RoboBoat 2023: Technical Design Report

Autoboat, Cornell University
Ithaca, New York, United States

I. Abstract

Autoboat is a Cornell University student project team which seeks to innovate novel maritime technology capable of complex autonomous behavior. We feature a trimaran-style boat with fiberglass hulls. The boat is equipped with a ZED 2i camera to collect environmental data and the Jetson Xavier NX and Arduino Mega microcontrollers to perform computations and control hardware. It is powered by 14.8 V batteries. Our system design prioritizes efficiency and safety while still maximizing efficacy. Our sleek design, advanced algorithms, creativity, and comradery will lead to success in our first in-person competition at RoboBoat

II. Competition Strategy

A. Approach

As a new team, our primary goal this year has been to establish a strong and flexible foundation to build upon in future years. We have learned from last year in which we competed virtually, and have since tried new techniques and experimented with different methodologies to learn what works best for our team. Our other high-level goals on Autoboat are to always move forward, become better engineers, and be proud of what we can accomplish in such short spans of time.

With respect to the competition, our strategy is to attempt all tasks except the Ocean Clean Up. To maximize our potential for points, we prioritized the optimization of our navigational abilities, catering software to the Panama Canal, Magellan's Route, Northern Passage, Beaching, and Explore the Coral Reef challenges. However, as there are points available for attempting all tasks, we also dedicated time to ensure our boat has the mechanical capabilities to shoot and aim both a water gun for the Fountain of Youth challenge and a Skee-Ball cannon for the Feed the Fish challenge.

B. Trade-off Studies

We spent a large portion of our time in the research phase, prioritizing knowledge accumulation over rushed decisions. Our research allowed us to weigh the trade-offs of various design choices such as switching from a catamaran to a trimaran hull design, utilizing a ZED 2i camera in place of additional sensors, adopting a ROS framework, incorporating a Pure Pursuit and PID controller, and more.

As our system's complexity increased, we modified our team's structure accordingly, delegating tasks to subgroups within each subteam and maintaining a clear leadership hierarchy.

III. Design Strategy

A. Mechanical Aspects

<u>Hulls</u>

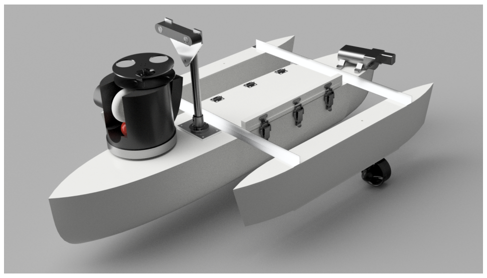

Figure 1: Assembly with skeeball, camera mount, main hull and amas.

The goal of our hull design is to maximize the stability of the boat and reduce the movement of the camera to improve image quality. For this reason, we opted for a trimaran design with a displacement main hull (Figure 1). A trimaran provides extra stability with the addition of the amas. If the boat starts to roll in the water from a turn, one of the amas will dip into the water and provide a restoring buoyant force upwards, which correlates to a righting moment on the main hull [1]. During prototype testing, it was found that we had an issue with the trim angle as the boat approached hull speed. It is important to keep the trim angle to a minimum so the camera remains flat relative to the horizon.

To fix this issue and reduce planing, we modified the new main hull to act as a displacement hull instead of a semi-planing hull. This is dictated by the Froude number, which depends on the waterline length (LWL). With a LWL of 54in, the Froude number comes out to about 0.40, pushing the hull to be a displacement hull. In order to reduce the Froude number, the water length of the hull needed to be maximized. Since the main hull is a displacement hull, it is important that we do not exceed the hull speed, or else the boat could start to climb up the back of its own bow wave. Importantly, this would also increase the trim angle, so we do not want to exceed this speed [1]. The calculated waterline is 3in from the bottom of the boat, meaning the waterline length is 54in, and the hull speed is 2.84knots or 1.46m/s.

The amas are long and skinny to optimize the volume of water displaced while keeping them hydrodynamic. Ideally, they should barely be in the water until the boat starts to roll. This maximizes the amount of extra volume that could be displaced while not inducing extra drag. By having the amas skinny, we can also extend them further from the boat and increase the righting moment on the main hull [1]. However, the closer the amas are the easier it is to navigate the buoys.

Hull Manufacturing

The general idea for the manufacturing plan of the main hull was to cut out profiles of the boat to use as guides to sand down a foam mold. We were inspired by the methods in [2].

After the foam was sanded to shape, a fiberglass layup was done on the mold. Next, fiberglass was denibbed and lightly sanded before applying a light body filler. The body filler was then thoroughly sanded down and the foam was removed from the fiberglass. Wooden ribs were inserted for structural support and the wooden hull top was secured to the top of the boat. A white gel coat was applied to the hull for a clean finish.

The amas were created in a similar fashion, except the foam did not need to be removed after the fiberglass layup, and we used more epoxy instead of a light body filler to obtain a smoother finish.

Simulations of Design

The software ANSYS Fluent was used to analyze the fluid flow around the boat. Due to the 512k cell limit imposed by the student license, it is important to create a uniform mesh that is not computationally expensive. To achieve this, multiple body of influence meshes were used around the free surface and hull geometry in order to create a more complicated mesh near the areas of interest (Figure 2) [3]. It is important to note that a facesize mesh and a curvature mesh were then used on both the main hull and the amas to accurately capture the curvature of the hulls (Figure 3).

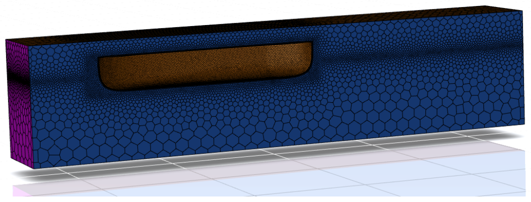

Figure 2: Mesh of entire domain with complicated mesh near free surface and hull.

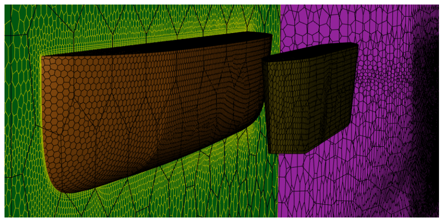

Figure 3: Inside view of the domain showing the main hull and ama mesh.

For the solution, the most important thing to look at was the velocity vectors of the fluid flow around the boat (Figure 4). The velocity of the simulation was set to the hull speed of the boat, which was 1.46m/s. This solution shows that water flows smoothly around the amas and the main hull, with an area of low turbulence by the stern of the main hull. In future iterations, the main hull should be optimized so the velocity changes gradually around the main hull. The sudden change in velocity near the stern should be minimized to maximize the pressure recovery [1].

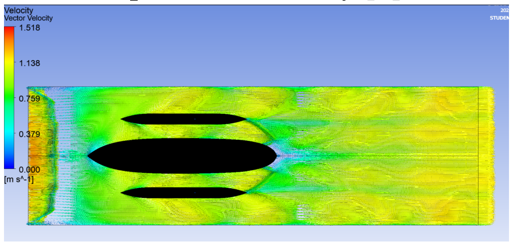

Figure 4: Velocity vectors of fluid flow around the boat.

Skeeball Launcher

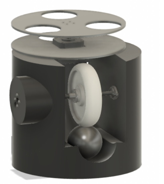

Figure 5: Skeeball Assembly.

The skeeball mechanism utilizes a single vertical rubber flywheel to launch the ball for the shooter task (Figure 5). The wheel, attached to a metal axle, is powered by a F2838-350kv-Z underwater DC motor. For the indexer, a gravity-powered turntable feeding mechanism is used to store the game pieces. It is powered by a Nema 17 Bipolar 1.8deg 26Ncm stepper motor which allows for the balls to be fed to the flywheel incrementally. The ball then rolls down the ramp and is ejected from the system. The skeeball launcher system has a rotating base powered by a Nema 23 Bipolar 1.8deg 1.26Nm stepper motor which allows it to rotate horizontally for aiming.

Cooling System

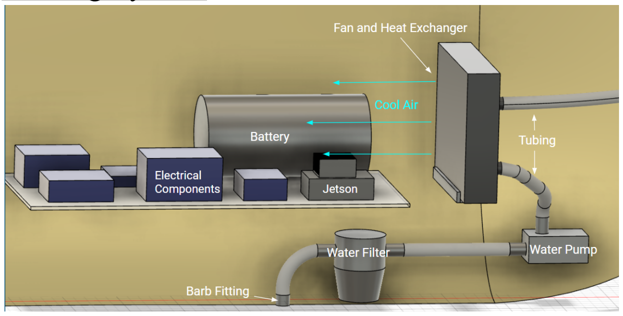

Figure 6: Cooling system assembly next to the electrical bay.

The electrical bay temperature is regulated via an open water cooling system (Figure 6). In an open water system, water is pumped in from the lake and passed through a heat exchanger to cool the air blowing over the electrical components. Using the environment to its advantage, the cooling system will effectively regulate the electrical bay temperatures using as little as 10W of power. This cooling system is also small and light, weighing about 1.5lbs and taking up minimal space in the back of the boat hull.

Propulsion

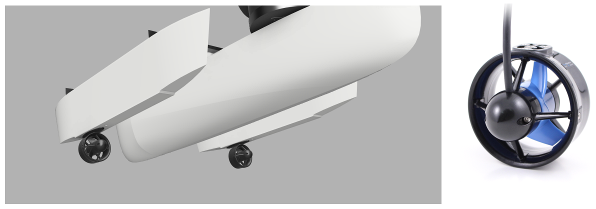

Figure 7: Blue Robotics T200 Thruster and mounting location underneath the amas.

The boat uses dual T200 thrusters that are mounted on the back of each ama (Figure 7). With two thrusters set 2 feet apart, the boat maneuvers with differential steering; capable of moving forwards, backwards, spinning in place, and any combination of the three. Thrusters are controlled via Electronics Speed Controllers and Pulse Width Modulation (PWM) through the Arduino.

B. Electrical Aspects

Sensor Hardware

The core of our sensor system is the ZED 2i camera from Stereolabs. This camera provides critical environmental information. Key features include its dual, wide-angle cameras and built-in 9-DoF IMU. Combined with the provided SDK, these features result in custom object detection, depth perception from 0.2 meters to 20 meters, and 3D position tracking and mapping. This detailed awareness of its surroundings enables the boat to make task completion decisions.

The boat is also equipped with sensors to monitor battery health and evaluate if the battery is safe to operate.

Computer Hardware

The primary on-board computer is a Jetson Xavier NX, where all the sensor and control information is received and processed. This computer has advanced AI performance, detailed graphics rendering (384-Core NVIDIA Volta GPU), 8 GB of RAM, and impressive computing power (6-Core NVIDIA Carmel ARM v8.2 64 bit), which means it is perfect for interfacing with the ZED 2i and supports all of our CV and AI requirements. The Jetson communicates with an Arduino Mega microcontroller through UART serial in order to control low-level hardware, which includes the T200 thrusters, stepper motors for the aiming mechanism, and motors for the water pump and skeeball flywheel. The Arduino communicates back to the Jetson with RC information, battery activity, and the status of the boat in general (e.g. killswitch activated or any errors). The Jetson Xavier NX has an installed wifi module so it can communicate with our base station through ROS.

Power System

The boat and its electrical components are powered by a 14.8V, 15.6Ah lithium-ion battery. This high-capacity battery will provide more than enough power for the boat to operate throughout the competition and hours of testing. The power system also includes a series of voltage and current regulators to deliver power to our 24V, 12V, and 5V, subsystems. Additionally, a kill switch circuit has been implemented to disconnect power from the thrusters in situations where the boat acts unexpectedly. All the relays and regulators featured are rated for their respective voltages and maximum currents that each electrical component draws upon.

C. Software Aspects

System Overview

To support the autonomous capabilities of the boat, our software consists of two major systems: the computer vision system and the motor control system. The computer vision system is responsible for constructing models to detect objects in the competition. The motor control system consists of the algorithms behind the autonomous decision making mechanisms and control systems for signaling the different motors. These two systems communicate via a ROS framework. Our ZED 2i camera is also part of the ROS system and is crucial in providing information for localization. The software system from a guidance, navigation, control (GNC) perspective is depicted in Figure 8.

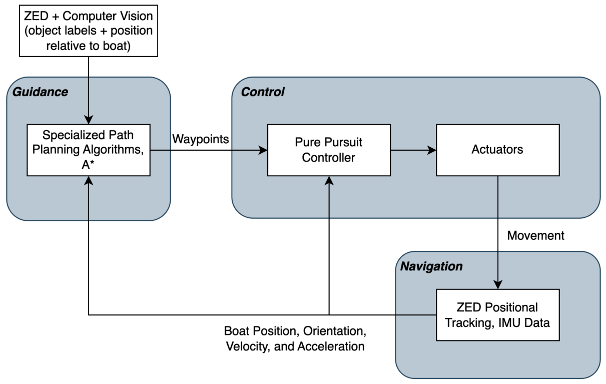
ZED + Computer Vision (object labels + position relative to boat)
Guidance
Specialized Path Planning Algorithms, A*
Waypoints
Control
Pure Pursuit Controller
Actuators
Movement
Navigation
ZED Positional Tracking, IMU Data
Boat Position, Orientation, Velocity, and Acceleration

Figure 8: Software full system overview from a GNC perspective.

Computer Vision

In conjunction with the ZED camera, our computer vision system enables the boat to detect and localize objects in its frame of view. Our team's workflow consisted of uploading data, labeling objects, and using the results to compare then improve the model.

Roboflow was used to create a custom dataset. We uploaded videos of the buoys in different environments and from a variety of angles. The videos were parsed into images based on the number of frames per second we chose. The images were then uploaded in Roboflow to manually label and draw bounding boxes around the target objects (such as the different classes of buoys). These labels were used to train the model.

YOLOv5 is a computer vision model [4]. We specifically tested the nano, small, and medium models, all of which had their own advantages and tradeoffs. Most notably, as the model grew in size, the potential for accuracy improved, but the real-time speed of detection slowed. Additionally, training time increased significantly. Using a GPU through Google

Colaboratory, however, made the training process quite fast, even with the larger models.

Having tried several different models, our most up-to-date version uses YOLOv5m (medium), and is quite effective at recognizing buoys. It is trained on over 1,000 different images (and counting) over 50 epochs. The model precision for object detection is around 99%. Confidence levels for detection are also similarly close to 1.

Motor Control

To support autonomous movement, the motor control system has two subgroups: path planning and path execution. Path planning encompasses the task-specific algorithms to create a list of waypoints defining the ideal path of the boat. To calculate these waypoints we utilize two frames of reference (Figure 9). The local frame is defined with the ZED camera as the origin, and is how obstacle coordinates are represented in the list we receive of objects detected by our model. The global frame is initialized where the ZED is turned on, and represents a more GPS-like frame which the boat moves through. The waypoints are global coordinates, yet require local information to calculate, so we utilize a mapping function using the current global position and orientation of the boat provided by the ZED.

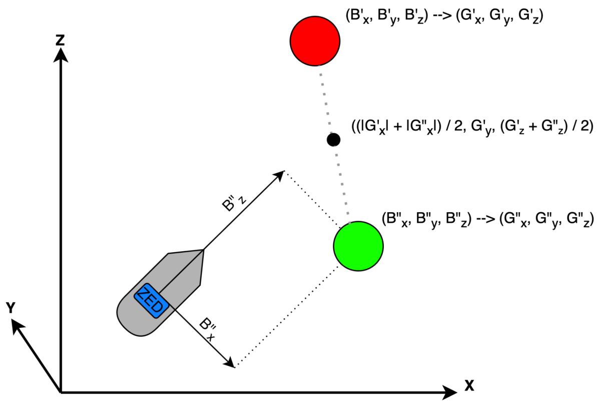
$(B'_x, B'_y, B'_z) \dashrightarrow (G'_x, G'_y, G'_z)$
Z
$((|G'_x| + |G''_x|) / 2, G'_y, (G'_z + G''_z) / 2)$
B''_z
$(B''_x, B''_y, B''_z) \dashrightarrow (G''_x, G''_y, G''_z)$
ZED
Y
B''_x
X

Figure 9: Aerial view depiction of the global and local coordinate frames.

Following a literature review of various path planning algorithms including [5], [6], we decided the waypoint creation will be done via specialized selection and calculation relative to given objects for simpler tasks (Panama Canal, Northern Passage, Beaching), and by an A* algorithm [7] for more complex tasks (Magellan's Route, Explore the Coral Reef). We handle task transitions by assuming a set order of tasks to complete, so after finishing one we know immediately which is next. This strategy was chosen for simplicity, yet we hope to create transition sequences more robust to edge cases in the future. The boat continuously executes an abstract loop of observing its surroundings, creating a path, executing the path, and determining if the task has been completed. Each task has a specific criteria for completion.

To execute the waypoint-defined path we are adapting a Pure Pursuit path follower, chosen for its success in differential-drive vehicles, ability to recover from drifting off path, and existing MatLab implementation [8]. The controller outputs the ideal linear and angular velocity the boat should maintain to reach a lookahead point some distance along the waypoint-defined path until a goal state is achieved. We use this information in conjunction with the IMU data provided by the ZED camera and PID control to send signals to the motors.

For the shooting tasks, we plan on selecting a strategic waypoint to orient the boat in front of the targets. Based on our distance and orientation from the target, we will aim and control the shooting speed of the water gun and skee ball shooter.

IV. Testing Strategy

The testing phase was an integral part of our development plan, understanding that verifying and correcting the behavior of our designs is critical to our success. Our testing plan consists of simulation, live-camera, and in-water components.

The simulation testing is purely software. For the task completion algorithms, simulation testing consisted of an expansive test suite for each challenge. Each test suite is a thorough set of test cases presenting different scenarios that the boat could experience as it progresses through the challenge. Specifically, the input for each test case is the boat's current position, orientation, and a list of course obstacles paired with their relative positions to the boat. The test case would then run the execution algorithm for the challenge on the provided scenario and verify that the execution algorithm selected the appropriate destination waypoints or executed the correct motor actions.

For the object detection system, we analyzed the accuracy of our trained models. The computer vision libraries that we used included methods for evaluating model performance and scripts for drawing bounding boxes, which we relied heavily on. We also manually and visually verified our models with live camera tests and ensured that the detection and position data provided by the ZED camera was accurate.

The in-water testing is a full-system software and hardware test, and was the final phase of our testing. Having verified that our software plans the correct course of action, the in-water testing verified that the boat would correctly execute that course of action on a physical course. We were able to check that the object detection, depth perception, path planning, and motor execution were accurate in this stage. The in-water testing began with isolating each challenge (and in some cases, specific aspects of each challenge). This eventually grew to combining multiple courses together so that full autonomous functionality was tested.

V. Acknowledgements

We would like to acknowledge and thank all those who supported our team, both monetarily and otherwise. We appreciate the Cornell University College of Engineering for funding our team's material and travel costs, providing facilities used in the construction and design of the boat, and offering course credit for those participating in the team. We would also like to thank all of our individual supporters including friends, family members, and alumni of the team who have graciously donated to Autoboat. Doctor Hunter Adams, our faculty advisor, has also been instrumental in our team's success by providing advice and guidance throughout our research and design processes. We also thank RoboNation's RoboBoat program for giving us an opportunity to challenge ourselves and grow as both engineers and teammates. Lastly, we must acknowledge the hard work and dedication of past Autoboat team members whose important contributions allowed us to be where we are today.

VI. References

[1] T. J. Grafton, "The Roll Motion of Trimaran Ships." Order No. U593304, University of London, University College London (United Kingdom), England, 2008.

[2] Easy Composites Ltd, "Mouldless Carbon Fibre Technique for One-Off and Prototype Components", *Youtube*, May 24, 2022. Available: https://www.youtube.com/watch?v=0Yaggj16S08&t=2s.

[3] F. Ahlstrand and E. Lindbergh, "Methods to Predict Hull Resistance in the Process of Designing Electric Boats", Dissertation, 2020.

[4] "Revolutionizing the world of Vision Ai," *Ultralytics*. [Online]. Available: https://ultralytics.com/yolov5.

[5] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, p. 7898, 2021. https://doi.org/10.3390/s21237898.

[6] Vagale, A., Oucheikh, R., Bye, R.T. *et al.* "Path planning and collision avoidance for autonomous surface vehicles I: a review," *J Mar Sci Technol* 26, 1292–1306 (2021). https://doi.org/10.1007/s00773-020-00787-6.

[7] N. Swift, "Easy A* (star) pathfinding," *Medium*, 29-May-2020. [Online]. Available: https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2.

[8] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," The Robotics Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, Tech. Rep. CMU-RI-TR-92-01, 1

VII. Appendix A: Component List

Component Name	**Vendor**	**Model/Type**	**Specs**	**Custom/Pur-chased**	**Cost**	**Year of Purchase**
Jetson	Stereolabs	Xavier NX	- NVIDIA® Jetson™ TX2-NX - GPU: 256-Core NVIDIA® Pascal™ - CPU : Dual-Core NVIDIA Denver 2 64-Bit and Quad-Core ARM Cortex-A57 MPCore - Memory : 4GB LPDDR4 - 51.2 DB/s	Purchased	$1,390	2023
Arduino Mega	Arduino	2560 Rev3	- Operating voltage (5V) - Input Voltage (7-12V) - Flash Memory (256 KB of which 8 KB used by bootloader) - SRAM (8 KB) - Digital I/O Pins (54, 15 PWM) - Analog Input Pins (16) - Clock Speed (16MHz)	Purchased	$45	2022
Camera	Stereolabs	ZED 2i Stereo Camera	120 FOV, built-in IMU, Barometer, Magnetometer, depth sensing, positional tracking, object detection, IP66-rated enclosure https://www.stereolabs.com/zed-2i/	Purchased	$499	2021
ASV Hull Form/Platform	Self Developed	N/A	54in x 29in x 8in	N/A	$400	2023
Propulsion	Blue Robotics	T200	Up to 5 kg Thrust / Each	Purchased	$400	2021
Power System	Blue Robotics	Lithium-ion Battery 14.8V, 15.6Ah	14.8V, 15.6Ah Max draw 60A Max Burst 132A	Purchased	$330	2021
Stepper Motor 1	Stepperonline	Nema 17 Bipolar	Step Angle: 1.8 deg Holding Torque: 26Ncm/36.8oz.in Weight: 230g	Purchased	$10	2021

Stepper Motor 2	Stepperonline	Nema 23 Bipolar	Step Angle: 1.8 deg Holding Torque: 1.26Nm/178.4oz.in Weight: 0.7kg	Purchased	$15	2021
Remote Controller	FLYSKY	2.4G FS-CT6B 6 Radio Model RC Transmitter & Receiver	8 model memory, digital control, full 2.4GHz 6-channel radio, 4-Model memory, integrated timer, throttle cut, computer programmable, USB Socket	Purchased	$50	2021
Motion Controllers	MATLAB	Pure Pursuit controller object, PID controller object	N/A	N/A	N/A	N/A
Vision	Self Developed	N/A	N/A	N/A	N/A	N/A
Localization and Mapping	Zed 2i and Self Developed	N/A	N/A	N/A	N/A	N/A
Autonomy	Self Developed	N/A	N/A	N/A	N/A	N/A
Programming Languages	Python 3, ROS, Arduino/C++	N/A	N/A	N/A	N/A	N/A
Programming Packages and Open Source Software	Numpy, Pytorch, MATLAB engine, ZED SDK, Google Collaboratory	N/A	N/A	N/A	N/A	N/A
Simulation Software	ANSYS Fluent	N/A	N/A	N/A	N/A	N/A