MuddSub: Exploring Vehicular Autonomy

Kehlani A. Fay James Clinton Eric Chen Chris Herrera Diego Herrera MuddSub Team Members

***Abstract*—This report presents the comprehensive scope of work carried out by MuddSub, a fully autonomous underwater robotics team at Harvey Mudd College. As a student-led and student-run organization, MuddSub aims to foster students' interests in robotics, autonomy, and research by employing experimental methods. This report highlights MuddSub's competition strategy, which encompasses fundamental principles, novel contributions to vehicular autonomy, and technical advancements through systems engineering. By developing two autonomous underwater robots, Crush and Alfie, MuddSub successfully demonstrates its ability to facilitate short-term real-world developments, long-term technical developments, and novel research initiatives using distinct physical platforms. This work begins by introducing Crush, a new experimental underwater robot, followed by improvements made to the team's initial robot, Alfie. The technical contributions discussed in this paper encompass various domains, including controls, navigation, state estimation, simultaneous localization and mapping (SLAM), mechanical design, printed circuit board (PCB) design, and computer vision. Each contribution is thoroughly detailed, providing insights into the research methods deployed. Additionally, both high-level and low-level design decisions are documented, underscoring the team's meticulous approach to system development. Through these efforts, this research paper presents MuddSub's commitment to advancing underwater robotics by leveraging student-led innovation, experimental methodologies, and the integration of diverse technical domains. The findings provide valuable insights to MuddSub's experimental work in the field of vehicular autonomy and robotics.**

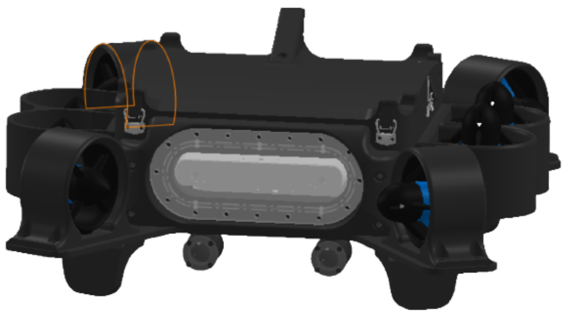


Fig. 1. Crush front profile.

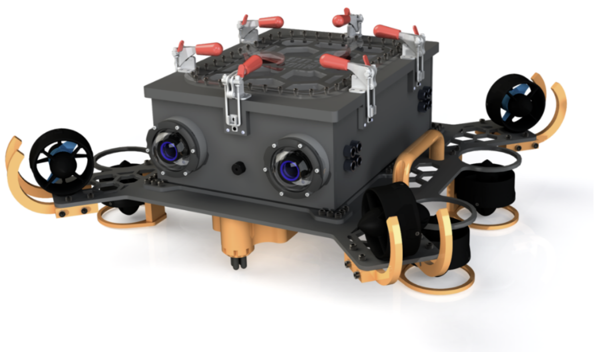


Fig. 2. Alfie.

I. Introduction

The domain of underwater autonomous robotics presents a distinctive opportunity for students and researchers to devise innovative approaches to surmount the demands of noisy and harsh environments. These challenges include accurate object segmentation and identification in cloudy, low lighting environments and extreme limitations in hardware due to pressure and high impact of unintended flooding. Furthermore, the scarcity of reliable grounding references in external stimuli poses significant hurdles to achieve accurate localization and mapping. Additionally, the influence of ocean currents and turbidity can lead to undesired drift with often no method of detection beyond internal sensors. To succeed in this domain, students must learn and utilize advanced research methodologies that push the boundaries of their understanding, enabling them to unravel the fundamental principles of autonomy and undertake research endeavors that address future advancements and unresolved challenges.

II. Competition Strategy

To address these fundamental challenges, MuddSub has undertaken the development of two distinct underwater robots, Alfie and Crush. The original robot, Alfie, was purposefully designed with simplicity in mind to emphasize reliability and encourage real-world deployment sustainability. Alfie's design incorporates a factor of four safety margins for all hardware components, utilizes off-the-shelf components for efficient electrical signal filtering, and adopts a straightforward software framework. MuddSub aims to make Alfie an approachable, easy to use robot for incoming students and to test new ideas through a modular software and electrical framework.

Conversely, Crush, the latest addition to MuddSub's robotic fleet, serves as an experimental platform that facilitates long-term research and development of novel methodologies through fundamental design modifications. Crush represents a highly experimental undertaking, featuring a reduced size, custom-designed printed circuit boards (PCBs), integration of stereo cameras, and the incorporation of cutting-edge sensors such as Doppler Velocity Log (DVL) and hydrophones. Additionally, Crush implements an experimental software framework that integrates newly developed research methods specific to underwater robotics. Through these two distinct robots, MuddSub aims to address both short-term success and long-term research objectives, catering to the multifaceted demands of underwater robotics exploration. Crush allows students to advance to complex research ideas and ensure students can test cutting edge methods.

Since our first year in 2019, MuddSub has focused on reliability and building within inherent constraints. Building a simple robot has allowed for easily onboarding students to the basics of autonomy, quick repair after failure, and let students try how ideas compare to the real world. MuddSub is organized into fourteen key subteams which develop clear functional needs of the robot. Each specializes in a challenge in underwater robotics with students who share interdisciplinary roles between teams to ensure easy transfer of data and that challenges are solved between subfields. Student leads evaluate functional performance of the robot alongside their subteam and determine future directions of development. In addition, students are encouraged to pursue projects of interest related to autonomous robotics alongside peers, leading to the development of several of these teams. MuddSub aims to enable as much learning and exploration as possible through encouraging peer learning, exploration beyond immediate performance tasks, and funding experimental mini projects.

To accomplish its goals, MuddSub aims to pursue three main objectives: short-term real-world developments, long-term technical developments, and novel research initiatives.

III. Design Strategy

A. Designing a New Robot: Crush

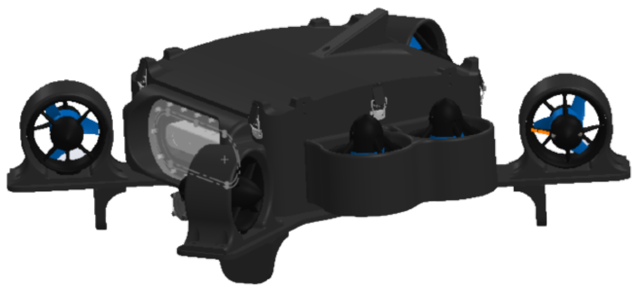


Fig. 3. Crush front profile.

In order to facilitate experimentation and enable the testing of advanced methodologies, Crush was developed with significant fundamental modifications when compared to Alfie. One notable distinction is the incorporation of two wide windows in Crush, allowing for the deployment of a diverse array of stereo cameras. These have enabled students to explore visual sensor fusion between two stereo camera systems to build complex discrete maps. With deployed cross comparisons of techniques, fusion between multiple cameras, IMU, gyro, and depth, this has lead to improved maps. An example of the team's tested IMU to Visual and depth data is included.

Crush is approximately half the size of its predecessor, Alfie, which not only promotes a compact form factor but

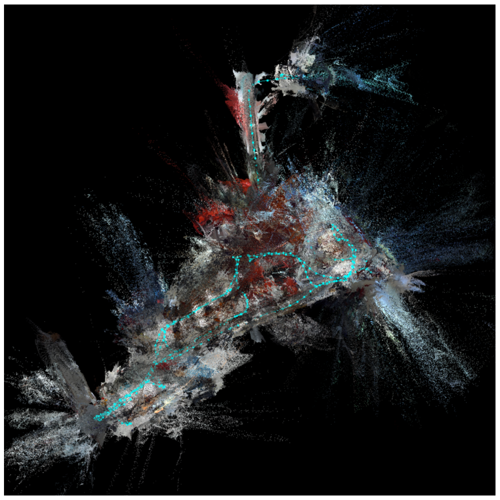


Fig. 4. Fused IMU, depth, RGB SLAM result from testing sensor fusion accuracy. Map of estimated path taken is in blue.

also necessitates the design of small, modular printed circuit boards (PCBs) dedicated to power distribution, noise filtering, and signal processing. The reduced size and weight of Crush have been carefully considered to ensure safety during pool testing, enabling enhanced maneuverability and operational versatility in tight underwater environments. By optimizing the design and employing lightweight carbon fiber materials, Crush exhibits increased agility and improved responsiveness. The body of Crush is fabricated from a shelled out aluminum frame with carbon fiber printed sensor and motor mounts.

For increased fine-grained control, Crush uses a Linear Quadratic Regulator (LQR) with motion matrices found through data collected in fluid simulation and real world force to motion testing [1]. This helps account for drift. The team also experimented with Model Predictive Control but found it was very computationally intensive and requires online computation at each time step for optimization.

$$J = \int_0^\infty -\frac{d}{dt}\left(\mathbf{x}^\top \mathbf{S}\mathbf{x}\right) dt = -\mathbf{x}^\top \mathbf{S}\mathbf{x}\Big|_0^\infty = \mathbf{x}^\top(0)\mathbf{S}\mathbf{x}(0) \quad (1)$$

$$\mathbf{A}\top\mathbf{S} + \mathbf{S}\mathbf{A} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{S} + \mathbf{Q} = \mathbf{0} \quad (2)$$

$$\text{optimal} \quad \mathbf{K}^* = \mathbf{R}^{-1}\mathbf{B}^\top\mathbf{S} \quad (3)$$

The team also experimented with learned navigation through reinforcement learning. However, we found this was not reliable enough with noisy and extremely limited real world data. Experiments were run in Gazebo for passing through the gate with input images and output PWM values in sim. These were not found to be successful largely due to latency.

B. Electrical : Crush and Replacing Alfie Components

The MuddSub electrical team focused on learning, bring-up, and future development this past year. Initial efforts focused on learning how Alfie's subsystems work, to bring incoming team members up to speed on the current state of the electronics. There, the team championed a collaborative effort to teach

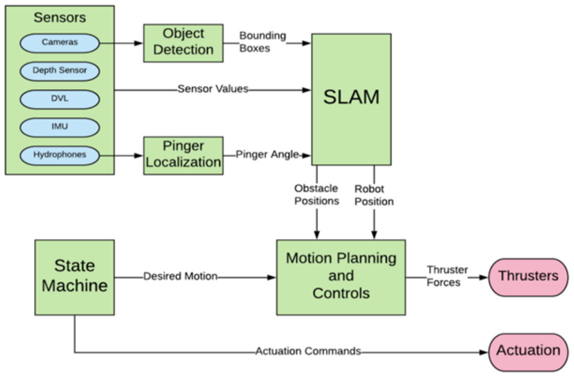


Fig. 5. Overview of the whole system.

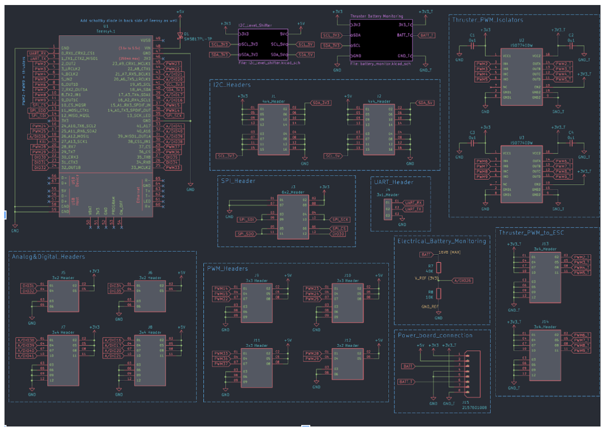


Fig. 6. Schematic for signal board.

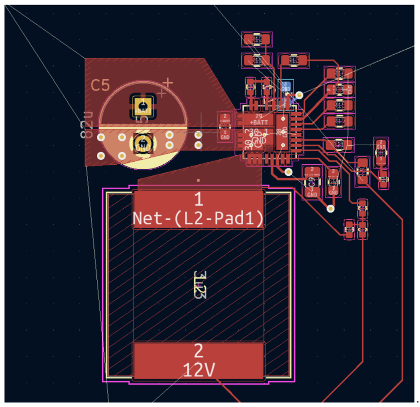


Fig. 7. Example of regulator board layout.

upcoming engineers the basics of communication protocols, BLDC motor driving, schematic capture, PCB layout, breadboard prototyping, and more. This resulted in a group of students who had the drive and some background knowledge to pursue the larger goal for the year: revamping the internals of the robot with a vision of implementing them in Crush.

Further efforts focused on designing the signal and power distribution boards for Crush. Previous electrical infrastructure for Alfie has typically been large, clunky, and undocumented, making it hard for newer members to contribute meaningfully to the project. To aid these goals, board designs were done carefully by justifying design decisions and having the documentation to back it up. The boards were designed with simple goals in mind: the boards must be small and easy to use.

The signal board is the heart of the low-level software for the robot: it hosted the Teensy 4.0 microcontroller, signal isolation to prevent ground loops, all the ports and I/O the robot would ever need, and a battery monitoring circuit. This board was primarily designed by the team's newer members, as the intention was to have them build a simpler system and learn as much as possible. The signal board handles powering the thrusters, sensors like the depth sensor, and servos to power grippers, torpedo, and marker actuators. KiCad was used for the board design, as it is capable yet accessible to learners of EDA software. Board layout is in progress, as supply chain issues forced the team to change critical components such as digital isolators multiple times throughout the design process.

The power distribution is responsible for giving appropriate power to every component inside of Crush. This includes the Jetson AGX Xavier computer, DVL, servo motors, and other power-hungry systems. This board takes unregulated power from the 4S LiPo battery and splits it into 12V, 9V, and 5V rails for the robot's systems. The 12V and 5V rails need to be capable of high power, as the Jetson and DVL run on the 12V rail, and all servos run off of 5V. The 9V rail uses less power, so the design requirements are less strenuous. The team estimated that having both the 12V and 5V rails be capable of withstanding 10A would be sufficient for the robot's needs. This meant that the board required special care in design due to high power draw. The team opted to use buck regulators, as linear regulators would dissipate too much heat by stepping 16V down to 5V at 10A. The increase in efficiency of using buck regulators comes with the trade-off of design ease, as signal integrity and EMI have much stronger effects in buck versus linear regulators. The team could have opted for buying existing regulator boards, but they were too large and expensive. However, the main reason for doing the design from scratch was that designing high power buck regulators would be an excellent learning experience for all of those involved, as any circuit designer will benefit from having power regulation knowledge.

Special care will be taken when testing this board, as it powers expensive components. The team plans to stress-test the board under typical and extreme loading conditions, which include pulling the maximum current for extended periods of time as well as testing quickly switching loads. The output voltage will be closely monitored in these scenarios to ensure that high transient voltages or over-current events do not damage the systems. The board layout requires more care than the signal board, as buck regulators require tight component placement to minimize parasitic inductance and good routing to ensure that feedback loops do not become unstable. An example of the layout for the 12V regulator is shown below.

C. Marker and Gripper : Alfie and Crush

The marker and gripper team focused on developing systems to be used in competition, developing mechanical design skills and creating immediate implementations.

The marker team experimented with different shapes for the marker itself with the goal of finding the optimal shape for accurate positioning. The team considered spheres and torpedo-type shapes of various designs, but eventually settled on dodecahedrons. Tests showed that dodecahedrons went down the water with the straightest trajectory and had the least chance to roll off if they landed on the edge of the target. The marker holder was designed to be as small as possible to hold both dodecahedrons on top of each other so they drop in the same place. The markers are held by a bar, which is rotationally actuated by a single servo to release the markers. The servo is programmed to move at the highest velocity to minimize the impact on the marker's intended trajectory. The team's next steps include testing to make the markers as dense as possible, so that any currents in the water do not affect the trajectory as much. The mechanical design of the arm to hold the markers will also be further investigated in hopes of improving the performance.

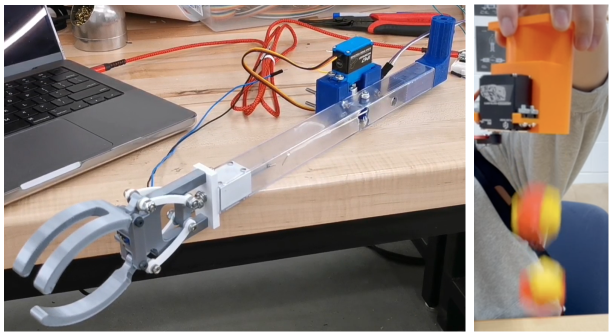


Fig. 8. Picture of gripper and marker prototypes.

The gripper team focused on firmware development and mechanical design. Many spring stiffnesses were tested for the gripper, as a spring is used to hold the claws back in their "holding" state. A servo then pushes against the springs to open the claws. The gripper team also worked to develop firmware for the gripper. Near the end of the year, many of the components fell apart, so future work will involve investigating how the gripper mechanism can be made more robust.

D. Simultaneous Localization and Mapping (SLAM) : Alfie

This year, the team continued development on our SLAM sub-system, after first implementing it in 2021. SLAM utilizes sensor inputs in order to simultaneously perform localization, determining where the robot is relative to its surroundings, and mapping, determining where a robot's surroundings are relative to it. This sub-system is essential to our competition strategy, as it enables short-term success in competition by aiding our navigation sub-system, allows for continued development of our robot's autonomy in the future with enhanced localization and mapping capabilities, and provides opportunities for research work on SLAM methods.

We have continued with the algorithm we identified in 2021, FastSLAM 2.0 [2] [3]. Previous versions of Alfie have primarily used vision for sensing, but with Alfie we also have a Doppler Velocity Log (DVL), an Inertial Measurement Unit (IMU), as well as a depth/pressure sensor. FastSLAM is a good choice over visual SLAM methods, as it has allowed for the extensibility we now make use of in our current design iteration, and will allow for us to make use of the stereo vision capabilities we have been working on.

There are two main parts of FastSLAM [4]: The first is the motion model which describes how the robot's state evolves over time:

$$p(s_t|u_t, s_{t-1}) = h(u_t, s_{t-1}) + \delta_t. \tag{4}$$

The second is the measurement model which describes how the sensors generate measurements given the state of the robot and environment:

$$p(z_t|s_{t-1}, \Theta, n_t) = g(s_t, \theta_{n_t}) + \varepsilon_t. \tag{5}$$

For development of this system, the team worked to move the system from design choice to implementation. We worked to develop a prototype implementation of the algorithm in both Python and C++. In addition, we have worked to produce a final implementation of the algorithm for use in Alfie, with hopes of making use of it in competition.

Finally, we have worked to further our understanding of the fundamentals of SLAM. Since the subteam contained experienced team members, they dedicated time to teaching new students what they had learned about state estimation. The new students were taken through a SLAM crash course, learning about probabilistic robotics algorithms such as Kalman filters, particle filters, and grid mapping. They also implemented particle filter localization for themselves, and contributed to the implementation of FastSLAM.

E. Navigation and Controls : Alfie

The navigation sub-system is responsible for planning the robot's trajectory. This sub-system works closely with the state machine and SLAM sub-systems. The state machine tells the robot where it should go in order to accomplish its current task, and the SLAM sub-system tells the robot where it currently is in its environment. Navigation combines these pieces of information to produce a series of poses for the robot to follow to get from a starting location to some goal location. The Navigation subteam implemented the A* algorithm for motion planning. It takes as its input a grid map showing what parts of the space are occupied and outputs a path along that grid that the robot can safely take. This past year, we experimented with different heuristic functions. The subteam also wrote code to generate trajectories that have specific shapes. For example, it is useful to have the robot move in a sine wave in order to take measurements of obstacles at different angles to gather increased information for SLAM. Another example is planning a circular path around some obstacle in order to allow the cameras to view it from many different angles. For controls, Alfie uses PID on its thrusters to achieve desired thrust.

F. State Machine: Alfie

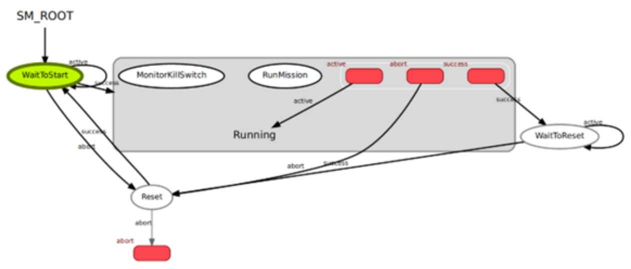


Fig. 9. State machine diagram

The state machine is responsible for directing robot behavior during tasks and emergencies. The team has implemented several important aspects of the state machine in smach. The state machine starts in the "WaitToStart" state then provides into the task layer when prompted. Each task in the competition shares three steps, which are represented by three states: "SearchForTarget," "GoToTarget," and "TaskSpecific." This modular structure allows for code reusability for different tasks. Specifically, only the "TaskSpecific" state needs to be modified for each task. Finally, the task layer is used concurrently with the "MonitorKillSwitch," which responds to emergency shutdown situations.

G. Computer Vision: Alfie

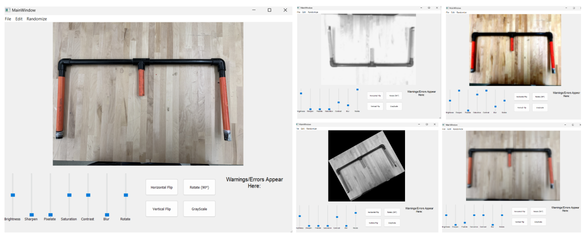


Fig. 10. A GUI for testing object detection models using data augmentation on demand.

Alfie runs YOLOv3 [5] based object detection. The object detection model identifies whole game pieces as well as their internal parts. For example, it detects both the Gate as well as the middle bar of the Gate. Detecting main game pieces allows Alfie's to find landmarks from afar while the interior detections allow for fine grain control at near range. The module publishes confidence, classes and bounding boxes which are used by SLAM and the state machine. To implement this system, the team fabricated sample game pieces and created a labeled dataset. A YOLOv3 model was trained on this dataset using Google Colab. The current model is able to detect gates but produces many extra incorrect detections. We hope to continue tuning this model to improve its precision.

A next step for object detection is to transition to object tracking using Deep Simple Online and Realtime Tracking (SORT) [6]. The envisioned object tracking pipeline will start with YOLOv3 to detect bounding boxes, a feature extractor to extract appearances from objects in the bounding boxes, and finally the SORT algorithm to track similar objects in subsequent frames.

IV. Testing Strategy

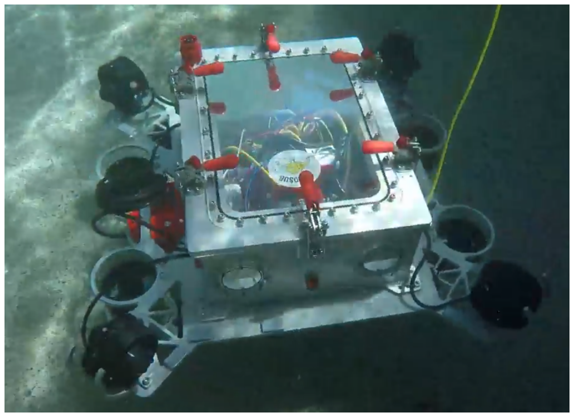


Fig. 11. Alfie underwater.

A. SLAM

The SLAM subteam validated the implementation of the FastSLAM 2.0 algorithm against the MRCLAM dataset [7], a dataset published by the University of Toronto. It consists of measurements taken by wheeled robots with cameras to identify landmarks, as well as groundtruth location data. The estimated locations of the robots and landmarks output by the FastSLAM 2.0 algorithm could be compared against the true locations recorded by external sensors.

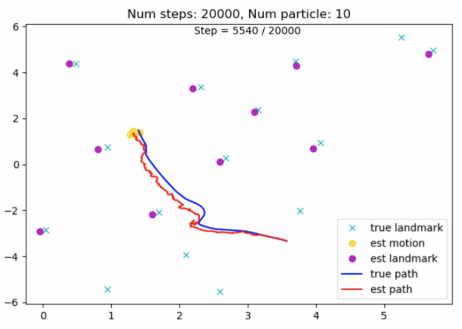


Fig. 12. SLAM results on MRCLAM dataset.

This dataset provided an easy way to validate the algorithm's implementation and was vital in testing and debugging the code. It is also much easier to do so with a dataset rather than the robot because it allows for testing of the SLAM code before the rest of the systems necessary for the code to work are in place. One large downside though is that the dataset does not accurately represent the 3D underwater environment present in the RoboSub competition.

B. Navigation

Mock grid maps were created for the purposes of testing the navigation subsystem. The grid maps were populated with obstacles and different heuristic functions were tested with them. Certain heuristic functions resulted in less searching before a path was found.

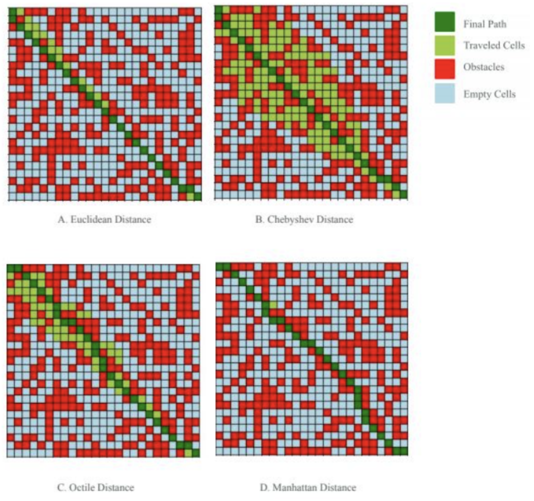


Fig. 13. Navigation comparisons with variable distance functions.

C. Controls

The control system consists of two components – PWM and PID. To test the PWM, we implemented a rqt GUI that publishes thruster values (0-100) to each motor individually. To test the PID offline, we provide an endpoint to the controls service. We create a mapping from thruster values to linear and angular velocities. Using these velocities, we can calculate where we expect the robot to be and publish this information to the controls system. Metrics we use are the rise time, settling time, overshoot, and stable state errors.

D. Computer Vision

Initially, the computer vision model's IoU, precision, and recall are tested on a test dataset of 100 labeled images. We further test our models using a data augmentation gui. This method provides more control to the user. Finally, the models are tested on robot in an underwater environment. The camera footage is recorded in rosbags for offline testing.

E. State Machine

The state machine is tested with scripts that mock sensor and subsystem outputs, such as detected objects and robot position. These messages forces state machine into predicable states. Finally, we verify the state machine behavior through logs.

F. Hardware

All hardware testing will be done before any other components are tested, to ensure that issues we encounter are not hardware-related.

The most important thing to test for an underwater robot is its waterproofing. When testing our O-ring seals, we take out all electrical components and submerge the robot into the water, then ensure no flooding occurs. To test our electrical system, we test each sensor individually using a microcontroller or computer, depending on what hardware interface the sensor uses. In addition, basic actuation of mechanisms will be tested with a microcontroller.

G. Overall System

In order to ensure that all subsystems work together, many rounds of testing are required. The first round focuses on making sure that all robot hardware is operational underwater. This includes testing motors and collecting all sensor data in ROS bag files, which is essential for debugging later. The computer vision subsystem also requires testing at this point, which is made easier when real data is collected by the robot. The next round of testing focuses on state estimation. Getting the SLAM subsystem working early is essential because nearly every other subsystem relies on it in some way. Without an estimate of the robot's location and current state, it is impossible to plan what tasks should be performed and impossible to navigate reliably. After state-estimation comes controls, i.e. the subsystem that enacts the commands given to the robot. Some control system is necessary in order for the navigation subsystem to work. Finally, the last subsystem to be integrated and tested is the state machine. This subsystem relies on all other subsystems to varying degrees. However, it is possible to forego some of the more complex navigation algorithms for path planning for the purposes of testing the state machine. For example, the robot could just be commanded to follow a straight line until it reaches the destination for the next task.

Acknowledgment

The authors would like to acknowledge the Shanahan Grant for providing funding and support, enabling the development of a new autonomous underwater robot within MuddSub. This grant, along with a second grant for creating an introductory course on autonomy through DuckieBots, has played a crucial role in fostering students' passion for robotics and autonomy. The authors express deep gratitude to the Shanahan Grant for their commitment to scientific exploration and education. These grants in combination with support from the Hartman Grant have allowed MuddSub to gain stereo cameras, create a research course, and build a new robot. Additionally, the authors would like to thank the faculty, staff, and team members at Harvey Mudd College for their guidance, mentorship, and collaborative efforts. In particular, we wish to thank our advisor, Professor Zachary Dodds, for his guidance, Xavier Walter, for his assistance with troubleshooting, and Drew Price, for his help in manufacturing efforts. The authors are also grateful to the research participants and individuals who provided valuable feedback and support throughout the project. The contributions of all these individuals and organizations have been instrumental in the success and impact of this work.

References

[1] A. Bright, ”E102 Modern Control Engineering: Class Notes.

[2] Michael Montemerlo, Sebastian Thrun, Daphne Roller, and Ben Wegbreit. 2003. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In Proceedings of the 18th international joint conference on Artificial intelligence (IJCAI'03). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1151–1156.

[3] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. 2002. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In Eighteenth national conference on Artificial intelligence. American Association for Artificial Intelligence, USA, 593–598.

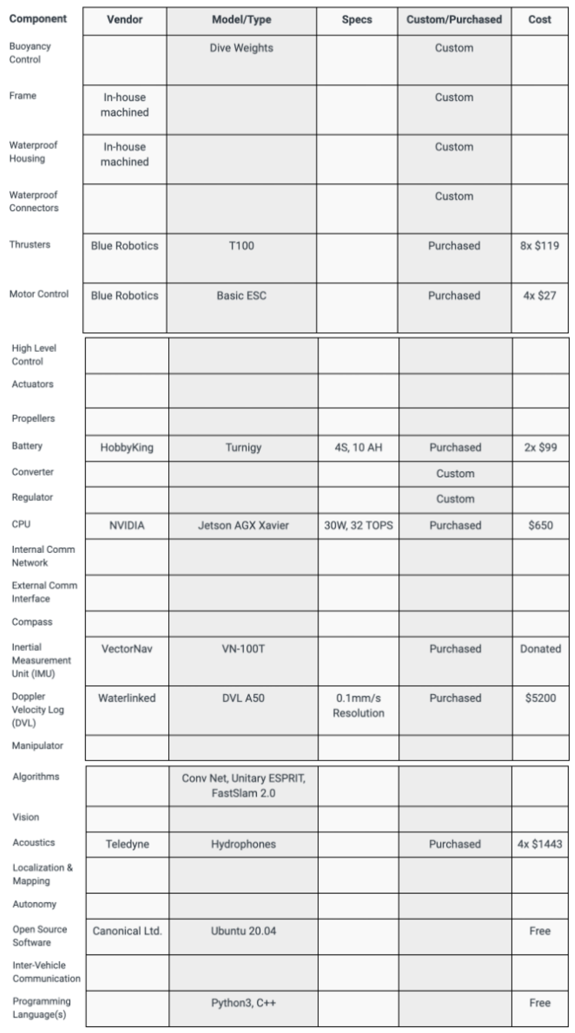
[4] Sebastian Thrun, Wolfram Burgard, Dieter Fox. Probabilistic robotics. Intelligent robotics and autonomous agents, MIT Press 2005, ISBN 978-0-262-20162-9

[5] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." arXiv, 2018. doi: 10.48550/ARXIV.1804.02767.

[6] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric." arXiv, 2017. doi: 10.48550/ARXIV.1703.07402.

[7] Leung K Y K, Halpern Y, Barfoot T D, and Liu H H T. "The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset". International Journal of Robotics Research, 30(8):969–974, July 2011.

Appendix A



Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost
Buoyancy Control		Dive Weights		Custom	
Frame	In-house machined			Custom	
Waterproof Housing	In-house machined			Custom	
Waterproof Connectors				Custom	
Thrusters	Blue Robotics	T100		Purchased	8x $119
Motor Control	Blue Robotics	Basic ESC		Purchased	4x $27
High Level Control					
Actuators					
Propellers					
Battery	HobbyKing	Turnigy	4S, 10 AH	Purchased	2x $99
Converter				Custom	
Regulator				Custom	
CPU	NVIDIA	Jetson AGX Xavier	30W, 32 TOPS	Purchased	$650
Internal Comm Network					
External Comm Interface					
Compass					
Inertial Measurement Unit (IMU)	VectorNav	VN-100T		Purchased	Donated
Doppler Velocity Log (DVL)	Waterlinked	DVL A50	0.1mm/s Resolution	Purchased	$5200
Manipulator					
Algorithms		Conv Net, Unitary ESPRIT, FastSlam 2.0			
Vision					
Acoustics	Teledyne	Hydrophones		Purchased	4x $1443
Localization & Mapping					
Autonomy					
Open Source Software	Canonical Ltd.	Ubuntu 20.04			Free
Inter-Vehicle Communication					
Programming Language(s)		Python3, C++			Free