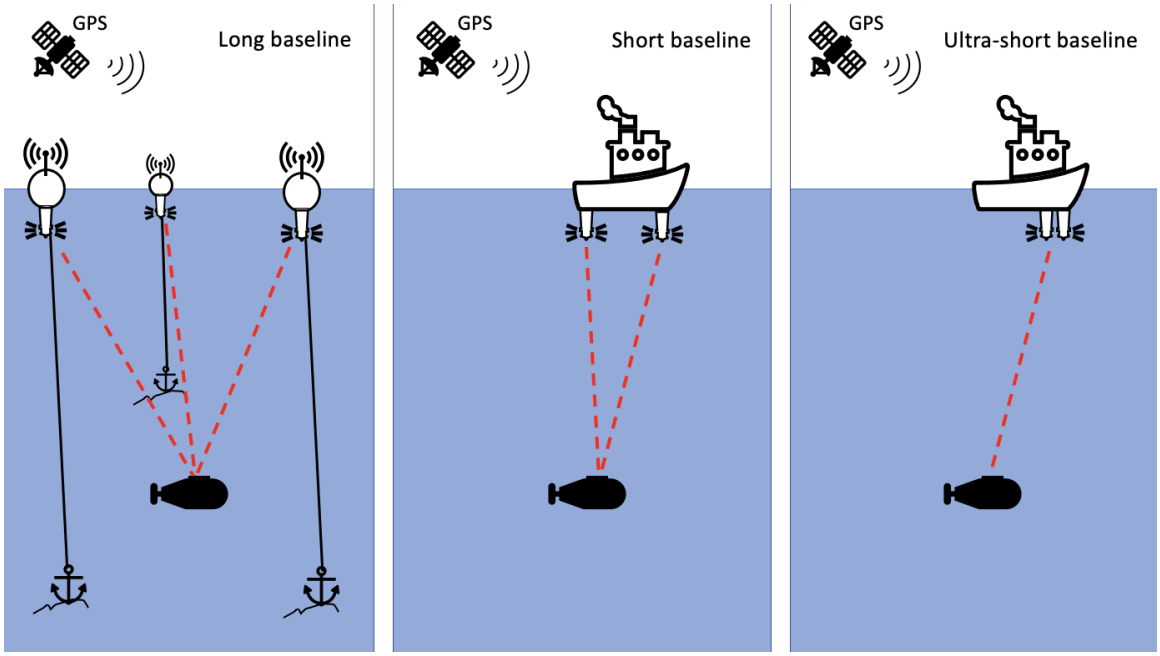


Figure 1-1: AUV acoustic localization methods [11]

plan adjustments. Conversely, terrestrial teams take advantage of accurate localization and low latency, high data-rate communications using the electromagnetic (EM) spectrum.

The infrastructure broadly available on land is less accessible to divers and autonomous underwater vehicles (AUVs) because EM signals attenuate rapidly in water. Standard underwater acoustic localization techniques involve pre-positioned long baseline (LBL), short baseline (SBL), or ultra-short baseline (USBL) beacons. Using acoustic transducers, divers and AUVs can determine their position using the range or bearing from these known beacon locations. However, in addition to requiring a significant time investment to install, these methods limit the survey site to the area within the beacons' acoustic range. Another option involves SBL or USBL systems on ships. Figure 1-1 illustrates these methods.

Autonomous surface vehicles (ASVs) can also employ USBL arrays. However, a significant issue with putting a USBL array on a surface vessel, particularly a small one, is that it requires a highly accurate inertial measurement unit (IMU) to compensate for the surface motion. Additionally, regardless of whether the vehicle is human-crewed or autonomous, the reliance on these surface platforms limits the system's scalability and applicability due to the communication architecture.

Fortunately, AUV navigation has improved with advances in inertial navigation systems (INSs) and simultaneous localization and mapping (SLAM) [11]. Provided the AUV can de-

termine its location with sufficient accuracy, a dive team can localize its position relative to the AUV. Therefore, it is possible to develop an accurate navigation system for divers without relying on pre-positioned beacons, ASVs, or surface ships, thus maintaining reliability while increasing versatility.

This thesis focuses on improving the diver's ability to navigate to a target. Đ. Nađ et al. [12] created an AUV-guided navigation system for divers that relied on ASVs and visual cues from the AUV. By eliminating these constraints, the HAT can be more robust and serve in more diverse environments, such as turbid or dark waters. As previously mentioned, removing the ASV would simplify the communication architecture and increase applicability. Suppose a search and rescue operation occurred in a busy shipping lane. If the system required an ASV, it would be necessary to cordon off the search area to ensure there are no collisions. However, without the ASV, all elements are subsurface, so the operation could be conducted without disturbing ship traffic.

Lastly, this thesis seeks to progress the underwater HAT, where heterogeneous members with dynamically changing roles and responsibilities cooperate to achieve the objective. Recent research in this area has utilized specially designed AUVs [13] [14]. However, these AUVs are not widely available, and their designs limit their capabilities outside the human-robot relationship. Leading commercial AUVs have the means to enable human-autonomy teaming while also providing a suite of advanced sensors for environmental mapping and other tasks. More capable AUVs expand the potential functions they can serve within the team. Therefore, while this thesis focuses on navigating to a target location, it serves as an early step in the broader underwater HAT's objective to understand their operational domain and perform a set of diverse roles to accomplish the mission.

1.2 Thesis Overview

This thesis evaluates the feasibility and effectiveness of a dive navigation method that provides significant endpoint accuracy improvement over dead reckoning (DR) using subsurface human-autonomy teaming. This approach relies on the Woods Hole Oceanographic Institution (WHOI) Micromodem 2 [15], the Georgia Tech Smoothing and Mapping (GTSAM) factor graph-based state estimation library [16], and the Robot Operating System (ROS™) middleware system for robotics [17]. A custom ROS package, called `ros_hat` [18], was de-

signed to integrate all required sensors and software to allow teaming between a Remote Environmental Monitoring Units (REMUS) AUV [19] and a diver. Although the eventual goal is to have this software run on a submersible dive tablet and REMUS 100 [20], this thesis sought to prove and evaluate this concept using kayaks as proxies for the diver and AUV. Consequently, this novel communication architecture opens many new opportunities to progress the undersea HAT, thereby increasing the diver's capabilities and safety.

Chapter 2 provides the background for this thesis by reviewing underwater human-robot teaming, range-only single-beacon (ROSB) navigation, acoustic telemetry, and nonlinear least-squares (NLS) state estimation using factor graphs. In addition to introducing two critical components to our HAT navigation architecture, namely the Micromodem 2 and incremental smoothing and mapping 2 (iSAM2), this chapter includes a brief history of how divers have utilized technology to increase their localization, navigation, and communication capabilities. It is important to note that the most recent HAT research efforts use custom-built AUVs focused on providing safety monitoring and task assistance to divers at very close distances. Instead, we view the AUV as a versatile element, having the ability to perform both cooperative and individual tasks that contribute to completing the overall mission.

Chapter 3 outlines the software, hardware, and equations that enable our human-autonomy teaming technique. This chapter introduces our software architecture, messaging sequence, and messaging structure. Within the call-and-response communication between the AUV and diver, the diver's state estimate is updated using odometry and range measurements, thereby providing real-time location updates for the diver to navigate to a geographic coordinate of interest.

Chapter 4 discusses the full-scale kayak-to-kayak experiments used to validate the functionality and accuracy of this diver navigation method. Along with describing the final experiment's protocols, equipment, and setup, this chapter describes the preliminary field tests utilized to find the Micromodem 2's two-way-travel-time (TWTT) ranging accuracy. The range data collected in both experiments ensured accurate noise modeling for simulations, which sought to test the performance of the cooperative navigation method in more challenging situations than those experienced in the field. This chapter concludes by analyzing the results from the simulations and field experiments. To the best of our knowledge, there are no similar diver-AUV cooperative navigation results to compare with our method. Therefore, we compare our technique to DR methods available on state-of-the-art

submersible dive tablets. Although this is not a perfect comparison, it shows the endpoint accuracy improvement through human-autonomy teaming.

Lastly, Chapter 5 presents conclusions from the simulations and field experiments. However, adding an AUV to the dive team is not only intended to improve the dive team's navigation ability; the dive team adds the entire library of AUV capabilities, from automatic target recognition to bathymetric surveying to environmental monitoring. Along with discussing the profusion of opportunities unlocked through underwater HATs, this chapter closes by recommending future research to improve our system's cooperative navigation function through environmentally adaptable autonomous behaviors.

Chapter 2

Background

This chapter reviews diver navigation methods and current underwater human-autonomy teaming research. It also introduces acoustic communication and state estimation techniques foundational to this thesis.

2.1 Diver Navigation and Underwater Human-Robot Teaming

This section provides a brief history of utilizing technology to improve a diver's capabilities. It is critical to understand this progression to appreciate the difficulties inherent to the ocean domain. First, we review the development of acoustic devices for improving a diver's localization and communication. Then, we discuss recent underwater human-robot teaming efforts and examine state-of-the-art submersible tablets.

The United States Navy pioneered research into acoustic-based diver navigation and communication. In 1970, the Department of the Navy filed for a patent of a novel diver navigation interface that relied on vibrating sensors [21]. This work built on a previous patent for converting a message into vibratory stimuli for long-range communication that required a quick response [22]. The inventor, Joseph Hirsch, claimed humans respond to tactile information faster than auditory or visual information. Hirsch [23] implemented this in a system where vibrations notify pilots of potential collisions. A back pad placed beneath the diver's wetsuit brought this communication method into the ocean domain. Acoustic messages sent by a remote station were detected by the receivers on the diver's belt and translated into vibrations. The vibration location on the diver's back corresponded to navigation corrections - top meaning swim forward, right side meaning turn right, etc.

However, the remote command station could only send these navigation corrections if it knew the diver's position and heading. Sonar determined the diver's location, and the diver sent their heading using four pushbuttons, with each one corresponding to a unique acoustic signal representing one of the cardinal directions. The acoustic transmitter on the diver's belt sent this message to the command station, which could then acoustically respond with navigation corrections provided it had an accurate diver location from sonar [21].

The Navy continued to invest in diver navigation in the late 1970s when it requested the Applied Physics Laboratory at the University of Washington develop a system capable of monitoring divers conducting underwater construction tasks. Four requirements for the project were that it tracked up to eight targets, had a range of one mile, had a circular error probable (CEP) of five feet, and included a radio frequency link that allowed the command station to see the diver's path from up to five miles away. The resulting system, called the portable acoustic tracking system (PATS), relied on an array of six fixed transducers at known locations. The inventors estimated it would take about an hour to deploy and retrieve the array. The deployment team used multiple small boats to place four buoyant survey transducers in a square with 250-foot sides. Of those four, the two on opposite corners were about 40 feet higher than the others to provide depth dispersion to resolve the diver's depth. They then set the two reference transducers approximately 750 feet apart on opposite sides of the square. Each diver wore an acoustic pinger on their back, which was about the size of a SCUBA tank. The system relied on time-division multiple access (TDMA) to allow eight divers to use the fixed transducers synchronously. At their assigned time, the diver's pinger transmitted two pulses. Each survey transducer utilized the first pulse to determine the signal's travel time and the delay between the first and second to resolve the diver's depth. The survey transducers then sent their timing information to a surface buoy, which transmitted 180-byte packets via radio link to a shoreside or seaside receiving unit. The receiving unit transformed the timing information into the diver's local three-dimensional coordinates [24]. The 1970s also saw the Navy invest in handheld diver navigation hardware and software.

A patent granted to the United States Navy in 1978 improved the diver's ability to swim along their desired course. The handheld system combined the outputs from a magnetic compass and Doppler sonar to display the heading the diver should swim to get from one point to another while compensating for ocean current. It also provided course corrections

to the diver. Before this, divers performed mental math to calculate their adjusted course based on readings from a drift meter [25]. However, the Navy was not the only group in the late 20^{th} century seeking to file patents related to improving diver navigation capabilities.

In 1993, the Hughes Aircraft Company, owned by Howard Hughes, filed a patent for an improved acoustic navigation system for divers. Increased Global Positioning System (GPS) capabilities allowed this system to find the diver's global coordinates instead of their local coordinates like previous methods. The system relied on two buoys to locate the divers and provide two-way communication between the command station and the diver. The command station was located at the bottom of the base buoy and monitored by the divemaster. At their assigned time slot, each diver's acoustic transducer communicated a unique identification signal along with its last known location. The buoys received this message and responded with their GPS coordinates. A processor attached to the diver calculated the range from the two buoys using the water's sound speed and the signal's TWTT. Then, the processor combined the range from the two buoys with the diver's depth sensor reading to determine the diver's position in the global coordinate system. The diver's location information could then be acoustically transmitted to the base buoy and passed through a cable to the command station to populate the computer's screen. This process allowed the divemaster to track the divers. Additionally, the divemaster at the command station could use a keyboard to send messages to the base buoy, which then relayed those messages to the divers. Each diver had a wrist-worn interface that made this two-way communication possible; it processed incoming messages, displayed messages received, and provided a keyboard for the diver to send simple messages. For example, the watch showed navigation corrections sent by the divemaster via the base buoy [26]. Figure 2-1 compares this method with PATS. Overall, this technology advanced diver navigation by providing global positioning and improving two-way communication.

Early efforts by the United States Navy and defense contractors decidedly improved a diver's ability to navigate underwater accurately. From the early 1970s to the late 1990s, acoustic ranging methods and GPS allowed divers to navigate and localize themselves within a global coordinate system instead of a local coordinate system. Command stations also gained the ability to monitor the diver's location, log the diver's path, and communicate with the diver. However, all these early systems relied on pre-staged surface beacons. As previously mentioned, reliance on those types of beacons comes with significant constraints.

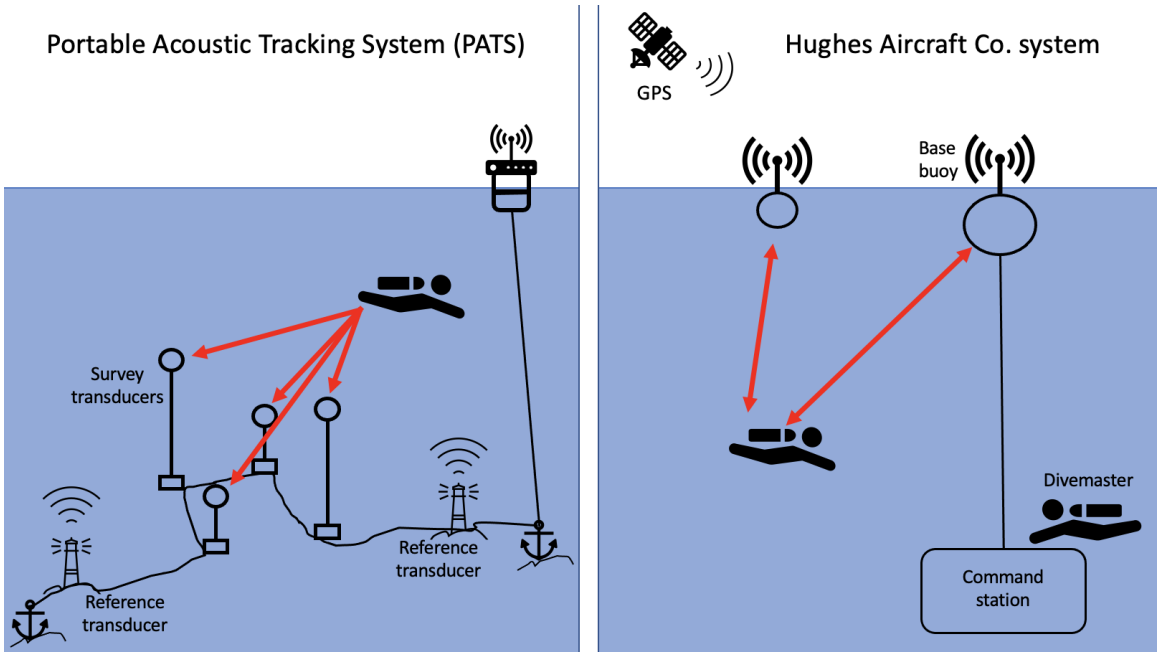


Figure 2-1: Early diver navigation systems [24] [26]

For example, it took about an hour for a series of small boats to set up or break down PATS [24]. Additionally, the fixed transducers' range and location determined the diver's survey area, requiring a costly time expense to move the dive spot. This scenario is unlikely in underwater construction tasks where the location of the pipe or cable is known. However, in complex search and rescue operations, a small, portable system capable of rapid implementation with negligible surface disruption is desirable. Fortunately, 21st-century advances in acoustic communication and autonomous vehicles enable mobile navigation systems and human-robot teaming.

Two significant research efforts into robotic dive buddies over the last decade are Cognitive Autonomous Diving Buddy (CADDY) [13] and Advancing Diver-Robot Interaction Capabilities (ADRIATIC) [14]. Since 2014, the CADDY project has improved an AUV's ability to track a diver. Early research into camera-based diver tracking relied on detecting the periodic kicking motion of the diver in video streams [27]. More recently, CADDY researchers relied on a combination of pressure sensors, stereo cameras, forward-looking multibeam sonar, and USBL systems to track the diver [28] [29]. They also invented DiverNet, consisting of 17 inertial sensors placed on the diver, to give the most reliable relative location and pose data. The sensors provide the body parts' absolute orientations, which are then translated into the relative orientation and sent via an acoustic modem to the

AUV [30]. However, the update rate is too low for the AUV to maintain the correct orientation and distance from the diver at all times. Between DiverNet updates, the sonar and stereo cameras on the AUV work together to estimate the diver's pose. The eigenvectors from the stereo camera's point cloud serve as inputs to a long short-term memory (LSTM) recurrent neural network (RNN), which outputs the diver's heading. Although this system affords a high update rate, the stereo cameras only give reliable estimations when the AUV is within three meters of the diver [29]. In low visibility conditions, one could expect that stereo cameras would have even less range. Therefore, cameras do not provide a dependable solution for a system intended for use in all visibility conditions.

Active and passive sonar for use in harbor surveillance systems is a rich field [31]. However, using an AUV's sonar for diver detection and tracking is less studied. The benefit of using sonar over camera images is that sonar provides greater ranges and is not dependent on ocean visibility. Most research has used forward-looking multibeam sonar for this application. DeMarco et al. [32] trained Hidden Markov Models (HMMs) to perform diver detection and tracking by classifying blob clusters extracted from a sonar image. Neural network approaches are also common. Kvasić et al. [33] trained five high-performing object detection and localization networks on sonar images instead of camera images. These convolutional neural network (CNN) algorithms were Tiny YOLO v3, YOLO v2, YOLO v3-SPP, YOLO v3, and VGG-16. However, research associated with the CADDY project found that neural networks are not reliable detection algorithms for sonar because the images are low quality, and noise affects the neural network efficacy [28]. Additionally, a primary issue with using multibeam sonar for this application is that if the diver leaves the AUV's field of view, it is difficult for the AUV to reacquire the diver, especially if there are other moving objects in the area.

CADDY researchers created an algorithm that fused USBL and sonar sensor outputs to compensate for these weaknesses. Instead of relying on a neural network approach, they relied on clustering contours and finding the cluster that matched the expected target description. The first step in their process is to convolve the sonar image with a two-dimensional Gaussian function. After this blurring step, binarization is performed, and contours are clustered together in the resulting image. Each cluster is ranked based on how closely it matches the expected target and its distance from where the USBL measurement places the diver. After successfully detecting the diver, an extended Kalman filter (EKF)

tracks the diver [28].

USBL systems are accurate and precise but prone to outlier measurements [29]. The CADDY AUV can augment USBL measurements with cameras and sonar because it is near the diver. Depending on the frequency selected, their forward-looking sonar (FLS) has a range between five and 15 meters [28]. However, when more considerable distances separate the diver and AUV, acoustic methods similar to USBL must be more heavily relied upon even though they have higher latency and lower update rates.

Aside from detecting and tracking divers, the CADDY and ADRIATIC projects have improved divers' safety and advanced communication between divers and AUVs. One explored method of communication is through wearable sensors on the diver's body. In addition to DiverNet's ability to send pose information to the AUV, adding a heart rate monitor or a breathing monitor to the system allows it to communicate the diver's physiological status [30]. ADRIATIC applied the concept of wearable sensors to gloves, which can acoustically send hand signals to the AUV using strain sensors and an IMU. While water visibility and the necessity for both divers to be looking decreases hand signals' effectiveness, this sensing glove recognizes the hand gesture and acoustically sends it to the AUV [14]. Therefore, they ensure that the diver can communicate with the AUV even when visibility is low and diver-AUV orientation is not ideal. Although the ADRIATIC research relied upon standard PADI-approved diver signals, CADDY researchers have established a gesture-based human-robot language called CADDIAN.

CADDIAN increases the depth of communication possible between the diver and AUV. As of publication, the language contained 51 commands throughout five core categories: problems, movement, setting variables, feedback, and tasks. More complicated messages, such as retasking the AUV to photograph a point of interest, are accomplished via a series of hand gestures. The operator must meticulously execute these hand gestures when communicating via CADDIAN to ensure the AUV has the best chance at understanding the message [34]. Aside from using kinesics, dynamic retasking of underwater robots has been accomplished by divers using fiducial markers [35] and dive tablets [36]. Dive tablets provide many useful tools to increase a diver's communication, navigation, and survey capabilities.

Although many dive computers are available to assist with tracking decompression stops, surface intervals, temperature, bottom time, and depth, very few handheld tools are available that help divers navigate or communicate with a robot. The AQUATablet, developed over

a decade ago, allows a diver to interact with an underwater vehicle via a tether or visually. When tethered to the robot, the tablet serves as a joystick, allowing the diver to drive the robot to deeper depths or more dangerous areas. For tether-less mode, the dive tablet displays fiducial markers that the diver shows the AUV. These markers can be pre-made or dynamically generated, allowing the diver to create a custom fiducial marker representing their desired instruction for the AUV. Additionally, the number of bits of information in the fiducial tag is dependent on the water visibility; the marker can store more data if the water clarity is high and vice versa [36]. However, the AUV's need to be within visual range or tethered to the diver restricts the team's capabilities.

The CADDY project utilized a dive tablet to enable acoustic communication between the diver, AUV, and ASV. In addition to allowing divers to send commands to the AUV, the support team on the surface could send messages to the diver via the acoustic modems on the ASV and the dive tablet. The tablet could also gather the diver's physiological status from the DiverNet sensors and notify the surface team if the diver's conditions deteriorated [13]. Stilinović et al. [37] discusses how the CADDY ASV remains above the diver's position, thereby allowing the diver's tablet to determine its relative position from the ASV without the risk of multipath propagation. Using this relative position, the tablet can then display the diver's global location on its map. Many research efforts rely on custom dive tablets for their experiments. However, Shark Marine Technologies [38] provides commercially available diver-held systems.

The three dive tablets available through Shark Marine are the E-TAC, the Dive Tablet 2, and the Navigator [38]. The E-TAC is the smallest and simplest of the three. It offers multi-domain navigation with a depth rating of 50 meters and an altitude rating of 15,000 meters. A magnetic compass, gyroscope, and accelerometer provide DR capabilities. Although DR navigation provides inaccurate location estimates, the diver can remove the accrued error by getting a fix with the tablet's GPS sensor. Additionally, the depth sensor and camera make it a useful tool for beach profiling operations [39]. The Dive Tablet 2 offers more capabilities than the E-TAC. Although larger and heavier, it provides more attachments, such as a Doppler velocity log (DVL) and an FLS. Along with this system's improved compass and orientation capabilities, these two tools make the tablet more suitable for diver DR navigation than the E-TAC [40]. However, neither offer as much as the Navigator, which is the company's most modular tablet. The Navigator was explicitly designed for

mine countermeasure (MCM) and search and rescue (SAR) operations. In addition to the advanced DR and FLS capabilities available on the Dive Tablet 2, the Navigator's attachments include multiple types of bathymetric sonars, a side-scan sonar, and an acoustic positioning system compatible with LBL beacons. It also has a head-mounted display to allow the diver to see the screen in zero visibility conditions [41]. Even though the three tablets offer different capabilities, they all run the same software.

DiveLog provides a standard interface between all three Shark Marine tablets. It allows the user to create waypoint navigation paths within standard map formats or custom map files. If a sonar sensor is attached, DiveLog will calculate the declared targets' location and save that information with a picture of the sonar image. Additionally, DiveLog includes a Tactical Navigation Screen, which displays the diver's heading, depth, and estimated position on the map to assist with DR navigation. DiveLog can also calculate the diver's course over ground (COG) if a DVL is attached to the tablet [42]. However, although the Shark Marine products' capabilities are impressive, the diver's positional error still increases without bound over time without GPS fixes or external information from acoustic transducers.

ASVs and GPS intelligent buoys (GIBs) can serve as navigation aids for divers. One benefit of using an ASV over a buoy is that if multiple divers are in the water, it can change its position to optimize the diver's acoustic communication capability. A patent published in 2020 and assigned to the United States Navy describes an improved diver navigation method enabled by a single GIB or ASV. This patent's goals are similar to the other systems - improve the diver's safety, provide a communication link between a command station and the diver, globally localize the diver, and provide navigation guidance to the diver [43]. However, this approach is unique compared to other recent underwater HAT navigation techniques because the mobile robot does not need to be in proximity to the diver.

Figure 2-2 shows the communication architecture of the patent described in the previous paragraph. The buoy or ASV gets its location from GPS and interacts with the diver through an acoustic modem and USBL array consisting of four hydrophones. The acoustic modem starts the localization process by sending an acoustic pulse to the diver. The transducer on the diver's tablet detects this pulse and acoustically responds with their depth and unique identification message. The USBL array uses the ocean's sound speed, round-trip travel time, and phase difference at each hydrophone to determine the diver's distance and bearing. The diver's location information is then sent via radio frequency link to a distant command

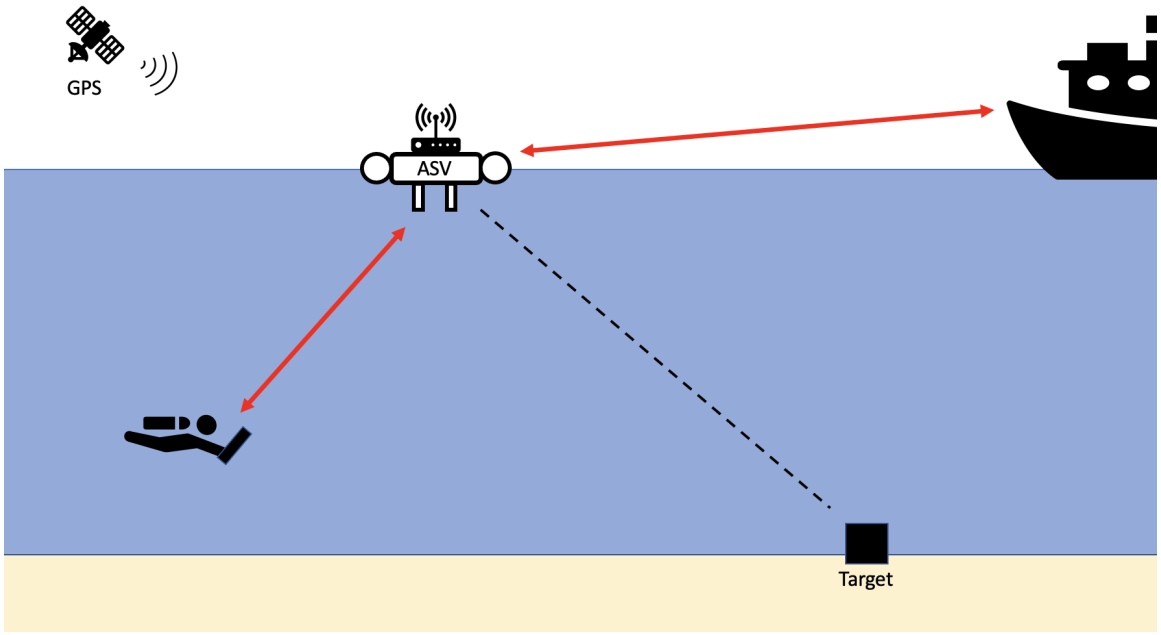


Figure 2-2: Communication architecture of the system described in the "Diver navigation, information, and safety buoy" patent [43]. The red arrows denote communication between those elements.

station at sea or on land, allowing them to track the diver in the water. The acoustic modem also sends the diver location information back to the diver, whose tablet has a processor capable of calculating its estimated global latitude and longitude. These coordinates serve as an input to a Kalman filter, which outputs the diver's predicted position. Additionally, the command station can send navigation directions through the ASV or GIB to the diver using simple 40-bit messages. These messages are then displayed on the diver's tablet [43].

The combination of an AUV and ASV also provides an accurate navigation aid for divers. In 2020, CADDY researchers published a paper detailing their human-robot navigation technique called the pointer strategy. Although CADDY research frequently utilized dive tablets, this approach does not require one. Instead, this method relies on a complex control system that allows their specially designed over-actuated AUV to maintain a constant range from the diver. Calculating the diver's relative position from the AUV is crucial to this ability and is done using a combination of depth sensors, stereo cameras, sonar, and USBL measurements. Proper processing of the stereo camera's point cloud results in the diver's distance, azimuth, elevation, and orientation. The sonar and USBL systems serve as alternate diver position detectors if they are outside the camera's range. Additionally,

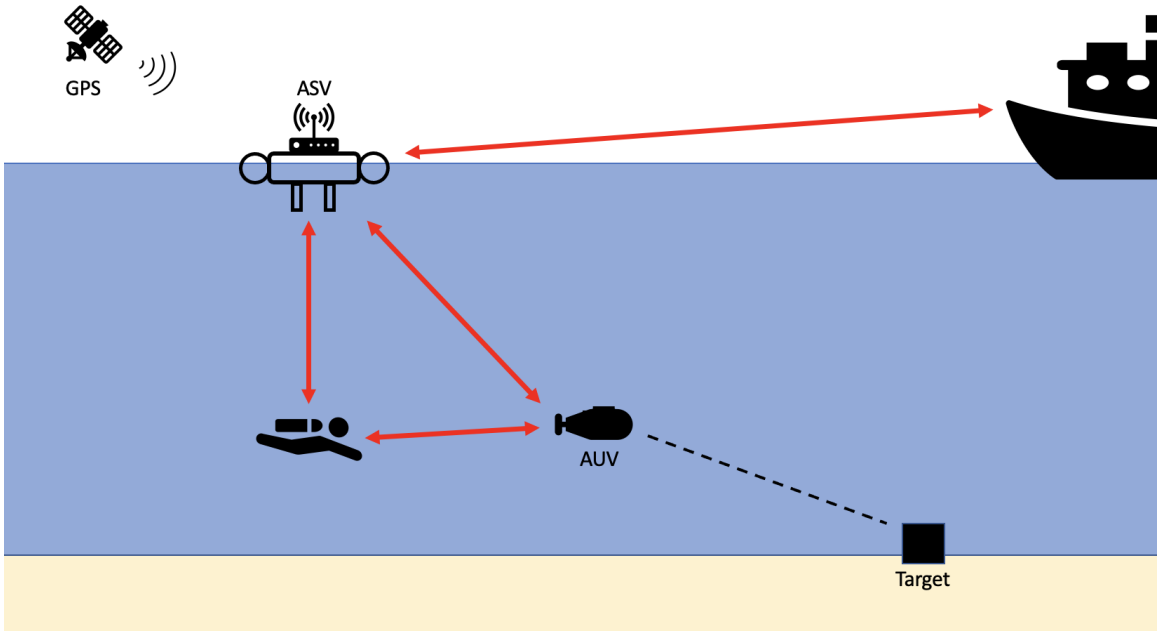


Figure 2-3: CADDY pointer strategy [12]. The red arrows denote communication between those elements.

the diver's depth is acoustically transmitted to the AUV to ensure they are both at the same depth. An ASV positioned above the human sends its GPS location to the AUV. After calculating its position, the AUV can determine the diver's global coordinates using the relative position information retrieved from its sensor suite. Provided it also knows the target location, the AUV can place itself between the diver and target so that the diver can swim in the AUV's direction to travel towards the target. In this way, the AUV serves as a pointer to the target. Figure 2-3 illustrates this orientation. As the diver swims towards the AUV, it adjusts its position to maintain a constant standoff distance from the diver until it is within a pre-determined range of the target. At that point, the guidance mission concludes [12]. Due to the diver's visual requirement, this system's effectiveness is dependent on water visibility. However, another recent HAT navigation system does not require visual communication between the diver and robot.

Like CADDY, the ADRIATIC project uses a novel over-actuated AUV explicitly designed to assist the diver with tasks and safety monitoring. Their dive buddy AUV concept is a handheld dive tablet attached to propulsion skids [14]. The tablet is a Kenautics D2 Diving Navigation System that includes a DVL, microelectromechanical systems (MEMS) attitude and heading reference system (AHRS), pressure sensor, FLS system, low-light video camera,

and GPS [44]. The diver can use the tablet on the vehicle to perform dynamic mission programming and setting reconfiguration. Additionally, the D2 AUV has an acoustic modem to communicate with the diver's acoustic modem. The diver's modem transmits messages formed via the previously mentioned sensing glove. These modems are also used for range-only localization to augment the camera and sonar measurements used to track the diver. Although this AUV concept is in its infancy and just recently built for the first time, the eventual goal is for it to have the ability to lead a diver to a target location [14]. Overall, this system's communication architecture and capabilities are the most similar found during the literature review to this thesis. However, the D2 AUV is not widely available, and the sensors onboard are less capable than those on leading commercial AUVs. Developing a range-only diver navigation method using established high-performing AUVs will result in a more skillful and available HAT.

Since the 1970s, diver navigation systems have improved in similar ways to many other technologies - the size has decreased, and the autonomy has increased. Early examples, such as PATS and the Hughes Aircraft Company system, employed pre-staged surface buoys. Improved robot autonomy and acoustic techniques allow modern methods to rely on AUVs and ASVs. Furthermore, recent AUV-guided diver navigation methods have relied on novel AUVs. However, improved AUV DR navigation in concert with modern acoustic, state estimation, and diver technologies enable a reliable range-only diver navigation system without any surface presence using commercial AUVs.

2.2 Range-Only Single-Beacon Navigation

Using range measurements calculated via an acoustic signal's travel time to localize an underwater vehicle is common and offers significant navigation improvements over dead reckoning methods. Although range-only localization initially relied on fixed beacons, research has extended this approach to moving vehicles within the past two decades. One example is the homing problem, where a single vehicle uses range measurements from a static beacon to travel to its location. A more recent technique is the moving long baseline (MLBL) concept, where AUVs with less accurate navigation systems localize themselves using range measurements from AUVs with highly capable navigation systems. These concepts demonstrate the benefit of using heterogeneous underwater elements to cooperatively accomplish a task and

provide the inspiration for extending ROSB navigation to teams involving humans.

Early research into underwater ROSB navigation came in the form of the homing problem. An inherent issue with range data is that a single observation does not provide the receiving vehicle with enough information to determine its position or the direction to travel towards the beacon. Vaganay et al. [45] determined that a vehicle path perpendicular to the direction of the fixed beacon maximized the Fisher information matrix, which quantifies the amount of information an observation contains about the state estimate [46]. Therefore, driving a heading perpendicular to the bearing of the beacon provides the best configuration to localize it. However, as this research focused on the homing problem, maintaining that heading is insufficient because it would cause the AUV to drive a fixed-radius circle around the static beacon. Instead, the authors implemented an EKF onboard the AUV to estimate its error covariance ellipsoid and the beacon's location. Upon each range measurement, these state variables helped determine the heading the AUV should steer. This behavior formed a spiral-like trajectory, with the AUV initially driving nearly perpendicular to the beacon's direction and, eventually, driving directly at the beacon as its confidence in the beacon's location increased [45]. Research into range-only measurements eventually advanced from the homing problem to cooperative navigation between multiple AUVs.

Vaganay et al. [47] first introduced the multi-vehicle relative navigation technique known as the MLBL concept. These authors recognized that removing the requirement for pre-surveyed beacons made this concept less logistically demanding and more appropriate for large area surveys than traditional LBL methods. Their MLBL system relied on three types of AUVs: the communication and navigation aid (CNA) vehicles, the search, classify, and map (SCM) vehicles, and the reacquire and identify (RI) vehicles. Each vehicle type served a specific role within the mission. Specifically, CNA vehicles enabled the MLBL concept by serving as moving beacons, SCM vehicles provided automatic target recognition (ATR) using sonar, and RI vehicles identified the targets at the locations of interest communicated by the SCM vehicles [47]. Accomplishing a complex mission through a heterogeneous AUV infrastructure served as an inspiration for this thesis.

For the MLBL research mentioned above, the two CNAs were positioned behind and on each side of the SCM and RI vehicles. Figure 2-4 illustrates this vehicle configuration. The less capable vehicles received two minutes of range measurements to the CNAs every 15 minutes throughout the transit. Simulations showed that the SCM vehicle could have an

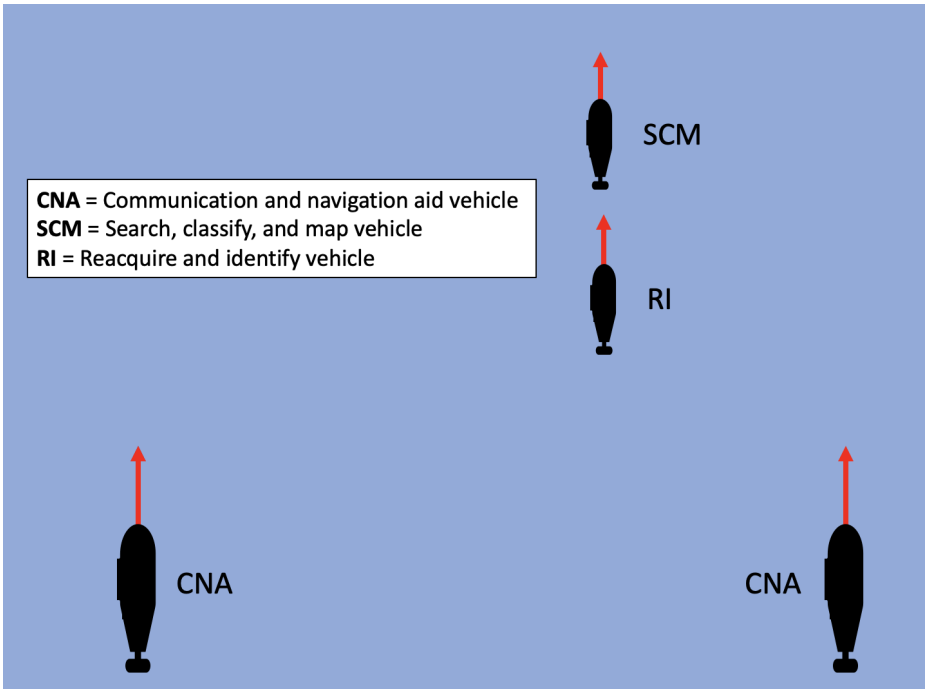


Figure 2-4: Moving long baseline (MLBL) vehicle configuration during transit. The communication and navigation aid (CNA) vehicles are 500 meters apart. Additionally, the reacquire and identify (RI) vehicles are 250 meters ahead of the CNAs and the search, classify, and map (SCM) vehicles are 350 meters ahead of the CNAs [47].

absolute position error as high as 60 meters after 60 minutes of travel if relying exclusively on its DR navigation solution with DVL bottom-lock. However, when using their MLBL filter to incorporate the ranges from the two CNAs, each of the 100 simulations resulted in an absolute position error below 15 meters after 60 minutes [47]. It is important to note that this configuration relied on range measurements from two CNAs. ROSB navigation only uses one CNA, thus requiring more complex autonomous behaviors to eliminate the spatial ambiguity problem.

Fallon et al. [48] extended the MLBL concept to a single ASV serving as the CNA for multiple AUVs. This research comments on the importance of a CNA trajectory that benefits localization for other AUVs with pre-planned paths. They determine that a non-linear estimator's measurement function can provide accurate and unambiguous location estimates using range measurements to a CNA if the two vehicles' relative position changes throughout the transit. Conversely, if their relative motion remains constant, the system's uncertainty grows. Therefore, they recommend two relative geometry configurations for an AUV navigating relative to an ASV: one is that the ASV zigzags across the AUV's path,

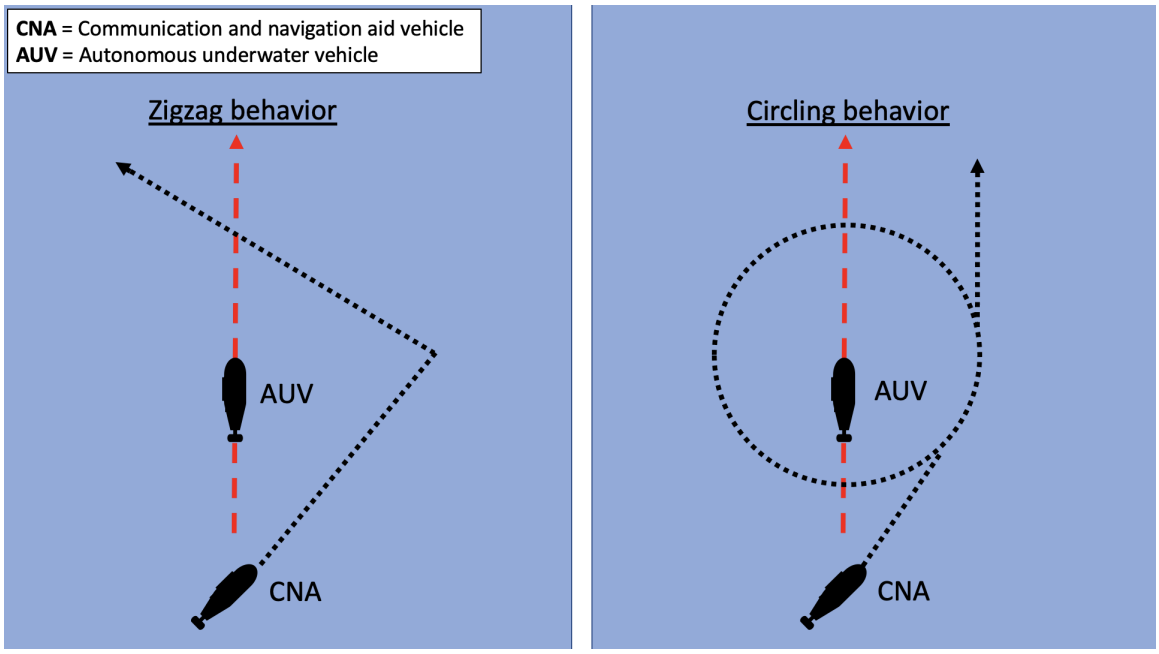


Figure 2-5: The zigzag behavior (**left**) and circling behavior (**right**) provide beneficial range-only single-beacon relative geometry between a communication and navigation aid vehicle and a second autonomous vehicle [48].

and the other is that the ASV encircles the AUV continuously [48]. Figure 2-5 provides a visual of these two methods. However, ROSB navigation performance degrades without successful acoustic communication regardless of the relative motion between elements.

2.3 Acoustic Communication and the Woods Hole Oceanographic Institution Micromodem 2

Underwater acoustic telemetry requires adaptations to traditional digital communication techniques due to the ocean channel's inherent rapid fading and time dispersive nature. It is essential to understand these principles to appreciate the difficulty of underwater communication. First, we introduce the environmental factors that impact the transmitted signal. For example, attenuation is the weakening of the signal as it travels through the ocean and multipath propagation means the signal takes multiple indirect paths to the receiver, with each one requiring a different amount of time. Additionally, receivers must also account for frequency spreading caused by the transmitter's and receiver's relative motion, called the Doppler shift. Then, we discuss the signal processing techniques required to overcome these difficulties and introduce the WHOI Micromodem 2 [15], a state-of-the-art acoustic

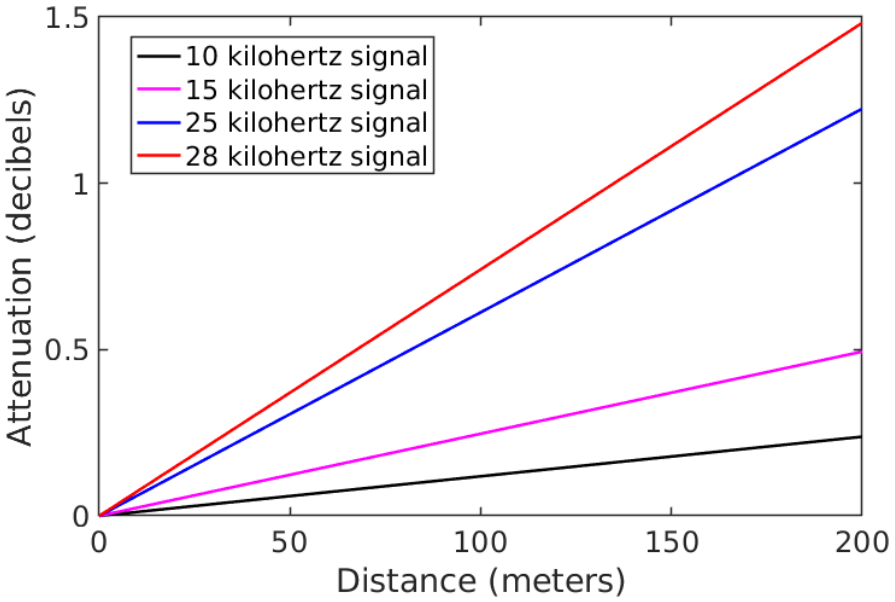


Figure 2-6: Acoustic attenuation at relevant frequencies and ranges.

communication device.

Radio channels operate using high-frequency EM waves with the potential to carry large quantities of data. However, EM waves rapidly attenuate in water, thus requiring underwater communication to rely on acoustic transmissions. Although acoustic waves also experience attenuation as they travel through the ocean due to absorption and scattering, it is at a much lower degree than EM waves. An approximation for the attenuation coefficient of ocean acoustic signals is

$$\alpha' \simeq 3.3 \times 10^{-3} + \frac{0.11 f^2}{1 + f^2} + \frac{44 f^2}{4100 + f^2} + 3.0 \times 10^{-4} f^2, \tag{2.1}$$

where α' is in decibels per kilometer and f is the frequency in kilohertz [49]. This attenuation is a primary reason acoustic communication is generally performed below 30 kilohertz, reducing the data each signal can carry compared to higher frequency radio waves.

Acoustic modems on AUVs typically use transducers rated for frequencies between 10 and 28 kilohertz [50]. Figure 2-6 uses Equation 2.1 to show the estimated attenuation at ranges relevant to our cooperative navigation approach. Since the attenuation is low across all frequencies shown in that figure, it is beneficial to use higher frequencies because the signal can carry more data.

One can visualize the multipath propagation of acoustic transmissions through a ray-

tracing approach. For an omnidirectional source, the signal spherically expands from the transmission point and eventually bounces off the ocean surface and seafloor. Multipath can even occur with a theoretically perfect planar wave because sound bends towards regions of low sound speed. Snell's law models this phenomenon with

$$\frac{cos(\theta)}{c} = \text{constant}, \tag{2.2}$$

where θ is the ray angle with respect to the horizontal plane and c is the local sound speed [49]. Since each of the multiple paths is a different length, the receiving transducer detects numerous arrivals of the same packet. These arrivals cause intersymbol interference (ISI), which makes equalization, or distortion reversal, difficult. Although ISI occurs in airborne radio channels, it is generally no larger than two or three symbol intervals. However, in the underwater acoustic channel, ISI can span several tens of symbol intervals [51].

Frequency spreading caused by the relative motion of the source and receiver also adversely affects communication. Accounting for Doppler shift is especially important with communication between two AUVs or an AUV and a diver, where the relative speed and relative heading between both elements rapidly varies. Kilfoyle and Baggeroer [52] wrote that the one-way Doppler shift can be estimated by

$$\triangle f_{doppler} \approx \frac{0.35Hz}{V * f_{source}}, \tag{2.3}$$

where V is the relative speed of the two elements in knots and f_{source} is the frequency in kilohertz of the acoustic signal at the source. This Doppler spreading causes a frequency translation as well as continuous frequency spreading, the latter of which is much more difficult for a receiver to mitigate [52]. Overall, acoustic communication receivers must be robust enough to operate in these harsh environments and adaptable enough to account for the constantly changing nature of the medium.

Early acoustic telemetry devices relied on frequency-shift keying (FSK), where modification of the carrier signal frequency represents changes in the digital signal. Receivers of this signal rely on energy detection, or incoherent, algorithms, which are designed to be robust to the channel's multipath and Doppler spreading. To decode the incoming message, the signal passes through narrow-band filters, and the receiver makes decisions based on the energy output of those filters. Incoherent methods do not compensate for ISI but are still

Figure 2-7: Signal frame consisting of a channel probe, training sequence, and data block [53].

subject to its effects. Consequently, they suffer from bandwidth inefficiency and, therefore, are inappropriate for larger data rates [52].

Before the 1990s, phase-shift keying (PSK) modulation was deemed ill-suited for acoustic communication due to the ocean channel's reverberation and spatiotemporal fluctuation. In PSK, the digital signal is encoded in the carrier signal's phase. Therefore, receivers require coherent, or phase detection, algorithms to decode these incoming transmissions [52]. Although more complex, PSK offers higher data rates and power efficiency than FSK. Stojanovic et al. [51] [53] demonstrated the first successful coherent receiver for ocean acoustic communication. The contribution of this influential research was an adaptable receiver that jointly performed synchronization, where the timing is aligned between the sender and receiver to assist in data extraction, and equalization, which seeks to remove the ISI [54]. This process relies on the signal frame shown in Figure 2-7. Although the following paragraph presents a brief overview of the process, Stojanovic et al. [51] [53] and Johnson et al. [54] provide more in-depth descriptions.

To start the process, the incoming signals are sampled at the Nyquist rate or higher, brought to baseband, and then decimated. Next, they are frame synchronized through matched filtering to the known channel probe at the beginning of each message. Then, the training sequence allows the receiver to coarsely estimate and, therefore, remove the mean Doppler shift. Any remaining Doppler shift is eliminated by a phase-locked loop [54]. Each input signal is then coherently combined in the feedforward section, after which decisions are made on the encoded information. Finally, the receiver updates its parameters, such as the carrier phase estimate and feedback equalizer coefficients, by analyzing the minimum mean square error (MMSE) of its symbol prediction [51]. Stojanovic et al. [53] provides an equation to calculate the MMSE:

$$MMSE = E\{|d_n - \hat{d}_n|^2\}, \tag{2.4}$$

where d_n is the transmitted symbol and $\hat{d}_n$ is the receiver's estimated symbol. These combiner and channel estimate parameters are rapidly changed to track the varying acoustic channel. Experiments with this new receiver structure showed successful PSK modulation, with 1,000 symbols per second transmitted over 48 nautical miles [51]. An additional benefit to this method is it takes advantage of the multipath nature of the environment by combining the signal's various arrivals to increase the signal-to-noise ratio (SNR) [55]. Since the 1990s, coherent communication technology has improved and is now considered the more robust acoustic messaging method.

The WHOI Micromodem 2 [15], first introduced in 2010, takes advantage of these incoherent and coherent communication techniques to provide a compact, low-power, and robust acoustic messaging platform. The Micromodem 2 offers increased performance over the WHOI Micromodem 1 [56]. For example, the Micromodem 2 has improved processing power and decreased power consumption compared to its predecessor. It also allows both frequency-hopping FSK (FH-FSK) and PSK modulation, with FH-FSK offering a data rate of 80 bits per second and PSK offering a burst data rate as high as 5,388 bits per second. The modem transmits data in packets, which may be broken up into multiple frames depending on the selected data rate and quantity of data. There are seven data rates, with Rate 0 being the only one to use FSK modulation. Rate 1 through Rate 6 utilize quadrature phase-shift keying (QPSK), where each phase symbol is ninety degrees apart and represents two bits. The data rates use various error correction codes and cyclic-redundancy checks to ensure the fidelity of every frame. Recently, WHOI engineers within the Acoustic Communication Group developed the Flexible Data Protocol (FDP), which provides greater flexibility and utility than their standard communication protocol. Overall, the Micromodem 2 provides a high-performing acoustic communication device specifically designed for underwater networking and navigation [57].

One Micromodem 2 function involves pinging another modem system to determine the signal's travel time. Specifically, the ping function determines the TWTT, which equals the time it takes for a signal to travel from a transmitting transducer to a second transducer and, then, for a reply signal from the second transducer to reach the original transmitter. Upon receiving the reply, the first modem system subtracts the turnaround time, or the time it takes for the second modem system to generate a reply signal, from the TWTT and divides the result by two to determine the one-way-travel-time (OWTT). Then, the OWTT can be

multiplied by the sound speed to calculate the range between the transducers [48]. State estimation algorithms can utilize these range measurements to achieve accurate localization of autonomous vehicles.

2.4 Nonlinear Least-Squares Estimation and Factor Graphs

This section provides the background on NLS analysis and factor graphs required to understand the state estimation method used to localize the diver. First, we discuss why we chose an NLS approach. Then, we show the process to get from the maximum a posteriori (MAP) equation to the NLS cost function. We conclude by discussing standard optimization methods to solve the NLS cost function.

We build on the research from Fallon et al. [48], which compared the use of an EKF, particle filter (PF), and NLS approach for cooperative navigation between an AUV and an ASV using range measurements. Their problem statement is very similar to ours in the reliance on ranging data between autonomous elements. Using a 70-minute transit where two autonomous kayaks localized themselves off a single CNA, those authors concluded that the PF outperformed the EKF, the online NLS algorithm outperformed the PF, and the post-processed NLS solution performed the best. Since the EKF has a fixed linearization point, imprecise range and odometry measurements during the transit resulted in sporadic position estimates requiring extended time periods to reconverge back to an accurate value. Conversely, NLS methods reoptimize their entire path upon each new measurement, resulting in better performance throughout the test. Additionally, this method relied heavily on nonlinear range measurements, which are more accurately captured by the PF and NLS approaches. However, the PF requires more computational power than the NLS method. Although not accomplished in this thesis, our process will eventually operate on a computer attached to a submersible dive tablet. For that reason, a more efficient state estimation algorithm is beneficial due to processing constraints on the tablet's computer. Overall, Fallon et al. [48] recommended NLS approaches for future work in cooperative navigation. Since our HAT navigation architecture shares many similarities with their method, we take their advice and rely on NLS estimation.

The iSAM2 algorithm [58] is a state-of-the-art NLS method that uses a factor graph-based approach to provide real-time location estimates and efficient incremental optimiza-

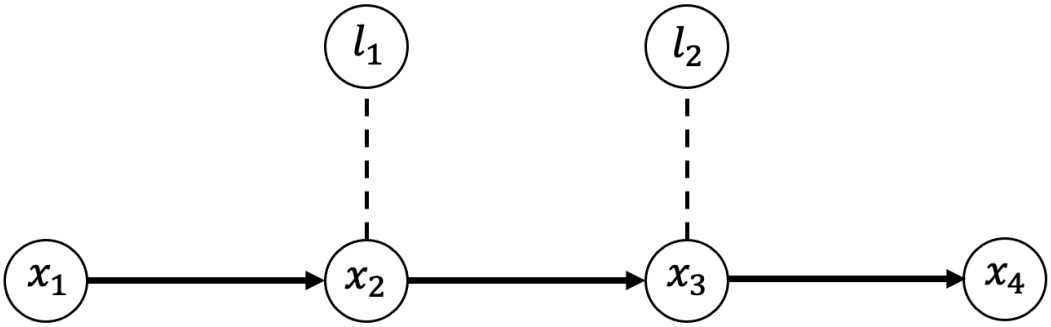


Figure 2-8: Simple cooperative navigation state estimation example.

tion. Similar to Dellaert and Kaess [59], we introduce the foundational elements of that algorithm through a simple example. However, our example is representative of the AUV-ASV cooperative navigation method from the previous paragraph.

As the AUV traveled through the ocean, OWTT range measurements to the ASV augmented its dead reckoning solution. Figure 2-8 represents the first few steps of this problem. At position x_1, a GPS fix determines the AUV's initial position. Following that fix, the AUV travels forward and receives two range measurements to the ASV, located at l_1 for the first and l_2 for the second. An effective state estimation algorithm would find all the AUV's unknown state variables, such as its position, given all the measurements it receives. The most common algorithm to determine these variables is the MAP estimate, which seeks to solve the equation

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, p(X|Z), \tag{2.5}$$

where X is the unknown state variables, Z is the measurements, and $p(X|Z)$ is the posterior density representing the probability of the state variables given the measurements [59].

Bayes' theorem states that

$$p(X|Z) = \frac{p(Z|X)p(X)}{p(Z)}, \tag{2.6}$$

but since all the measurements Z are given, $p(Z)$ is ignored as a normalization factor that does not impact the MAP estimate [59]. Additionally, rewriting $p(Z|X)$ as a likelihood underscores that the value is a function of the state variables (X) because the measurements

(Z) are provided and merely serve as parameters. Therefore, since the likelihood $(l(X;Z))$ is proportional to $p(X|Z)$, the restructured MAP equation is

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, l(X;Z)p(X), \tag{2.7}$$

which provides the best form of this estimate to visualize its connection to factor graphs [59]. However, before describing factor graphs, we find the posterior for our AUV-ASV example using Bayes' theorem to serve as a comparison. In Figure 2-8, the posterior density $(p(X|Z))$ is proportional to

$$l(X;Z)p(X) = l(x_1;z_1)l(x_2,l_1;z_2)l(x_3,l_2;z_3) \tag{2.8}$$

$$\times\, p(l_1)p(l_2)p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3), \tag{2.9}$$

where z_1 is the GPS measurement, z_2 is the first range measurement to l_1, and z_3 is the second range measurement to l_2.

A factor graph is a bipartite graph that seeks to represent a global function as a series of simpler local functions. Within a factor graph, there are factors (ϕ_i) and variables (x_j). The relationships between factors and variables are expressed through edges (e_{ij}), which show that each factor is only a function of the variables adjacent to it [60]. Overall, the factor graph defines the global function as a product of the local factors through

$$\phi(X) = \prod_i \phi_i(X_i), \tag{2.10}$$

with $X_i \subseteq X$ being the subset of variables in X that neighbor the factor ϕ_i [59].

The factor graph from our simple example is seen in Figure 2-9, where the small numbered black circles represent factors, the large white circles represent variables, and the black lines represent edges. This graph is represented by the equation

$$\phi(X) = \phi_1(x_1)\phi_2(x_2,l_1)\phi_3(x_3,l_2) \tag{2.11}$$

$$\times\, \phi_4(l_1)\phi_5(l_2)\phi_6(x_1)\phi_7(x_2,x_1)\phi_8(x_3,x_2)\phi_9(x_4,x_3). \tag{2.12}$$

The factor order in Equation 2.11 and 2.12 mimics Equation 2.8 and 2.9, respectively, so it is clear which values the factors correspond to in the original posterior density calculation.

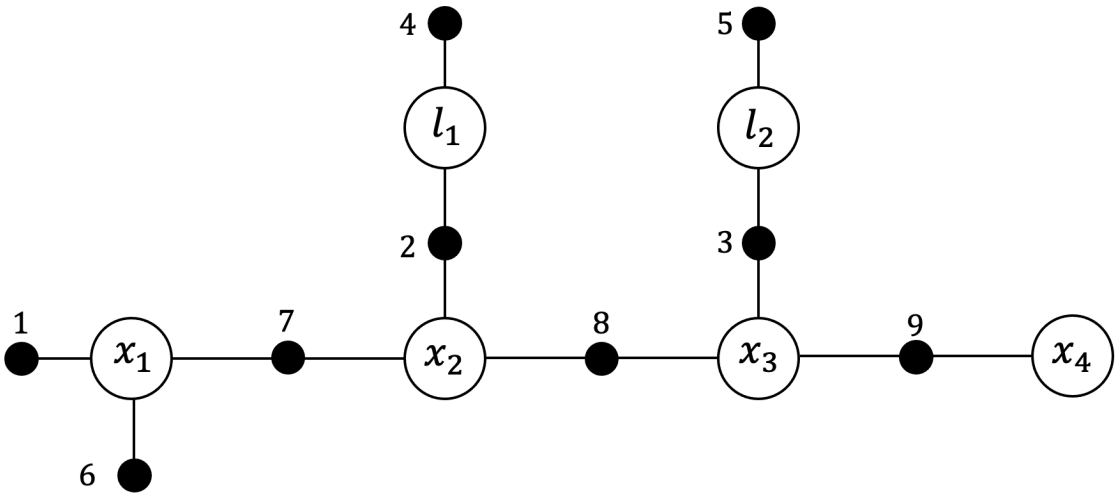


Figure 2-9: Simple cooperative navigation factor graph example.

Therefore, Dellaert and Kaess [59] show that, since the product of factors is proportional to the posterior density, Equation 2.5 can be rewritten as

$$X^{MAP} = \underset{X}{\operatorname{argmax}} \prod_{i} \phi_i(X_i). \tag{2.13}$$

Odometry and two-dimensional range factors are the primary measurements in our simple example and, more broadly, this thesis. In Figure 2-9, odometry factors link adjacent poses x_{i-1} to x_i and range factors connect a pose x_i to a landmark l_{i-1}. The measurement model $(h_i(X_i))$ for an odometry factor is defined as

$$h(x_{i-1}, x_i) \triangleq \ominus x_{i-1} \oplus x_i, \tag{2.14}$$

where, for this example, x_{i-1} and x_i represents subsequent global coordinates of the AUV, $\ominus$ represents pose inversion, and $\oplus$ represents pose composition [61]. Additionally, being global coordinates, the poses are in the special Euclidean group SE(2), which is the set of rigid body motions in two dimensions. The measurement model for a range factor is

$$h(x_i, l_{i-1}) \triangleq \| \, t_{x_i} - t_{l_{i-1}} \, \|_2, \tag{2.15}$$

where t_{x_i} represents the AUV's global translation vector at x_i, $t_{l_{i-1}}$ represents the ASV's global translation vector at l_{i-1}, and $\| \, e \, \|_2$ represents the Euclidean distance between those

two points [61].

Throughout this thesis, we assume measurements are corrupted by additive Gaussian noise with known covariance. Therefore, the equation for each factor is

$$\phi(X_i) \propto \exp\Big\{ -\frac{1}{2} \parallel h_i(X_i) - z_i \parallel^2_{\Sigma_i} \Big\}, \tag{2.16}$$

where $h_i(X_i)$ is the model that predicts the measurement given X_i, z_i is the measurement from the sensor and $\parallel e \parallel^2_{\Sigma_i} \triangleq e^T \Sigma_i^{-1} e$ is the squared Mahalanobis distance with covariance matrix Σ_i [58]. However, by removing the $1/2$ and taking the negative log of Equation 2.16, Dellaert and Kaess [59] show it can be substituted into Equation 2.13 to convert the MAP estimate to an NLS problem:

$$X^{MAP} = \underset{X}{\operatorname{argmin}} \sum_i \parallel h_i(X_i) - z_i \parallel^2_{\Sigma_i} . \tag{2.17}$$

As mentioned previously, the measurement functions are nonlinear. Therefore, to solve Equation 2.17, the measurement function ($h_i(.)$) must be linearized using the Taylor expansion

$$h_i(X_i) = h_i(X_i^0 + \Delta_i) \approx h_i(X_i^0) + H_i \Delta_i, \tag{2.18}$$

where H_i is the measurement Jacobian, X_i^0 is the linearization point, and Δ_i is the state update vector [59]. The measurement Jacobian is the partial derivative of the measurement function at the linearization point. Substituting the above equation into Equation 2.17 results in

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \sum_i \parallel h_i(X_i^0) + H_i \Delta_i - z_i \parallel^2_{\Sigma_i}, \tag{2.19}$$

where Δ^* is the state update solution at this linearization point [59]. The covariance (Σ_i) can be eliminated from the above equation by, instead, pre-multiplying each term by $\Sigma_i^{-1/2}$. This is possible because, as shown by Dellaert and Kaess [59], the Mahalanobis norm can be rewritten as

$$\parallel e \parallel^2_{\Sigma} \triangleq e^\top \Sigma^{-1} e = \left(\Sigma^{-1/2} e\right)^\top \left(\Sigma^{-1/2} e\right) = \parallel \Sigma^{-1/2} e \parallel_2^2 . \tag{2.20}$$

Then, by reordering Equation 2.19 and setting $A_i = \Sigma_i^{-1/2} H_i$ and $b_i = \Sigma_i^{-1/2} \left(z_i - h_i(X_i^0) \right)$, that equation becomes

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \parallel A\Delta - b \parallel_2^2, \tag{2.21}$$

where $A \in \mathbb{R}^{m \times n}$ is the sparse measurement matrix containing all Jacobians (A_i) and b is a right-hand-side vector containing all prediction errors (b_i) [59]. An important relation to recognize is that the measurement matrix (A) represents the original factor graph linearized at X_i^0. The measurement matrix for our simple cooperative navigation problem is shown below.

$$[A|b] = \begin{array}{c} \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \end{array} \left(\begin{array}{cccccc|c} \delta l_1 & \delta l_2 & \delta x_1 & \delta x_2 & \delta x_3 & \delta x_4 & b \\ \hline & & A_{13} & & & & b_1 \\ A_{21} & & & A_{24} & & & b_2 \\ & A_{32} & & & A_{35} & & b_3 \\ A_{41} & & & & & & b_4 \\ & A_{52} & & & & & b_5 \\ & & A_{63} & & & & b_6 \\ & & A_{73} & A_{74} & & & b_7 \\ & & & A_{84} & A_{85} & & b_8 \\ & & & & A_{95} & A_{96} & b_9 \end{array} \right)$$

In equation 2.21, Δ is equal to $\left(\delta l_1^\top, \delta l_2^\top, \delta x_1^\top, \delta x_2^\top, \delta x_3^\top, \delta x_4^\top \right)$. Once Δ is found, the new state estimate is calculated by $X_i^0 \oplus \Delta$, where $\oplus$ is simple addition for our scenario. This new state estimate is then used as the linearization point in the following iteration [58].

Dellaert and Kaess [59] describe multiple nonlinear optimization methods to find Δ using the NLS cost function ($c(X)$) derived from Equation 2.17:

$$c(X) \triangleq \sum_i \parallel h_i(X_i) - z_i \parallel_{\Sigma_i}^2 . \tag{2.22}$$

An important reminder is that the right side of Equation 2.22 corresponds to the summation of factors within the nonlinear factor graph. Therefore, our example's NLS problem aims to find the global minimum of the sum of the odometry and two-dimensional range factors.

This thesis relies on three nonlinear optimization methods: gradient descent, Gauss-

Newton, and Powell's dogleg algorithm. Each of these start with an initial state estimate (X_0) and revise that state using an update (Δ) to find the subsequent state estimate (X_1). This process continues to get from each state (X_t) to the following state (X_{t+1}).

Gradient descent is a popular technique in machine learning algorithms. Dellaert and Kaess [59] show this method calculates the update value (Δ) with the equation

$$\Delta = -\alpha \nabla c(X)|_{X=X_t}. \tag{2.23}$$

The step size (α) is a user-selected parameter that determines whether large or small steps should be taken in the negative gradient direction. An issue with this method is it converges slowly near the global minimum.

The Gauss-Newton method offers fast convergence by relying on direct methods for least squares. The update is found using the normal equations

$$A^\top A\Delta = A^\top b, \tag{2.24}$$

where A is the measurement Jacobian and b contains the prediction errors [59]. The Hessian, being the set of all second partial derivatives taken with respect to Δ, is equal to $A^T A$ and determines whether the solution is a global minimum, local minimum, or saddle point. For example, if the Hessian is positive semi-definite, it is a global minimum. This technique can result in diverging estimates if the objective function is not quadratic and the prior is inaccurate [59].

Although both gradient descent and Gauss-Newton have drawbacks, Powell's dogleg algorithm [62] combines both to efficiently and accurately perform NLS optimization. This algorithm computes the gradient descent update and takes that step. Next, it calculates the Gauss-Newton update and moves from the gradient descent solution toward the Gauss-Newton solution but stops at a trust-region boundary. At each iteration, Powell's dogleg algorithm determines the size of the trust-region based on a gain ratio that signifies whether the cost function from Equation 2.22 has decreased [59]. Another NLS optimization method is Levenberg-Marquardt, which also relies on a trust region. However, Powell's dogleg is more efficient, which is beneficial due to the eventual computational constraints of the computer on the diver's submersible tablet.

Overall, factor graph-based NLS estimation using Powell's dogleg algorithm provides

efficient and accurate state updates for underwater cooperative navigation problems relying on range measurements. The measurement matrix, representing the underlying factor graph, serves as a foundational element of iSAM2 to provide efficient real-time location updates. For more in-depth coverage of these methods, we recommend Kaess et al. [58], Dellaert and Kaess [59], and Kschischang et al. [60].

2.5 Incremental Smoothing and Mapping 2

Assuming Gaussian noise and measurement models with known covariances, iSAM2 tries to solve the linear least-squares problem shown in Equation 2.21. Two graphical models important to iSAM2 are the Bayes net and the Bayes tree. A Bayes net represents the graphical model version of the square root information matrix (R), which has been used in SLAM problems to solve the robot trajectory and map [63]. A Bayes Tree is a directed data structure with nodes that represent the cliques within the Bayes net [64]. It is important to introduce these models to understand iSAM2 and why it provides an advantageous state estimation algorithm for our proposed system.

The measurement matrix, introduced in the previous section, represents the system's Gaussian factor graph with the number of rows equal to the number of measurements. As shown by Dellaert and Kaess [59], QR factorization converts the measurement matrix (A) into a Bayes net. This matrix decomposition method seeks to solve

$$A = QR, \tag{2.25}$$

where A is the measurement matrix, Q is an orthogonal matrix, and R is an upper triangular matrix associated with the Bayes net [59]. Therefore, this step converts a Gaussian factor graph, represented by A, into a chordal Bayes net (R) through elimination. Figure 2-10 shows the Bayes net for our simple cooperative navigation example. The R matrix representing that Bayes net is below, where the Xs represent non-zero entries.

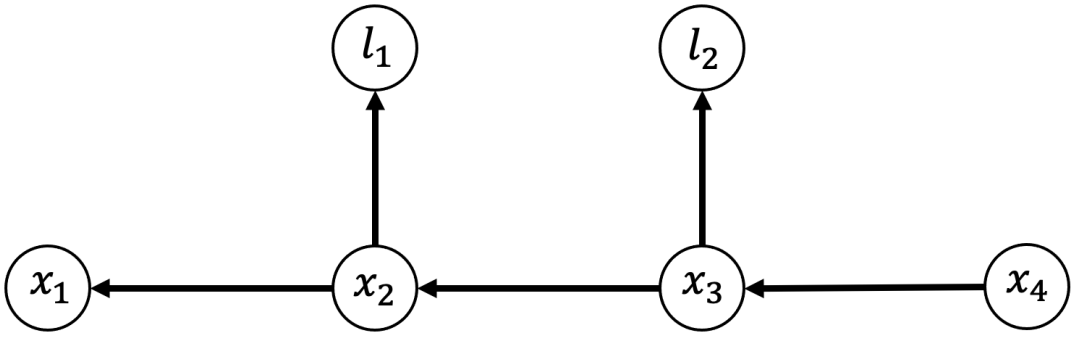


Figure 2-10: Bayes net for our simple cooperative navigation example.

$$R = \begin{array}{c} \begin{array}{cccccc} l_1 & l_2 & x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} X & & & X & & \\ & X & & & X & \\ & & X & X & & \\ & & & X & X & \\ & & & & X & X \\ & & & & & X \end{pmatrix} \end{array}$$

The key graphical model within iSAM2 is the Bayes tree. Cliques are groups of fully connected variables. The example we have been utilizing through these past two sections has a relatively uninteresting clique structure considering there are no more than two variables in a clique. Two examples from our Bayes net are the root clique $C_1 = \{x_3, x_4\}$ and $C_2 = \{l_2, x_3\} = \{l_2 : x_3\}$. The tree's root clique represents the last variables eliminated. Additionally, C_2 is written as $\{l_2 : x_3\}$ to delineate the separator (x_3), which serves as the intersection between that clique and its parent clique [59]. Figure 2-11 shows the full Bayes tree for the simple ASV-AUV cooperative navigation example. Kaess et al. [64] provides more complicated examples and explains the full algorithm to turn a chordal Bayes net into a Bayes tree.

Bayes trees are useful because they encode the problem's joint density $P(X)$, where X is the set of variables in the tree [59]. However, other graphic models accomplish this as well. The true utility of the Bayes tree within the SLAM problem is that the structure

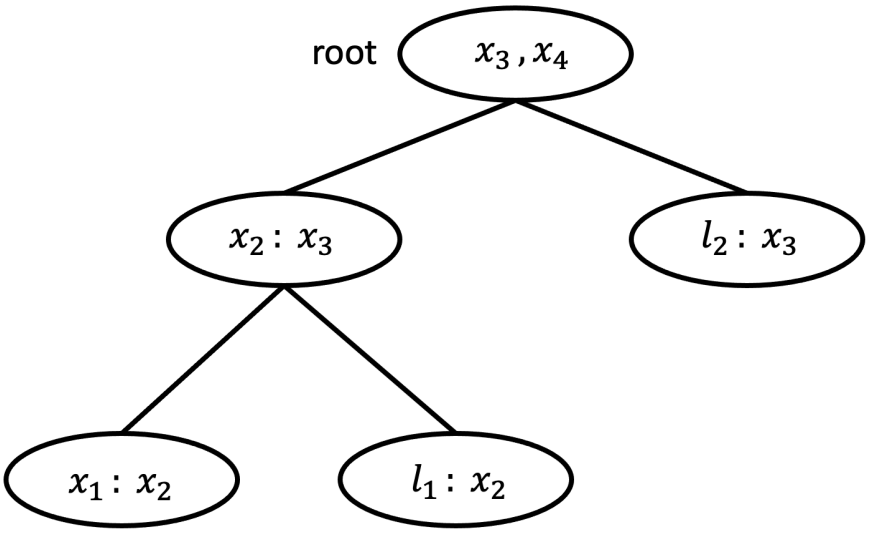


Figure 2-11: Bayes tree for our simple cooperative navigation example.

Algorithm 1 One step of the iSAM2 algorithm [58]

In/Out: Bayes tree (τ), nonlinear factors (ϕ), linearization point (X_i^0), update (Δ)
In: New nonlinear factors (ϕ'), new variables ($X_i^{0\prime}$)
1: Add new odometry or range factors ($\phi := \phi \cup \phi'$)
2: Initialize new variables ($X_i^0 := X_i^0 \cup X_i^{0\prime}$)
3: Update linearization point using fluid relinearization
4: Update Bayes tree by recalculating cliques affected by new variables and relinearization
5: Complete partial state update by solving for Δ using Powell's dogleg algorithm
6: Calculate the current state estimate $\Delta^* = X_i^0 + \Delta$

captures the flow of information during elimination, thereby causing newly added factors to have minimal impact baring a loop closure [58]. To add a factor between variables x_j and x_k, all cliques involving those variables up through the root are converted back to a factor graph. After the measurement is added, the factors are relinearized, and the full Bayes tree is recreated through elimination [59]. The parts of the Bayes tree below the affected cliques and not containing x_j and x_k are untouched. For example, to add a new odometry factor $\phi(x_4, x_5)$ in our simple example, only the root clique would be affected. Kaess et al. [64] contains more detailed information on this process and on how the Bayes tree promotes efficient incremental inference.

The Bayes tree enables iSAM2's fluid relinearization and partial state updates to provide a computationally efficient and accurate SLAM method. Whereas an EKF has the potential to diverge due to its fixed linearization point, iSAM2 tracks each variable's linearization

point and only relinearizes when necessary. When the state update (Δ_i) is calculated at each linearization point (X_i^0), iSAM2 marks variables in Δ_i that exceeded threshold β compared to the previous state update vector (Δ_{i-1}). Only the linearization point for these marked variables is updated, saving computational power over other methods that relinearize every variable. Similarly, iSAM2 only performs nonlinear optimization on variables impacted by the addition of the new factor. This partial state update starts performing full back-substitution at the root but stops when it encounters a clique with all variables in Δ changing less than threshold α. Therefore, along with providing real-time state estimates, it smooths the entire path at each step in a computationally efficient manner [58]. We summarize iSAM2's main steps in Algorithm 1. Some of the steps provide information specific to this thesis, such as the relevant factor types and the nonlinear optimization method.

Along with providing more detail on iSAM2's algorithmic process, Kaess et al. [58] compares it to other state-of-the-art SLAM methods and shows that iSAM2 is the most computationally efficient and accurate. Therefore, iSAM2 provides the best online NLS state estimation algorithm for the diver.

Chapter 3

Methods

This chapter details the software, hardware, cooperative navigation approach, and communication architecture that enables our human-autonomy teaming technique.

3.1 Software and Hardware

Cooperative navigation between the AUV and diver is enabled by the ROS™ middleware for robotics [17]. The highest software organization unit within ROS is the package, and the three most important ROS packages for our HAT architecture are `ros_remus` [65], `ros_acomms` [66], and `ros_hat` [18]. Both `ros_remus` and `ros_acomms` are existing repositories managed by members of The Acoustic Communications Group at WHOI. The former provides autonomous mission planning capabilities to a REMUS AUV [65], and the latter allows ROS messages to be encoded and transported across an acoustic link via the Micromodem 2 [66]. However, during this research, we created the ROS Human-Autonomy Teaming repository, abbreviated as `ros_hat`, to enable cooperative navigation between a REMUS AUV and a diver. Being familiar with these repositories and how they interact with sensors is foundational to understanding our HAT navigation approach.

During all testing and simulation, our computers had Ubuntu 20.04 Linux operating systems using ROS Noetic Ninjemys, which was released on May 23rd, 2020 [67]. Common components of ROS packages are nodes, messages, and services. Each node is an executable that performs specific tasks contributing to the overall goal of the package. Nodes communicate with each other through messages and services. By subscribing to a specific category of message, known as a topic, a node receives messages published by other nodes to that topic.

Services are different and operate in a request and response manner [68]. These capabilities are foundational to the function of `ros_remus`, `ros_acomms`, and `ros_hat`.

The proven performance of the REMUS AUV series, combined with its hardware and software capabilities, makes it our choice for this HAT navigation technique. Gallimore et al. [69] discusses the REMUS's autonomy computer, which allows us to send commands to the vehicle's operating system and receive data from its sensor suite while ensuring the primary computer retains control of its safety. Colloquially, the operating system is referred to as the front seat computer, while the autonomy computer is referred to as the back seat computer. The reason for these terms can be understood by envisioning someone driving a car with a passenger in the back seat providing navigation instructions; even though the passenger is telling the driver where to go, the driver operates the vehicle and is responsible for its safety.

Existing software enables the communication between the front seat and back seat computers. The back seat computer runs `ros_remus` to provide mission planning capabilities via actions sent to the REMUS's front seat using the Remote Control (RECON) protocol [70]. The specific node that accomplishes this is the `REMUS Node` [65]. Important actions for this thesis are transits, where the REMUS travels from one point to another, and circles, where the REMUS circles a specific coordinate. The front seat controls the vehicle via a series of preprogrammed objectives, which include the action to perform as well as depth, sensor configuration, and other necessary parameters. Once a ROS node posts an action on the back seat computer, it is converted from Python data types to American Standard Code for Information Interchange (ASCII) messages with eight-bit checksums. The RECON client sends this information to the front seat over a User Datagram Protocol (UDP) socket [69]. Once the action reaches the front seat, it overrides the current vehicle behavior. In this way, the back seat computer can control the REMUS. However, once this action finishes, the front seat computer regains control, provided the back seat does not send another action.

Sensor data follows the reverse process from the vehicle's operating system to the autonomy computer. For example, location data from the INS gets passed through RECON to the `REMUS Node` and posted within the `status` topic for other ROS nodes to access [65]. These nodes can then determine a specific action to post based on the sensor data. Farrell et al. [71] defines an AUV behavior as converting sensor readings to vehicle actions. Behavior-based autonomy takes a series of behaviors and arranges them together in an appropriate order to

navigate the vehicle and accomplish a mission. An early version of RECON demonstrated the impact of behavior-based autonomy through chemical plume tracing. Specifically, it allowed a REMUS to enter an operating area, detect a chemical plume, and follow that plume to its source [71]. Although HAT navigation is a different mission, we utilize data from sensors and acoustic messages to influence the vehicle's behavior.

The WHOI Micromodem 2 is the standard acoustic communication device on REMUS AUVs [20]. Acoustic communication between two autonomous elements is enabled by `ros_acomms`. The four important nodes within `ros_acomms` are the `Acomms Driver Node`, the `Packet Dispatch Node`, the `Message Queue Node`, and the `TDMA Node`. Although the primary roles of each are below, the `ros_acomms` documentation [66] provides more detailed node descriptions and capabilities as well as information on other nodes within the repository that enable simulation.

1. `Acomms Driver Node`
    - Communicates with the Micromodem 2 using `pyAcomms`
2. `Packet Dispatch Node`
    - Uses the appropriate codec to decode the bits sent from the `Acomms Driver Node` into the proper ROS message
3. `Message Queue Node`
    - Uses the appropriate codec to encode a ROS message into its compact bit-packet form
4. `TDMA Node`
    - Queries the `Message Queue Node` at a predetermined interval
    - Passes messages to the `Acomms Driver Node` for transmission

The `Message Queue Node` subscribes to a group of topics the user wants acoustically transmitted. Once another ROS node posts a message to one of those topics, the `Message Queue Node` uses a series of codecs to encode the data into a compact bit-packet representation. For our HAT navigation method, the `TDMA Node` queries the `Message Queue Node` every 15 seconds looking for new data to transmit because that slot duration provided the

fastest communication between the diver and AUV. If there are packets in the queue, the highest priority message gets passed to the `Acomms Driver Node` for transmission. However, if that message is less than the rate's packet size, the `Message Queue Node` bundles as many messages as will fit. Sending the packet to the modem is assisted by `pyAcomms` [72], which serves as the interface between the Micromodem 2 and `Acomms Driver Node`. On the receiving modem, the `Acomms Driver Node` handles the incoming packet from the Micromodem 2 and passes the data to the `Packet Dispatch Node`, which accomplishes the reverse of the `Message Queue Node` to publish the received ROS message for other nodes to access [66]. In this way, `ros_acomms` allows ROS messages to be sent underwater from one computer to another.

Our novel ROS package, called `ros_hat`, relies on `ros_remus` and `ros_acomms`. On the diver side, the `ros_hat` software is designed to run on a processor attached to a submersible dive tablet and interact with the tablet's sensors. For the REMUS, `ros_hat` operates on the back seat computer, utilizing existing `ros_remus` capabilities to collect important sensor data and control the vehicle. Specifically, `ros_remus` communicates the autonomous decisions made by `ros_hat` to the vehicle's front seat computer using RECON. Similarly, `ros_acomms` serves as the interface between `ros_hat` and the Micromodem 2.

Although `ros_hat` runs on both the diver's tablet and the REMUS, it serves very different roles on each. The three main nodes within `ros_hat` are the `Navigator Node`, `Director Node`, and `Estimator Node`. The `Navigator Node` operates on the REMUS's back seat computer, while the `Director Node` and `Estimator Node` run on the diver's submersible tablet. The primary roles and responsibilities of each are shown below.

1. `Navigator Node`
    - Receive message data from the Micromodem 2 using `ros_acomms`
    - Determine the next coordinate the REMUS should travel to based on the diver's estimated location
    - Ensure the back seat computer retains control of the REMUS by initializing and updating a transit action using `ros_remus`
    - Collect necessary sensor outputs using `ros_remus` for transmission to the diver
    - Transmit the appropriate message to the diver using `ros_acomms`

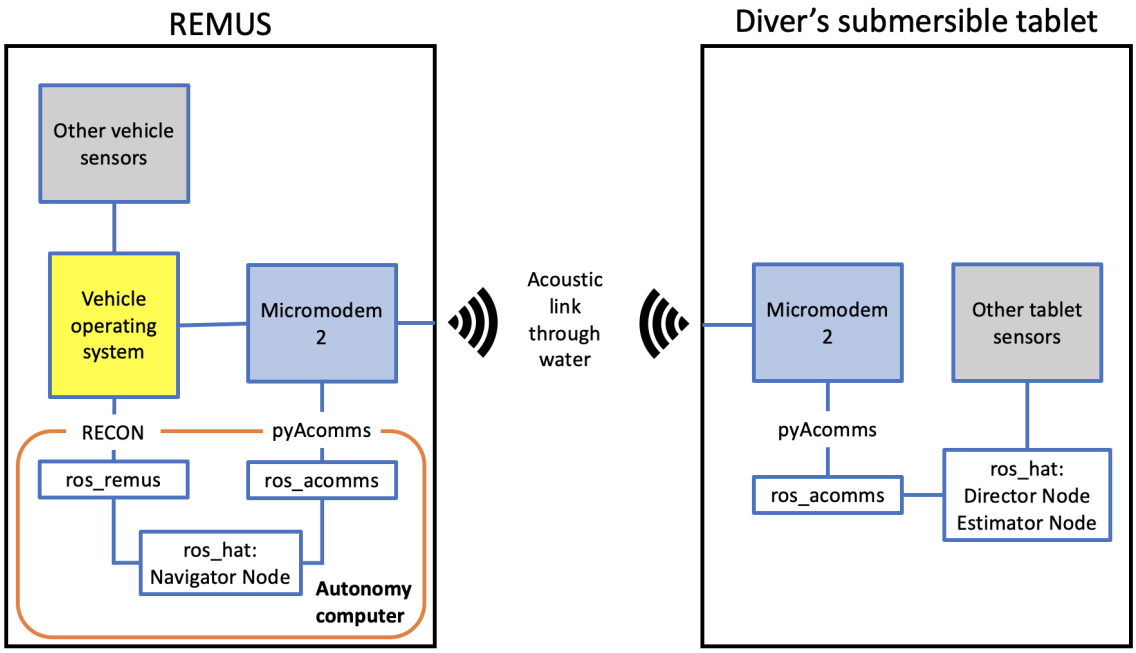


Figure 3-1: Overview of our system's hardware and software architecture.

2. `Director Node`

   - Receive message data from the Micromodem 2 using `ros_acomms`
   - Calculate range measurements using data sent by the REMUS
   - Pass range data to the `Estimator Node`
   - Collect necessary data from sensors and nodes for transmission to the REMUS
   - Transmit the appropriate message to the REMUS using `ros_acomms`

3. `Estimator Node`

   - Receive range data from the `Director Node`
   - Receive odometry data from the AHRS
   - Given an initial fix, provide online diver location estimates using odometry and range measurements

Figure 3-1 provides an overview of our navigation method's hardware and software architecture. Although not completed in this thesis, we plan to have an ADLE3800PC computer [73] run the required software on the diver's submersible tablet. Aside from the Micromodem 2, the diver's tablet requires a depth sensor, a GPS or Global Navigation Satellite

System (GNSS) receiver, and an AHRS. Depth data from both elements are needed to turn a three-dimensional range into a two-dimensional range, as will be explored in Section 3.3. The GPS or GNSS receiver is required to get the diver's initial position before they submerge for the transit. Additionally, the AHRS we utilized for this research is the Sparton M2 AHRS [74]. Fortunately, WHOI's Deep Submergence Lab had written a ROS node for the Sparton M2 [75], which we used to get accurate heading data to other `ros_hat` nodes.

On the REMUS, `ros_hat` nodes require data from the GPS, INS, and conductivity, depth, and temperature (CTD) sensors in addition to the Micromodem 2. The AUV needs a GPS or GNSS receiver for the same reason as the diver. The INS offers a high-accuracy navigation solution to the REMUS using a fiber optic gyroscope (FOG) and DVL [76]. As will be discussed in Chapter 4, a more accurate REMUS location estimate is beneficial to the performance of our HAT navigation method. Appendix B provides additional details on the nodes and topics within `ros_hat` as well as in-depth diagrams showing the specific software and hardware architecture proposed for our HAT navigation method.

3.2 Communication Sequence

This section describes the communication sequence between the diver and REMUS AUV throughout the transit. It is crucial to understand how data is exchanged between both elements because the diver's state estimation algorithm and the REMUS's autonomous behavior are dependent on receiving updated information from their teammate. First, we discuss how the Micromodem 2 transmits four critical message types between the diver and REMUS using `ros_acomms`. Then, we discuss how the `ros_hat` nodes control the message traffic between the diver and AUV.

The `Navigator Node` and `Director Node` control the transmitted packet's type and content on the REMUS and diver, respectively. We use the Micromodem 2's Rate 1 FDP communication, which utilizes QPSK modulation and Bose–Chaudhuri–Hocquenghem error-correcting code [57]. There are four unique acoustic messages sent between the diver and AUV: an Initialization Packet, a Follower Packet, a Leader Packet, and a Terminate Packet. Figure 3-2 shows the contents of these messages. The diver's submersible tablet sends the Initialization Packet, Follower Packet, and Terminate Packet, while the REMUS always responds with a Leader Packet.

Initialization Packet	
float	diver_depth
float	diver_latitude
float	diver_longitude
float	target_latitude
float	target_longitude

Follower Packet	
float	diver_speed_over_ground
float	diver_depth
float	diver_latitude
float	diver_longitude
unsigned integer	diver_time_of_fix

Leader Packet	
float	remus_depth
float	remus_uncertainty
float	remus_latitude
float	remus_longitude
float	sound_speed
float	one_way_travel_time

Terminate Packet	
unsigned integer	terminate

Figure 3-2: Initialization, Follower, Leader, and Terminate Packet contents.

```
diver_depth: #m
  codec: float
  min_value: 0.0
  max_value: 30.0
  precision: 1
diver_latitude:
  codec: float
  min_value: -90.0
  max_value: 90.0
  precision: 6
diver_longitude:
  codec: float
  min_value: -180.0
  max_value: 180.0
  precision: 6
target_latitude:
  codec: float
  min_value: -90.0
  max_value: 90.0
  precision: 6
target_longitude:
  codec: float
  min_value: -180.0
  max_value: 180.0
  precision: 6
```

Figure 3-3: Message codec for the Initialization Packet. The precision parameter represents the number of decimal points desired for the variable.

As discussed in the previous section, the `Message Queue Node` within `ros_acomms` uses codecs to convert ROS messages into a compact bit-packet representation for acoustic transmission. Conversely, the `Packet Dispatch Node` uses the same codecs to decode the received data into a ROS message [66]. Within a message codec, the user defines the range of values each variable can hold. The largest message sent in our HAT navigation process is the Initialization Packet due to the two pairs of latitudes and longitudes with six decimals of precision, which offer about four inches of accuracy per value [77]. Figure 3-3 shows the message codec for this ROS topic. Given the precision we set for each variable, the transmitted message can be expressed with 16 bytes. However, the packet also requires an additional nine bytes for the header, one byte to identify `ros_acomms` sent the packet, and another byte to identify the message type within the codec. Therefore, the Initialization Packet requires the modem to transmit 27 data bytes.

Each FDP packet sent by the Micromodem 2 can hold eight mini-frames, with the first mini-frame holding nine data bytes and the others holding 13. Consequently, it takes three mini-frames to transmit the Initialization Packet. Since we utilize Rate 1 communication, each mini-frame contains 896 symbols, so three mini-frames have 2,688 QPSK symbols [57].

As discussed in Section 2.3, the Micromodem 2's signal frame contains more than just the data; it includes a frequency-modulated probe, null time, and overhead symbols, which include the training data. We used the default probe length and null time of 200 symbols and 250 milliseconds, respectively. Additionally, there are 702 overhead symbols. Therefore, the signal frame requires an additional 902 symbols along with the data symbols.

As shown in the Micromodem 2 User Guide [57], the packet duration (d) can be found by

$$d = \frac{\text{packet length (in QPSK symbols)}}{\text{bandwidth}}. \tag{3.1}$$

Given the 2,688 data symbols, 200 probe symbols, and 702 overhead symbols, the Initialization Packet contains a total of 3,590 symbols. Therefore, it takes 718 milliseconds to transmit that quantity of QPSK symbols with a five-kilohertz bandwidth. By adding the 250-millisecond null time, we calculate the transmission duration for the Initialization Packet as 968 milliseconds. Table 3.1 shows a similar analysis for the other transmission types relevant to our algorithm.

Packet Analysis						
Type	Initialization	Leader	Follower	Terminate	Ping	Ping reply
Probe symbols	**200**	**200**	**200**	**200**	**200**	**200**
Overhead symbols	**702**	**702**	**702**	**702**	**702**	**702**
Header bytes	9	9	9	9	6	7
`ros_acomms` byte	1	1	1	1	0	0
Message type byte	1	1	1	1	0	0
Data bytes	16	12	13	1	0	0
Total bytes	27	23	24	12	6	7
Mini-frames	3	3	3	2	1	1
Data symbols	**2688**	**2688**	**2688**	**1792**	**896**	**896**
Total symbols	3590	3590	3590	2694	1798	1798
Symbol time (ms)	718	718	718	539	360	360
Null time (ms)	250	250	250	250	250	250
Duration (ms)	968	968	968	789	610	610

Table 3.1: Packet lengths (in symbols) and durations (in milliseconds) for all transmissions [57]. The number of data symbols is dependent on the number of mini-frames, which is determined by the total data bytes. Additionally, the total number of symbols per transmission is the sum of the three bolded values in each column. Lastly, the packet duration is the sum of the time to transmit the symbols and the null time.

The communication sequence starts with the diver pressing a button on the submersible tablet to trigger the transmission of an Initialization Packet. Upon reception of the Initialization Packet, the REMUS responds with a Leader Packet. Then, a call-and-response process begins with the diver sending a Follower Packet and the REMUS responding with a Leader Packet. However, the REMUS must ping the diver to populate the variables within the Leader Packet. Algorithm 2 shows the Micromodem 2's ping sequence. Let the diver have Modem 0, and the REMUS have Modem 1.

Algorithm 2 Micromodem 2 Ping Sequence [57]

Modem 1 initiates the ping sequence
1: Modem 1 sends ping to Modem 0
2: Modem 0 detects ping
3: Modem 0 sends ping response to Modem 1
4: Modem 1 receives response and calculates OWTT

The "Follower, ping, Leader" cycle continues until the diver broadcasts a Terminate Packet, at which point the REMUS surfaces to conclude the trial. Algorithm 3 and Algorithm 4 show how the `Director Node` and the `Navigator Node` control the message traffic between the diver and AUV.

Algorithm 3 `Director Node` communication procedure

Require: Diver presses button on tablet to transmit Initialization Packet
  Send Initialization Packet
  repeat
    if Leader Packet received **then**
      Send Follower Packet
    end if
  until Diver presses button on tablet to transmit Terminate Packet
  Send Terminate Packet

Algorithm 4 `Navigator Node` communication procedure

  if Initialization Packet received **then**
    Send Leader Packet
  else if Follower Packet received **then**
    Conduct ping
    Send Leader Packet
  else if Terminate Packet received **then**
    Surface REMUS
  end if

3.3 Range calculation

The external measurement the diver relies on for localization is their range to the AUV. This section shows how the Micromodem 2 determines an acoustic signal's OWTT and how that OWTT is subsequently used to calculate the range between the diver and the REMUS. After discussing how the TWTT can be used to find the OWTT, we provide the equations and steps required for the diver to receive the OWTT information from the REMUS, convert a three-dimensional range into a two-dimensional range, and pass that data to the `Estimator_Node` for addition to the factor graph.

The Micromodem 2 offers two methods to get the range between transducers: one is based on a packet's arrival time, and the other utilizes a ping as described in Algorithm 2. Grund et al. [78] details the process to get OWTT measurements using synchronous navigation. Once the modem receives a pulse-per-second (PPS) signal from GPS, it can maintain that signal using its real-time clock (RTC). Therefore, given synchronous navigation is enabled, the modem transmits acoustic messages at the top of the second. The receiving modem reports the packet's arrival time in the reception statistics. Provided both modems' clocks are synchronized, and the transducers are within a second of acoustic travel distance, the OWTT is found by reading the fractional seconds of the reception time [57]. This technique

is acceptable for the ranges within our HAT navigation method. However, without an expensive precision clock, such as a chip-scale atomic clock (CSAC), the modem's RTC drifts at two parts per million [57], which could add 20 meters of range error per hour if both clocks drift in opposite directions with a sound speed of 1500 meters per second. Conversely, the modem's ping function removes the requirement to synchronize clocks and eliminates the impact of clock drift by using TWTT. Therefore, although it adds an additional step in the communication sequence, we chose the ping for its sustained accuracy independent of dive length.

Within Algorithm 4, the REMUS sends a Rate 1 FDP ping to the diver. As described by Fallon et al. [48], the Micromodem 2 uses the elapsed time between when it sent the ping and when it received the ping reply ($T_{elapsed}$) to calculate the OWTT using the following equations:

$$TWTT = T_{elapsed} - T_{turnaround}, \tag{3.2}$$

$$OWTT = \frac{TWTT}{2}, \tag{3.3}$$

where $T_{turnaround}$ equals 1,915 milliseconds, which is the known time it takes for the Micromodem 2 to detect a ping and send a Rate 1 FDP ping response [79]. Although we reference the OWTT frequently throughout the remainder of this document, it is important to note that it is calculated using the TWTT.

After calculating the OWTT, the Micromodem 2 posts it within a `CACMD` National Marine Electronics Association (NMEA) string. The `Navigator Node` monitors the modem's raw NMEA strings by subscribing to the appropriate `ros_acomms` topic. Therefore, when the `Navigator Node` detects the `CACMD` string, it retrieves the OWTT from that string and saves the REMUS's location and relevant sensor outputs for inclusion in the ensuing Leader Packet. Through this process, the Leader Packet's contents represent the REMUS's state upon receiving the OWTT measurement to the diver.

Similar to the `CACMD` string posted upon reception of the ping response, the modem outputs a `CACMA` string when it receives a ping. Upon detecting this NMEA string, the `Director Node` saves the diver's depth as well as other variables that will be discussed in the following section. These variables represent the diver's state upon receiving the ping. Therefore, the `Director Node` must simply wait until it receives the Leader Packet to have

a complete picture of the HAT's state during the previous ping.

When the `Director Node` receives the Leader Packet, it uses the OWTT to calculate the two-dimensional range to the REMUS's location when it received the ping response. Fallon et al. [48] shows that the following equation turns the OWTT into a three-dimensional range ($r_{3D,i}$):

$$r_{3D,i} = OWTT_i * c_i, \tag{3.4}$$

where $OWTT_i$ is the OWTT from the `CACMD` NMEA string and c_i is the sound speed. The `Director Node` retrieves both those values from the Leader Packet. The two-dimensional range ($r_{2D,i}$) is then calculated by an equation from Fallon et al. [48]:

$$r_{2D,i} = \sqrt{r_{3D,i}^2 - (d_{R,i} - d_{d,i})^2}, \tag{3.5}$$

where $d_{R,i}$ is the REMUS's depth as reported in the Leader Packet and $d_{d,i}$ is the diver's depth when they received the ping. After these calculations, the `Director Node` passes this two-dimensional range to the `Estimator Node`.

3.4 Online Diver Localization With Incremental Smoothing and Mapping 2

As discussed in Sections 2.4 and 2.5, our state estimation method is iSAM2 using Powell's dogleg algorithm for NLS optimization. The GTSAM library [16] allows these algorithms to be implemented on the `Estimator Node` to provide computationally efficient location updates for the diver throughout the entire dive. We utilize three factors in GTSAM: the prior factor, the odometry factor, and the two-dimensional range factor. First, the prior factor initializes the diver's position. Then, odometry and range factors progress the diver from that initialization point to the target. This section also details how NMEA strings posted by the Micromodem 2 ensure range factors are added at the correct location within the factor graph. Once the diver's location is predicted, `ros_hat` software posts navigation instructions to guide the diver to the target. Overall, the `Estimator Node` is the keystone of our system; all other algorithms are designed to provide the `Estimator Node` with the appropriate information so it can accurately predict the diver's location throughout the

transit.

The `Estimator Node` is dormant until the diver elects to send an Initialization Packet. At that point, a prior factor is added to the formerly empty factor graph. Since GTSAM operates in a local coordinate system, this prior factor sets the diver's location at (0.0, 0.0). However, the diver must be receiving GPS signals when the Initialization Packet is sent because the `Estimator Node` saves the latitude and longitude to assist in future conversions between local and global coordinates. After setting the prior, the system no longer needs GPS data, so the diver can submerge to commence the transit. Algorithm 5 shows the steps to add the initial prior factor.

Algorithm 5 Add initial prior factor

Require: Prior noise model (Λ_p)
In: Nonlinear factor graph (Φ), variables (X), global heading from AHRS (θ_i)
1: Convert θ_i to local frame heading (Θ_i)
2: Current local pose (X_i) = (0.0, 0.0, Θ_i)
3: Format X_i as a prior factor (Φ_i) with Λ_p
4: Add Φ_i to Φ ($\Phi := \Phi \cup \Phi_i$)
5: Initialize new variables ($X := X \cup X_i$)
return Φ, X

After adding the prior factor, the `Estimator Node` begins adding odometry factors every second. An odometry factor requires heading and speed. The global heading data is gathered from an AHRS and converted to the local reference frame. Conversely, the system does not have a method of measuring the diver's speed, so we rely on a constant speed approximation. Specifically, the diver's speed is assumed to be zero until the `Estimator Node` detects a specific ROS message. At that point, the diver's speed is assigned a positive value set within the node's parameters. The diver can toggle their speed estimate between zero and that constant value throughout the dive using ROS messages. As will be discussed in Chapter 4, this is currently conducted using a Linux laptop's terminal. However, we eventually envision the submersible tablet having a button to accomplish this. The constant speed approximation adds error to the odometry factors due to inconsistent swimming and unknown ocean currents. To compensate for this, we place low confidence in these factors by artificially increasing their covariance. Algorithm 6 shows how the `Estimator Node` adds an odometry factor.

As shown in Algorithm 6, the `Estimator Node` provides the diver's entire smoothed path once they elect to send a Terminate Packet. Although iSAM2 optimizes the diver's

Algorithm 6 Add odometry factor

Require: Odometry noise model (Λ_o), diver's speed (υ), diver's initial global coordinates (C_o)
In: Nonlinear factor graph (Φ), variables (X), local heading from last factor (Θ_{i-1}), global heading from AHRS (θ_i), time since last factor (δT)
1: Convert θ_i to local frame heading (Θ_i)
2: Calculate change in local heading ($\delta\Theta$) = Θ_i - Θ_{i-1}
3: Calculate change in distance (δD) = $\upsilon \times \delta T$
4: Format odometry factor (Φ_i) using $\delta\Theta$, δD, and Λ_o
5: Add Φ_i to Φ ($\Phi := \Phi \cup \Phi_i$)
6: Predict current pose (X_i) using X_{i-1}, δD, and $\delta\Theta$
7: Initialize new variables ($X := X \cup X_i$)
8: Calculate local iSAM2 estimate (X_i) using incomplete linear Δ found through fluid relinearization
9: Convert X_i to global coordinates (C_i) using C_o
if Terminate Packet sent **then**
  return Φ, X, C_i, smoothed path
else
  return Φ, X, C_i
end if

entire path every time it is updated, the `Estimator Node` only publishes the current location estimate until the diver is ready to end the mission, presumably indicated by the transmission of a Terminate Packet. Therefore, once navigation is complete, the diver receives the most accurate representation of their trajectory.

GTSAM also offers the ability to place factors retroactively. This is beneficial for our HAT navigation method because REMUS coordinates sent within the Leader Packet are outdated. As discussed in Section 3.2, the REMUS sends a Leader Packet containing state variables from when it had received the ping response. However, due to the slot duration we set for the `TDMA Node`, there could be up to 15 seconds between when the `Navigator Node` sets the variables in the Leader Packet and when it gets transmitted to the diver. By the time the diver's `Director Node` performs the range calculation and passes the result to the `Estimator Node`, that information is more than 15 seconds old. In that time, odometry factors have been added to the factor graph, progressing the diver's location estimate forward from the location they were when the REMUS performed the ping. Fortunately, the capabilities of the Micromodem 2 and GTSAM overcome this potential error.

The previous section discussed how the `Director Node` saves the diver's depth upon detecting a ping via the `CACMA` string. Along with saving the depth, it also marks the current

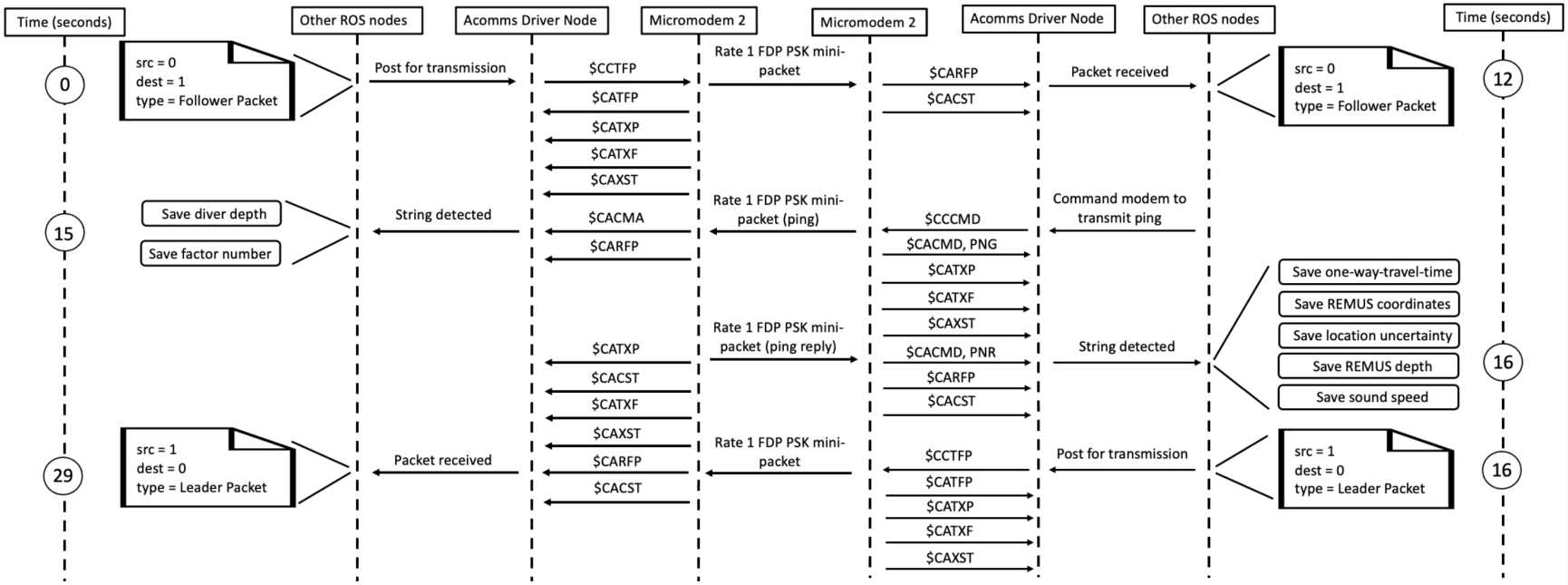
factor number on the `Estimator Node`. This factor number serves as a location placement for adding the eventual range factor. Therefore, when the Leader Packet arrives about 15 seconds later, the `Director Node` includes this factor number in the range data sent to the `Estimator Node`. Instead of adding the range factor to the most recent odometry factor, the `Estimator Node` adds it at the correct location in the graph. This process ensures the range factors are as accurate as possible. Algorithm 7 shows the steps for adding a two-dimensional range factor on the `Estimator Node`.

Algorithm 7 Add two-dimensional range factor

Require: Range noise model (Λ_r), diver's initial global coordinates (C_o)
In: Nonlinear factor graph (Φ), variables (X), two dimensional range (r_{2D}), factor number (ϕ) when `CACMA` string posted, data from Leader Packet $\rightarrow$ REMUS coordinates (C_R), REMUS location uncertainty (σ_R)
1: Convert C_R to local coordinates (X_R) using C_o
2: Format X_R as a prior factor (Φ_R) with σ_R
3: Add Φ_R to Φ ($\Phi := \Phi \cup \Phi_R$)
4: Initialize new variables ($X := X \cup X_R$)
5: Format two-dimensional range factor (Φ_r) using Λ_r, r_{2D}, and X_R
6: Add Φ_r to Φ at ϕ ($\Phi := \Phi \cup \Phi_r$)
7: Calculate local iSAM2 estimate (X_i) using complete Δ from full back-substitution
8: Convert X_i to global coordinates (C_i) using C_o
return Φ, X, C_i

Figure 3-4 details the sequence of Micromodem 2 commands and responses that occur during a single "Follower, ping, Leader" cycle [57] [79]. Although the time stamps were created using a single cycle, the timing was consistent across all cycles. Additionally, the time delay between when a `ros_hat` node posts a packet and when they are transmitted is due to the `TDMA Node`'s timing. As a result, the `Director Node` gets a Leader Packet about every 29 seconds. Fortunately, the ping operates independent of the `TDMA Node`, so its inclusion does not require additional time. Therefore, after the Initialization Packet is sent, the `Estimator Node` adds an odometry factor every second and a two-dimensional range factor every 29 seconds.

Algorithm 8 shows the full process running on the `Estimator Node`. An important parameter is the number of ranges to receive before adding them as factors to the graph. Since range measurements can lead to ambiguity without proper location diversity, this parameter seeks to mitigate that uncertainty by receiving ranges to multiple locations before updating the graph. For the first R ranges, the `Estimator Node` saves the range information



NMEA string	Purpose
$CACST	Statistics of received packet
$CCCMD	Transmit mini-packet ping
$CACMD	$CACMD, PNG = echo of ping command; $CACMD, PNR = reply to ping received
$CACMA	Received ping
$CARFP	Received FDP data packet

NMEA string	Purpose
$CCTFP	Transmit FDP data packet
$CATFP	Response to CCTFP command
$CATXP	Transmission started
$CATXF	Transmission ended
$CAXST	Statistics of transmitted packet

Figure 3-4: The sequence of Micromodem 2 commands and responses that occur during a single "Follower, ping, Leader" cycle [57] [79]. Let the diver have Modem 0, and the REMUS have Modem 1. Additionally, we set the time the `Director Node` posts the Follower Packet to 0 seconds to show the time it takes to complete a "Follower, ping, Leader" cycle.

in a Python dictionary. For the cooperative navigation field trials and simulations discussed in Chapter 4, we set R equal to three. On range $R+1$, all previous ranges are added to the factor graph in succession. Subsequent ranges are added immediately to the graph. This capability causes early iSAM2 estimates to be less erratic.

Algorithm 8 `Estimator Node` process

Require: Number of ranges to receive before adding to graph (R)
  In: Nonlinear factor graph (Φ), variables (X)
  Out: Diver's estimated global coordinates; diver's smoothed path
  Initialize: $\Phi = \emptyset, X = \emptyset$
  if Initialization Packet sent **then**
    Add prior factor according to Algorithm 5
    repeat
      if ≥ 1 second since last odometry factor was added **then**
        Add odometry factor according to Algorithm 6
      end if
      if Range data received from `Director Node` **then**
        if Number of ranges received $\leq R$ **then**
          Save range data
        else if Number of ranges received $= R+1$ **then**
          for Each saved range **do**
            Add range factor according to Algorithm 7
          end for
        else
          Add range factor according to Algorithm 7
        end if
      end if
    until Terminate Packet sent
  end if

Once the `Director Node` receives the diver's estimated coordinates from the `Estimator Node`, it guides the diver according to Algorithm 9. Eventually, the submersible tablet's screen will show these directions. However, as Chapter 4 will discuss, it is currently accomplished using a Linux laptop terminal.

Algorithm 9 `Director Node` guidance to diver

Require: Target coordinates (C^T)
  In: Diver's estimated global coordinates (C_i)
  1: Calculate distance (d_T) to target using C_i and C^T
  2: Find bearing (θ_T) from C_i to C^T
  Publish: d_T, θ_T

3.5 Range-Only Single-Beacon Autonomous Behavior Algorithm

The cooperative navigation research detailed in Section 2.2 serves as the inspiration for our ROSB approach. Within that research, the authors recommended two CNA behaviors to ensure beneficial relative geometry between the CNA and the AUV throughout the transit: the zigzag behavior and the circling behavior [48]. This section discusses why we chose the circling behavior. Then, it details the algorithms behind the three stages of our autonomous circling behavior, as shown in Figure 3-5. First, the REMUS circles the start location. Then, the REMUS circles the diver throughout the transit once they depart. Lastly, the REMUS circles the target. This autonomous behavior aims to provide the diver with range measurements to diverse locations to benefit their state estimation.

During our AUV behavior development, we sought to keep the diver and vehicle close enough to decrease acoustic transmission power but far enough apart to ensure safety. Size, weight, and power are critical measurement metrics for any underwater vehicle system. Keeping the diver and AUV relatively close makes it possible to use less acoustic transmission power for communication, thus saving energy for other sensors. Our goal was to keep the AUV and diver between 15 and 100 meters apart throughout the transit. We initially favored the zigzag behavior but discovered in simulation that the REMUS's operating system frequently rejected subsequent waypoints for violating the vehicle's turn radius. Once a point was rejected, the back-seat computer abandoned control to the front seat, causing the vehicle to perform undesired actions, such as returning to the start location. Conversely, the circling behavior did not have this issue and met all the requirements necessary for our HAT algorithm. Additionally, simulations and field tests by Masmitja et al. [80] [81] support this choice by concluding a circular trajectory centered around the target provides the best ROSB tracking with AUVs for static and dynamic targets.

The `Navigator Node` provides the user with a parameter to adjust the circle's size. Through simulation, we found that a 25-meter radius circle accomplished the goal of keeping the diver and AUV between 15 and 100 meters apart. Our autonomous behavior also maintains two meters of vertical distance between each element. When the `Navigator Node` receives the Follower Packet, it uses the diver's depth value to adjust the vehicle's position in this manner. Therefore, even if they are at the same location, vertical dispersion

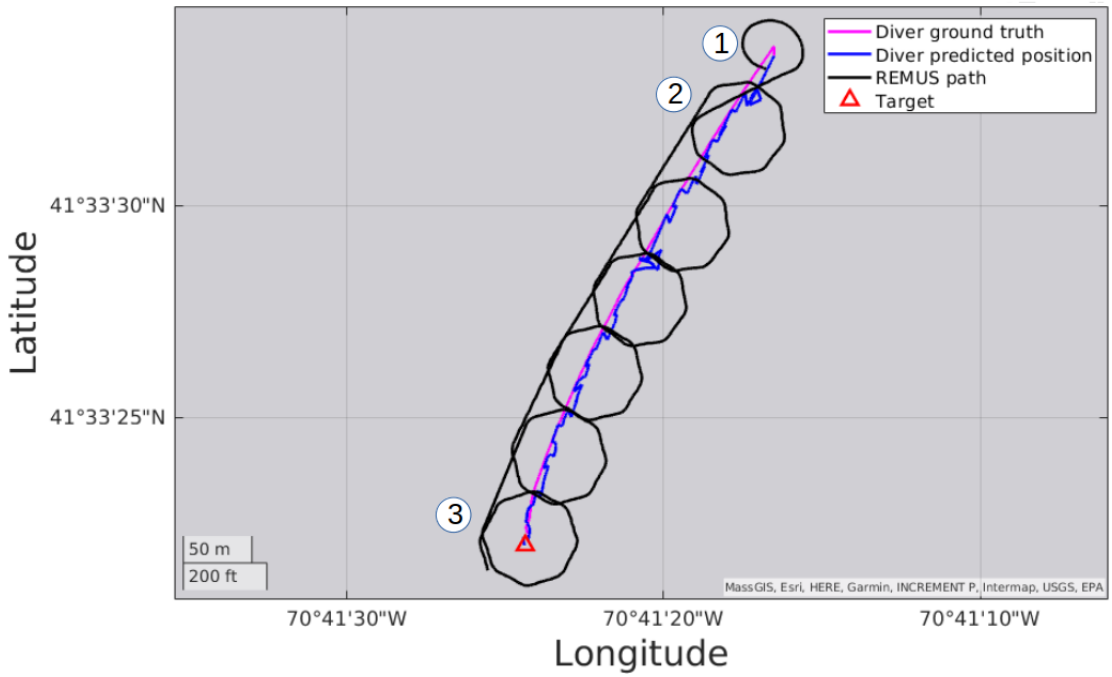


Figure 3-5: Example of our autonomous circling behavior for the REMUS. During Step 1, the REMUS circles the starting location. Step 2 involves the REMUS circling the diver throughout the transit. Lastly, the REMUS circles the target location at Step 3.

keeps the diver safe.

The REMUS circles the insertion point via a preprogrammed objective on the vehicle's operating system. At that point, `ros_hat` is passive and has not taken control of the vehicle. Upon receiving the Initialization Packet, the `Navigator Node` marks the diver's starting coordinates and the target coordinates. Both of these are used throughout the transit to predict the diver's current and future location. The REMUS continues to circle the start location until the first Follower Packet's arrival. Then, the `Navigator Node` takes control of the vehicle.

Figure 3-5 shows both the diver's ground truth path and online estimated coordinates to highlight that the REMUS determines its next waypoint using the estimated value. Although the behavior aims to drive a circle, it is accomplished through transit actions, which instruct the REMUS to travel from one coordinate to another. This technique provided the best way to ensure the `Navigator Node` retained control of the vehicle throughout the transit. It also resulted in the most consistent vehicle movement. Early efforts using circle actions resulted in unpredictable motion because it was difficult to control where the vehicle entered and

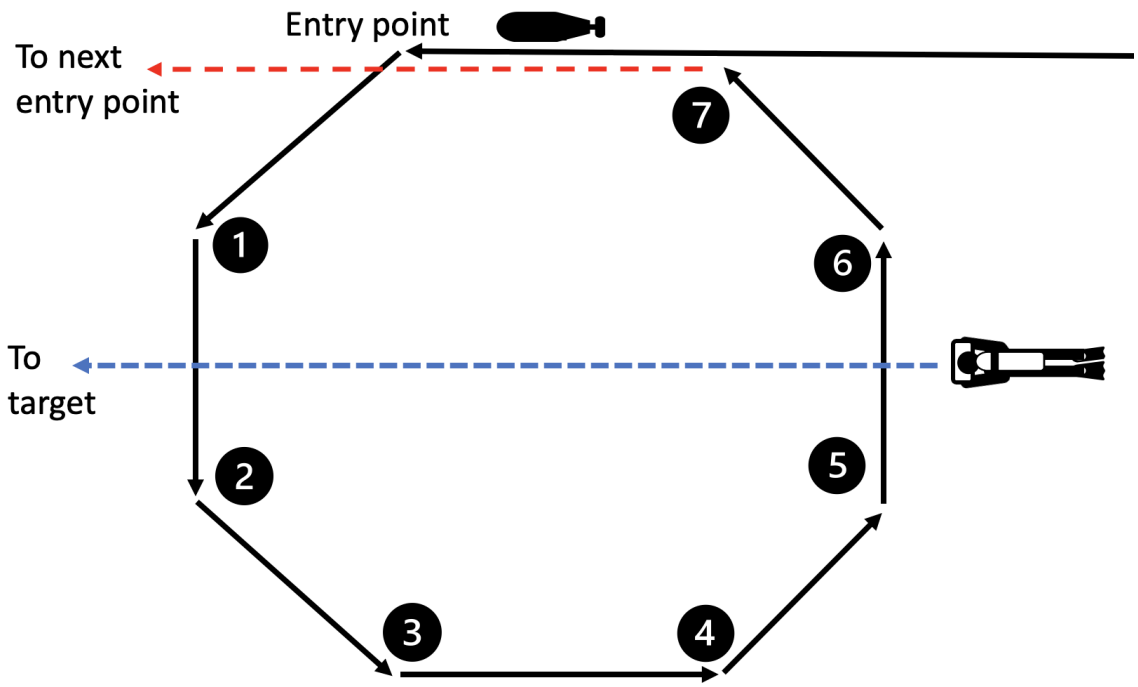


Figure 3-6: Close-up view of the circling behavior the REMUS performs during the dive.

departed each circle. Erratic paths from one circle to the next made timing difficult, and the diver was frequently outside the circle when REMUS started its pattern. Conversely, transit actions afford the most control, making it the best choice for this behavior. Although this approach makes our circle look like an octagon, we still refer to it as the circling behavior.

Figure 3-6 shows a more detailed look at the circle the REMUS performs throughout the dive. We define two types of waypoints within this autonomous behavior: entry points and intermediate points. The entry point serves as the location where the REMUS starts the circle. The intermediate points are the numbered vertices of the octagon. The `Navigator Node` uses two different processes to find each waypoint type. However, throughout the entire dive, only one transit action gets posted to the front-seat computer because the `Navigator Node` updates the goal coordinates of that transit action instead of posting a completely new action. Algorithm 10 summarizes how it calculates each entry point.

After finding the first entry point, the `Navigator Node` calculates the first intermediate point. However, it does not update the action yet. Instead, it begins to monitor the AUV's distance from the entry point. Once it determines it is within 15 meters of that location, it updates the transit action. This process is used at every waypoint, ensuring the transit action never completes and the back-seat computer retains control throughout the dive.

Algorithm 10 Find entry point

Require: Circle radius (r), target coordinates (C^T)
In: REMUS's current location (C_o^R); recent Follower Packet data → diver's estimated coordinates (C_o^d), depth (D_d), speed (s_d), and time of estimation (t_d)
Assume: Diver travels at heading from C_o^d to C^T
1: Set initial time allotted (T) to find entry point {We start with 5 seconds}
repeat
2: Predicted time at entry point (T_E) = current time + T
3: Predict the diver's location (C_i^d) at T_E using C_o^d, s_d, and t_d
4: Set C_i^d as the base of the circle
5: Calculate the entry point (C_E^R) to the circle using r and C_i^d
6: Find the distance between C_o^R and C_E^R
7: Find the speed required (s_R) to reach C_E^R in T seconds
8: T += 1 second
until 1.4 m/s $\leq s_R \leq$ 1.6 m/s {goal speed is 1.54 m/s}
9: Calculate depth (D_R) using D_d to maintain proper depth dispersion
if this is the first entry point **then**
return Post transit action with C_E^R and D_R
else
return Update transit action with C_E^R and D_R
end if

Based on the goal to perform a 25-meter radius circle, the distance between intermediate points is about 30 meters. Therefore, the REMUS travels about 240 meters each circle, which takes about 2 minutes and 35 seconds at 3 knots. Algorithm 11 and Algorithm 12 show how the REMUS performs the circle around the diver. Like Figure 3-6, these algorithms describe a counterclockwise circle. However, the `Navigator Node` is agnostic to the direction. Whether the circles are clockwise or counterclockwise is determined based on the relative locations of the diver and REMUS at the beginning.

Algorithm 11 Find intermediate point 1

Require: Intermediate leg length (L), target coordinates (C^T), diver's initial coordinates (C^d)
In: REMUS's current location (C_o^R), entry point coordinates (C_E^R), recent Follower Packet data → diver's depth (D_d)
1: Calculate transit bearing (θ_t) from C^d to C^T
2: Calculate heading to intermediate point 1 (θ_1) = $\theta_t - 45^o$
3: Find intermediate point 1 (C_1^R) using C_E^R, L, and θ_1
4: Calculate depth (D_1^R) using D_d to maintain proper depth dispersion
repeat
5: Calculate distance (d) between C_o^R and C_E^R
until $d \leq$ 15 meters
return Update transit action with C_1^R and D_1^R

Algorithm 12 Complete the circle

Require: Intermediate leg length (L)
In: REMUS's current location (C_o^R), intermediate point 1 coordinates (C_1^R), bearing from entry point to intermediate point 1 (θ_1), recent Follower Packet data $\rightarrow$ diver's depth (D_d)
for $i = 1$ to 6 **do**
1: Heading to next point $(\theta_{i+1}) = \theta_i - 45^o$
2: Find intermediate point $i+1$ (C_{i+1}^R) using C_i^R, L, and θ_{i+1}
3: Calculate depth (D_{i+1}^R) using D_d to maintain proper depth dispersion
repeat
4: Calculate distance (d) between C_o^R and C_i^R
until $d \leq 15$ meters
return Update transit action with C_{i+1}^R and D_{i+1}^R
end for

To get from the entry point to the first intermediate point in Figure 3-6, the `Navigator Node` finds the heading 45-degrees less than the bearing from the diver's initial position to the target and calculates the coordinate 30 meters from the entry point at that heading. Getting from intermediate point to intermediate point is accomplished similarly, as the `Navigator Node` finds the coordinates 30 meters away at a heading 45 degrees less than the heading to its previous point. When the REMUS determines it is at intermediate point seven, it follows Algorithm 10 to find the entry point to the next circle.

Throughout the dive, the `Navigator Node` monitors the REMUS's distance from the target. Once within a certain range set by the operator, it triggers the REMUS to continuously circle the target, as shown in step three of Figure 3-5. Algorithm 13 shows how the `Navigator Node` finds the entry point to circle the target.

Algorithm 13 Find entry point to circle the target

Require: Circle radius (r), target coordinates (C^T)
In: Recent Follower Packet data $\rightarrow$ diver's depth (D_d)
1: Set C^T as the circle's center
2: Calculate the entry point (C_E^R) to the circle using r and C^T
3: Calculate depth (D_R) using D_d to maintain proper depth dispersion
return Update transit action with C_E^R and D^R

The autonomous behavior enabled by the `Navigator Node` uses all the algorithms within this section to circle the diver throughout the dive. Algorithm 14 shows the full process.

Algorithm 14 HAT navigation autonomous behavior

Require: Circle radius (r), initial circle objective on front-seat computer
Initialization: Performing preprogrammed circle objective, Initialization Packet received, first Follower Packet received
1: Find first entry point using Algorithm 10
repeat
while Distance from target $\leq$ 50 meters **do**
2: Find intermediate point 1 using Algorithm 11
3: Complete the circle using Algorithm 12
4: Find next entry point using Algorithm 10
end while
5: Find entrance point to circle target using Algorithm 13
6: Find intermediate point 1 using Algorithm 11
7: Complete the circle using Algorithm 12
until Terminate Packet received

3.6 Summary

This chapter outlined our proposed HAT navigation method. The communication sequence discussed in Section 3.2, implemented using the software and hardware from Section 3.1, serves as the heartbeat to this process. The REMUS transits from the start location to the target using the behavior detailed in Section 3.5. Concurrently, the diver navigates according to directions based on their estimated coordinates, as provided by the algorithms in Section 3.4 using range calculations discussed in Section 3.3. In this way, every aspect of our method works together to get the diver to the target location without a surface presence or ocean current data.

Chapter 4

Experiments and Simulations

This chapter outlines the experiments and simulations used to validate the methods proposed in Chapter 3. First, we performed preliminary field tests to determine important ranging accuracy and speed parameters within our software. Then, we used kayaks as proxies for the REMUS and diver to test our HAT technique by performing real-time navigation assistance throughout a 400-meter transit. Lastly, simulations tested our method's performance in more adverse conditions than those experienced in the field. Throughout this chapter, we label the kayak serving as a proxy for the diver as the "diver kayak" and the kayak acting as a proxy for the REMUS as the "REMUS kayak."

4.1 Equipment Overview

Before discussing the field experiments performed during this research, it is important to review the equipment that allowed us to test the Micromodem 2's ranging performance and evaluate the methods from Chapter 3 using kayaks as proxies for the REMUS and diver. This critical equipment is shown in Figure 4-1 and listed below.

- Micromodem 2 25-kilohertz PSK deck box [82]
- 25-kilohertz transducer towfish [50]
- Arrow 100 GNSS receiver [83]
- Garmin GPS 18x receiver [84]
- Sparton M2 AHRS [74]

Figure 4-1: Key equipment used during our tests and experiments. **Top left:** Micromodem 2 25-kilohertz PSK deck box [82]. **Top center:** Garmin GPS 18x receiver [84]. **Top right:** Sparton M2 AHRS [74]. **Bottom left:** 25-kilohertz transducer towfish [50]. **Bottom center:** Arrow 100 GNSS receiver [83]. **Bottom right:** Sea-Bird Scientific SBE 37 SMP pumped MicroCAT [85].

- Sea-Bird Scientific SBE 37 SMP pumped MicroCAT [85]

As previously mentioned, our software runs on Linux laptops using ROS Noetic Ninjemys [67]. Custom `ros_hat` nodes designed for these tests send ping requests and messages via `ros_acomms` to the deck box, which contains the Micromodem 2 power amplifier [86], coprocessor [87], and digital signal processor (DSP) mainboard [88]. The Micromodem 2 then passes the data to the transducer for transmission. On the receiving side, the transducer converts the acoustic signal to an electrical signal and sends it to the Micromodem 2 within the deck box. The packet data is then made accessible to the `ros_hat` nodes with the help of `ros_acomms`.

The other sensors connect directly to the laptops and provide data for the `ros_hat` nodes. The Garmin GPS receiver and Arrow 100 GNSS receiver supply ground truth coordinates. The Garmin has an R95 accuracy of three meters [84], while the Arrow 100 GNSS receiver uses Differential GPS (DGPS) to provide an R95 accuracy of less than one-meter [83]. These accuracies mean 95% of the location values provided by the device would fall within that distance of the actual coordinates. Additionally, the Sparton M2 AHRS provides heading

measurements, and the Sea-Bird SBE 37 SMP provides sound speed data to convert OWTT values into ranges.

4.2 Static Ping Test

This section details the ping test used to verify the Micromodem 2's performance. Specifically, we sought to answer two questions: what is the Micromodem 2's static ranging accuracy, and is that accuracy range-dependent? First, we list the equipment required for this test. Then, we describe the experimental setup and process. As a reminder, the Micromodem 2 uses its ping function to find the TWTT, which is then used to calculate the OWTT. We conclude this section by discussing the results and how they compared to the modem's advertised static ranging accuracy.

4.2.1 Equipment

Table 4.1 shows the equipment for this test.

Static Ping Test Equipment			
Quantity	**Equipment**	**Specification**	**Purpose**
2	Deck Box	25 kHz PSK	Micromodem 2
2	Towfish	25 kHz	Transducer
1	Arrow 100	Single reading per location	Ground truth
2	Laptop (Ubuntu 20.04)	ROS Noetic Ninjemys	Execute software

Table 4.1: Equipment for the static ping test on September 22$^{\text{nd}}$, 2021.

4.2.2 Description

On September 22$^{\text{nd}}$, 2021, we conducted three static ping tests at a dock in Woods Hole, Massachusetts. As mentioned in Chapter 3, our goal was to keep the diver and REMUS between 15 and 100 meters apart. However, the largest dock we could access was 60 meters long. Therefore, the goal of this test was to determine the ping-derived OWTT accuracy at multiple ranges between zero and 60 meters.

The iSAM2 algorithm assumes measurements are corrupted by additive Gaussian noise with known covariance. The Micromodem 2 advertises an OWTT positive bias of 84 microseconds and a standard deviation of 42 microseconds at four-kilohertz bandwidth [57].

Figure 4-2: Static ping test set-up.

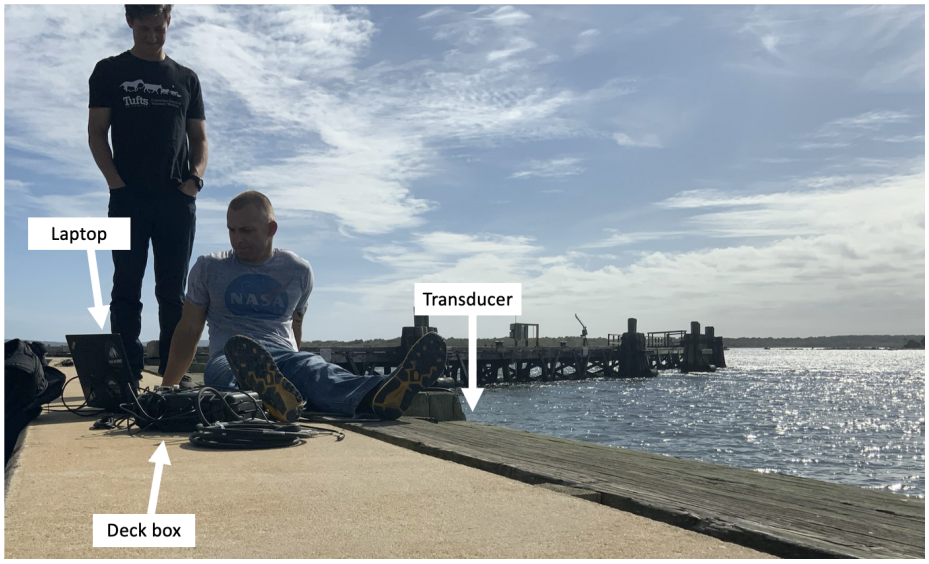


Figure 4-3: Picture from the static ping test performed on September 22nd, 2021.

However, those values were established using synchronized clocks and packets transmitted through a coaxial cable connecting side-by-side modems. Therefore, by corroborating these values using pings in the ocean, we could calculate the proper covariance matrix for OWTT-derived range factors with increased confidence.

Figure 4-2 shows this test's intended set-up and Figure 4-3 provides a picture of our actual environment. In that image, we are located with one of the deck boxes observing data posted to its associated laptop. Additionally, a towfish is hung off the cleat immediately to our left. The second laptop, deck box, and towfish are behind us on the same dock.

The ROS node written for this test ran on one of the laptops and used `ros_acomms` to trigger the modem to send a ping every five seconds. The OWTT value calculated from each ping sequence was saved in a bag file, which is a type of ROS file that records all messages posted to user-specified topics [89]. Using bag files is beneficial because MATLAB® [90]

functions can easily access the saved message data using the ROS Toolbox [91].

Utilizing this ROS node, we performed twenty minutes of ping data collection at three different distances. The transducer locations were determined using the Arrow 100, which provided R95 accuracies between 0.48 meters and 0.71 meters at these locations. The steps for each trial are shown below.

1. Place transducers at desired locations
2. Use Arrow 100 to determine each location's coordinates
3. Run software to conduct 20-minute ping data collection

After all three trials, we used MATLAB to compare the OWTT-derived range to the ground truth range calculated using coordinates provided by the Arrow 100. One issue with this test was that we did not have a CTD to measure the sound speed. As seen in Equation 3.4, this is a critical value in calculating the range. Although we set the depth of each modem at two meters and found the water temperature using local buoy data, we did not have an accurate salinity value. Fortunately, about a week after this test, we received the Sea-Bird SBE 27-SMP MicroCAT and measured the water's salinity at this location. Therefore, the sound speed utilized for this test was calculated with the correct temperature and depth but a questionable salinity reading. Regardless, we do not believe this uncertainty adds significant error due to the short distances between the modems.

4.2.3 Results

Table 4.2 shows the data gathered from this test. It is difficult to resolve the range bias (μ) as the Arrow 100 does not provide exact locations, making the ground truth ranges imperfect. However, the bias and standard deviation, or sigma (σ), values show that the modem's ranging accuracy is independent of range within the distances relevant to us. Therefore, by subtracting each trial's ground truth OWTT from the modem's OWTT data, we could combine each trial's OWTT errors into a single figure to get the total standard deviation value. As shown in Figure 4-4, our static ping test validates the Micromodem 2's advertised 42-microsecond standard deviation. This value helped calculate the range factor noise model for our kayak-to-kayak cooperative navigation experiment.

Static Ping Test			
Metric	**Trial 1**	**Trial 2**	**Trial 3**
Pings transmitted	287	289	267
Successful pings	279	260	267
Ping success percentage	97.21	89.97	100
Ground truth range (meters)	8.40	30.42	55.24
Modem range bias (centimeters)	-5.54	2.20	-11.01
Modem range sigma (centimeters)	3.63	3.83	3.35
Maximum absolute error (centimeters)	13.70	13.03	19.04

Table 4.2: Results from static ping test on September 22nd, 2021

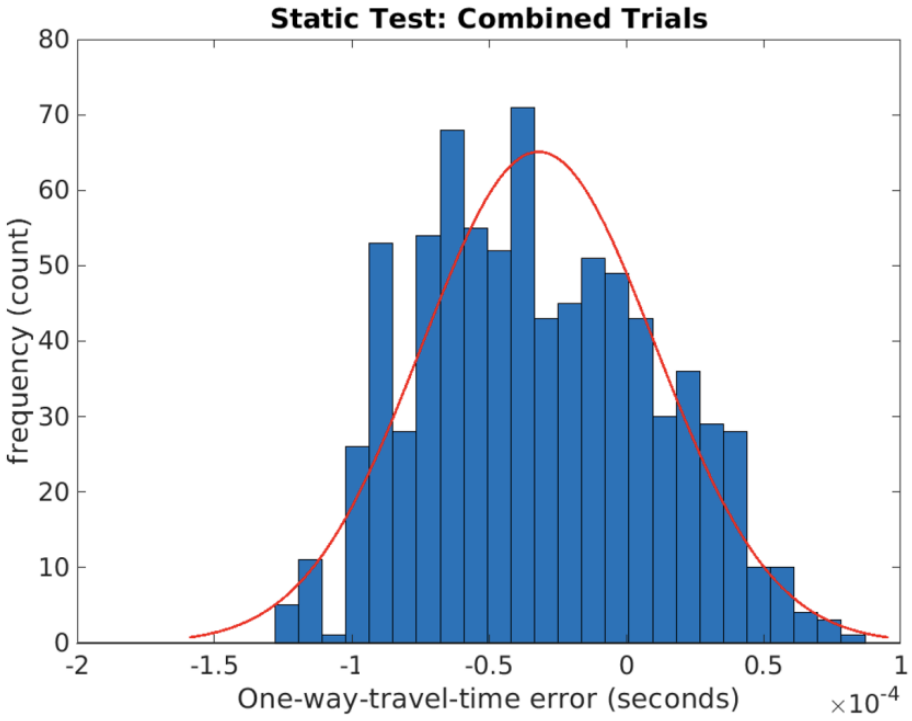


Figure 4-4: Combined static ping test data: μ = -31.89 microseconds; σ = 42.47 microseconds

4.3 100-Meter Paceline Test

As mentioned in Chapter 3, our HAT navigation method relies on a constant speed approximation within the odometry factors. Traditionally, divers estimate their speed through paceline dives, where they swim along a 100-meter line and time how long it takes to get from one end to the other. After multiple trips back and forth, they calculate their average time in seconds per trial and divide 100 by that number to get their average speed in meters per second. This test sought to duplicate that process with kayaks. First, we list

the required equipment for a 100-meter paceline test with a kayak. Then, we discuss the experimental process and provide the results.

4.3.1 Equipment

Table 4.3 shows the equipment for this test.

Paceline Test Equipment			
Quantity	**Equipment**	**Specification**	**Purpose**
1	Kayak	Human operated	Mobility
1	Survey tape	100-meter	Measure distance
1	Buoy	With anchor	Mark 100-meter point
1	Timer	N/A	Time trial

Table 4.3: Equipment for the paceline test on October 24$^{\text{th}}$, 2021.

4.3.2 Description

On October 24$^{\text{th}}$, 2021, we conducted the 100-meter paceline test at Bourne's Pond in Falmouth, Massachusetts. Our full-scale experiments used kayaks as proxies for the REMUS and diver. Therefore, we performed the paceline process to find the kayak's average paddling speed. To set up the test, we attached the end of a survey line to the corner of a dock and paddled to the 100-meter mark, where we placed a buoy. For each trial, the kayak operator determined the time to paddle from the dock's corner to the buoy or vice versa.

4.3.3 Results

Table 4.4 shows the results of the five paceline trials. This 0.92-knot average speed value served as the constant speed parameter for the diver kayak in the experiments and the diver in the simulations. According to the Navy Dive Manual, a comfortable swimming speed for a diver is 0.8 knots [10]. Therefore, our estimated kayak speed is similar to that of a diver.

Paceline Results					
Metric	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**
Time (seconds)	215	207	224	205	210
Speed (meters per second)	0.47	0.48	0.45	0.49	0.48
Speed (knots)	0.91	0.93	0.87	0.95	0.93
Average speed = 0.92 knots					

Table 4.4: Results from the 100-meter paceline test on October 24$^{\text{th}}$, 2021

4.4 Dead Reckoning Experiment

This experiment's goal was to serve as a comparison to the HAT method by showing the navigation accuracy without range data from the REMUS kayak. Along with providing the equipment required to perform this experiment, this section discusses the experimental setup and the algorithms within the `Dead Reckoning Node`, which was created specifically for this test. After setting the initial position with a GPS fix, the `Dead Reckoning Node` uses time, speed, and heading measurements to predict the diver's location throughout the transit. We conclude this section by comparing the kayak's final estimated position to the corresponding ground truth location.

4.4.1 Equipment

Table 4.5 shows the equipment for this experiment. Both the Garmin GPS 18x and the Sparton M2 AHRS were connected to the laptop via Universal Serial Bus (USB) ports.

Diver Kayak Equipment for the Dead Reckoning Experiment			
Quantity	**Equipment**	**Specification**	**Purpose**
1	Kayak	Human operated	Diver proxy
1	Sparton M2 AHRS	Data rate: 1 Hz	Global heading
1	Garmin GPS 18x	Data rate: 1 Hz	Ground truth
1	Laptop (Ubuntu 20.04)	ROS Noetic Ninjemys	Execute software

Table 4.5: Dead reckoning equipment for the diver kayak on November 6$^{\text{th}}$, 2021.

4.4.2 Description

On November 6$^{\text{th}}$, 2021, we conducted the 400-meter DR experiment at Bourne's Pond in Falmouth, Massachusetts, using the coordinate information in Table 4.6. The first trial started at Location 1 and set Location 2 as the target, while the second trial proceeded in the opposite direction.

400-meter Dead Reckoning Experiment Coordinates		
Label	**Latitude (degrees North)**	**Longitude (degrees West)**
Location 1	41.5703500	70.5537100
Location 2	41.5668011	70.5529324

Table 4.6: Dead reckoning experiment coordinates for November 6$^{\text{th}}$, 2021.

Our custom-made `Dead Reckoning Node` enabled these trials. The kayak's operator

interacted with the node by posting ROS messages via the Linux laptop's command line. One ROS message started the DR process, while the second stopped the DR process. Table 4.7 shows the steps to complete a 400-meter DR trial.

Dead Reckoning Experiment Process	
Step	**Event**
1	Paddle kayak to start location
2	Launch `Dead Reckoning Node`
3	Post ROS message to start the navigation assistance
4	Paddle to the target based on guidance from the `Dead Reckoning Node`
5	Post ROS message to stop the navigation assistance

Table 4.7: Steps to complete a dead reckoning trial.

Once the operator starts the DR process, the `Dead Reckoning Node` sets the kayak's initial location using a GPS fix. Then, the node provides guidance using Algorithm 15, which repeats every second. Therefore, every second, the distance to the target, bearing to the target, and current Sparton M2 heading are published within a ROS message and posted to the terminal window for the kayak operator to see. To paddle to the target, the operator matches their current heading with the target bearing until the distance to the target equals zero. Theoretically, this should bring the kayak to the desired location. However, inconsistent paddling and unknown ocean currents push the kayak off course from where the `Dead Reckoning Node` believes they are.

Algorithm 15 One step of the `Dead Reckoning Node`

Require: Target coordinates (C_T), diver kayak's estimated speed (υ) {0.92 knots}
In: Global heading from Sparton M2 AHRS (θ_i), estimated diver kayak location (C_i)
1: Change in time (δT) = 1 second
2: Calculate change in distance (δD) = $\upsilon \times \delta T$
3: Estimate new diver kayak location (C_{i+1}) using C_i, δD, θ_i
4: Find distance to target (d_T) using C_{i+1} and C_T
5: Find bearing to target (θ_T) using C_{i+1} and C_T
6: $C_i = C_{i+1}$
Publish: d_T, θ_T, θ_i

Algorithm 15 uses geodesy in steps three through five. Step three utilizes the `destination` function within the `GeoPy` Python client [92] to calculate new coordinates given a bearing and distance from an original location. Specifically, `GeoPy` performs this function using geodesic algorithms discussed by Karney [93] with the World Geodetic System 1984 (WGS 84) ellipsoid model [94]. Step four relies on the `Vincenty` Python library [95] to calculate

the distance between two coordinates accurately. This method also uses WGS 84 as the reference ellipsoid to provide distances within one millimeter of the ground truth value, which is more accurate than the great-circle or Euclidean approach [95]. Lastly, step five uses the following equations to calculate the bearing to target (θ_T):

$$X = \cos(lat_A)\sin(lat_B) - \sin(lat_A)\cos(lat_B)\cos(lon_B - lon_A), \tag{4.1}$$

$$Y = \sin(lon_B - lon_A)\cos(lat_B), \tag{4.2}$$

$$\theta_T = \text{atan2}(Y, X), \tag{4.3}$$

where (lat_A, lon_A) are the diver kayak's predicted location in radians and (lat_B, lon_B) are the target's coordinates in radians [96]. These equations are derived from the haversine formula.

4.4.3 Results

Figure 4-5 and 4-6 compare the DR path to the ground truth path for both trials. The DR path represents where the `Dead Reckoning Node` believed the kayak was throughout the transit, while the ground truth path depicts the coordinates reported by the Garmin GPS 18x.

Table 4.8 shows the results from both trials. The accuracy values were determined using the `vdist` MATLAB function [97], which calculates the distance between two coordinates using Vincenty's formula. We define the DR to target endpoint error as the difference between the `Dead Reckoning Node`'s final location prediction and the target coordinates. Similarly, the ground truth to target endpoint error represents the kayak's actual distance from the target at the end of the trial. However, the most important value is the difference between the final DR prediction and the last ground truth coordinate because it reflects the performance of the navigation method. A perfect endpoint solution would have the final estimate equal the corresponding ground truth location. Yet, although the `Dead Reckoning Node` reported the kayak was at the target location, it was actually much further away.

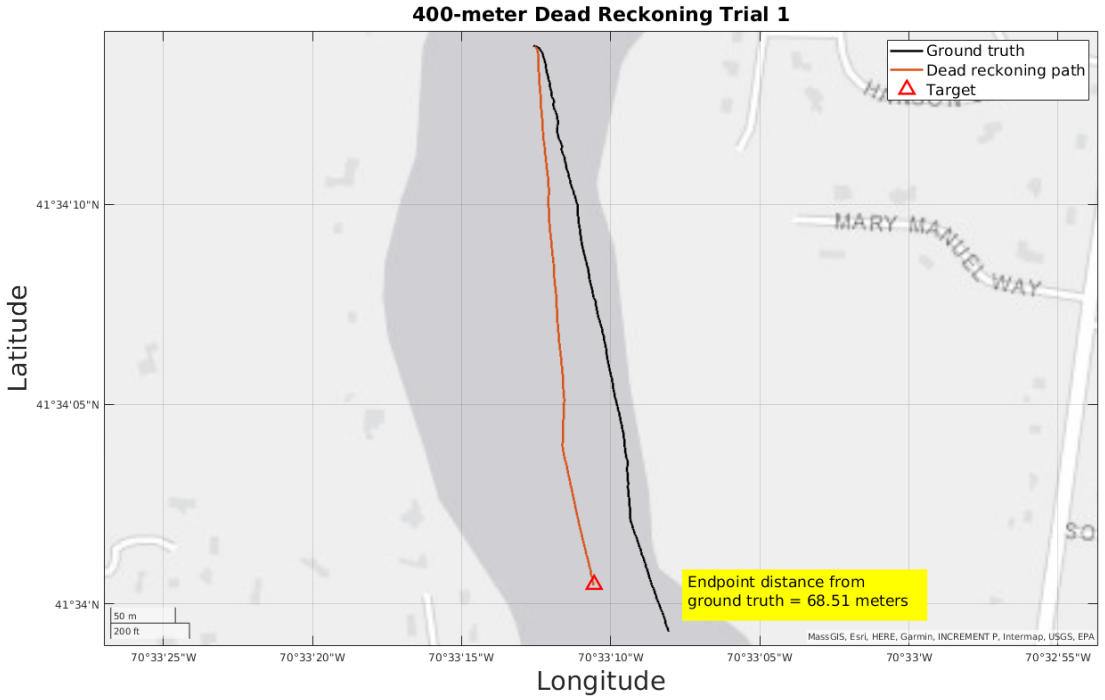


Figure 4-5: 400-meter dead reckoning trial 1 results from November 6$^{\text{th}}$, 2021.

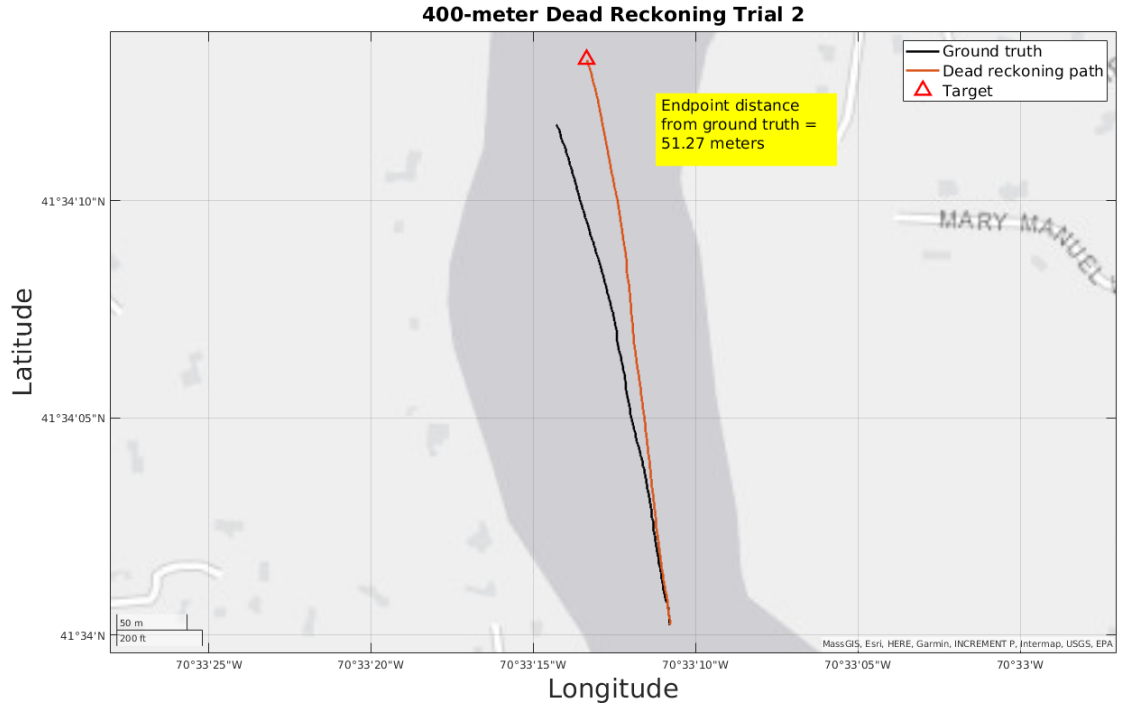


Figure 4-6: 400-meter dead reckoning trial 2 results from November 6$^{\text{th}}$, 2021.

Endpoint Error (in meters)			
Label	**DR to target**	**Ground truth to target**	**DR to ground truth**
Trial 1	0.92	68.28	68.51
Trial 2	0.66	51.35	51.27

Table 4.8: Dead reckoning experiment results (in meters) from November 6$^{\text{th}}$, 2021.

4.5 Cooperative Navigation Experiment

Through these cooperative navigation trials, we sought to implement and prove the validity of the methods described in Chapter 3. First, we list the equipment used onboard the two kayaks that served as proxies for the REMUS and the diver. However, some aspects of our method were no longer possible because we used kayaks. Therefore, we examine these changes and discuss their impact on the experiment. Then, we explain the experimental setup and detail the steps performed by both kayak operators. Lastly, we evaluate the performance of our system during this field experiment by analyzing how well it guided the diver kayak to the target and how effectively it calculated the diver kayak's path throughout the transit.

4.5.1 Equipment

REMUS Kayak Equipment			
Quantity	**Equipment**	**Specification**	**Purpose**
1	Kayak	Human operated	REMUS proxy
1	Deck Box	25 kHz PSK	Micromodem 2
1	Towfish	25 kHz	Transducer
1	Arrow 100	Data rate: 1 Hz	Ground truth
1	Laptop (Ubuntu 20.04)	ROS Noetic Ninjemys	Execute software

Table 4.9: Cooperative navigation equipment for the REMUS kayak on November 6$^{\text{th}}$, 2021.

This experiment relied on the diver kayak and the REMUS kayak. Table 4.9 lists the equipment for the REMUS kayak. Furthermore, Figure 4-7 provides a visual of the REMUS kayak's setup and shows how the electronics were connected. The transducer's cable was secured to the front of the kayak such that the towfish hung a meter below the waterline. Additionally, the GPS or GNSS receiver was fastened directly above the towfish. Therefore, the ground truth coordinates reflected the transducer's location. It is important to note that the image in Figure 4-7 was taken before the November 6$^{\text{th}}$ experiment. As such, it

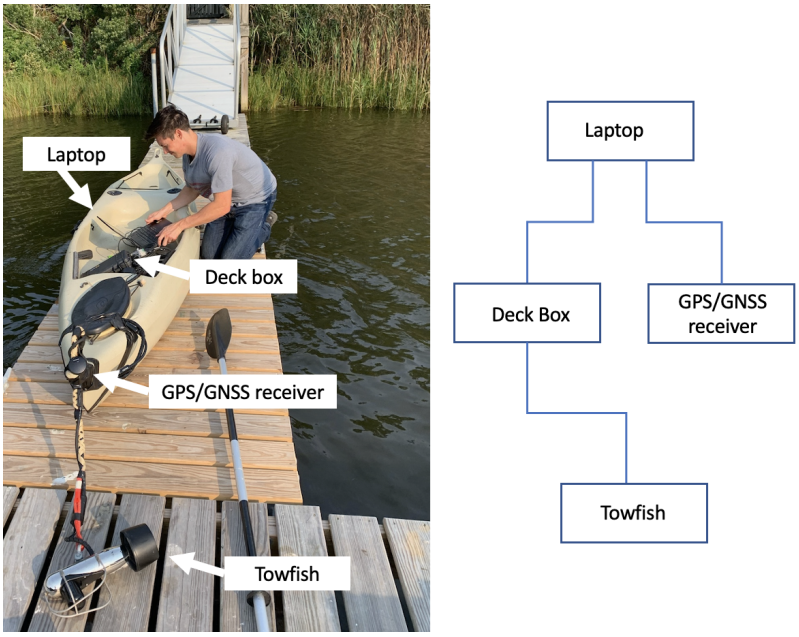


Figure 4-7: **Left:** REMUS kayak set-up for the cooperative navigation experiment. Although this figure shows a Garmin GPS 18x receiver, the November 6th experiment replaced that with an Arrow 100 GNSS receiver. **Right:** Diagram displays how the hardware was connected onboard the kayak.

shows the Garmin GPS 18x receiver, which was replaced by the Arrow 100 GNSS receiver for the November 6th experiment.

Diver Kayak Equipment			
Quantity	**Equipment**	**Specification**	**Purpose**
1	Kayak	Human operated	Diver proxy
1	Deck Box	25 kHz PSK	Micromodem 2
1	Towfish	25 kHz	Transducer
1	Sparton M2 AHRS	Data rate: 1 Hz	Global heading
1	Garmin GPS 18x	Data rate: 1 Hz	Ground truth
1	Laptop (Ubuntu 20.04)	ROS Noetic Ninjemys	Execute software

Table 4.10: Cooperative navigation equipment for the diver kayak on November 6th, 2021.

Table 4.10 lists the equipment for the diver kayak. Additionally, Figure 4-8 displays the diver kayak on Bourne's Pond and shows how the electronics were connected. Like the REMUS kayak's image in Figure 4-7, this picture was taken at an earlier trial and did not perfectly reflect the setup used on November 6th. The towfish, laptop, GPS receiver, and deck box remained in the exact location, with the towfish secured to the back of the kayak and the GPS receiver fastened to the kayak above the towfish. However, while Figure 4-8 shows the Sparton M2 next to the laptop, we moved it towards the front of the kayak on

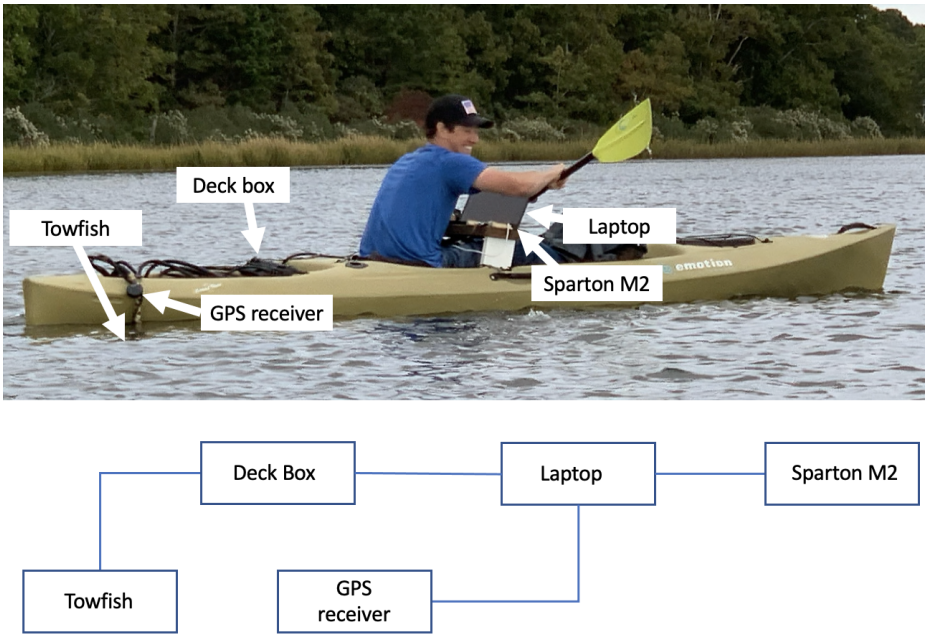


Figure 4-8: **Top:** Diver kayak set-up for the cooperative navigation experiment. Although this figure shows the Sparton M2 next to the laptop, it was moved to the front of the kayak for the November 6$^{\text{th}}$ experiment. **Bottom:** Diagram displays how the hardware was connected onboard the kayak.

November 6$^{\text{th}}$. This change was because we believe electromagnetic interference from the computer caused the Sparton M2 to output incorrect heading data during an earlier trial. After performing a better calibration and increasing its distance from the laptop, it acted appropriately.

4.5.2 Adjustments to Facilitate Experimentation With Kayaks

Adjustments to the `Director Node` and `Navigator Node` were made to facilitate experimentation with the kayaks. However, the communication process discussed in Section 3.2 and the state estimation process detailed in Section 3.4 were unchanged. Table 4.11 summarizes the reasons for the modifications as well as their impact.

One change is that we did not include depth sensors in this experiment. However, since both towfish remained at constant depths of one meter, the three-dimensional range calculation in Equation 3.4 was equivalent to the two-dimensional range. Therefore, the `Director Node` calculates the two-dimensional range ($r_{2D,i}$) using a modified version of the equation discussed by Fallon et al. [48]:

Impacts of Using Kayaks as Proxies		
Kayak	**Change**	**Consequence**
REMUS	No CTD	Constant sound speed and towfish depth
REMUS	No INS	Arrow 100 supplies coordinates for Leader Packet
REMUS	No autonomy	Operator visually navigates kayak
Diver	No depth sensor	Constant towfish depth
Diver	No dive tablet	Interaction with operator occurs via laptop terminal

Table 4.11: Impacts of using kayaks as proxies for the diver and REMUS.

$$r_{2D,i} = OWTT_i * c_i, \tag{4.4}$$

where $OWTT_i$ is the OWTT and c_i is the sound speed from the most recently received Leader Packet. Additionally, unlike the REMUS, which provides updated sound speed values throughout the transit, our sound speed value was constant and determined using a single measurement taken with the Sea-Bird SBE 27-SMP MicroCAT immediately before the start of the experiment. Table 4.12 shows this measurement.

Sea-Bird SBE 27-SMP MicroCAT measurement	
Salinity	27.45 parts per thousand
Temperature	14.12 degrees Celsius
Pressure	0.80 decibars
Sound speed	1495.03 meters per second

Table 4.12: Salinity, temperature, pressure, and sound speed readings from the Sea-Bird SBE 27-SMP MicroCAT on November 6$^{\text{th}}$, 2021.

Another impact of replacing the REMUS with a kayak is that the `Navigator Node` no longer needed to create and update transit actions to implement an autonomous behavior. Instead, the REMUS kayak's operator visually adjusted their course throughout the transit to mimic the autonomous behavior as best they could. However, it was much more complicated than anticipated to perform perfect 25-meter-radius circles. As such, the OWTT-derived ranges were between 6.29 meters and 36.63 meters, indicating the kayaks were closer than our 15-meter minimum distance goal.

Lastly, since the REMUS kayak does not have an INS to predict its location, we populated the Leader Packet coordinates using the Arrow 100, which had an average R95 accuracy of 0.54 and 0.69 meters for the first and second trials, respectively. Therefore, the coordinates within the Leader Packet were more accurate than an INS would have provided.

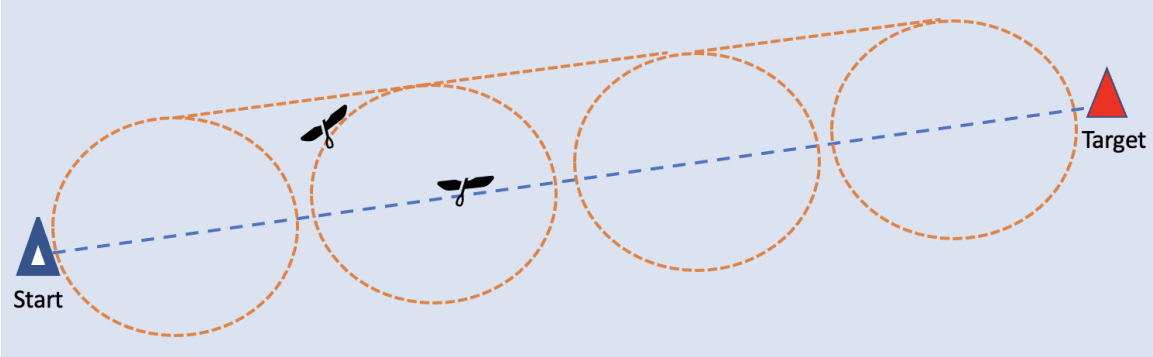


Figure 4-9: Cooperative navigation experiment scheme of maneuver.

4.5.3 Description

On November 6th, 2021, we conducted the 400-meter cooperative navigation experiment at Bourne's Pond in Falmouth, Massachusetts. Table 4.13 provides the coordinates for the two trials. The first started at Location 1 and set Location 2 as the target, while the second proceeded in the opposite direction. These coordinates are the same as the dead reckoning experiment. Additionally, both experiments were conducted within a two-hour window to replicate the environmental conditions as much as possible.

400-meter Cooperative Navigation Experiment Coordinates		
Label	**Latitude (degrees North)**	**Longitude (degrees West)**
Location 1	41.5703500	70.5537100
Location 2	41.5668011	70.5529324

Table 4.13: Cooperative navigation experiment coordinates for November 6th, 2021.

Figure 4-9 shows this experiment's intended setup, and Figure 4-10 provides a picture of our actual environment at Bourne's Pond. Although Bourne's Pond has a depth between one and three meters, it offered a favorable experiment location due to a lack of boat traffic, which could have impeded our transit or hindered our acoustic communication. Furthermore, as the WHOI Micromodem 1 had performed well in 1999 communicating with another modem kilometers away in water between three and eight meters deep [98], we were confident the Micromodem 2 would perform well at Bourne's Pond.

The diver kayak's operator interacts with the `ros_hat` nodes through ROS messages posted via the command line. The operator posted messages to four topics during these trials. When the `Director Node` receives a message on the `HNI_trigger` topic, it triggers the Initialization Packet to be transmitted. Posting to the `swim` topic sets the kayak's estimated

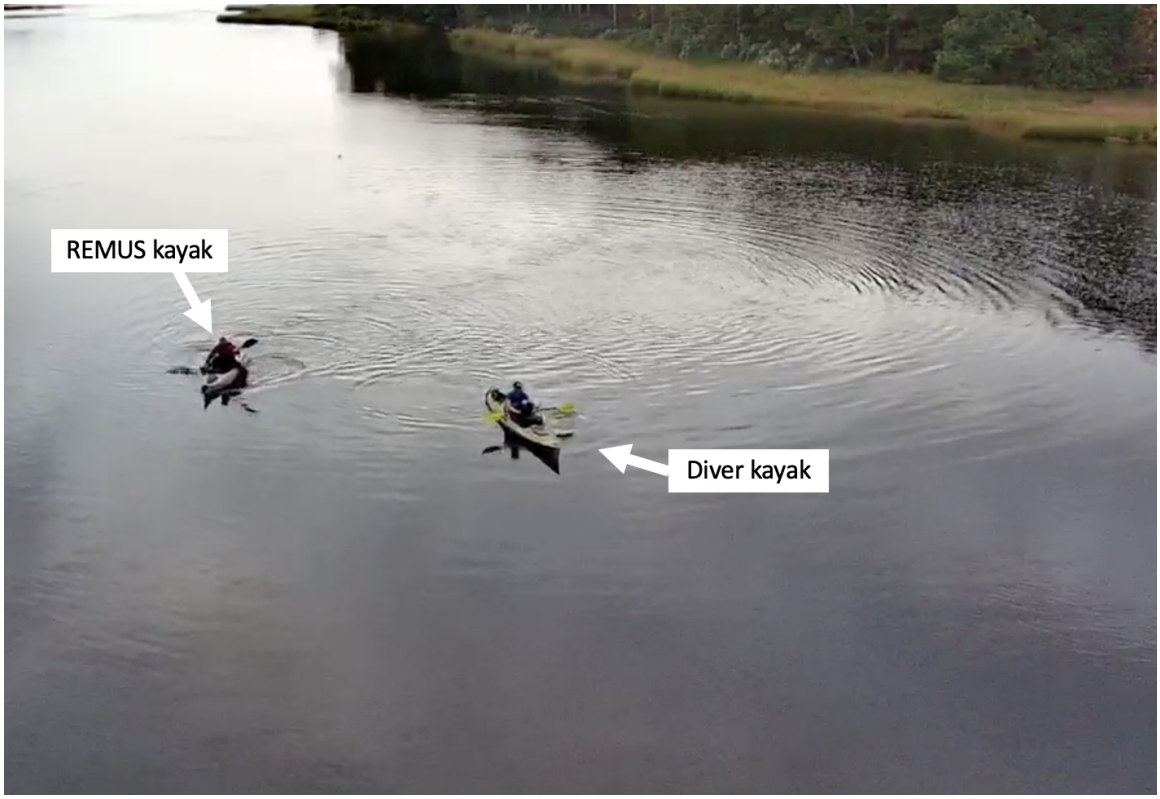


Figure 4-10: Image showing our cooperative navigation environment at Bourne's Pond. The diver kayak travels toward the target using directions posted to the laptop terminal, while the REMUS kayak moves to provide diverse ranging locations.

speed to 0.92 knots, which is the value determined with the paceline tests. Additionally, a message on the `stop_swim` topic resets the kayak's estimated speed to zero, and a message on the `HNT_trigger` prints iSAM2's smoothed path to a comma-separated values (CSV) file. Table 4.14 summarizes the steps performed by the diver kayak's and REMUS kayak's operators to complete a cooperative navigation trial.

Diver Kayak Process	
Step	**Event**
1	Paddle kayak to start location
2	Launch software
3	Post `HNI_trigger` message to send Initialization Packet
4	Once first Leader Packet received, post `swim` message to update kayak speed
5	Paddle to the target according to guidance provided by the `Director Node`
6	Once at target location, post `stop_swim` message to set kayak speed to zero
7	Post `HNT_trigger` message to have the `Estimator Node` save the smoothed path coordinates to a CSV file
REMUS Kayak Process	
Step	**Event**
1	Paddle kayak to location about 25-meters from diver kayak's start point
2	Launch software
3	Once first Follower Packet received, begin transit
4	Paddle around the diver kayak in 25-meter-radius circles

Table 4.14: Diver kayak and REMUS kayak steps to conduct a cooperative navigation trial.

As described in Algorithm 9, the `Director Node` uses the diver kayak's estimated position to calculate the distance to the target and the bearing to the target. Like the `Dead Reckoning Node`, the `Director Node` uses the `Vincenty` Python library [95] to calculate the distance between the estimated kayak location and the target coordinate. Additionally, the `Director Node` uses Equations 4.1 through 4.3 to calculate the bearing to the target. These values get posted to the terminal window along with the current Sparton M2 heading. Therefore, to travel to the target, the diver kayak's operator matches their current heading with the target bearing until the distance to the target equals zero.

4.5.4 Results

Figures 4-11 and 4-12 display the results of these two trials. Within each figure, there are two images. The left image compares the real-time estimates calculated via iSAM2 with the ground truth path according to the Garmin GPS 18x. Every time a new odometry or range factor is added to the graph, the `Estimator Node` predicts the kayak's current location

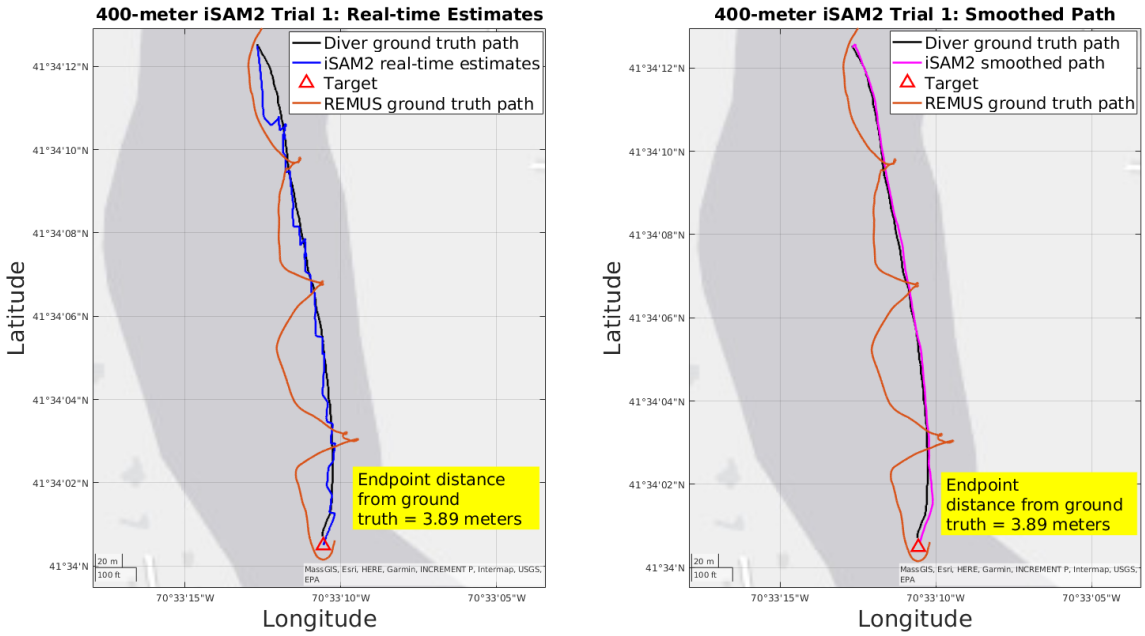


Figure 4-11: Cooperative navigation trial 1 results from November 6$^{\text{th}}$, 2021. **Left:** Real-time kayak location estimates. **Right:** Kayak's final smoothed path.

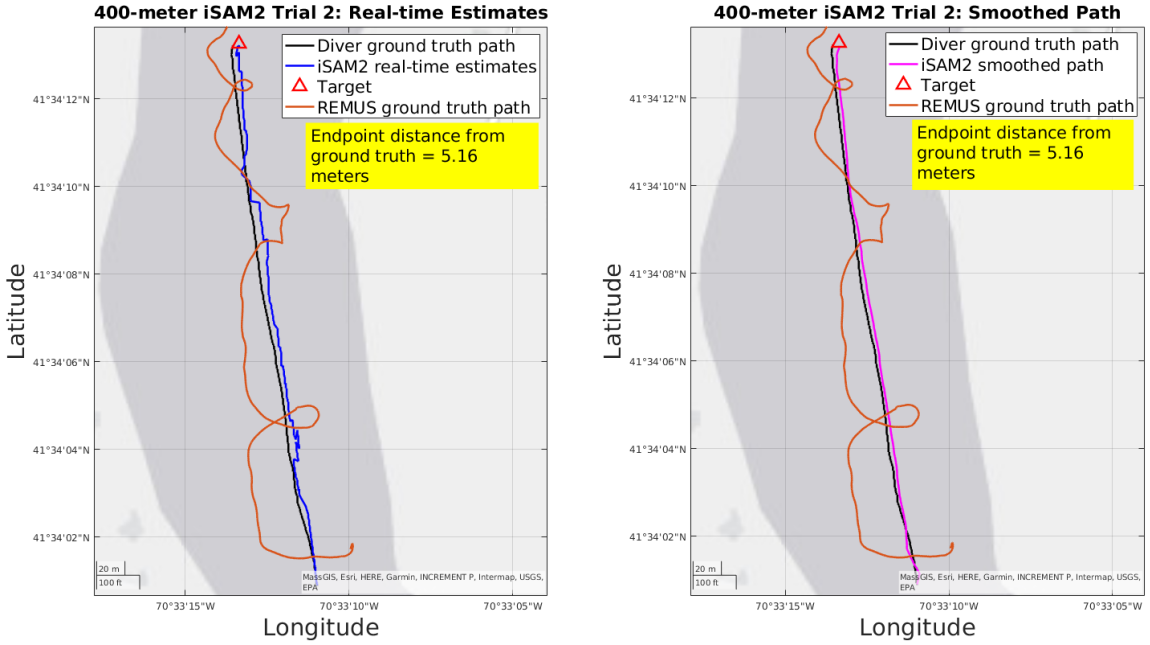


Figure 4-12: Cooperative navigation trial 2 results from November 6$^{\text{th}}$, 2021. **Left:** Real-time kayak location estimates. **Right:** Kayak's final smoothed path.

according to Algorithm 8. Therefore, the real-time estimates show where iSAM2 thought the kayak was during the trial. Conversely, the image on the right shows the final smoothed path of the diver kayak.

As mentioned in Section 2.4, NLS state estimation algorithms attempt to find the path that best represents all the measurements it receives. Figure 4-13 helps visualize how iSAM2 uses the two-dimensional range factors from the entire transit to calculate the smoothed path. Although the final coordinates of the real-time estimates and the smoothed path are the same, Figures 4-11 and 4-12 show the trajectory of the smoothed solution matches the ground truth path much better.

Table 4.15 quantifies the navigation accuracy of this experiment. We define the iSAM2 to target endpoint error as the difference between the `Estimator Node`'s final location prediction and the target coordinates. Similarly, the ground truth to target endpoint error represents the diver kayak's actual distance from the target. However, as previously mentioned, the critical value in determining the efficacy of the navigation approach is the distance between the final iSAM2 prediction and the last ground truth coordinate.

Endpoint Error (in meters)			
Label	**iSAM2 to target**	**Ground truth to target**	**iSAM2 to ground truth**
Trial 1	3.62	6.81	3.89
Trial 2	2.21	4.63	5.16

Table 4.15: Cooperative navigation experiment endpoint results (in meters) from November 6$^{\text{th}}$, 2021.

Path Accuracy (in meters)				
	Real-time		Smoothed Path	
Label	**Max distance**	**Avg distance**	**Max distance**	**Avg distance**
Trial 1	22.89	6.17	4.50	1.42
Trial 2	12.70	6.40	4.99	2.42

Table 4.16: Cooperative navigation experiment path results from November 6$^{\text{th}}$, 2021.

The results in Table 4.16 show how capable our navigation method is at determining the kayak's location throughout the trial. We assess this performance using the maximum and average distances between the real-time estimates and their corresponding ground truth locations. For example, the maximum difference between iSAM2's real-time estimate and the kayak's actual location for trial one was 22.89 meters, which Figure 4-11 shows as occurring near the beginning of the transit.

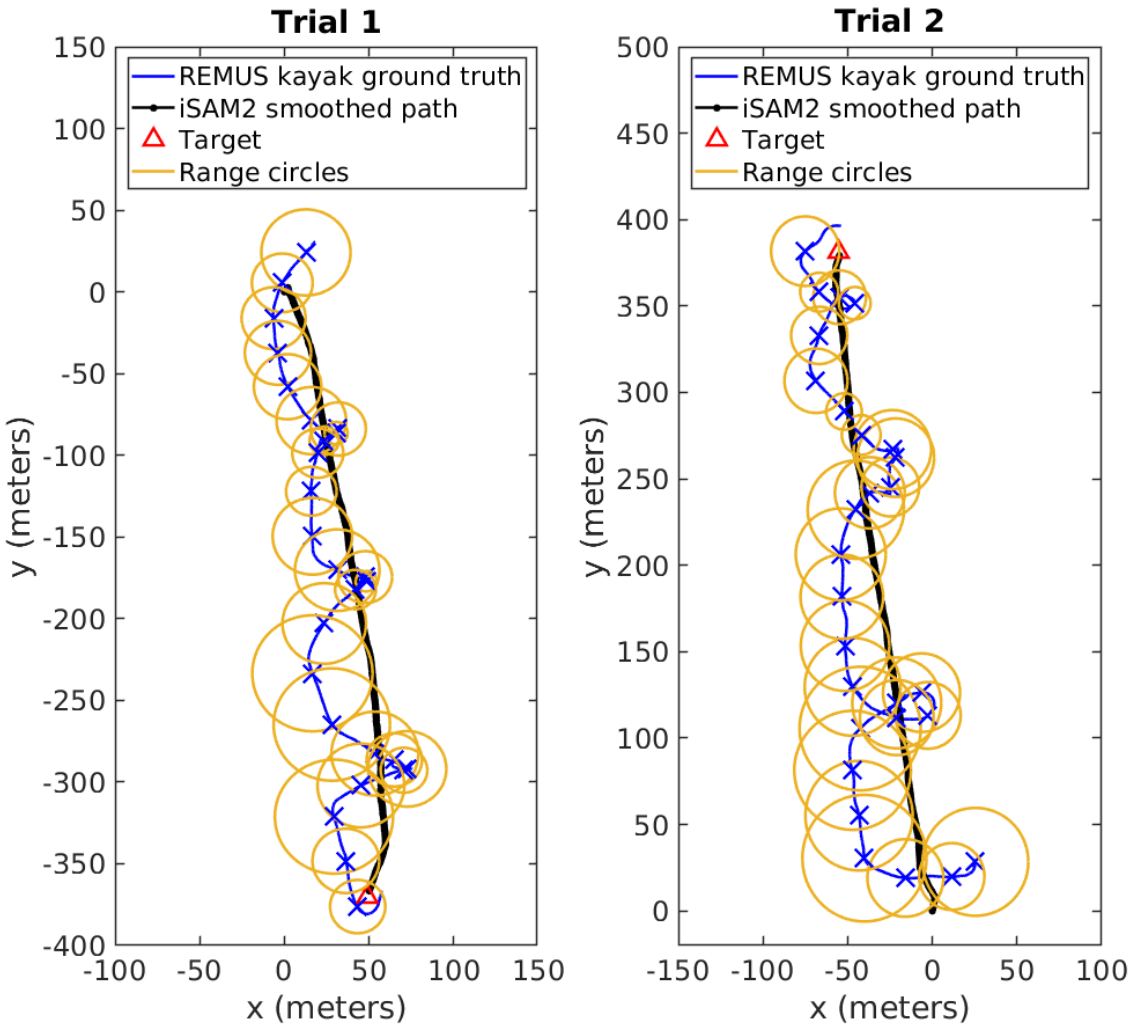


Figure 4-13: Local iSAM2 solution with range rings for both trials from November 6$^{\text{th}}$, 2021. Each blue X marks the REMUS kayak's location upon completing a ping sequence. These coordinates served as the landmark locations for the two-dimensional range factors within the factor graph on the `Estimator Node`. **Left:** Cooperative navigation trial 1 . **Right:** Cooperative navigation trial 2.

Table 4.16 also displays values representing the smoothed path's accuracy. At the end of each trial, we use MATLAB to create a vector of values representing the minimum distance between each point of the smoothed path and the ground truth track. These values determine how accurately the smoothed solution represents the ground truth trajectory. For example, if both paths lined up perfectly at a location, the distance between them would be zero. Then, we quantify the accuracy of the smoothed solution by finding the maximum and average values within that vector.

4.6 Micromodem 2 Ping, Packet, and Range Analysis

This section analyzes the acoustic performance of the Micromodem 2. First, we discuss the cooperative navigation experiments' ping and packet success rates. Then, we evaluate the same metrics across all field trials performed during this research and comment on our decision to change from Rate 5 to Rate 1 pings. Lastly, we compare the ground truth range measurements to the OWTT-derived range measurements to assess the modem's ranging performance. This analysis provided critical parameters for the simulations discussed in the following sections.

The November 6$^{\text{th}}$ cooperative navigation experiment relied on the Micromodem 2's Rate 1 communication with a 25-kilohertz center frequency and a five-kilohertz bandwidth. As mentioned in Section 3.2, this Rate 1 FDP communication uses QPSK modulation and Bose–Chaudhuri–Hocquenghem error-correcting code [57]. Table 4.17 shows the communication success rates for this experiment.

Communication Performance on November 6$^{\text{th}}$			
Category	**Transmitted**	**Successful**	**Percent success**
Ping	57	57	100.00
Packet	116	116	100.00

Table 4.17: Ping and packet success rates for both cooperative navigation trials on November 6$^{\text{th}}$, 2021.

Throughout all our experiments, we relied on Rate 1 packets. However, early testing used Rate 5 pings. Rate 5 utilizes QPSK modulation but performs error correction with Hamming code to enable faster data transmission. For example, the burst rate for an FDP mini-packet is 508 bits per second with Rate 1 and 2,797 bits per second with Rate 5 [57]. However, Rate 5 is less robust to acoustic clipping, which Mei et al. [99] defines as a type

of waveform distortion that occurs when an amplifier tries to increase the electrical signal's amplitude past its maximum capability.

Table 4.18 and Table 4.19 quantify the Micromodem 2's communication performance throughout this research. Most missed pings came during an early October cooperative navigation trial in Woods Hole, Massachusetts. Chapter 5 discusses that trial in more detail. After experiencing those issues, we switched from Rate 5 pings to Rate 1 pings, which had an increased success rate albeit with a smaller sample size.

Ping Success			
Type	**Transmitted**	**Successful**	**Percent success**
Rate 1	57	57	100.00
Rate 5	919	1002	91.72
Combined	976	1059	92.16

Table 4.18: Ping success rate throughout all field tests and experiments.

Packet Success			
Type	**Transmitted**	**Successful**	**Percent success**
Rate 1	713	717	99.44

Table 4.19: Packet success rate throughout all field tests and experiments.

Given the 42.47-microsecond ping-derived OWTT standard deviation from our September 22^{nd} test and 1495.03 meters per second sound speed, the range factor's standard deviation for the November 6^{th} cooperative navigation trials was set equal to 6.35 centimeters. However, as Fallon et al. [48] states, a modem's ranging accuracy is expected to decrease when using TWTT due to the relative motion of the vehicles as the transmitted ping travels from one vehicle to the other and back.

Figure 4-14 shows the range errors for every range factor during the cooperative navigation experiment. Each time the REMUS kayak transmitted a ping, the `Navigator Node` saved the coordinates posted by the Arrow 100. Similarly, every time the diver kayak received a ping, the `Director Node` saved the coordinates provided by the Garmin GPS 18x. Within that figure, we define the GPS-derived range as the distance between those two coordinates and the OWTT-derived range as the distance calculated using Equation 4.4.

Figure 4-15 displays a histogram of the data from Figure 4-14. The x-axis is defined as the ground truth range minus the modem range. The ground truth range is calculated using coordinates provided by the GPS and GNSS receivers, and the modem range is calcu-

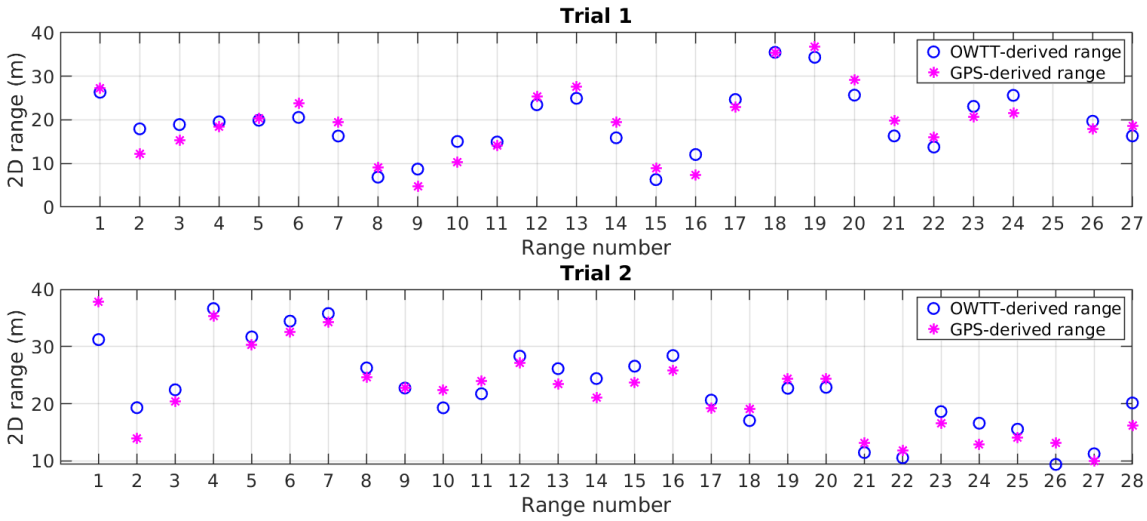


Figure 4-14: Range error data for both cooperative navigation trials on November 6$^{\text{th}}$, 2021. Within each figure, the range measurements are in the order they occurred during the trial. For example, range number 1 is the first range measurement during the trial.

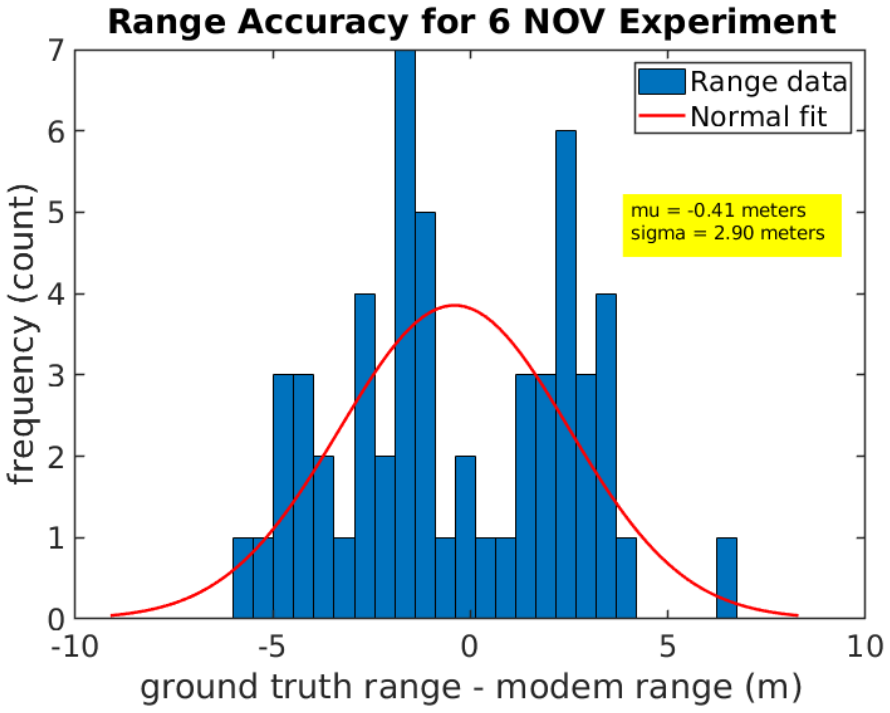


Figure 4-15: Range error histogram for both cooperative navigation trials on November 6$^{\text{th}}$, 2021. Range bias (μ) = -0.41 meters, and range standard deviation (σ) = 2.90 meters.

lated using the OWTT. During these trials, the Arrow 100 supplied coordinates with R95 accuracies between 0.49 and 0.73 meters, while the coordinates from the Garmin GPS 18x had R95 accuracies between 3.30 and 3.90 meters. Due to the uncertainty of the locations involved in calculating the ground truth range, it is difficult to make confident conclusions about the range bias (μ) and range sigma (σ). However, given our cooperative navigation experiment data, we calculate the Micromodem 2's dynamic ranging accuracy as having a negative bias of 0.41 meters and a standard deviation of 2.90 meters.

4.7 Simulating the Diver's Ground Truth Path

Although the kayak provided the ground truth path in the field experiments, simulations require software to fabricate the diver's actual coordinates during the transit. Critical parameters for the simulations were the ocean current speed, ocean current heading, and diver's estimated kick speed, which we define as the speed they would travel unhampered by ocean currents. Although the ocean current was assumed constant throughout our simulations, we added noise to the diver's estimated kick speed and heading to replicate real-world variability. This section discusses how we added noise to those values and details the algorithm we used to simulate the diver's ground truth trajectory. Properly simulating the diver's ground truth path was critical to the overall HAT navigation simulation.

Usually, divers use 100-meter paceline tests to estimate their kick speed. However, their average pace during the dive is unlikely to equal that value. To replicate that bias, we add zero-mean Gaussian noise to the original speed estimate, which we set equal to 0.92 knots to mimic the DR and cooperative navigation experiments. Additionally, we assume the standard deviation of the added noise is 0.05 knots. Therefore, at the start of each simulation, the new mean kick speed (υ_N) is found by

$$\upsilon_N = 0.92 + \mathcal{N}(0, 0.05^2). \tag{4.5}$$

There is also variability in a diver's pace during the dive itself. Consequently, the diver's kick speed (υ_i) at each time interval is found by adding zero-mean Gaussian noise with a 0.05-knot standard deviation to the noisy mean kick speed:

$$\upsilon_i = \upsilon_N + \mathcal{N}(0, 0.05^2). \tag{4.6}$$

Noise is also added to the diver's global heading because they are unlikely to hold the submersible tablet perfectly straight throughout the transit, thereby introducing a temporary heading bias. Additionally, momentary or persistent uneven kicking can add inaccuracy even if the diver is holding the tablet properly. Given these error sources, we assume the diver maintains a course within five degrees of the desired heading before incorporating the ocean current's impact. For example, if the diver holds 207 degrees on their submersible tablet, we assume the ground truth bearing is between 202 and 212 degrees. With these assumptions, we calculate the actual heading for each time step by

$$\Theta_i = \theta_i + \mathcal{U}(-5, 5), \tag{4.7}$$

where θ_i is the diver's global heading according to the simulated dive tablet and $\mathcal{U}(-5, 5)$ represents a uniform distribution between negative five and five degrees.

Algorithm 16 shows the steps needed to generate the diver's ground truth path given these assumptions and noise models. During step three, the diver's speed over ground (SOG) and course over ground (COG) are calculated by adding one vector representing the noisy kick speed and heading to a second vector signifying the constant ocean current.

Algorithm 16 Find the Diver's Next Ground Truth Location

Require: Ocean current speed (v^O) and heading (θ^O), diver's new mean kick speed (v_N)
In: Diver's global heading in degrees (θ_i) and current ground truth location (C_i^{GT})
1: Current kick speed (v_i) = $v_N + \mathcal{N}(0, 0.05^2)$ {σ=0.05 knots}
2: Current global heading (Θ_i) = $\theta_i + \mathcal{U}(-5, 5)$ {added noise in degrees}
3: Calculate diver's speed over ground (SOG) and course over ground (COG) using v_i, Θ_i, v^O, and θ^O
4: Calculate diver's new ground truth location (C_{i+1}^{GT}) using SOG, COG, and C_i^{GT}
return C_{i+1}^{GT}

4.8 Dead Reckoning Simulation

The purpose of the DR simulations is to serve as a comparison to the HAT cooperative navigation simulations discussed in the following section. Using MATLAB, we analyzed the performance of DR navigation in 13 different environments using 26 separate simulations. Our software performs similarly to the `Dead Reckoning Node` used in the field. Specifically, given a starting coordinate, the MATLAB script uses time, speed, and heading measure-

ments to predict the diver's location throughout the transit. After describing the algorithm used for these simulations, we conclude this section by evaluating the DR method's performance.

4.8.1 Description

A custom MATLAB script simulated the diver's DR and ground truth paths using Algorithm 17. We assume the diver holds the bearing from their estimated position to the target throughout the simulation. Therefore, each time a new DR estimate is made, θ_i in Algorithms 16 and 17 is set equal to the new bearing to the target. Appendix C shows the entire MATLAB script used to perform these simulations.

Algorithm 17 One Step of the Dead Reckoning Simulation

Require: Target coordinates (C_T), ocean current speed (v^O) and heading (θ^O), diver's kick speed estimate (v_E), diver's new mean kick speed (v_N) {calculated using Equation 4.5}
In: Dead reckoning estimate (C_i^{DR}), ground truth location (C_i^{GT})
1: Change in time (δT) = 1 second
2: Calculate bearing to target (θ_i) using C_i^{DR} and C_T
{Assume diver holds bearing θ_i}
3: Calculate diver's dead reckoning estimate (C_{i+1}^{DR}) using δT, v_E, θ_i, and C_i^{DR}
4: $C_i^{DR} = C_{i+1}^{DR}$
5: Calculate ground truth location (C_{i+1}^{GT}) using Algorithm 16
6: $C_i^{GT} = C_{i+1}^{GT}$

Both the dead reckoning estimate and ground truth location start at an initial user-defined coordinate. Then, the algorithm updates the dead reckoning estimate using the kick speed estimate and bearing to the target. However, simulated ocean currents, inconsistent kick speeds, and imprecise headings push the diver off course from the dead reckoning estimate. Therefore, the algorithm updates the diver's ground truth location at each step using the diver's SOG and COG, which reflect the impacts from the ocean current and added noise. The trial ends once the dead reckoning estimate is within two meters of the target.

We conducted 26 MATLAB simulations according to the properties listed in Table 4.20. Regardless of the transit length, the bearing from the diver's starting location to the target was 207 degrees. Therefore, a 207-degree ocean current is in the direction of the diver's transit, a 27-degree ocean current is opposite the direction of the diver's transit, and a 117-degree ocean current provides a right-to-left crosscurrent. The goal of these dead reckoning simulations was to serve as a comparison to the cooperative navigation simulations.

Simulation Parameters			
Trial	**Distance**	**Ocean current speed**	**Ocean current heading**
1, 2	400 meters	0.1 knots	27 degrees
3, 4	400 meters	0.1 knots	117 degrees
5, 6	400 meters	0.1 knots	207 degrees
7, 8	400 meters	0.3 knots	27 degrees
9, 10	400 meters	0.3 knots	117 degrees
11, 12	400 meters	0.3 knots	207 degrees
13, 14	400 meters	0.5 knots	27 degrees
15, 16	400 meters	0.5 knots	117 degrees
17, 18	400 meters	0.5 knots	207 degrees
19, 20	400 meters	0.2 knots	117 degrees
21, 22	600 meters	0.2 knots	117 degrees
23, 24	800 meters	0.2 knots	117 degrees
25, 26	1000 meters	0.2 knots	117 degrees

Table 4.20: Simulation parameters.

4.8.2 Results

Table 4.21 shows the results for the 13 different sets of simulation parameters provided in Table 4.20. The endpoint error metrics within that table are equivalent to those from the November 6$^{\text{th}}$ dead reckoning experiment. Additionally, the values in Table 4.21 represent the average results of the two trials performed for each set of simulation parameters. However, Appendix D provides the data for each individual trial.

Dead Reckoning Simulation Endpoint Results (in meters)			
Trial	**DR to target**	**Ground truth to target**	**DR to ground truth**
1, 2	1.88	40.25	38.37
3, 4	1.88	44.51	44.32
5, 6	1.88	65.69	67.57
7, 8	1.88	125.15	123.26
9, 10	1.88	131.02	131.19
11, 12	1.88	116.01	117.89
13, 14	1.88	218.80	216.92
15, 16	1.88	218.26	218.21
17, 18	1.88	215.71	217.59
19, 20	1.64	87.51	87.62
21, 22	1.88	144.32	143.87
23, 24	1.65	197.65	198.54
25, 26	1.88	221.29	221.57

Table 4.21: Endpoint results (in meters) from the dead reckoning simulations. The displayed values represent the average results from the trials in the left-most column.

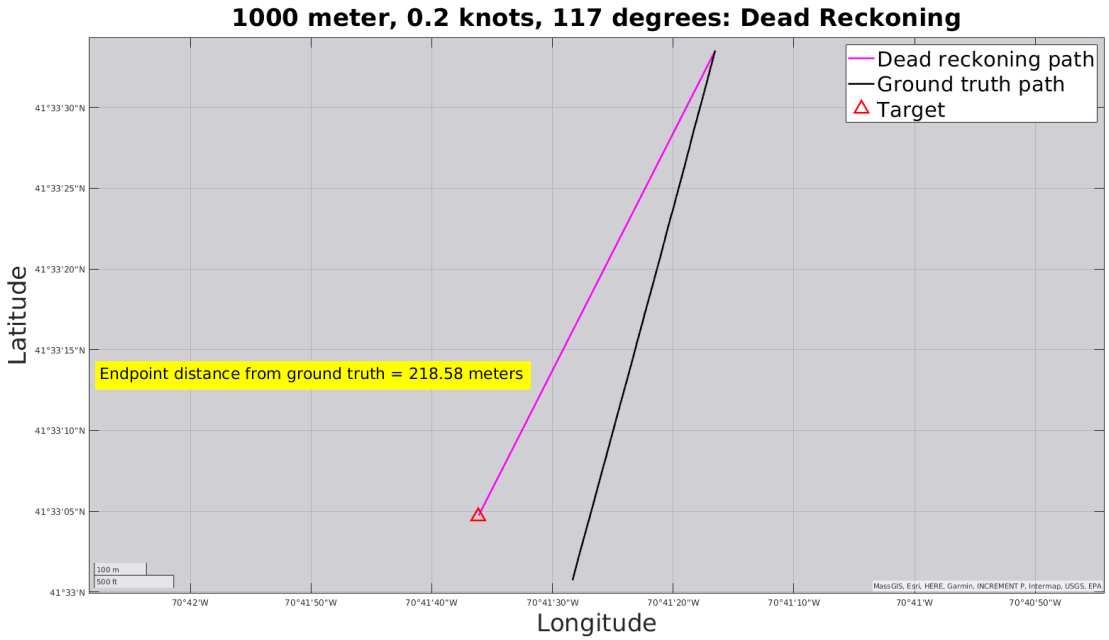


Figure 4-16: One-kilometer dead reckoning simulation. Ocean current speed = 0.2 knots. Ocean current heading = 117 degrees.

Figure 4-16 displays trial 25's results by plotting the DR path and the ground truth path. The ocean current for this simulation was set at 0.2 knots with a 117-degree heading.

4.9 Cooperative Navigation Simulations

Eventually, we intend for our HAT navigation method to be utilized during multi-kilometer transits in all environments safe for diving. Therefore, through these simulations, we sought to test how our system performed in more difficult environments than those experienced in the field. First, we list the equipment used to conduct these simulations and discuss how the operator can control the simulated diver by manipulating the Sparton M2 AHRS. Then, we explain how the REMUS's location uncertainty was simulated and how the Micromodem 2 performance from the cooperative navigation field trials helped model range measurements for the simulations. After describing the process required to conduct a simulation, we evaluate how well our system guided the simulated diver to the target and how effectively it calculated the diver's path throughout the transit. This section concludes by comparing the ping and packet success rates from the simulations to those from the field experiments.

4.9.1 Equipment

The cooperative navigation simulations required a Sparton M2 AHRS, a Software Development Box, a core-board simulator, and a computer. Table 4.22 provides the specification and purpose for each piece of equipment.

Simulation Equipment			
Quantity	**Equipment**	**Specification**	**Purpose**
1	Sparton M2 AHRS	Data rate: 1 Hz	Global heading
1	Software Development Box	Dual Micromodem 2	Communication
1	Core-board simulator	REMUS 100	Simulate AUV
1	Computer (Ubuntu 20.04)	ROS Noetic Ninjemys	Execute software

Table 4.22: Equipment to simulate cooperative navigation trials.

A REMUS 100 core-board simulator imitated the vehicle's front seat computer and sensors. Along with the standard sensor suite, our core-board build had a Phins C3 INS [76] and RECON. The desktop computer serves as the simulated vehicle's back seat computer. Therefore, RECON provides the link between the computer and the core-board, which can post messages and respond to commands the same way a REMUS would during field use.

The REMUS Vehicle Interface Program (VIP) is a graphical user interface that provides mission planning capabilities, posts sensor data, and plots the vehicle's location on a map [100]. We used VIP to program the initial circling objective and visualize the vehicle's path throughout the simulation. Therefore, combining the core-board and VIP software allowed us to plan and execute missions like a real REMUS 100.

A Sparton M2 AHRS provided the diver's global heading data throughout the simulation. Therefore, the person running the simulation could manually adjust the diver's heading by rotating the AHRS. This capability allowed them to respond to `ros_hat`'s navigation guidance as a diver would.

The Linux desktop computer has a Ryzen 9 5950x 16-core, 32-thread processor with 62.7 gibibytes (GiBs) of random access memory (RAM). Although it runs software to simulate both the REMUS and diver, the nodes for each are compartmentalized. Communication between the two groups is accomplished with acoustic messaging via the Dual Micromodem 2 DSP Software Development Box [101], which has two Micromodem 2 stacks that communicate through coaxial cables. These modems post the same NMEA strings as they would in the field, which is crucial for `ros_hat` to function correctly. Through this equipment, we

tried to provide as authentic a simulation as possible.

4.9.2 Simulating Vehicle Location Error and Two-dimensional Ranges

Although we let the coordinates provided by the core-board simulator be the REMUS's ground truth location, we still predict the vehicle's actual location error throughout the dive. We assume the simulated REMUS 100 has a Phins C3 INS and that its DVL has bottom lock throughout the dive. We also assume that the C3 accrues error according to its area survey pattern model, which advertises a CEP accuracy equal to 0.04% of the total distance traveled provided its DVL has bottom lock [76]. However, even with this excellent navigation solution, the REMUS's error is unbounded.

This additive uncertainty builds on the REMUS's original location accuracy, which we assume is provided by the Arrow 100. Consequently, every time the REMUS received a fix within this simulation, the vehicle's location uncertainty reset to 0.73 meters, which was the greatest R95 error the Arrow 100 experienced during the cooperative navigation experiment on November 6^{th}. However, the core-board simulator does not calculate the vehicle's location uncertainty itself. Instead, the `Navigator Node` used Algorithm 18 to find this value.

Algorithm 18 Simulate REMUS 100 uncertainty

Require: Initial R95 uncertainty (U_o)
  Assume: Phins C3 CEP accuracy = 0.04% of distance traveled
  In: REMUS's initial global coordinates (C_o)
  1: Convert Phins C3 CEP accuracy to R95 accuracy
  2: Current REMUS location (C_i) = C_o
  3: Current R95 uncertainty (U_i) = U_o
  repeat
    {The following steps repeat every second}
    4: INS provides next REMUS location (C_{i+1})
    5: Calculate distance (d_i) between C_i and C_{i+1}
    6: Calculate added uncertainty (U^+) using d_i and Phins C3 R95 accuracy
    6: New R95 uncertainty (U_{i+1}) = U_i + U^+
    7: Convert U_{i+1} to RMS uncertainty (U_{RMS}) {RMS format required by GTSAM}
    8: $U_i = U_{i+1}$
    return U_{RMS}
  until Trial ended

The REMUS's location error is included in the Leader Packet and transmitted to the diver. As stated in Chapter 3, this value exclusively serves to calculate a landmark location's covariance within the factor graph. However, the location error in these simulations also

impacts the two-dimensional range calculation because the Software Development Box's published OWTT does not reflect the distance between the simulated REMUS and diver. Therefore, inexact two-dimensional ranges reflecting the modem's ranging accuracy and the REMUS's location error must be simulated instead.

To simulate noisy range measurements, we assume the range error and the REMUS's location uncertainty are linearly related, which allows us to increase the range error throughout the transit based on how much the REMUS's location uncertainty has increased. Each time the REMUS transmits a ping, the ground truth range is calculated using the ground truth diver and REMUS coordinates. Then, the `Director Node` calculates imprecise two-dimensional ranges by adding zero-mean Gaussian noise to this value. We let the standard deviation of the added noise depend on the modem's range accuracy and the REMUS's location uncertainty. Additionally, we assume the modem's range accuracy has a constant standard deviation of 2.90 meters to reflect the results of our cooperative navigation experiment. However, the REMUS's location uncertainty increases throughout the dive. Therefore, each time the `Director Node` receives a Leader Packet, it calculates how much the REMUS's location error has increased since its last GPS fix. Then, the combined standard deviation ($\sigma_{f \otimes g}$) is found by convolving two Gaussian distributions representing the error sources. Bromiley [102] provides the equation to accomplish this:

$$\sigma_{f \otimes g} = \sqrt{\sigma_f^2 + \sigma_g^2}, \tag{4.8}$$

where σ_f and σ_g are the two distribution's standard deviations. In our case, we set σ_f equal to the change in the REMUS's root mean square (RMS) location error and σ_f equal to 2.90 meters. Lastly, the `Director Node` calculates the noisy two-dimensional range (R_N) using

$$R_N = R_{GT} + \mathcal{N}(0, \sigma_{f \otimes g}^2), \tag{4.9}$$

where R_{GT} is the ground truth two-dimensional range and $\sigma_{f \otimes g}$ is the combined standard deviation calculated using Equation 4.8. This range is passed to the `Estimator Node` for inclusion in the range factor.

4.9.3 Description

Using the steps listed in Table 4.23, we completed 13 trials with the same environmental conditions as the dead reckoning simulations, which are provided in Table 4.20. The simulation's operator controlled the diver and interacted with the `ros_hat` nodes through ROS messages posted via the command line. These simulations utilized the same ROS messages as the cooperative navigation experiment. Additionally, the operator manipulated the diver's heading within the simulation using the Sparton M2. Similar to the field experiments, the `Director Node` posts the recommended heading, distance to the target, and current Sparton M2 heading values to the terminal screen. Therefore, the operator aligns the Sparton M2 heading with the recommended heading to steer the simulated diver towards the target.

REMUS Simulation Process	
Step	**Event**
1	Launch REMUS software
Diver Simulation Process	
Step	**Event**
1	Launch diver software
2	Post `HNI_trigger` message to send Initialization Packet
3	Once first Leader Packet received, post `swim` message to update diver speed
4	Manipulate the Sparton M2 according to guidance posted to the terminal
5	Once at the target location, post `stop_swim` message to set diver speed to zero
6	Post `HNT_trigger` message to have the `Estimator Node` save the smoothed path coordinates to a CSV file

Table 4.23: Steps to conduct a cooperative navigation simulation.

It is also important to note that, although the core-board simulator and VIP collaborate to simulate the REMUS's ground truth path, it cannot perform the same function for the diver. Consequently, along with the software discussed in Chapter 3, these simulations require an extra ROS node that fabricates the diver's ground truth location according to Algorithm 16. Therefore, the same noise is added to the diver's speed and heading for the dead reckoning and cooperative navigation simulations. However, the `Estimator Node` is unaware of these changes and adds odometry factors using the Sparton M2 heading and 0.92-knot speed estimate.

4.9.4 Results

Table 4.24 quantifies the navigation accuracy of the 26 simulations, while Table 4.25 compares the real-time estimates to the smoothed path. The metrics within both tables are equivalent to those from the November 6th cooperative navigation experiment results described in Section 4.5.4. Additionally, the values in Tables 4.24 and 4.25 represent the average results of the two trials performed for each set of simulation parameters listed in Table 4.20. However, Appendix D provides the data for each individual trial.

Cooperative Navigation Simulations: Average Endpoint Results (in meters)			
Trial	**iSAM2 to target**	**Ground truth to target**	**iSAM2 to ground truth**
1, 2	1.55	1.67	2.25
3, 4	5.18	9.27	7.13
5, 6	5.97	2.75	3.88
7, 8	3.23	6.84	9.36
9, 10	4.42	9.68	10.85
11, 12	5.64	6.82	5.92
13, 14	8.70	23.66	15.44
15, 16	8.85	20.41	12.04
17, 18	9.84	19.20	9.79
19, 20	1.42	6.88	5.62
21, 22	1.48	5.26	3.82
23, 24	1.58	6.26	5.81
25, 26	1.77	6.36	4.89

Table 4.24: Endpoint results (in meters) from the cooperative navigation simulations. The displayed values represent the average results from the trials in the left-most column.

As a comparison to Figure 4-16 in Section 4.8.2, Figure 4-17 shows the real-time location estimates and final smoothed path for a one-kilometer simulation. Similar to the field experiments, these figures were made with the help of bag files and MATLAB's ROS Toolbox) [91]. Specifically, a MATLAB script plotted the paths and evaluated the endpoint error using the `vdist` function [97] after the simulation concluded.

4.9.5 Micromodem 2 Ping, Packet, and Range Analysis

Due to the perfect communication performance during the November 6th cooperative navigation tests, we did not intentionally drop packets or pings throughout the simulations. However, communication using the Software Development Box was naturally imperfect. Therefore, comparing Table 4.26 to Table 4.18 shows the simulation's ping success rate was higher than the Rate 5 ping rate but lower than the Rate 1 ping rate. Additionally, the

Cooperative Navigation Simulations: Path Results (in meters)				
	Real-time		Smoothed Path	
Trial	**Max distance**	**Avg distance**	**Max distance**	**Avg distance**
1, 2	9.72	3.97	6.27	1.66
3, 4	11.31	5.17	6.89	2.03
5, 6	10.55	4.17	4.86	1.71
7, 8	28.92	9.75	6.43	1.63
9, 10	27.03	11.21	5.15	1.75
11, 12	33.07	13.00	9.70	2.45
13, 14	51.25	19.32	6.80	2.13
15, 16	51.26	19.10	7.18	1.75
17, 18	53.65	20.46	6.90	2.16
19, 20	16.43	6.82	7.23	2.02
21, 22	17.95	8.10	5.16	1.83
23, 24	30.08	8.73	6.66	1.70
25, 26	17.78	7.78	7.08	1.99

Table 4.25: Path estimation results (in meters) from the cooperative navigation simulations. The displayed values represent the average results from the trials in the left-most column.

differences between Table 4.26 and Table 4.19 show the simulation's packet success rate was lower than we experienced in the field.

Software Development Box Communication Performance			
Category	**Transmitted**	**Successful**	**Percent success**
Ping	1083	1049	96.86
Packet	2202	2168	98.46

Table 4.26: Ping and packet success rates for the cooperative navigation simulations.

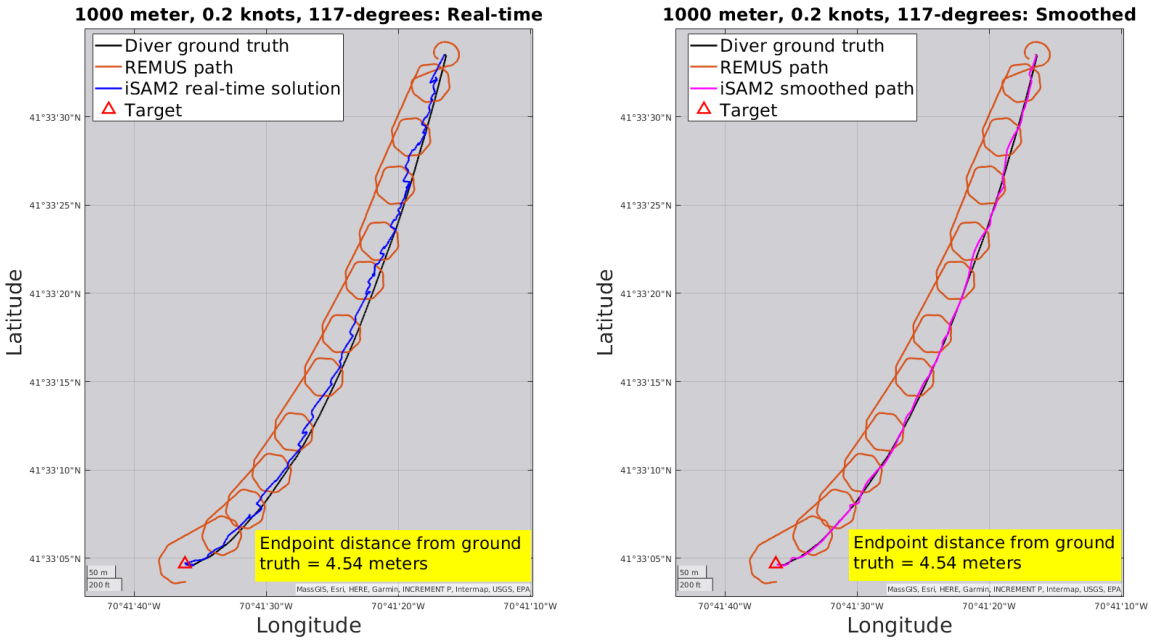


Figure 4-17: One-kilometer cooperative navigation simulation. Ocean current speed = 0.2 knots. Ocean current heading = 117 degrees. **Left:** Real-time diver location estimates. **Right:** Diver's final smoothed path.

4.10 Analysis of Results

This section evaluates the efficacy of our proposed HAT cooperative navigation method by answering the following questions using the results from the field exercises and simulations:

1. How much better is the cooperative navigation method than the dead reckoning approach?
2. How does the ocean current speed impact the cooperative navigation performance?
3. How does the ocean current heading impact the cooperative navigation performance?
4. How does the dive distance impact the cooperative navigation performance?
5. How close should a diver expect to be from their ground truth position while using the cooperative navigation method in real-time?
6. How accurate should the diver expect their smoothed iSAM2 solution to be?

We compared the endpoint errors of the cooperative navigation and DR approaches to quantify the relative performance. The endpoint error equals the distance between the

method's final diver location estimate and the diver's corresponding ground truth coordinate. Additionally, we define the cooperative navigation improvement as the difference between the DR endpoint error and the iSAM2 endpoint error.

Table 4.27 provides the improvement values for both field trials and the 13 different sets of simulation parameters provided in Table 4.20. Our proposed HAT cooperative navigation approach provides superior navigation based on those results, with the minimum and maximum average improvements equaling 36.11 and 216.67 meters, respectively. However, the performance of both methods is impacted by the environment and dive length.

Improvement = Dead reckoning endpoint error - iSAM2 endpoint error (in meters)				
Trial	**Distance**	**Ocean Current**		**Improvement**
Field 1	400 meters	Unknown	Unknown	64.62
Field 2	400 meters	Unknown	Unknown	46.11
1, 2	400 meters	0.1 knots	27 degrees	36.11
3, 4	400 meters	0.1 knots	117 degrees	37.18
5, 6	400 meters	0.1 knots	207 degrees	63.68
7, 8	400 meters	0.3 knots	27 degrees	113.89
9, 10	400 meters	0.3 knots	117 degrees	120.33
11, 12	400 meters	0.3 knots	207 degrees	111.97
13, 14	400 meters	0.5 knots	27 degrees	201.47
15, 16	400 meters	0.5 knots	117 degrees	206.17
17, 18	400 meters	0.5 knots	207 degrees	207.80
19, 20	400 meters	0.2 knots	117 degrees	82.00
21, 22	600 meters	0.2 knots	117 degrees	140.05
23, 24	800 meters	0.2 knots	117 degrees	192.73
25, 26	1000 meters	0.2 knots	117 degrees	216.67

Table 4.27: Endpoint error differences between the dead reckoning and cooperative navigation solutions (in meters). All improvement values are averages except for the field trials.

Figure 4-18 compares the ocean current speed with the iSAM2 endpoint error, which represents the ability of our navigation method to predict the diver's final location. Since we did not conduct simulations with every ocean current speed between zero and 0.5 knots, we did a linear fit of our data to extrapolate the results. The linear fit predicts an endpoint error of 4.37 meters at 0.1 knots and 12.50 meters at 0.5 knots. Therefore, we can infer that increased ocean current speeds decrease the performance of our HAT cooperative navigation method.

Although there is a clear correlation between the ocean current speed and performance, Figure 4-19 shows that the effect of ocean current heading is less severe. The x-axis represents

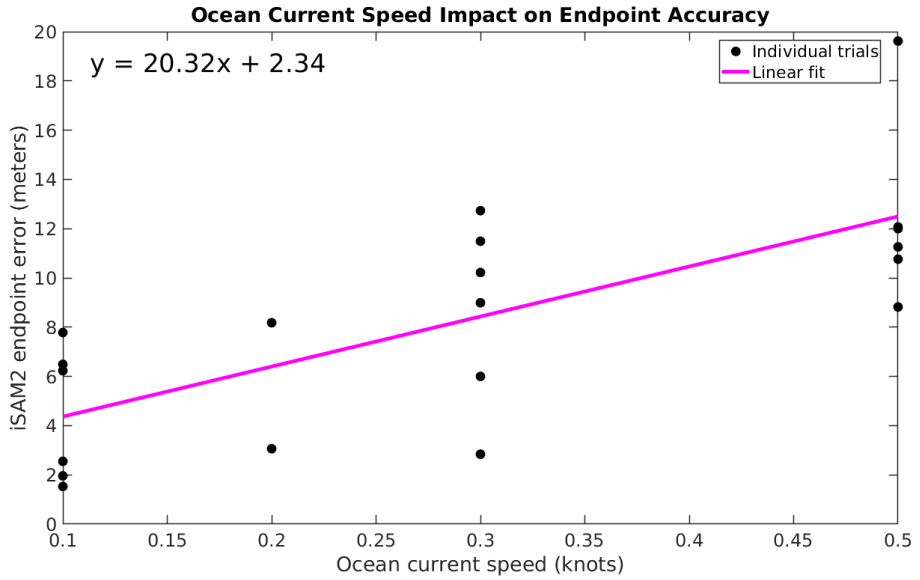


Figure 4-18: Impact of ocean current speed on the endpoint error of our cooperative navigation approach.

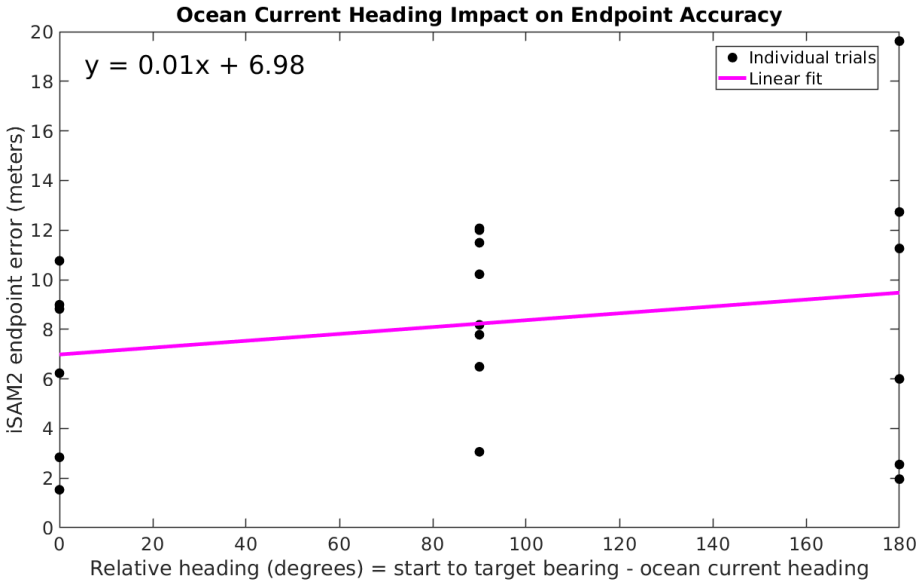


Figure 4-19: Impact of relative ocean current heading on the endpoint error of our cooperative navigation approach.

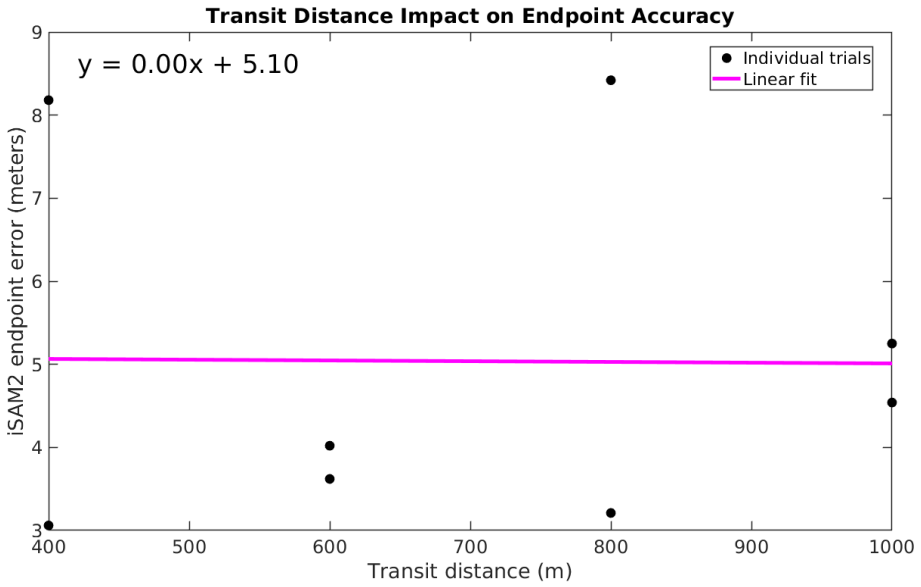


Figure 4-20: Impact of transit distance on the endpoint error of our cooperative navigation approach.

the ocean current's relative heading, defined as the difference between the bearing from the start location to the target and the ocean current heading. Therefore, a zero-degree relative heading means the ocean current flows in the direction of the diver's travel, while a 180-degree relative heading implies the opposite. Due to the shallow slope of the linear fit, it does not appear there is a strong connection between the endpoint error and relative heading.

The last variable changed during the simulations was the dive length. Figure 4-20 compares the dive length with the endpoint error using simulation trials 19 through 26. Specifically, those simulations maintain a 0.2-knot ocean current at a 117-degree bearing with dive distances between 400 meters and one kilometer. Although there is a significant difference between the two trial's endpoint accuracies for the 400- and 800-meter dive lengths, the linear fit suggests the dive length does not impact the endpoint error at the simulated distances.

Along with analyzing how varying environmental factors impacted iSAM2's endpoint error, we also examined how increasing ocean current speeds, varying ocean current headings, and longer dive distances impact the real-time estimates during the dive and the smoothed solution at the end of the dive. The accuracy of these solutions is measured using four metrics. The first and second metrics are called the online maximum distance and the online average distance, respectively. These values assess iSAM2's real-time performance

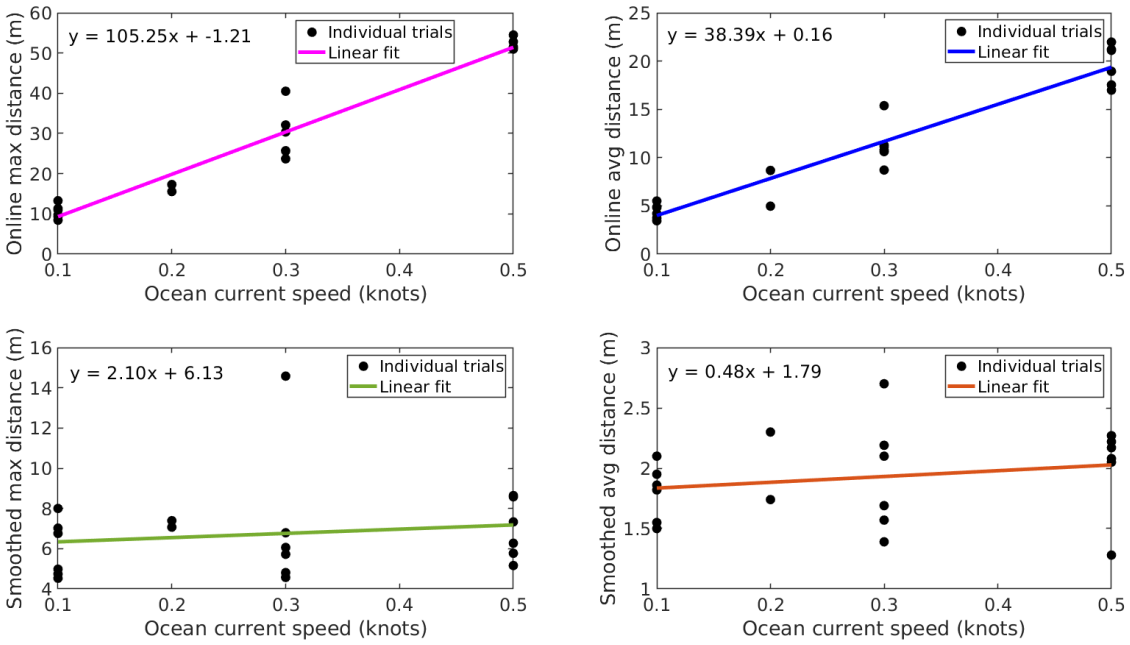


Figure 4-21: Impact of ocean current speed on the online maximum distance (**top left**), online average distance (**top right**), smoothed path maximum distance (**bottom left**), and smoothed path average distance (**bottom right**).

by finding the maximum and average differences between the diver's estimated location and their ground truth position during the dive. The third and fourth are the smoothed path maximum and the smoothed path average distances, respectively. These metrics quantify the accuracy of the smoothed solution by finding its maximum and average distances from the ground truth path.

Figure 4-21 compares the ocean current speed with the accuracy of the cooperative navigation method according to the metrics defined in the previous paragraph. We can infer that our HAT cooperative navigation method degrades as the ocean current speed increases by performing linear fits on our simulation results. The online performance is most impacted because the odometry factors assume the diver travels at 0.92 knots at the heading provided by the AHRS. This assumption becomes less and less accurate as the ocean current speed increases. Although our algorithm intentionally places low confidence in the odometry factors, the speed and heading inaccuracies still cause erratic iSAM2 estimates during the dive. Fortunately, the smoothed solution is impacted significantly less by higher ocean current speeds.

Figure 4-22 and Figure 4-23 demonstrate the effect increased ocean current speeds have

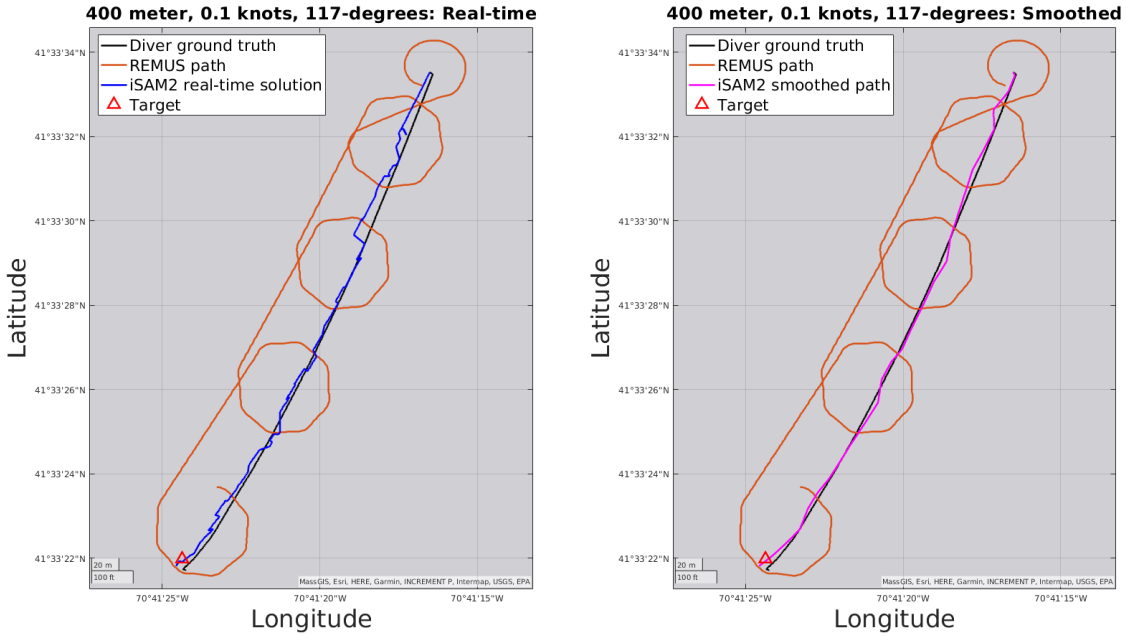


Figure 4-22: 400-meter cooperative navigation simulation results. The ocean current speed and heading are 0.1 knots and 117 degrees, respectively. A 117-degree ocean current heading represents a right-to-left crosscurrent for the diver. **Left:** Real-time diver location estimates. **Right:** Diver's final smoothed path.

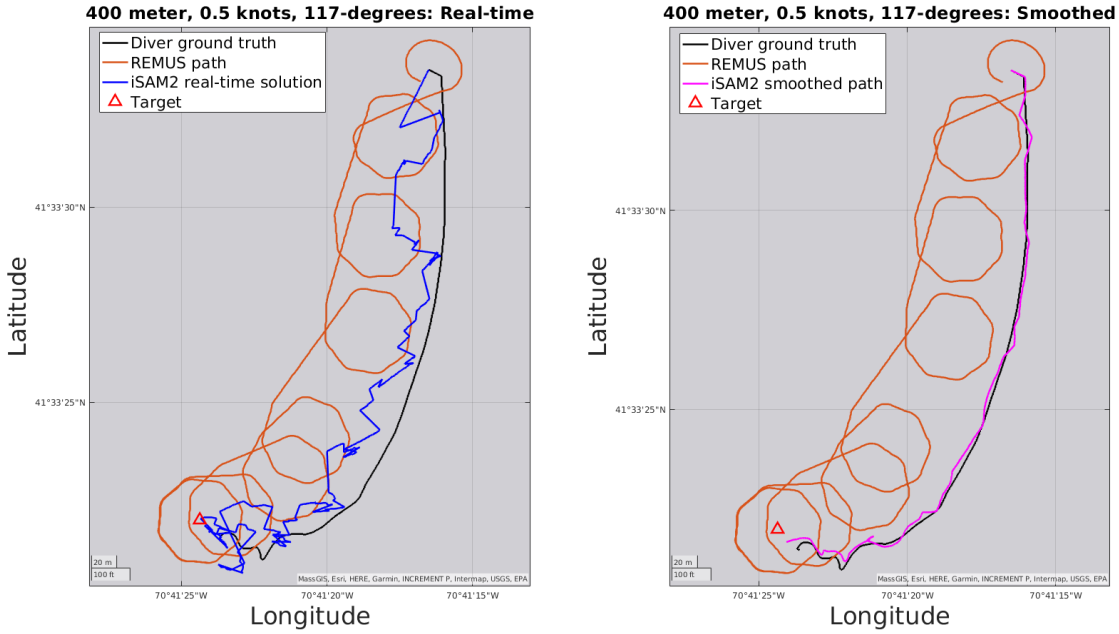


Figure 4-23: 400-meter cooperative navigation simulation results. The ocean current speed and heading are 0.5 knots and 117 degrees, respectively. A 117-degree ocean current heading represents a right-to-left crosscurrent for the diver. **Left:** Real-time diver location estimates. **Right:** Diver's final smoothed path.

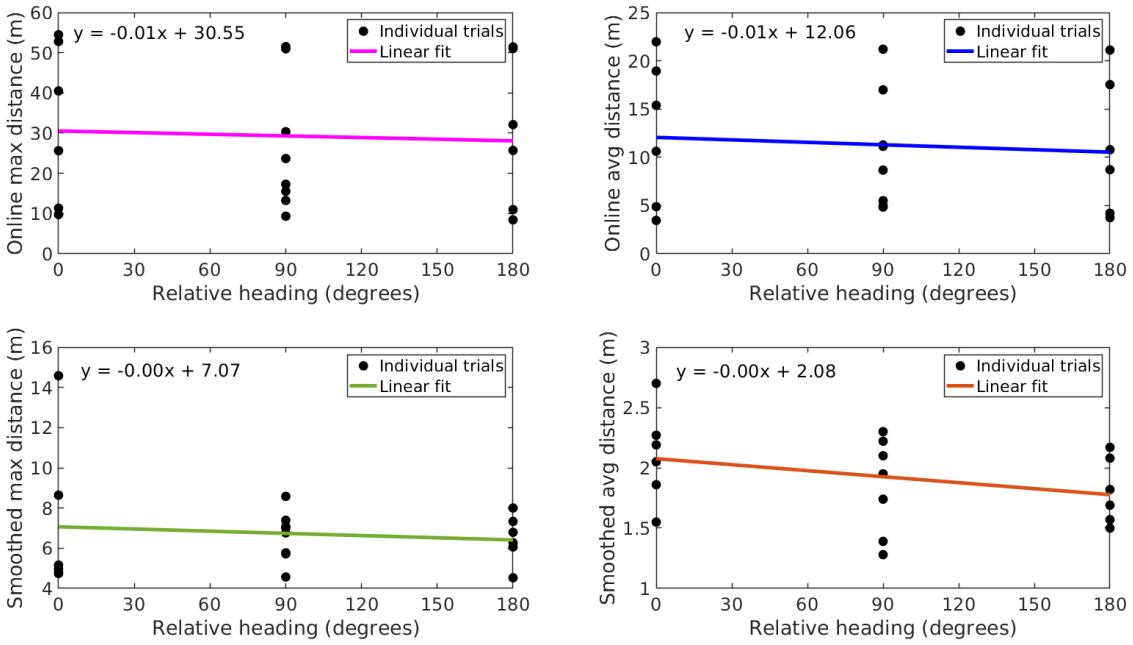


Figure 4-24: Impact of ocean current heading on the online maximum distance (**top left**), online average distance (**top right**), smoothed path maximum distance (**bottom left**), and smoothed path average distance (**bottom right**). We define the relative heading as the bearing from the diver's start location to the target minus the ocean current heading.

on the real-time and smoothed solutions. Both figures show a 400-meter dive length with a 117-degree ocean current heading. However, the ocean current in Figure 4-22 is 0.1 knots, while the ocean current in Figure 4-23 is 0.5 knots. Clearly, the real-time solution in the latter figure is significantly less stable. Since the diver uses those estimates to navigate, their path to the target is negatively impacted. Specifically, the diver makes untraditional turns near the target. However, the smoothed solution matches the diver's irregular path to provide an accurate result.

Figure 4-24 shows that the relative heading of the ocean current has a negligible impact on the cooperative navigation performance. Additionally, Figure 4-25 shows that longer dive lengths may increase the online maximum distance. However, it does not appear that the online performance is impacted as much by the dive length as by the ocean current speed. Regardless, the linear fits for the other three metrics show minimal impact by the longer dive lengths.

Lastly, Table 4.28 provides maximum and average accuracy values for our cooperative navigation method. Although the average values provide useful information about the performance of our approach, the maximum distances are the most impactful. For example,

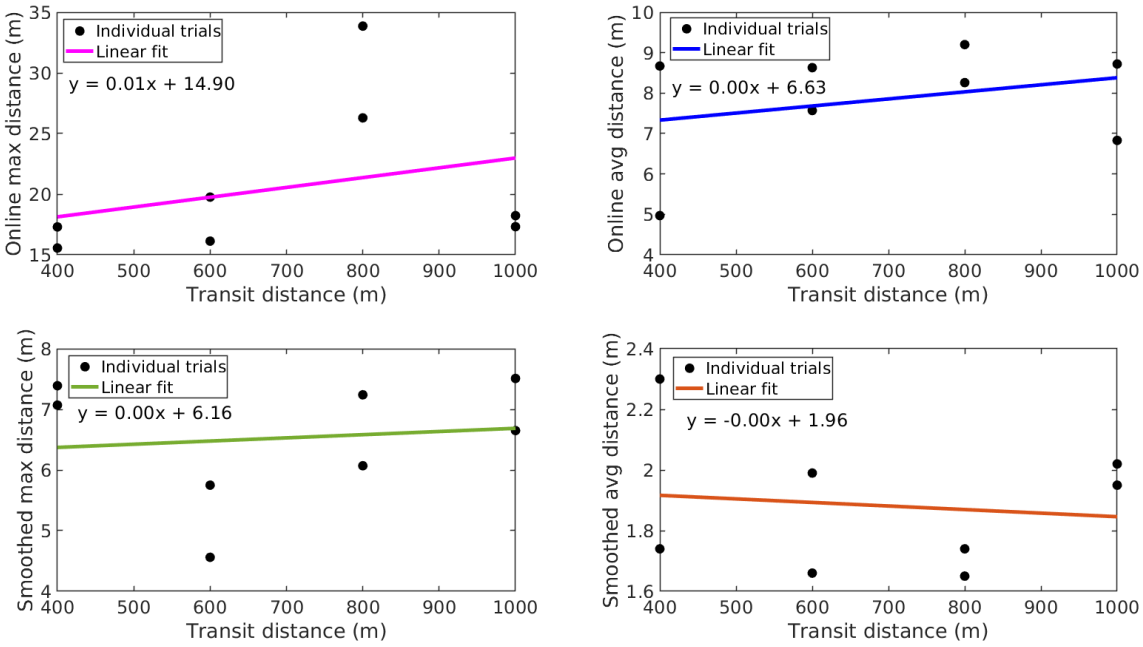


Figure 4-25: Impact of transit distance on the online maximum distance (**top left**), online average distance (**top right**), smoothed path maximum distance (**bottom left**), and smoothed path average distance (**bottom right**).

although divers can expect to be about seven meters away from their ground truth location at the end of the dive, it is critical to realize that they could be nearly 20 meters away. Similarly, during the transit, the diver should know that their estimated position could have an error of 55 meters, even though it is more likely to be around 10 meters.

Maximum and average accuracy values (in meters)		
Metric	**Maximum distance**	**Average distance**
Endpoint error	19.62	7.24
Real-time estimate	54.49	10.27
Smoothed path	14.58	1.91

Table 4.28: Maximum and average accuracy values (in meters) from the cooperative navigation field trials and simulations.

Chapter 5

Conclusions and Future Work

Chapter 5 concludes this thesis by discussing its contribution and proposing future work focused on increasing the capabilities of the diver-AUV team.

5.1 Contribution

This thesis introduced a novel cooperative navigation technique between a REMUS 100 and a diver. Our approach leverages new and existing software repositories to perform ROSB navigation through a new AUV behavior and acoustic communication architecture. Although we did not test our proposed system using a REMUS 100 or diver, we have shown through kayak-to-kayak field experiments and simulations that our approach dramatically improves navigation performance compared to traditional DR solutions. Additionally, while DR diverges from the ground truth location throughout the dive, our method combines all the range measurements throughout the transit to provide accurate trajectory mapping to the diver. Therefore, along with improving the diver's ability to navigate to the target, our system shows them how they arrived at that location.

5.2 Future Work

Future work in human-autonomy teaming between a diver and a REMUS AUV can be categorized in two ways: improving the team's cooperative navigation ability and expanding the team's capabilities beyond navigation. This section addresses the former by considering improvements to the autonomous behavior and the state estimation algorithm. Specifically,

we propose experiments to test our current autonomous behavior and recommend research into how it can incorporate more variables, such as distance from the diver and turning radius, to calculate its next waypoint. We also believe effort should be focused on how existing Micromodem 2 functions can be used to teach the AUV about the operating environment and calculate the diver's speed. After recommending adaptations to the diver's software to provide them with a confidence level of their state estimate, we conclude this section by introducing how the diver-AUV team could advance with soundscape analysis and cooperative surveys. The diver-AUV team is just beginning to develop, and there is much more work to be done.

5.2.1 Improving the Autonomous Behavior

Improving the autonomous behavior will progress the diver's navigation ability. First, we believe efforts should focus on validating the AUV's circular behavior. Then, we recommend incorporating intermittent surfacing, additional optimization parameters, and environmental monitoring to improve the cooperative navigation performance.

It is important to verify that a circling behavior provides adequate navigation solutions to the REMUS. The AUV rolls to one side while turning, potentially impacting the DVL's bottom lock and decreasing the INS's performance. Although we propose a circling behavior in this thesis, a rectangular path could provide similar location diversity to the diver while benefiting INS performance. Figure 5-1 provides a visual of the rectangle behavior.

Performing a field test to determine the impact of a circular versus a rectangular behavior on the INS would provide valuable information to improve our ROSB approach. The test would start with a REMUS 100 circling a coordinate for an extended period and then surfacing. Software onboard the back seat computer would compare the vehicle's estimated location to the GPS coordinates it received at the end of the trial. Then, the AUV would complete the same process but with a rectangular path instead. Less endpoint error indicates superior INS performance. If the rectangle behavior offers significant INS improvements, our autonomous behavior should be adapted to perform that pattern instead.

Another approach to decreasing the REMUS's location drift is to have the vehicle intermittently surface to get a GPS fix. The behavior to enable this capability already exists on the `Navigator Node`. Specifically, the software monitors the AUV's R95 uncertainty and triggers a porpoise behavior to surface the vehicle if it exceeds a user-defined value. After

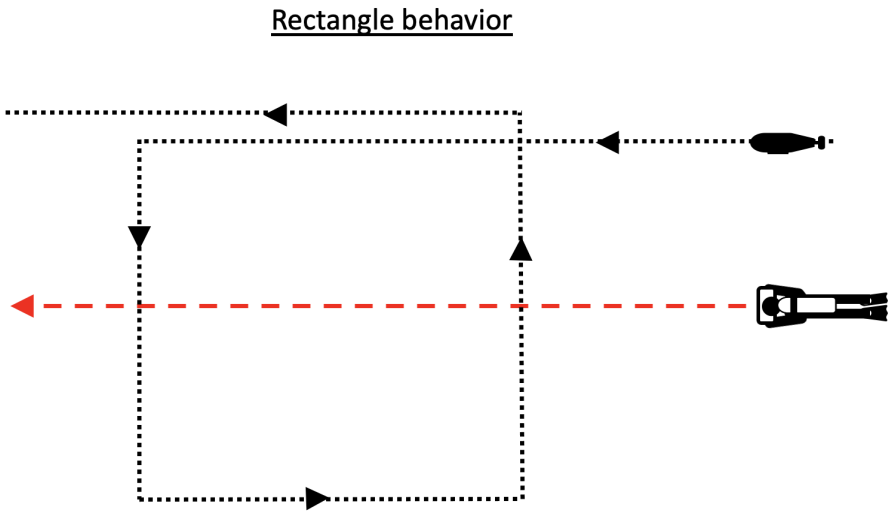


Figure 5-1: Cooperative navigation rectangle behavior.

receiving a fix and resetting the vehicle's location uncertainty, the behavior submerges the AUV back to the appropriate depth.

We did not utilize this capability during the simulations because we sought to assess the performance of a cooperative navigation approach with no surface presence during the transit. Additionally, simply surfacing the vehicle without knowing what is on the surface could result in a collision. Therefore more research should be done on using an upward-facing acoustic Doppler current profiler (ADCP), passive acoustic monitoring, or acoustic pinging to determine whether it is safe for an AUV to surface. After mitigating the collision risk, the AUV can safely surface during the transit to reset its location uncertainty, resulting in more accurate two-dimensional ranges for iSAM2. This behavior will likely improve the diver's location estimate and enable longer dives.

Future efforts should also adapt our autonomous behavior to calculate the REMUS's next waypoint using more variables. This thesis introduced a relatively simple behavior that predicts the diver's location, calculates a new waypoint based on that location, and performs a circle. Early simulations showed this behavior maintains the desired 15- to 100-meter REMUS-diver dispersion provided there is no ocean current. However, once the simulations incorporated ocean currents, there were multiple occurrences of the REMUS and diver being at the same location. Even with our autonomous behavior's two-meter depth dispersion, it is imperative to maintain the proper horizontal distance to ensure the diver's safety. Therefore, the `Navigator Node` should calculate its next waypoint in a way that

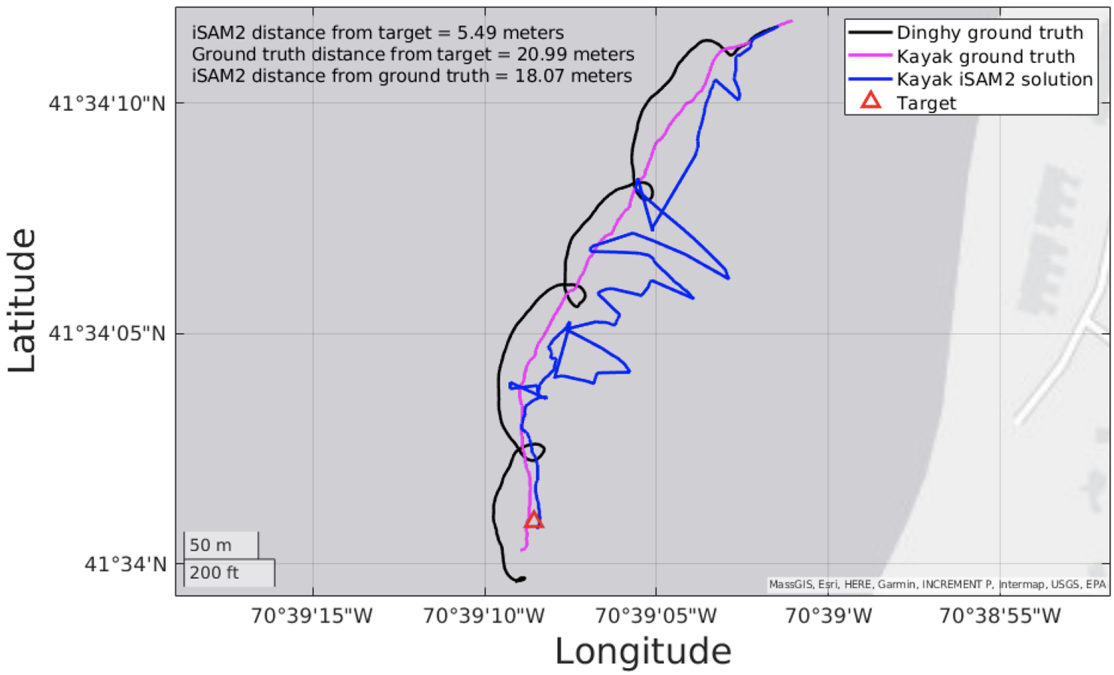


Figure 5-2: October 8th cooperative navigation trial.

provides beneficial range diversity while maintaining a safe distance from the diver.

Although we experienced perfect communication during the cooperative navigation experiment on November 6th, an earlier trial from October 8th showed the impact of unsuccessful pings on the navigation solution. Figure 5-2 shows the unstable real-time estimates and high endpoint error results from a trial where only six of the 15 pings were successful.

Our early experiments used a dinghy as a proxy for the REMUS 100. We believe the decreased acoustic communication performance resulted from a combination of acoustic clipping with Rate 5 pings, a high multipath environment, and the dinghy's motor having a frequency near our 25-kilohertz signals. Although the motor's impact is unconfirmed, substituting the dinghy for a kayak and moving the experiment to Bourne's Pond increased the ping success rate even though we were still using Rate 5 pings in a very shallow environment. However, divers that utilize our HAT cooperative navigation method will not have the luxury of removing noise sources or changing their environment. Therefore, future work should adapt our autonomous behavior to monitor the environment's acoustic properties and adjust the vehicle's location to benefit communication with the diver.

Fortunately, the Micromodem 2 automatically analyzes key acoustic communication metrics and publishes the results within a Communications Cycle Receive Statistics (CST)

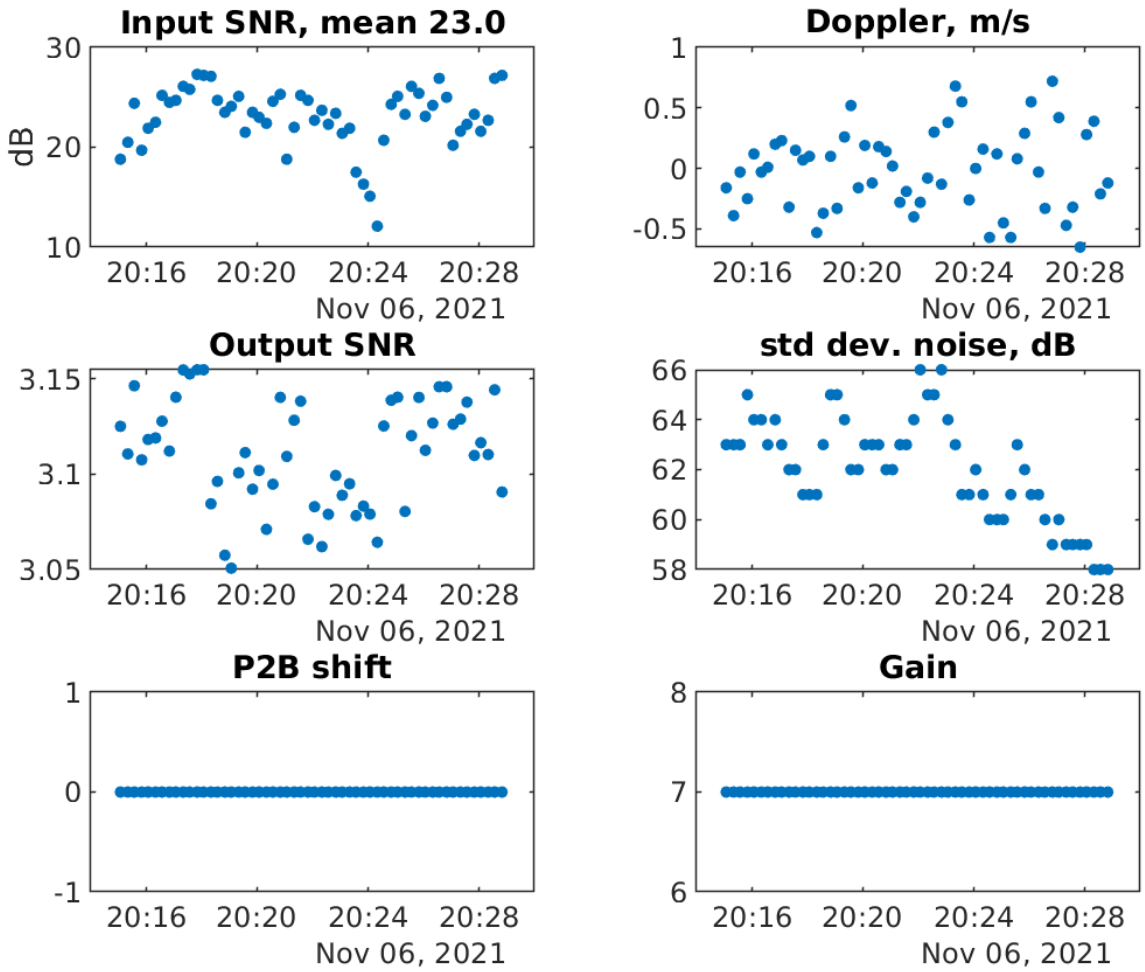


Figure 5-3: Modem statistics for all mini-packets transmitted and received by the diver kayak during the first November 6$^{\text{th}}$ cooperative navigation trial.

NMEA string. Figure 5-3 shows CST data from a November 6$^{\text{th}}$ cooperative navigation trial. The Micromodem 2 determines the input SNR after bandpass filtering the incoming signal and the output SNR after the adaptive equalizer attempts to eliminate the multipath and Doppler shift effects. A high input SNR and low output SNR indicate the equalizer is having difficulty tracking the channel, typically due to significant multipath [103]. Additionally, the P2B shift represents the modem converting the incoming data from 12-bits to 16-bits during basebanding. Lower P2B shifts indicate high signal strengths [57], and, when paired with a low gain, also suggest the signal is being clipped. Therefore, by analyzing Figure 5-3, we can determine that we are likely in an environment with high multipath and that we should decrease our transmission power to mitigate acoustic clipping.

The REMUS 100 could perform a similar analysis in real-time to evaluate the acoustic

environment and adapt its path to benefit communication. For example, if the AUV detected an area where the acoustic noise was high enough to impact communication, it could relocate itself and instruct the diver to avoid that area. This approach would result in the diver swimming indirect paths from the start to the target. However, our simulation results show iSAM2 still performs accurate trajectory mapping when the diver makes unconventional turns during the dive.

Although autonomously adapting to the acoustic environment in real-time is beneficial, the REMUS could also preemptively map the environmental properties of the operating area through a reconnaissance mission. This step would occur before the diver entered the water and require the REMUS to passively monitor the acoustic environment and ocean current field as it traveled a pre-determined route. Using the information from this behavior, the REMUS could calculate an optimal trajectory for the diver that accounts for the ocean current and avoids areas where poor acoustic communication is likely. Overall, an environmentally adaptable autonomous behavior is paramount to improving our HAT cooperative navigation approach and should be a primary area of future research.

5.2.2 Improving State Estimation on the Submersible Tablet

Future work should focus on implementing the `ros_hat` nodes on a submersible dive tablet and progressing the capabilities of the diver's software. Therefore, this subsection recommends incorporating existing Micromodem 2 and GTSAM capabilities on the submersible tablet. Specifically, estimating the diver's speed using the Doppler shift and quantifying iSAM2's confidence in the diver's location would improve navigation and build trust between the teammates.

As discussed in Section 4.10, increased ocean current speeds negatively impacted our method's performance more than varying ocean current heading or longer dive lengths. Since this thesis used a constant speed assumption to create the odometry factors within iSAM2, higher ocean current speeds caused greater discrepancies between the diver's estimated speed and heading and their actual SOG and COG. Adding odometry factors that reflect the diver's SOG and COG is likely to mitigate the negative impact from high ocean current speeds.

Fortunately, as shown in Figure 5-3, the Micromodem 2's CST message reports the Doppler shift between the transmitter and receiver as the relative speed between the two entities. Therefore, the diver's SOG and COG can be calculated using this relative speed

value along with the REMUS's SOG and COG. Incorporating the algorithms to determine the diver's SOG and COG within the `Director Node` would eliminate our system's reliance on the constant speed assumption, benefiting the state estimation algorithm and improving the real-time navigation performance.

Estimating the diver's SOG using the Doppler shift has other benefits for the diver. First, it would remove the requirement for the diver to manually notify the `Director Node` when they start or stop swimming. Instead, the diver's speed would automatically update every time a new packet or ping arrived. Second, the ocean current speed and heading could be estimated by comparing the diver's SOG and COG with their expected speed and AHRS heading. These ocean current values would help determine a recommended heading for the diver to swim the most direct route to the target.

Along with improving our system's performance, future work should go towards increasing its transparency. GTSAM provides functions to quantify the confidence of iSAM2's current location estimate [104]. A useful improvement would be to incorporate these functions into the `Estimator Node` and publish the data to the dive tablet's screen. To develop trust between the diver and this cooperative navigation method, the diver must know when the software is confident in the solution and vice versa. If the diver is provided with a solution of low confidence, they could perform certain actions to improve the result. For example, they could remain in a single location for a few minutes and wait for numerous range measurements to arrive, potentially increasing the `Estimator Node`'s confidence in their prediction. In this way, the HAT could use the diver's adaptability and creativity to improve the system's navigation performance.

5.2.3 Beyond Navigation

The future work recommended thus far has been focused on improving the performance and useability of our cooperative navigation algorithm. However, the HAT has additional benefits after reaching the target. Two of those are environmental monitoring and cooperative surveying.

The REMUS 100 could provide passive acoustic monitoring capabilities for a diver working at the target location. This function relies on algorithms trained to detect certain noises within an ocean soundscape. For example, if a neural network on the REMUS's back seat computer could determine whether a boat was nearby, it could notify the diver to maintain

an adequate depth.

Another future capability at the target location is that the REMUS and diver could cooperatively survey the target area. A survey with a robot and human working together is called a cyborg survey. Provided automatic target recognition capabilities are sufficient to allow the REMUS 100 to locate points of interest in real-time, the REMUS could navigate the diver to a survey box and then perform a cyborg survey. As the REMUS conducts a broad survey of the area, it marks points of interest and acoustically transmits their coordinates to the diver. The diver then travels to each location using our HAT cooperative navigation method to analyze the target more thoroughly. A potential application of this is within Project Recover, which seeks to locate and recover remains of lost World War II veterans in the Pacific Ocean [105]. Overall, cyborg surveys are a natural extension of the cooperative navigation approach proposed in this thesis.

5.3 Concluding Remarks

This thesis provides the software and communication architecture necessary to enable cooperative navigation between a REMUS 100 and a diver. Although developing a system that improves upon a diver's dead reckoning solution is beneficial, there are many options available to accomplish this, with some providing endpoint accuracy below a meter [106]. Therefore, this research separates itself from the other methods through underwater human-autonomy teaming. Additionally, unlike previous underwater human-robot teaming efforts, we rely on commercial AUV navigation capabilities and acoustic communication to provide subsurface navigation from the diver's starting location to the target. Overall, this research takes an early step towards developing a reliable and competent underwater HAT that improves the capabilities and safety of divers.

Appendix A

List of Acronyms

ADCP	acoustic Doppler current profiler
ADRIATIC	Advancing Diver-Robot Interaction Capabilities
AHRS	attitude and heading reference system
ASCII	American Standard Code for Information Interchange
ASV	autonomous surface vehicle
ATR	automatic target recognition
AUV	autonomous underwater vehicle
CADDY	Cognitive Autonomous Diving Buddy
CEP	circular error probable
CNA	communication and navigation aid
CNN	convolutional neural network
COG	course over ground
CSAC	chip-scale atomic clock
CST	Communication Cycle Receive Statistics

CSV	comma-separated values
CTD	conductivity, temperature, and depth
DGPS	Differential Global Positioning System
DR	dead reckoning
DSP	digital signal processor
DVL	Doppler velocity log
EKF	extended Kalman filter
EM	electromagnetic
FDP	Flexible Data Protocol
FH-FSK	frequency-hopping frequency-shift keying
FLS	forward-looking sonar
FOG	fiber optic gyroscope
FSK	frequency-shift keying
GiB	gibibyte
GIB	Global Position System intelligent buoy
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GT	ground truth
GTSAM	Georgia Tech Smoothing and Mapping
HAT	human-autonomy team
HMM	Hidden Markov Model

HNF	human-autonomy teaming navigation Follower Packet
HNI	human-autonomy teaming navigation Initialization Packet
HNL	human-autonomy teaming navigation Leader Packet
HNT	human-autonomy teaming navigation Terminate Packet
IMU	inertial measurement unit
INS	inertial navigation system
iSAM2	incremental smoothing and mapping 2
ISI	intersymbol interference
LBL	long baseline
LSTM	long short-term memory
MAP	maximum a posteriori
MCM	mine countermeasure
MEMS	microelectromechanical system
MIT	Massachusetts Institute of Technology
MLBL	moving long baseline
MMSE	minimum mean square error
NLS	nonlinear least-squares
NMEA	National Marine Electronics Association
OWTT	one-way-travel-time
PADI	Professional Association of Diving Instructors
PATS	portable acoustic tracking system

PF	particle filter
PPS	pulse-per-second
PSK	phase-shift keying
QPSK	quadrature phase-shift keying
RAM	random access memory
RECON	Remote Control
REMUS	Remote Environmental Monitoring Units
RI	reacquire and identify
RMS	root mean square
RNN	recurrent neural network
ROS	Robot Operating System
ROSB	range-only single-beacon
RTC	real-time clock
SAR	search and rescue
SBL	short baseline
SCM	search, classify, and map
SCUBA	self-contained underwater breathing apparatus
SE	special Euclidean
SLAM	simultaneous localization and mapping
SNR	signal-to-noise ratio
SOG	speed over ground

SPP	spatial pyramid pooling
TDMA	time-division multiple access
TWTT	two-way-travel-time
UDP	User Datagram Protocol
USB	Universal Serial Bus
USBL	ultra-short baseline
USN	United States Navy
VGG	Visual Geometry Group (Oxford University)
VIP	Vehicle Interface Program
WGS 84	World Geodetic System 1984
WHOI	Woods Hole Oceanographic Institution
YOLO	You Only Look Once

Appendix B

Software Architecture

B.1 Repository Breakdown

The lists below provide the nodes, services, messages, and actions our HAT navigation algorithm relied on within each software repository.

ros_acomms [66]

Nodes:

- Acomms Driver Node
- Packet Dispatch Node
- Message Queue Node
- TDMA Node

Services:

- /queue_tx_packet
- /get_next_packet_data
- /write_nmea_string

Messages:

- /nmea_from_modem
- /packet

- /from_acomms/initial_packet
- /from_acomms/follower_packet
- /from_acomms/leader_packet
- /from_acomms/terminate_packet

ros_remus [65]

Node:

- REMUS Node

Services:

- /mission_command
- /update_transit

Messages:

- /status
- /ctd
- /fix

Action:

- /transit

ros_hat [18]

Nodes:

- Navigator Node
- Director Node
- Estimator Node
- Stopwatch Node
- Depth Node

- Arrow Node

Messages:

- /HNI_trigger
- /HNT_trigger
- /swim
- /stop_swim
- /tablet_data
- /beacon_data
- /initial_packet
- /follower_packet
- /leader_packet
- /terminate_packet
- /diver_state
- /diver_depth
- /arrow
- /ping_trigger
- /leader_trigger

Other software repository [75]

Node:

- Sparton Node

Message:

- /spartonm2/ahrs

B.2 `ros_hat` Information

This section provides more information on the nodes and messages within `ros_hat`.

B.2.1 Node descriptions

Navigator Node

- Receive message data from the Micromodem 2 using `ros_acomms`
- Determine the next coordinate the REMUS should travel to based on the diver's estimated location
- Ensure the back seat computer retains control of the REMUS by initializing and updating a transit action using `ros_remus`
- Collect necessary sensor outputs using `ros_remus` for transmission to the diver
- Transmit the appropriate message to the diver using `ros_acomms`

Director Node

- Receive message data from the Micromodem 2 using `ros_acomms`
- Calculate range measurements using data sent by the REMUS
- Pass range data to the `Estimator Node`
- Collect necessary data from sensors and nodes for transmission to the REMUS
- Transmit the appropriate message to the REMUS using `ros_acomms`

Estimator Node

- Receive range data from the `Director Node`
- Receive odometry data from the AHRS
- Given an initial fix, provide online diver location estimates using odometry and range measurements

Stopwatch Node

- If a ping sequence fails, trigger the `Navigator Node` to send another ping
- If an acoustic message sequence fails, trigger the `Navigator Node` to send another Leader Packet
- Provide parameters for the user to adjust the time allotted for each of these triggers to occur

Depth Node

- Interface with a depth sensor to provide depth measurements to the `Director Node`

Arrow Node

- Interface with the Arrow 100 GNSS receiver to provide location data to the `Director Node`

B.2.2 Message content

/HNI_trigger

- string trigger_str

/HNT_trigger

- string trigger_str

/swim

- string trigger_str

/stop_swim

- string trigger_str

/tablet_data

- string sparton_heading
- string bearing_to_target
- string dist_to_target

/beacon_data

- float32 sound_speed
- float32 two_dim_range
- float32 beacon_lat
- float32 beacon_lon
- uint32 factor_number
- float32 beacon_loc_uncertainty
- uint32 time_of_fix

/initial_packet

- float32 diver_depth
- float32 diver_latitude
- float32 diver_longitude
- float32 target_latitude
- float32 target_longitude

/follower_packet

- float32 diver_sog
- float32 diver_depth
- float32 diver_latitude
- float32 diver_longitude
- uint32 diver_time_of_fix

/leader_packet

- float32 remus_depth
- float32 remus_loc_uncertainty_sigma

- float32 remus_latitude
- float32 remus_longitude
- float32 sound_speed
- float32 owtt

/terminate_packet

- uint8 terminate

/diver_state

- float32 diver_latitude
- float32 diver_longitude
- uint32 time_of_estimate
- uint32 factor_number

/diver_depth

- float32 diver_depth

/arrow

- float64 latitude
- float64 longitude
- int32 utc_epoch_time
- float32 horizontal_rms_error

/ping_trigger

- bool ping_trigger

/leader_trigger

- bool leader_trigger

B.3 Software Maps

The `ros_acomms` [66] and `ros_remus` [65] documentation pages were referenced to make the software maps below.

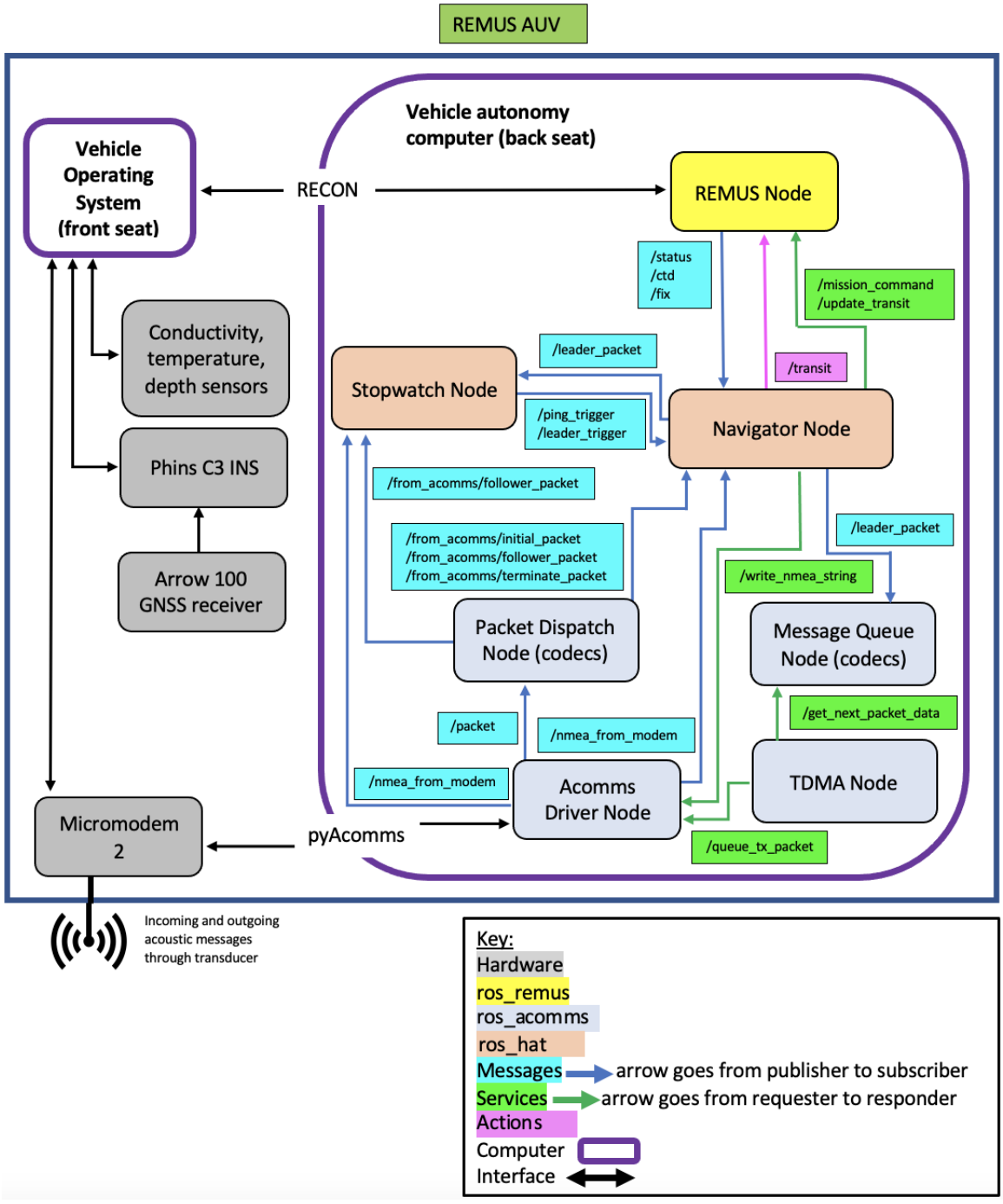


Figure B-1: Software map for the REMUS.

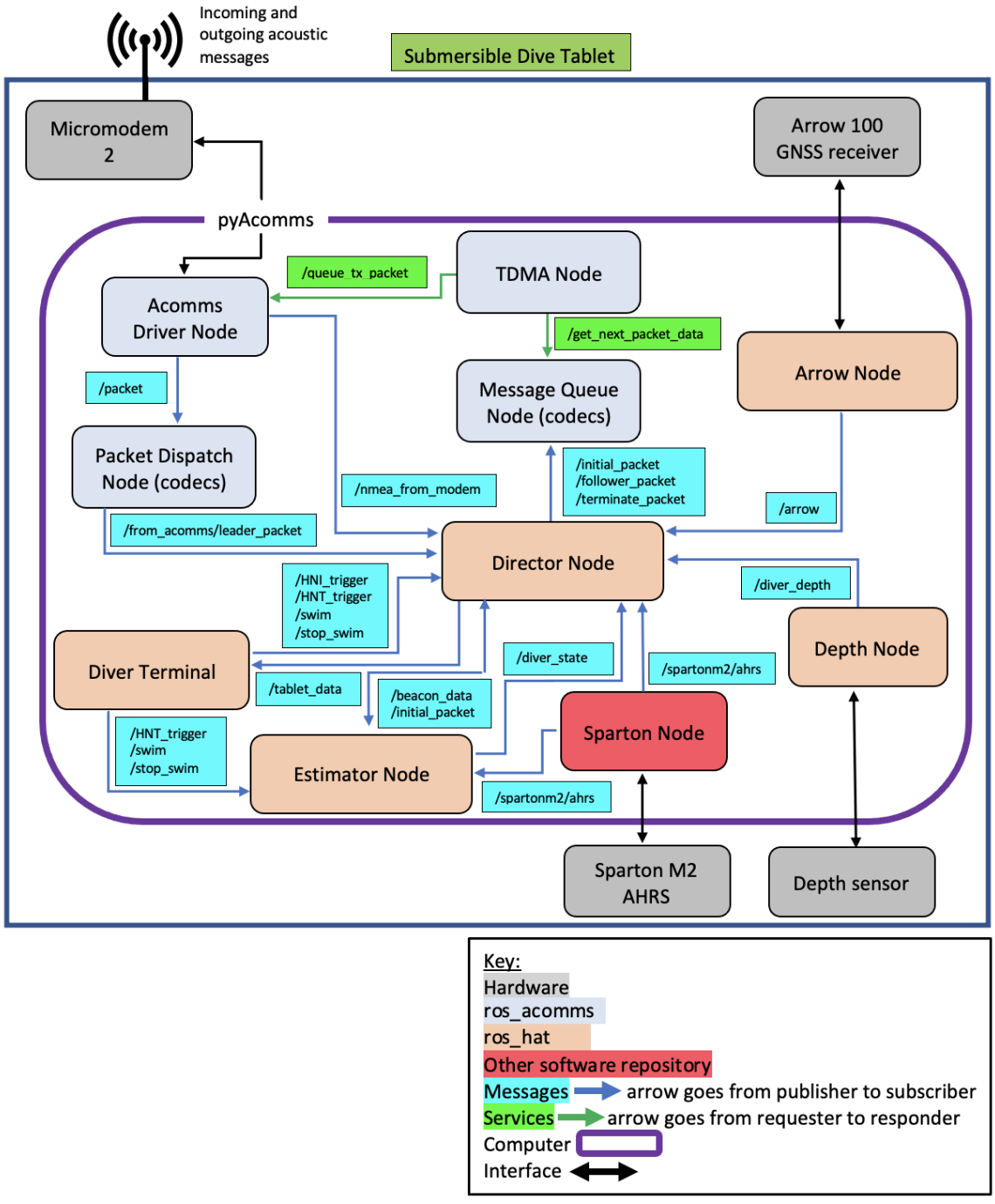


Figure B-2: Software map for the diver.

Appendix C

Dead Reckoning Simulation MATLAB Script

The MATLAB [90] code below was used to perform the dead reckoning simulations. The parameters shown in the example are for a 400-meter trial with an ocean current speed and heading of 0.2 knots and 117 degrees True, respectively. Aside from commonly available MATLAB functions, this script also uses `vdist` [97], which calculates the distance between two coordinates using Vincenty's formula. Additionally, the equations within the `next_coords` function are based on the haversine formula [96].

```
% parameters
start_lat = 41.5593119;
start_lon = -70.6879216;
target_coord = [41.5561076, -70.6901034]; % 400-meter trial
ocean_current_speed_kts = 0.2;
ocean_current_heading_deg = 117;
original_speed_estimate_kts = 0.92;

% initialize the kick speed and distance to target
mean_kick_speed_kts = original_speed_estimate_kts + normrnd(0,0.05);
dist_to_target = vdist(start_lat,start_lon,target_coord(1),target_coord(2));

% get the ocean current vector (assume constant ocean current speed and ...
    heading)
ocean_current_speed_ms = ocean_current_speed_kts * 0.5144;
```

```
converted_angle = mod(360 - ocean_current_heading_deg + 90,360);
ocean_curr_N = ocean_current_speed_ms * sin(converted_angle * pi/180);
ocean_curr_E = ocean_current_speed_ms * cos(converted_angle * pi/180);

% get the starting position
diver_gt_lat(1) = start_lat;
diver_gt_lon(1) = start_lon;
without_o_curr_lat(1) = start_lat;
without_o_curr_lon(1) = start_lon;

% initialize the previous coordinate as the starting position
prev_coord = [diver_gt_lat(1), diver_gt_lon(1)];
without_o_curr_prev_coord = [diver_gt_lat(1), diver_gt_lon(1)];

% while the DR solution is greater than 2 meters from the target
i = 1;
while dist_to_target > 2
    latA = without_o_curr_prev_coord(1) * pi/180;
    lonA = without_o_curr_prev_coord(2) * pi/180;
    latB = target_coord(1) * pi/180;
    lonB = target_coord(2) * pi/180;
    X = cos(latA)*sin(latB) - sin(latA)*cos(latB)*cos(lonB-lonA);
    Y = sin(lonB-lonA)*cos(latB);
    bearing_to_target = atan2(Y,X) * 180/pi;

    % add noise to diver's heading
    r = -5 + (5+5)*rand(1,1);
    diver_heading = bearing_to_target + r;
    converted_heading = mod(360 - diver_heading + 90,360);

    % add noise to diver's speed
    diver_kick_speed = (mean_kick_speed_kts + normrnd(0,0.05))*0.5144; % m/s

    % set the equations x(1) = diver_sog, x(2) = diver_cog (converted)
    F = @(x) [(diver_kick_speed * cos(converted_heading * pi / 180)) + ...
        ocean_curr_E - (x(1) * cos(x(2) * pi / 180));
        (diver_kick_speed * sin(converted_heading * pi / 180)) + ...
            ocean_curr_N - (x(1) * sin(x(2) * pi / 180))];

```

```
52      % make an initial guess
53      x0 = [diver_kick_speed;converted_heading];
54
55      % solve the equations
56      [x,¬] = fsolve(F,x0);
57
58      % calculate the distance the diver traveled over one second
59      distance = (x(1) * 1)/1000;
60      without_o_curr_dist = (original_speed_estimate_kts * 0.5144 * 1)/1000;
61
62      % convert answer back to compass angle reference frame
63      diver_cog = mod(450 - x(2),360);
64
65      % find the diver's new coordinates: next_coords(prev_lat, prev_lon, ...
            COG, distance)
66      [next_lat, next_lon] = next_coords(prev_coord(1), prev_coord(2), ...
            diver_cog, distance);
67      [without_o_curr_next_lat, without_o_curr_next_lon] = ...
            next_coords(without_o_curr_prev_coord(1), ...
68          without_o_curr_prev_coord(2), bearing_to_target, ...
                without_o_curr_dist);
69
70      % update the lat and lon matrices
71      diver_gt_lat(i+1) = next_lat;
72      diver_gt_lon(i+1) = next_lon;
73      without_o_curr_lat(i+1) = without_o_curr_next_lat;
74      without_o_curr_lon(i+1) = without_o_curr_next_lon;
75
76      % reset the previous coordinates to the new diver location
77      prev_coord = [next_lat, next_lon];
78      without_o_curr_prev_coord = [without_o_curr_next_lat, ...
            without_o_curr_next_lon];
79
80      % find new distance to target (what DR thinks)
81      dist_to_target = vdist(without_o_curr_next_lat, ...
            without_o_curr_next_lon,target_coord(1),target_coord(2));
82      i = i + 1;
83  end
84  fprintf('DR distance from target = %2.2f \n',dist_to_target);
```

```
actual_dist = vdist(diver_gt_lat(end), ...
    diver_gt_lon(end),target_coord(1),target_coord(2));
fprintf('GT distance from target = %3.2f \n',actual_dist);
dist_between = vdist(diver_gt_lat(end), ...
    diver_gt_lon(end),without_o_curr_lat(end),without_o_curr_lon(end));
fprintf('GT distance from DR = %3.2f \n',dist_between);
```

```
function [lat_final,lon_final]  = next_coords(lat, lon, bearing, distance)
  % bearing in degrees
  % distance in kilometers (km)
  R = 6371; % Earth's radius in km

  lat2 = asin(sin(pi / 180 * lat) * cos(distance / R) + cos(pi / 180 * ...
      lat) * sin(distance / R) * cos(pi / 180 * bearing));
  lon2 = pi / 180 * lon + atan2(sin( pi / 180 * bearing) * sin(distance / ...
      R) * cos( pi / 180 * lat ), cos(distance / R) - sin( pi / 180 * ...
      lat) * sin(lat2));

  lat_final = 180 / pi * lat2;
  lon_final = 180 / pi * lon2;
```

Appendix D

Simulation Results

Definitions

iSAM2-target endpoint accuracy = the distance between iSAM2's final prediction of the diver's location and the target coordinates.

GT-target endpoint accuracy = the distance between the diver's final ground truth location and the target coordinates.

iSAM2-GT endpoint accuracy = the distance between iSAM2's final prediction of the diver's location and the diver's final ground truth location.

Real-time maximum = the maximum distance throughout the transit between iSAM2's real-time prediction of the diver's location and the diver's ground truth coordinates.

Real-time average = the average distance throughout the transit between iSAM2's real-time prediction of the diver's location and the diver's ground truth coordinates.

Smoothed path maximum = the maximum distance between iSAM2's smoothed diver path and the diver's ground truth path.

Smoothed path average = the average distance between iSAM2's smoothed diver path and the diver's ground truth path.

DR-target endpoint accuracy = the distance between the dead reckoning algorithm's final prediction of the diver's location and the target coordinates.

DR-GT endpoint accuracy = the distance between the dead reckoning algorithm's final prediction of the diver's location and the diver's final ground truth location.

	Ocean Current		Endpoint			Real-time		Smoothed Path	
Length	Speed	Heading	iSAM2-target	GT-target	iSAM2-GT	Maximum	Average	Maximum	Average
400	0.1	27	2.47	0.76	2.55	8.44	3.74	4.53	1.50
400	0.1	27	0.64	2.58	1.96	10.99	4.20	8.00	1.82
400	0.1	117	6.76	8.85	7.78	13.27	5.50	7.02	1.95
400	0.1	117	3.60	9.69	6.49	9.34	4.84	6.76	2.10
400	0.1	207	8.08	2.47	6.23	9.75	3.46	4.98	1.86
400	0.1	207	3.87	3.03	1.53	11.34	4.88	4.74	1.55
400	0.3	27	5.83	7.31	12.73	32.12	10.78	6.79	1.57
400	0.3	27	0.64	6.36	6.00	25.71	8.71	6.06	1.69
400	0.3	117	6.62	8.15	11.49	30.37	11.28	4.57	1.39
400	0.3	117	2.21	11.22	10.22	23.69	11.13	5.72	2.10
400	0.3	207	5.69	5.40	8.99	25.64	10.62	4.81	2.19
400	0.3	207	5.60	8.25	2.84	40.50	15.38	14.58	2.70
400	0.5	27	12.30	22.92	11.26	51.06	17.53	7.33	2.17
400	0.5	27	5.11	24.39	19.62	51.44	21.11	6.27	2.08
400	0.5	117	12.00	23.55	12.00	50.98	21.20	8.58	2.22
400	0.5	117	5.70	17.27	12.07	51.53	16.99	5.77	1.28
400	0.5	207	10.07	18.77	8.82	52.81	18.94	8.64	2.27
400	0.5	207	9.61	19.62	10.76	54.49	21.97	5.16	2.05
400	0.2	117	0.00	3.06	3.06	15.56	4.97	7.07	2.30
400	0.2	117	2.84	10.70	8.18	17.30	8.67	7.39	1.74
600	0.2	117	0.00	3.62	3.62	16.13	7.57	4.56	1.66
600	0.2	117	2.97	6.90	4.02	19.76	8.63	5.75	1.99
800	0.2	117	1.81	2.85	3.21	26.29	9.20	6.07	1.65
800	0.2	117	1.34	9.67	8.42	33.86	8.26	7.24	1.74
1000	0.2	117	2.68	7.20	4.54	17.32	8.72	7.51	2.02
1000	0.2	117	0.85	5.52	5.25	18.23	6.83	6.65	1.95

Table D.1: iSAM2 results from all simulation trials. The ocean current speed is in knots and the ocean current heading is in degrees True. All other values are in meters.

	Ocean Current		Endpoint		
Length	Speed	Heading	DR-target	GT-target	DR-GT
400	0.1	27	1.88	46.91	45.03
400	0.1	27	1.88	33.58	31.70
400	0.1	117	1.88	46.61	46.03
400	0.1	117	1.88	42.40	42.60
400	0.1	207	1.88	75.68	77.56
400	0.1	207	1.88	55.69	57.57
400	0.3	27	1.88	133.30	131.41
400	0.3	27	1.88	116.99	115.10
400	0.3	117	1.88	130.30	130.34
400	0.3	117	1.88	131.73	132.03
400	0.3	207	1.88	128.97	130.85
400	0.3	207	1.88	103.05	104.93
400	0.5	27	1.88	201.99	200.11
400	0.5	27	1.88	235.60	233.72
400	0.5	117	1.88	218.82	218.65
400	0.5	117	1.88	217.69	217.77
400	0.5	207	1.88	229.82	231.70
400	0.5	207	1.88	201.60	203.48
400	0.2	117	1.88	88.16	88.38
400	0.2	117	1.88	86.86	86.86
600	0.2	117	1.65	157.71	156.79
600	0.2	117	1.65	130.92	130.95
800	0.2	117	1.88	197.21	198.11
800	0.2	117	1.88	198.08	198.97
1000	0.2	117	1.64	218.38	218.58
1000	0.2	117	1.64	224.19	224.55

Table D.2: Dead reckoning results from all simulation trials. The ocean current speed is in knots and the ocean current heading is in degrees True. All other values are in meters.

Bibliography

[1] G. Kasparov and M. Greengard, *Deep Thinking: Where Machine Intelligence Ends and Human Creativity Begins*. New York, NY: PublicAffairs, 2017.

[2] B. Chandrasekaran and J. M. Conrad, "Human-robot collaboration: A survey," *SoutheastCon 2015,* 2015, pp. 1-8, doi: 10.1109/SECON.2015.7132964.

[3] COL G.R. Gilbert, USA and CDR M.K. Beebe, USNR, "United States Department of Defense Research in Robotic Unmanned Systems for Combat Casualty Care," Telemedicine and Advanced Technology Research Center, Fort Detrick, MD, Tech. Report. RTO-MP-HFM-182, 2010 [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA526596.pdf

[4] Deputy Secretary of Defense B. Work, "Reagan Defense Forum: The Third Offset Strategy," presented at Reagan Presidential Library, Simi Valley, CA, Nov. 7, 2015 [Online]. Available: https://www.defense.gov/News/Speeches/Speech/Article/628246/reagan-defense-forum-the-third-offset-strategy/

[5] United States Department of Defense, "Summary of the 2018 National Defense Strategy of The United States of America," Washington, DC, USA, 2018 [Online]. Available: https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf. [Accessed: Jan. 30, 2021].

[6] J.B. Lyons, K. Sycara, M. Lewis and A. Capiola, "Human–Autonomy Teaming: Definitions, Debates, and Directions," *Frontiers in Psychology,* vol. 12, May 2021, doi: 10.3389/fpsyg.2021.589585.

[7] H. Moravec, *Mind Children: The Future of Robot and Human Intelligence*. Cambridge, MA: Harvard University Press, 1990.

[8] *Autonomy in Weapon Systems,* DOD Directive 3000.09, United States Department of Defense, Washington, DC, USA, 2012 [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/300009p.pdf

[9] United States Defense Innovation Board, "AI Principles: Recommendations on the Ethical Use of Artificial Intelligence by the Department of Defense," Washington, DC, USA, 2019 [Online]. Available: https://media.defense.gov/2019/Oct/31/2002204458/-1/-1/0/DIB_AI_PRINCIPLES_PRIMARY_DOCUMENT.pdf. [Accessed: Jan. 30, 2021].

[10] *US Navy Diving Manual, Rev.7 Change A,* SS521-AG-PRO-010, Naval Sea Systems Command, Washington, DC, USA, 2018 [Online]. Available:

http://www.navsea.navy.mil/Portals/103/Documents/SUPSALV/Diving/US%20DIVING%20MANUAL_REV7_ChangeA-6.6.18.pdf?ver=mJHYtu_ILh4DQu3V45PjjQ-%3d%3d. [Accessed Aug. 30, 2021].

[11] L. Paull, S. Saeedi, M. Seto and H. Li, "AUV Navigation and Localization: A Review," *IEEE Journal of Oceanic Engineering,* vol. 39, no. 1, pp. 131-149, Jan. 2014, doi: 10.1109/JOE.2013.2278891.

[12] Đ. Nađ, F. Mandić, and N. Mišković, "Using Autonomous Underwater Vehicles for Diver Tracking and Navigation Aiding," *Journal of Marine Science and Engineering*, vol. 8(6), no. 413, 2020, doi: 10.3390/jmse8060413.

[13] N. Mišković et al., "CADDY project, year 3: The final validation trials," *OCEANS 2017 - Aberdeen,* 2017, pp. 1-5, doi: 10.1109/OCEANSE.2017.8084715.

[14] Đ. Nađ et al. "Towards Advancing Diver-Robot Interaction Capabilities," *12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019,* 2019, pp. 199-204, doi: 10.1016/j.ifacol.2019.12.307.

[15] E. Gallimore, J. Partan, I. Vaughn, S. Singh, J. Shusta and L. Freitag, "The WHOI micromodem-2: A scalable system for acoustic communications and networking," *OCEANS 2010 MTS/IEEE SEATTLE,* 2010, pp. 1-7, doi: 10.1109/OCEANS.2010.5664354.

[16] "borglab/gtsam," *GitHub, Inc.* [Online]. Available: https://github.com/borglab/gtsam. [Accessed: Dec. 21, 2021].

[17] M. Quigley et al., "ROS: an open-source Robot Operating System," *ICRA workshop on open source software,* 2009, vol. 3, no. 3.2, [Online]. Available: https://ai.stanford.edu/ ang/papers/icraoss09-ROS.pdf

[18] "ros_hat," *WHOI Community GitLab.* [Online]. Available: https://git.whoi.edu/jpel/ros_hat. [Accessed: Dec. 21, 2021].

[19] "Unmanned Underwater Vehicles," *Huntington Ingalls Industries, Inc.* [Online]. Available: https://tsd.huntingtoningalls.com/what-we-do/unmanned-systems/unmanned-underwater-vehicles/. [Accessed: Nov. 9, 2021].

[20] "REMUS 100 UUV Variants," *Huntington Ingalls Industries, Inc.* [Online]. Available: https://tsd.huntingtoningalls.com/what-we-do/unmanned-systems/unmanned-underwater-vehicles/remus100variants/. [Accessed: Nov. 9, 2021].

[21] J. Hirsch, "Swimmers tactile command navigation apparatus," U.S. Patent 3736551A, May 29, 1973 [Online]. Available: https://patents.google.com/patent/US3736551A/en

[22] J. Hirsch, "Apparatus and method for communication through the sense of touch," U.S. Patent 2972140A, Feb. 14, 1961 [Online]. Available: https://patents.google.com/patent/US2972140A/en

[23] J. Hirsch, "Tactile communication system," U.S. Patent 3246323A, Apr. 12, 1966 [Online]. Available: https://patents.google.com/patent/US3246323A/en

[24] R. Gill, "Portable acoustic tracking system for divers (PATS)," *IEEE Journal of Oceanic Engineering,* vol. 5, no. 3, pp. 204-209, July 1980, doi: 10.1109/JOE.1980.1145466.

[25] C.A. Dildy, Jr. and L.C. Adair, "Diver navigation system," U.S. Patent 4103279A, Jul. 25, 1978 [Online]. Available: https://patents.google.com/patent/US4103279A/en

[26] C.B. McLaren, "Acoustic navigation and diving information system and method," U.S. Patent 5331602A, Jul. 19, 1994 [Online]. Available: https://patents.google.com/patent/US5331602A/en

[27] J. Sattar and G. Dudek, "Where is your dive buddy: tracking humans underwater using spatio-temporal features," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2007, pp. 3654-3659, doi: 10.1109/IROS.2007.4399527.

[28] F. Mandić, I. Rendulić, N. Mišković and Đ. Nađ, "Underwater Object Tracking Using Sonar and USBL Measurements," *Journal of Sensors*, vol. 2016, 2016, doi: 10.1155/2016/8070286.

[29] A. G. Chavez, C. A. Mueller, A. Birk, A. Babic and N. Mišković, "Stereo-vision based diver pose estimation using LSTM recurrent neural networks for AUV navigation guidance," *OCEANS 2017 - Aberdeen,* 2017, pp. 1-7, doi: 10.1109/OCEANSE.2017.8085020.

[30] G. M. Goodfellow, J. A. Neasham, I. Rendulić, Đ. Nađ and N. Mišković, "DiverNet — A network of inertial sensors for real time diver visualization," *2015 IEEE Sensors Applications Symposium (SAS),* 2015, pp. 1-6, doi: 10.1109/SAS.2015.7133640.

[31] R.T. Kessel and R.D. Hollett, "Underwater Intruder Detection Sonar for Harbour Protection: State of the Art Review and Implications," NATO Undersea Research Centre, La Spezia, Italy, Tech. Report. NURC-PR-2006-027, Aug. 2006 [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA457007.pdf

[32] K. J. DeMarco, M. E. West and A. M. Howard, "Sonar-Based Detection and Tracking of a Diver for Underwater Human-Robot Interaction Scenarios," *2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester,* 2013, pp. 2378-2383, doi: 10.1109/SMC.2013.406.

[33] I. Kvasić, N. Mišković and Z. Vukić, "Convolutional Neural Network Architectures for Sonar-Based Diver Detection and Tracking," *OCEANS 2019 - Marseille,* 2019, pp. 1-6, doi: 10.1109/OCEANSE.2019.8867461.

[34] D. Chiarella et al., "Gesture-based language for diver-robot underwater interaction," *OCEANS 2015 - Genova,* 2015, pp. 1-9, doi: 10.1109/OCEANS-Genova.2015.7271710.

[35] G. Dudek, J. Sattar and A. Xu, "A Visual Language for Robot Control and Programming: A Human-Interface Study," *Proceedings 2007 IEEE International Conference on Robotics and Automation,* 2007, pp. 2507-2513, doi: 10.1109/ROBOT.2007.363842.

[36] B. Verzijlenberg and M. Jenkin, "Swimming with robots: Human robot communication at depth," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2010, pp. 4023-4028, doi: 10.1109/IROS.2010.5652751.

[37] N. Stilinović, Đ. Nađ and N. Mišković, "AUV for diver assistance and safety — Design and implementation," *OCEANS 2015 - Genova,* 2015, pp. 1-4, doi: 10.1109/OCEANS-Genova.2015.7271670.

[38] "Diver Held Systems," *Shark Marine Technologies Inc.* [Online]. Available: http://www.sharkmarine.com/products/diver-held-systems/. [Accessed: Feb. 12, 2021].

[39] "E-TAC: Bringing diver navigation to new heights," *Shark Marine Technologies Inc.* [Online]. Available: http://www.sharkmarine.com/products/diver-held-systems/e-tac/. [Accessed: Feb. 12, 2021].

[40] "Dive Tablet 2," *Shark Marine Technologies Inc.* [Online]. Available: http://www.sharkmarine.com/products/diver-held-systems/dive-tablet/. [Accessed: Feb. 12, 2021].

[41] "Navigator: Diver Held Sonar Imaging and Navigation System," *Shark Marine Technologies Inc.* [Online]. Available: http://www.sharkmarine.com/products/diver-held-systems/navigator/. [Accessed: Feb. 12, 2021].

[42] "DiveLog," *Shark Marine Technologies Inc.* [Online]. Available: http://www.sharkmarine.com/products/software/divelog/. [Accessed: Feb. 12, 2021].

[43] L.A. Hamme and V. Djapic, "Diver navigation, information and safety buoy," U.S. Patent 10597128B2, Mar. 24, 2020 [Online]. Available: https://patents.google.com/patent/US10597128B2/en

[44] "DiNIS: Diver Navigation and Imaging System," *Kenautics, Inc.* [Online]. Available: http://www.kenautics.com/pdf/KenauticsD2-booklet-A4.pdf. [Accessed Mar. 29, 2021].

[45] J. Vaganay, P. Baccou and B. Jouvencel, "Homing by acoustic ranging to a single beacon," *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158),* 2000, pp. 1457-1462, vol.2, doi: 10.1109/OCEANS.2000.881809.

[46] H.J.S. Feder, J.J. Leonard and C.M. Smith, "Adaptive Mobile Robot Navigation and Mapping," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650-668, 1999, doi: 10.1177/02783649922066484.

[47] J. Vaganay, J. J. Leonard, J. A. Curcio and J. S. Willcox, "Experimental validation of the moving long base-line navigation concept," *2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No.04CH37578),* 2004, pp. 59-65, doi: 10.1109/AUV.2004.1431194.

[48] M.F. Fallon, G. Papadopoulos, J.J. Leonard, N.M. Patrikalakis, "Cooperative AUV Navigation using a Single Maneuvering Surface Craft," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1461-1474, 2010. doi:10.1177/0278364910380760.

[49] F.B. Jensen, W.A. Kuperman, M.B. Porter, and H. Schmidt, *Computational Ocean Acoustics,* 2nd ed. New York, NY, USA: Springer, 2011.

[50] "Transducers, Arrays, and Towfish," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/transducers-arrays-and-towfish/. [Accessed: Nov. 11, 2021].

[51] M. Stojanovic, J. Catipovic, and J.G. Proakis "Adaptive multichannel combining and equalization for underwater acoustic communications," *The Journal of the Acoustical Society of America,* vol. 94, no. 3, Aug. 1998, doi: 10.1121/1.408135.

[52] D. B. Kilfoyle and A.B. Baggeroer, "The State of the Art in Underwater Acoustic Telemetry," *IEEE Journal of Oceanic Engineering,* vol. 25, no. 1, pp. 4-27, Jan. 2000, doi: 10.1109/48.820733.

[53] M. Stojanovic, J. A. Catipovic and J.G. Proakis "Phase-coherent digital communications for underwater acoustic channels," *IEEE Journal of Oceanic Engineering,* vol. 19, no. 1, pp. 100-111, Jan. 1994, doi: 10.1109/48.289455.

[54] M. Johnson, L. Freitag and M.Stojanovic, "Improved Doppler tracking and correction for underwater acoustic communications," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol.1, pp. 575-578, 1997, doi: 10.1109/ICASSP.1997.599703.

[55] M. Stojanovic, J. Catipovic, J.G. Proakis, "Coherent Communications over Long Range Underwater Acoustic Telemetry Channels," *Acoustic Signal Processing for Ocean Exploration. NATO ASI Series (Series C: Mathematical and Physical Sciences),* vol. 388, pp. 607-612, 1993, doi: 10.1007/978-94-011-1604-6_57.

[56] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski and K. Ball, "The WHOI micromodem: an acoustic communications and navigation system for multiple platforms," *Proceedings of OCEANS 2005 MTS/IEEE,* 2005, pp. 1086-1092, vol. 2, doi: 10.1109/OCEANS.2005.1639901.

[57] "Micromodem-2 User's Guide," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/wp-content/uploads/sites/20/2021/08/Micromodem2-Users-Guide-v1.7.pdf. [Accessed: Nov. 3, 2021].

[58] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research,* vol. 31, no. 2, pp. 216–235, 2012, doi: 10.1177/0278364911430419.

[59] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics,* vol. 6, no.1-2, pp. 1-139, 2017, doi: 10.1561/2300000043.

[60] F. R. Kschischang, B. J. Frey and H. -A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory,* vol. 47, no. 2, pp. 498-519, Feb 2001, doi: 10.1109/18.910572.

[61] K. Doherty, "Robust Non-Gaussian Semantic Simultaneous Localization and Mapping," M. S. thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, and Woods Hole Oceanographic Institution, Woods Hole, MA, USA, 2019 [Online]. Available: https://hdl.handle.net/1912/24859

[62] M.J.D. Powell, "A New Algorithm for Unconstrained Optimization," in *Nonlinear Programming,* J.B. Rosen, O.L. Mangasarian and K. Ritter, Eds. Cambridge, MA, USA: Academic Press, 1970, pp. 31-65, doi: 10.1016/B978-0-12-597050-1.50006-3.

[63] F. Dellaert and M.Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *The International Journal of Robotics Research,* vol. 25, no. 12, pp. 1181-1203, 2006, doi: 10.1177/0278364906072768.

[64] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes tree: An algorithmic foundation for probabilistic robot mapping," *Algorithmic Foundations of Robotics IX. Springer Tracts in Advanced Robotics,* vol. 68, pp. 157-173, 2010, doi: 10.1007/978-3-642-17452-0_10.

[65] "ros_remus," *WHOI Community GitLab.* [Online]. Available: https://git.whoi.edu/ros/ros_remus. [Accessed: Aug. 30, 2021].

[66] "ros_acomms," *WHOI Community GitLab.* [Online]. Available: https://git.whoi.edu/acomms/ros_acomms. [Accessed: Nov. 9, 2021].

[67] "ROS Noetic Ninjemys," *ROS.org.* [Online]. Available: https://wiki.ros.org/noetic. [Accessed: Nov. 9, 2021].

[68] "ROS Tutorials," *ROS.org.* [Online]. Available: https://wiki.ros.org/ROS/Tutorials. [Accessed: Nov. 9, 2021].

[69] E. Gallimore, R. Stokey and E. Terrill, "Robot Operating System (ROS) on the REMUS AUV using RECON," *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV),* 2018, pp. 1-6, doi: 10.1109/AUV.2018.8729755.

[70] T. Austin and R. Stokey, "Integrated Mission, Vehicle, and Sensor Control of the iPUMA AUV," Woods Hole Oceanographic Institution, Woods Hole, MA, USA, ADA552460, 2011 [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA552460.pdf

[71] J. A. Farrell, S. Pang, W. Li and R. Arrieta, "Chemical plume tracing experimental results with a REMUS AUV," *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492),* 2003, pp. 962-968, vol.2, doi: 10.1109/OCEANS.2003.178458.

[72] "pyAcomms," *WHOI Community GitLab.* [Online]. Available: https://git.whoi.edu/acomms/pyacomms. [Accessed: Nov. 9, 2021].

[73] "ADLE3800PC," *ADL Embedded Solutions.* [Online]. Available: https://www.adl-usa.com/product/adle3800pc/. [Accessed: Aug. 30, 2021].

[74] "AHRS-M2," *Sparton Navigation and Exploration.* [Online]. Available: https://www.spartonnavex.com/product/ahrs-m2/. [Accessed: Aug. 30, 2021].

[75] "sentinel-aug/ros/sparton-m2-ahrs," *GitLab.* [Online]. Available: https://gitlab.com/sentinel-aug/ros/sparton-m2-ahrs. [Accessed: Nov. 9, 2021].

[76] "Phins Compact Series," *iXblue.* [Online]. Available: https://www.ixblue.com/products/phins-compact-series. [Accessed: Aug. 31, 2021].

[77] "Why is latitude/longitude decimal precision important?" *Trimble Inc.* Feb. 27, 2019. [Online]. Available: https://support.pcmiler.com/en/support/solutions/articles/19000084902-why-is-latitude-longitude-decimal-precision-important-. [Accessed: Dec. 21, 2021].

[78] M. Grund, J. Partan, P. Koski and L. Freitag, "Synchronous Navigation with the Micro-Modem," Woods Hole Oceanographic Institution, Woods Hole, MA, USA, 2005 [Online]. Available: https://acomms.whoi.edu/wp-content/uploads/sites/20/2014/09/401004d-SPEC-Synchronous-Navigation-With-MicroModem-RevD.pdf

[79] S. Singh, "Flexible Data Packets Protocol: Version 1.8.5," unpublished.

[80] I. Masmitja, S. Gomariz, J. Del Rio, B. Kieft and T. O'Reilly, "Range-only underwater target localization: Path characterization," *OCEANS 2016 MTS/IEEE Monterey,* 2016, pp. 1-7, doi: 10.1109/OCEANS.2016.7761246.

[81] I. Masmitja et al., "Range-Only Single-Beacon Tracking of Underwater Targets From an Autonomous Vehicle: From Theory to Practice," *IEEE Access,* vol. 7, pp. 86946-86963, 2019, doi: 10.1109/ACCESS.2019.2924722.

[82] "Micromodem Deck Box," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/micro-modem-deck-box/. [Accessed Nov. 11, 2021].

[83] "Arrow 100 Submeter GNSS Receiver," *Eos Positioning Systems, Inc.* [Online]. Available: https://eos-gnss.com/products/hardware/arrow-100. [Accessed: Nov. 11, 2021].

[84] "GPS 18x Technical Specifications," *Garmin International, Inc.* [Online]. Available: https://static.garmincdn.com/pumac/GPS_18x_Tech_Specs.pdf. [Accessed: Nov. 11, 2021].

[85] "SBE 37-SM, SMP, SMP-ODO MicroCAT," *Sea-Bird Scientific.* [Online]. Available: https://www.seabird.com/moored/sbe-37-sm-smp-smp-odo-microcat/family?productCategoryId=54627473786. [Accessed: Nov. 22, 2021].

[86] "Micromodem Power Amplifier," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/micromodem-power-amplifier/. [Accessed: Nov. 19, 2021].

[87] "Micromodem Coprocessor," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/micro-modem-co-processor/. [Accessed: Nov. 19, 2021].

[88] "Modem Mainboard," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/modem-mainboard/. [Accessed: Nov. 19, 2021].

[89] "Bags," *ROS.org.* [Online]. Available: https://wiki.ros.org/Bags. [Accessed: Nov. 20, 2021].

[90] "MATLAB," *The MathWorks, Inc.* [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: Nov. 22, 2021].

[91] "ROS Toolbox," *The MathWorks, Inc.* [Online]. Available: https://www.mathworks.com/help/ros/index.html?s_tid=CRUX_lftnav. [Accessed: Nov. 20, 2021].

[92] "geopy/geopy," *GitHub, Inc.* [Online]. Available: https://github.com/geopy/geopy. [Accessed: Nov. 20, 2021].

[93] C.F.F. Karney, "Algorithms for geodesics," *Journal of Geodesy,* vol. 87, pp. 43-55, 2013, doi: 10.1007/s00190-012-0578-z.

[94] "GeoPy," *GeoPy.* [Online]. Available: https://geopy.readthedocs.io/en/stable/. [Accessed: Nov. 20, 2021].

[95] M. Pietrzak, "maurycyp/vincenty," *GitHub, Inc.* [Online]. Available: https://github.com/maurycyp/vincenty. [Accessed: Nov. 20, 2021].

[96] C. Veness, "Calculate distance, bearing and more between Latitude/Longitude points," *Movable Type Scripts.* [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html. [Accessed: Nov. 20, 2021].

[97] themaze, "optimcode/vincenty," *GitHub, Inc.* [Online]. Available: https://github.com/optimcode/vincenty. [Accessed: Nov. 20, 2021].

[98] L. Freitag, M. Grund, S. Singh and M. Johnson, "Acoustic communication in very shallow water: results from the 1999 AUV Fest," *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158),* 2000, pp. 2155-2160, vol.3, doi: 10.1109/OCEANS.2000.882253.

[99] H. Mei, H. Sun, Q. Jie, X. Kuai and G. Ye, "The clipping and nonlinear distortion compensation for underwater acoustic OFDM system," *OCEANS 2014 - TAIPEI,* 2014, pp. 1-6, doi: 10.1109/OCEANS-TAIPEI.2014.6964288.

[100] R. P. Stokey, L. E. Freitag and M. D. Grund, "A Compact Control Language for AUV acoustic communication," *Europe Oceans 2005,* 2005, pp. 1133-1137, vol. 2, doi: 10.1109/OCEANSE.2005.1513217.

[101] "Micromodem Software Development Box," *Acoustics Communication Group.* [Online]. Available: https://acomms.whoi.edu/micro-modem/micro-modem-software-development-box/. [Accessed: Aug. 31, 2021].

[102] P.A. Bromiley, "Products and Convolutions of Gaussian Probability Density Functions," Imaging Sciences Research Group, Institute of Population Health, School of Medicine, University of Manchester, Manchester, UK, Tina Memo No. 2003-003, 2014 [Online]. Available: http://in.ruc.edu.cn/wp-content/uploads/2016/09/The-product-and-convolution-of-guassian-distributions.pdf

[103] L. Freitag, P. Koski, S. Singh, T. Maksym and H. Singh, "Acoustic communications under shallow shore-fast Arctic Ice," *OCEANS 2017 - Anchorage,* 2017, pp. 1-5.

[104] "gtsam::ISAM2 Class Reference," *GTSAM.* [Online]. Available: https://gtsam.org/doxygen/a04096.html. [Accessed: Nov. 30, 2021].

[105] E.J. Terrill et al., "Project Recover: Extending the Applications of Unmanned Platforms and Autonomy to Support Underwater MIA Searches," *Oceanography,* vol. 30, no. 2, pp. 150-159, Jun. 2017, doi: 10.5670/oceanog.2017.237.

[106] N. H. Kussat, C. D. Chadwell and R. Zimmerman, "Absolute positioning of an autonomous underwater vehicle using GPS and acoustic measurements," *IEEE Journal of Oceanic Engineering,* vol. 30, no. 1, pp. 153-164, Jan. 2005, doi: 10.1109/JOE.2004.835249.