Design and Development of the NaviGator Autonomous Maritime System

Keith Khadar, Andrew Knee, Adrian Fernandez, Lester Bonilla, Andres Pulido, Adam McAleer, Lorant Domokos, Cameron Brown, Daniel Parra, Kayleigh Beron, Adam Hamdan, Eric Schwartz

***Abstract*—NaviGator Autonomous Maritime System (AMS) includes collaborative autonomous aerial and surface vehicles. This paper details our strategy and process for upgrading the NaviGator AMS for the 2024 Maritime RobotX Challenge. Key enhancements include an improved racquetball launcher, a redesigned modular electrical architecture, advanced object detection algorithms, and the integration of an Unmanned Aerial Vehicle (UAV).**

I. Competition Strategy

The Maritime RobotX competition presents a unique set of challenges, requiring teams to balance system complexity with overall performance. Given the limited time available, our competition strategy has been carefully crafted to maximize efficiency and effectiveness. Unlike newly formed teams, we have leveraged the advantage of existing systems in our approach. Therefore, the University of Florida's NaviGator AMS (Figure 1) team has been focused on enhancing the reliability of established systems before progressively tackling tasks of increasing complexity. By placing an emphasis on modality we have been able to parallelize many of our tasks effectively, allowing different sub-teams to work simultaneously on separate components. This approach has enabled us to focus on completing core features first while still ensuring that we give time for more complex ones.

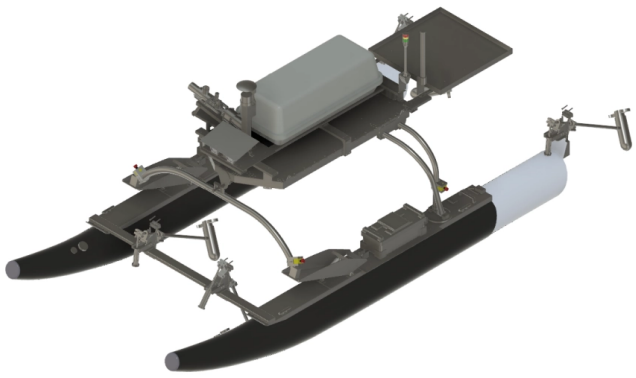

Fig. 1: 3D model of NaviGator assembly

To leverage the existing systems we had in place, we first needed to thoroughly test and verify their reliability. This was crucial, as our previous systems contained all the core functionality needed to complete most tasks. We began with a comprehensive inspection and testing process to identify which components required repair or replacement. From this assessment, we found several electrical components that needed replacement. The electrical team then performed a complete overhaul of the previous system, with the intention of improving modularity.

Focusing on testing the reliability of each subsystem not only simplified troubleshooting and development, but it also enabled us to perform maintenance and upgrades on individual components without affecting the entire system. The modular redesign allowed different sub-teams to work in parallel, making more effective use of our limited time. For instance, the electrical team could work on creating an RF controlled LED panel while the software team simultaneously refined and tested navigation and object detection algorithms. This parallelized workflow ensured that progress was continuous, minimizing downtime.

We carefully selected which tasks to parallelize and established deadlines based on estimated completion times and task dependencies. From the behavior standpoint, we defined a chart based on the dependencies of functionalities. The roots of the chart are navigation and object detection, and therefore we placed a high priority on completing them first (Figure 2). By ensuring the reliability of for example the navigation module, we could be confident that it would work effectively across all other tasks that depend on it as well as work independently on high-importance functionalities that did not depend on it (Figure 3).

In addition to core functionality, more complex tasks like Dock and Deliver needed to be started early due to their extensive requirements. Dock and Deliver, in particular, involved designing and manufacturing various components, such as the racquetball launcher, and producing new electrical boards, all of which required significant lead time. Additionally, the drone tasks had to be parallelized, as they were a high-scoring group of tasks that needed to be developed from scratch. By parallelizing these efforts, we ensured that critical components were ready on schedule, allowing us to meet the competition's requirements efficiently. With this approach, our strategy prioritized foundational tasks while simultaneously addressing complex, resource-intensive tasks in parallel.

Team NaviGator AMS is composed of three main divisions: Mechanical, Electrical, and Software. Within these divisions, there are multiple sub-teams. For example, the Mechanical

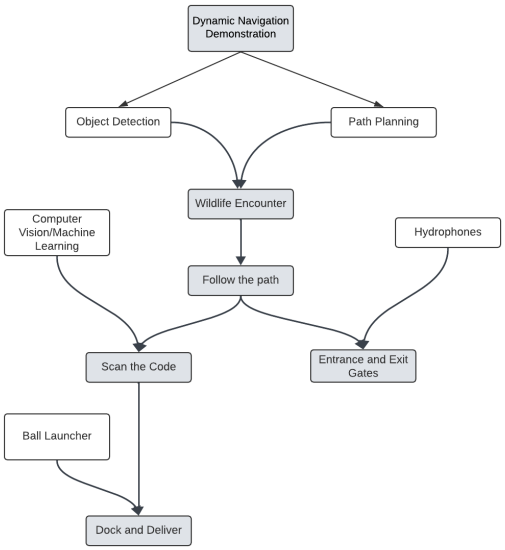

Fig. 2: Task Dependency.

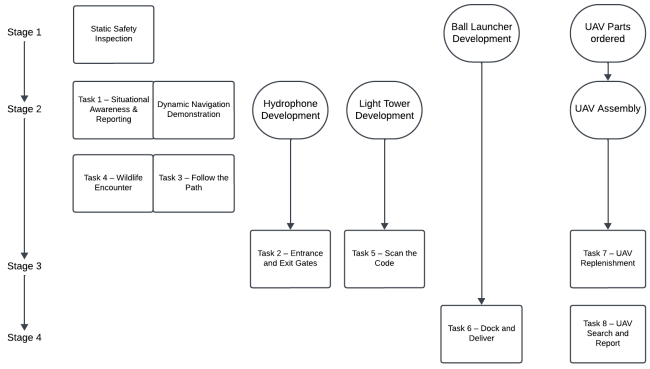

Fig. 3: Parallelized Development.

division includes sub-teams responsible for creating testing equipment, such as the light tower or buoys, and others focused on building actuators like the ball launcher. Similarly, the Electrical division has sub-teams working on developing each of the new modular printed circuit boards (PCBs). To manage the complexity of these tasks, we use GitHub as our primary tool for organization. We break down the competition into multiple actionable tasks/issues, which are then assigned to each sub-team or team member based on their skills and interests. Each division has its own GitHub repository where issues can be posted, and where project and Gantt charts are used to manage assignments effectively. All teams meet regularly to track progress and make adjustments to timelines as needed. In addition we used Discord for real-time communication across sub-teams, ensuring that all updates were logged and visible to every team member.

II. Design Strategy

A. Mechanical

1) Propulsion: NaviGator is designed to support holonomic motion. Four thrusters, two at the front, two in the back (Figure 4) are angled at 45 degrees to allow the boat to move omnidirectionally in the water plane while supporting simultaneous rotation. This configuration makes the boat highly maneuverable which is an advantage in in this competition. However, this comes at the cost of speed, as it sacrifices single-direction efficiency for greater maneuverability. Mounting four thrusters in this position does provide the advantage of redundancy as we observed after getting our boat in the water for the first time at the competition in 2022. During testing, we noticed a motor driver died and had to cut it without time to replace it. In the water, the controller was able to compensate and move the boat with only three motors, albeit in a sub-optimal way.

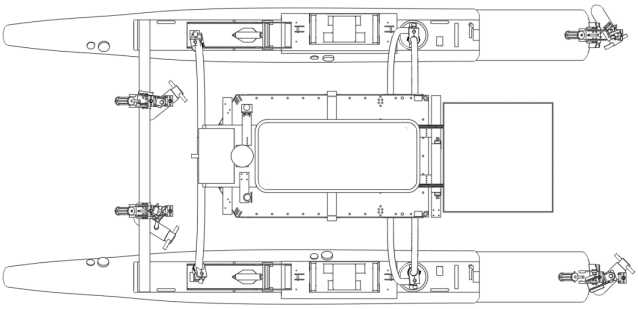

Fig. 4: Drawing of NaviGator assembly

Mounting the thrusters presented several challenges, primarily due to the structural constraints of the WAM-V and its transport trailer. We needed a design that allowed the thrusters to be repositioned for transport while maintaining precise alignment and orientation when deployed. The main issue arose from the fact that, in their operational position, the thrusters would collide with the trailer structure. To resolve this, we engineered a system that raises the thrusters while NaviGator is on the trailer and lowers them once it's deployed. For the front thrusters, we incorporated a piston mechanism that ensures both position and orientation are fixed during movement. This piston is connected to a mechanism that adjusts the angle of the thrusters as they are raised and lowered. The rear thrusters use a similar system, relying on a 3D-printed collar and block to maintain orientation when deployed, while also allowing them to be raised for transport.

2) Racquetball Launcher: The racquetball launcher (Figure 5) was redesigned for improved reliability, addressing issues with previous versions where wet components, caused by sea spray or rain, disrupted performance. While the new design retains the flywheel mechanism, it now features encased components and extended tubes at both the entry and exit points to reduce water exposure. Additionally, a mechanical guide system ensures only one ball enters the flywheel at a

time, eliminating reliance on electrical or software components for timing.

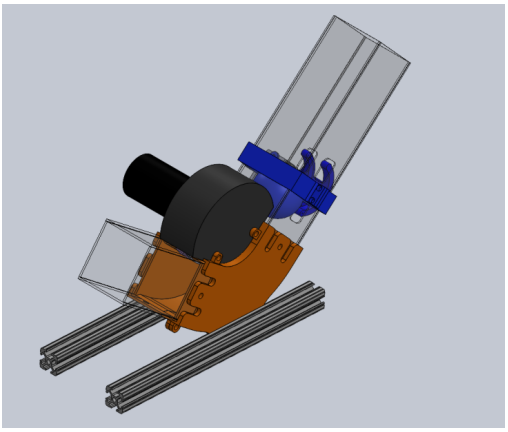

Fig. 5: 3D Model of Racquetball Launcher

3) Structures: In our design, one of the key structural elements is the front crossbar, which plays a crucial role in supporting two additional thrusters. The crossbar itself is designed to be both lightweight and sturdy, ensuring that it does not compromise the overall balance of the system while maintaining its structural integrity under load.

This year, we also integrated a new hydrophone system specifically developed for Task 2 – Entrance and Exit Gates. The hydrophone system is mounted on the front crossbar, which provides an ideal, stable platform for acoustic signal detection and localization. This system is designed to detect and track underwater pings that guide the vehicle through the gates. The secure placement of the hydrophones ensures they are positioned for accurate signal capture, without affecting the performance of the vehicle or its hydrodynamics.

Additionally, the drone landing pad (Figure 6) is another important aspect of the mechanical structure. We chose to construct it from wood, providing the necessary weight to stabilize the drone during takeoff and landing. Wood's workability allowed us to easily customize the landing pad to fit our specific design needs, ensuring it offers both durability and a stable platform for safe drone deployment.

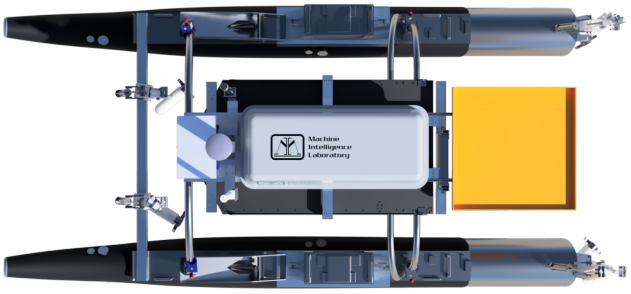

Fig. 6: Top Down View of NaviGator

B. Electrical

1) Open Architecture: In previous iterations of the electrical system, a single board was tasked with managing various critical components such as the relays, Visual Feedback System, and RF module. While this approach centralized control, it also introduced significant drawbacks. Any malfunction in one of the board's components necessitated a complete redesign and reassembly of the entire board. This not only increased downtime but also posed challenges in terms of reliability and repairability.

To address these issues, we opted to implement a modular approach in the current design. Rather than relying on a single board, we have divided the electrical system into several specialized boards (Figure 7), each responsible for a distinct subsystem. By decoupling these components, we ensure that a failure in one section. This modularity allows for quicker repairs and simplifies the upgrade process, ensuring greater flexibility and robustness in the overall design.

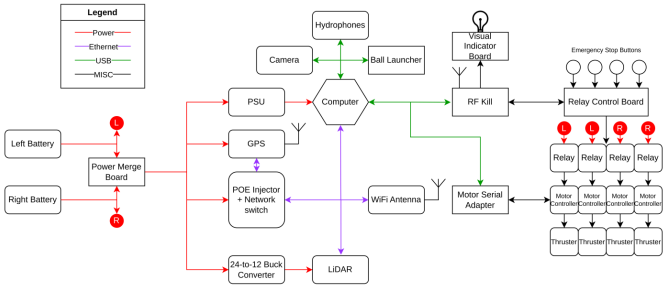

Fig. 7: Electrical Block Diagram

2) Power Merge Board: The key features of the power system include a dual battery power supply and a custom power merge board. The NaviGator AMS's power demands exceeded those of MIL's previous projects, both in terms of total power and the durability of the power sources required. Addressing these challenges involved exploring new battery suppliers beyond those traditionally used by MIL. To power four thrusters, along with the computers, sensors, and communication hardware, the AMS utilizes two Torqueedo Power 26-104 batteries. Each battery independently powers two thrusters while also contributing to the power rail that supports all other onboard devices.

A student-designed printed circuit board assembly (PCBA) known as the power merge board plays a critical role in this system. It employs two Texas Instruments LM5050 High Side OR-ing FET controllers to act as ideal diode rectifiers, balancing and paralleling the batteries into a single rail that provides four output ports. This design enhances fault tolerance, allowing for seamless battery swaps without shutting down the system.

3) Kill System: The hardware kill system consists of two student-designed PCBAs and four commercial off-the-shelf (COTS) twist-to-detent kill switches. The kill system also includes a software component. The system monitors five kill sources, which include the four mounted COTS kill switches

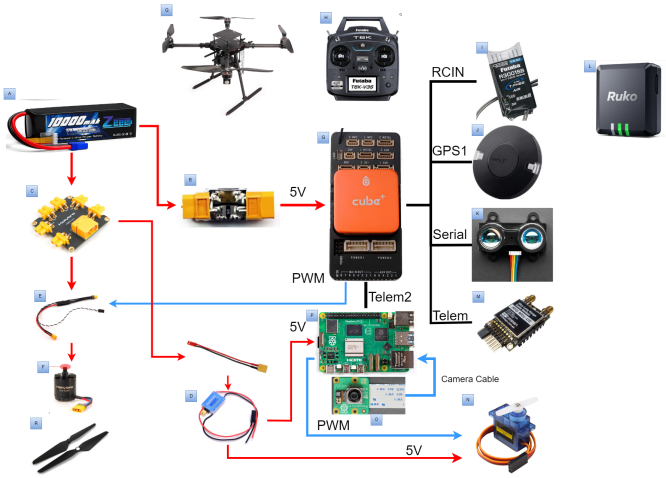

Fig. 8: UAV Component Diagram

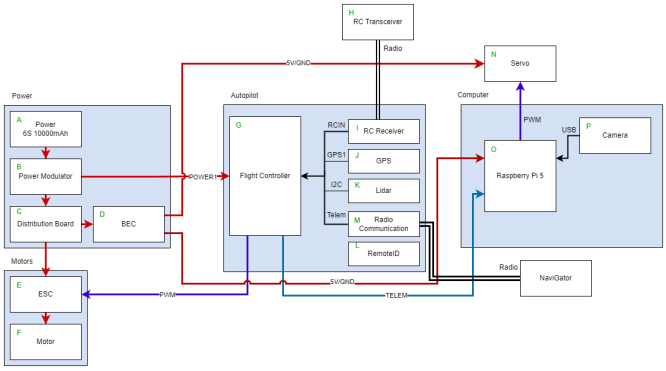

Fig. 9: UAV Block Diagram

and a remote kill switch. These inputs feed into a relay control board, which is a digital logic board designed to cut power to the thruster motor controllers when any kill source is triggered or becomes unresponsive.

The remote kill switch operates over a wireless link. While we've been experimenting with ZigBee, LoRa, and Nordic protocols, we are currently leaning toward LoRa for the communication link. Additionally, the system includes status monitoring for the vehicle's hardware kill status. Though the computer can also act as a kill source, its mechanism is currently separate from the relay-controlled system.

4) UAV System: Our UAV system is made from several components (Figure 8) that connect to a central flight controller. We opted to purchase a ready made frame kit as a base for our UAV, onto which we added the components necessary for autonomous missions. We opted for a mostly-ready-to-fly option after assessing the risk of a fully custom build. The cost of the kit was close enough to the cost of a fully custom build, without the benefit of flight testing from the manufacturer. With the recommended 10,000 mAHr battery, we reached close to the advertised 30 minute flight time. This flight time was much longer than the expected 10 minute flight of our expected custom build. Without a large increase in cost, the kit allowed us to reduce complexity in assembly of the drone as well a needed increase in flight time that will allow our missions to operate with a larger room for error.

Power is distributed from our 10,000mAHr 6S battery to the ESCs, Flight Controller, Raspberry Pi, and Servo from a power distribution board provided in the kit. We use Universal Battery Elimination Circuits (UBECs) to convert the 25.2V supply down to 5V to power the Raspberry Pi, Controller and a servo motor.

The sensors architecture (Figure 9) on our drone include GPS, Lidar Rangefinder, Raspberry Pi Camera, IMUs, Compass, barometer, DroneID, and radio communication. Our flight controller, a Cube Orange+, is accompanied by a Raspberry Pi 5. Our system utilizes a servo motor for use in the UAV replenishment task.

For programming the UAV missions, we used popular hobbyist software that is supported by the flight controller, ArduPilot, with Python libraries that facilitate communication in ArduPilot's communication protocol, MavLink. The Raspberry Pi connects to the flight controller and directs the UAV to GPS coordinates (Figure 9). Keeping with our design goal of simplicity first, our missions were developed with straight forward principles. Once we were flying manually, we created a simple program that would command the UAV to predetermined positions, proving how we would send messages from the Pi to dynamically operate the UAV. The next step was to create an OpenCV program that would detect the launchpad. Our first priority was to safely launch from, move away from, and land back onto NaviGator autonomously. This is the most mission critical aspect of the UAV, as it would determine success of the missions.

With safe return available to the UAV, we create a simple loop for the replenishment mission: launch from NaviGator, ascend high enough to where the replenishment pads can be seen, without changing altitude, hover above the launchpad. The UAV then descends incrementally, using CV to adjust itself to the center of the launchpad. Once we can reliably see the tins, we repeat descension, but centered on the tin. To keep our tin collection simple, we use a strong magnet. A servo controls the position of the magnet and slides the tin off the UAV when delivering. The UAV ascends, and verify that the number of tins on the platform has decreased. It will attempt pick up twice more before aborting mission and returning to NaviGator. Once the tin is secured, a similar algorithm is used to find the second launchpad. We take the same launch, detect, hover, descend approach for the UAV Search and Report task.

C. Software

1) Object Detection : The lowest level perception service available on NaviGator is the Occupancy Grid Server [1]. Occupancy grids are a two-dimensional grid-like representation of the environment generated by the sensor suit on the AMS. The generated map contains information about both the occupied and unoccupied regions in the environment. NaviGator utilizes a Velodyne VLP-16 LIDAR (Figure 10. A LIDAR

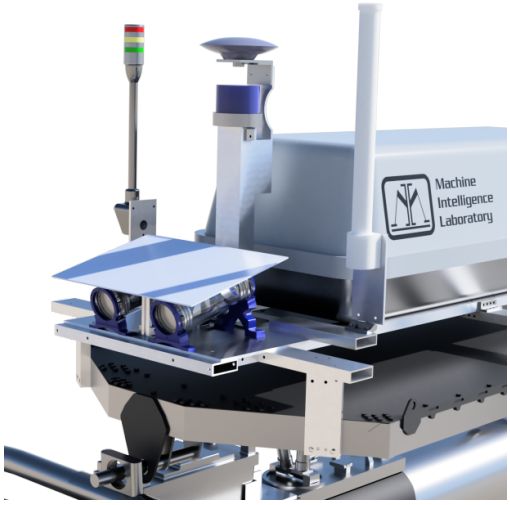

Fig. 10: LIDAR Sensor Mast

uses lasers to provide relatively dense range information of the environment. This information is then segmented by regions containing dense clusters of relatively close points. These bounding regions are treated as obstacles, and are placed in the occupancy grid. This information is then provided to higher level services such as the motion planner and classification server. From our extensive testing we discovered that the wake produced by NaviGator could in specific circumstances, like when moving backwards, be detected by the LIDAR and treated as obstacle leading to issues in our higher level services like in our motion planner. We solved this issue by checking if points detected while moving backwards are inside of our area of interest. Often times, light reflecting on the surface of the water would be relatively low intensity and trigger an object detection. So we additionally implemented a secondary point cloud filter to filter out low intensity points.

2) Classification: In our classification system, the goal is to label the clusters identified by the object detection system. We utilize a YOLOv7 model, one of the latest and most efficient real-time object detection systems in computer vision [2]. This YOLO model was trained on thousands of labeled images of buoys gathered from field tests and previous competition videos . The model continuously processes image streams from our cameras, identifying buoys by placing bounding boxes around them.

The 3D bounding boxes generated by the LIDAR system are transformed into camera space, providing an estimate of where each object would appear in the image. By combining these transformed LIDAR boxes with the YOLO bounding boxes, we can match each YOLO detection to the closest corresponding cluster in camera space, allowing us to accurately label each buoy or object.

3) Trajectory Generation: For the motion planning we currently utilize a rapidly-exploring random tree (RRT) algorithm [3]. The algorithm starts with a seed node representing NaviGator's initial state. The algorithm then randomly samples a state in the region of navigational interest. A nearness function is then applied to each node in the tree and that node is extended or steered towards the random state based on a policy function. The resulting endpoint is then added to the tree as a new node if it is allowable, i.e. the newly created edge does not cross through an obstacle (Figure 11). If the extension or or any intermediate state leading up to it is not allowable, that iteration is abandoned. This algorithm repeats for a specified time. Once a node reaches the goal, region, the tree is traced backward from the goal to the seed, and is classified as one solution to the planning problem. The best of the found solutions is defined as the one that takes the least amount of time. The goal region is likely to be reached because one can bias tree-growth towards it by shaping the probability density function from which random states are sampled.

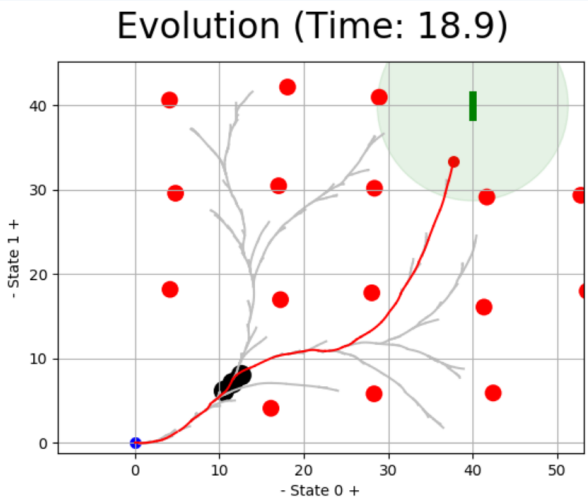

Fig. 11: An example of NaviGator's RRT algorithm in action

4) Motion Control: The controller takes in the trajectory generator as the reference states for the feedback controller to follow. We utilize a model-reference adaptive control (MRAC) architecture [4] for its prevalence in marine and aerial systems. MRAC brings steady-state error to negligible amounts in all cases without introducing oscillations. The design choice for the MRAC was to estimate the drag and inertial effects to be able to adapt external disturbances. The block diagram of the MRAC controller used for NaviGator is shown in Figure 12. In this diagram, Y_ref is the current state in the sequence generated by the motion planner, u is the control effort choice, and Y is the actual state. Finally, with the controller outputting desired wrenches (i.e., forces and moments), the last operation needed is to map that wrench to a thrust command for each thruster.

III. Testing Strategy

Our overall testing strategy matches our competition strategy. We test systems according to our specified deadline and collect data to help complete future tasks on time. This process is designed to ensure thorough validation of the system's functionality and readiness for the competition. Testing is conducted on a regular schedule, with major testing sessions

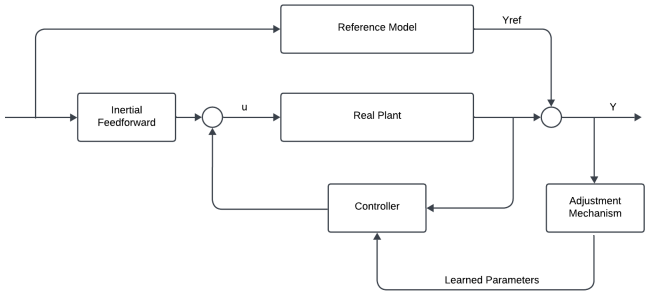

Fig. 12: An example of NaviGator's RRT algorithm in action

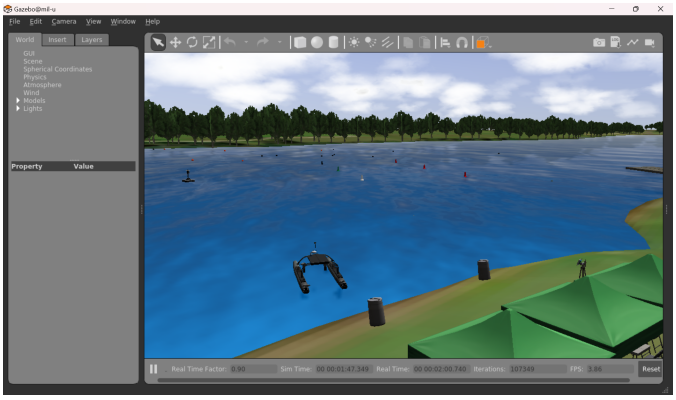

Fig. 13: Gazebo Simulation Testing

held every Sunday and smaller sessions on Wednesdays. The Sunday sessions focus on evaluating key competition tasks and major system changes, while the Wednesday sessions are dedicated to validating smaller updates and preparing for the Sunday testing sessions.

Between these testing sessions we make use of our simulator, Gazebo, to test changes in our software. We use the Virtual RobotX simulation from Open Source Robotics Foundation (Figure 13) as a foundation to design the test cases. We replicate much of the conditions and challenges that are present in RobotX 2024 allowing us to test each task thoroughly. This helps us find potential issues early and ensuring our physical testing sessions are productive.

Each testing session follows a standardized procedure. Prior to the testing sessions, we determine what systems and features we intent to test and verify. We typically go to our primary testing location, Lake Wauburg, or go to our backup location, Lake Orange, when needed. Once on site, we deploy NaviGator into the water as shown in Figure 14. Afterwards, a team is sent out on a kayak to help monitor and assist the boat. The kayak crew is also responsible for positioning the buoys and other task specific items into the water. Clear communication is established between the ground team and water team through the use of walkie-talkies.

Our testing sessions are collaborative, with team members working together to troubleshoot and resolve any issues that arise. We strive to replicate the competition conditions as

Fig. 14: Testing NaviGator at the Lake

closely as possible, ensuring that our tests provide accurate insights into the system's performance under realistic scenarios.

Following each test, we write a detailed summary report. This report details what was accomplished, any issues encountered, and what changes or fixes are needed. These reports allow us to reflect on our work and progress. They are vitally important as they are key indicators on if we need to adjust out timeline and priorities to stay on track for the competition.

ACKNOWLEDGMENT

Team NaviGator AMS, a project of the Machine Intelligence Laboratory (MIL) would like to extend our sincerest gratitude to everyone who has supported the team. We extend our thanks to the University of Florida's departments of Electrical & Computer Engineering and Mechanical & Aerospace Engineering for their continued support. We would also like to thank the CIMAR lab for providing the essential resources and facilities for our work. We are also deeply appreciative of our major industry sponsors: L3Harris Corporation, Texas Instruments, and Sylphase. A special thank you to our advisers, Dr. Eric Schwartz, Dr. Carl Crane, and Andres Pulido, for their invaluable guidance and mentorship throughout this project.

REFERENCES

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," in Computer, vol. 22, no. 6, pp. 46-57, June 1989, doi: 10.1109/2.30720.

[2] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-ofthe-art for real-time object detectors," arXiv preprint arXiv:2207.02696, 2022.

[3] S. M. LaValle, "Rapidly-exploring random trees : A new tool for path planning," The annual research report, 1998.

[4] A. Shekhar and A. Sharma, "Review of Model Reference Adaptive Control," 2018 International Conference on Information , Communication, Engineering and Technology (ICICET), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICICET.2018.8533713.

APPENDIX A

	Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost (total)	Year of Purchase	Reasoning
NaviGator ASV	Propulsion	Bass Pro Shops	Minn Kota Riptide Transom-Mount Saltwater Trolling	Volts: 24v, Thrust (lbs.):80	Purchased	$2,200	2022	Movement
	Batteries	Torqeedo	Power 26-104 Lithium-Manganese 24 Volt Battery	Amp Hours (Ah): 104Ah (2,685Wh) Nominal Voltage: 25.9V DC	Purchased	$4,600	2016	Power
	Motor Controls	RoboteQ	MDC1460	Brushed DC Motor Controller, Single Channel, 1 x 120A, 60V, USB, CAN, 8 Dig/Ana IOr	Purchased	$370	2016	Motor controller
	Teleoperation	Ubiquiti	Ubiquiti Rocket AC Wireless Access Point (R5AC-LITE)	5 GHz frequency band 500+ Mbps throughput	Purchased	$135	2016	Communications
	CPU	Intel	i7-8700k	6 core 4.7 GHz	Purchased	$360	2022	Computation
	GPS/INS	Sylphase	Infix-1	0.6m absolute position error	Custom	Donated	2022	Odometry
	Camera(s)	Dell	UltraSharp Webcam	4k Ultra HD, 1/2.8" Sensor	Purchased	$340	2024	Computer Vision
	Hydrophones	Teledyne	Teledyne Reson TC4013	frequency range of 1Hz to 170kHz	Purchased	$1710	2022	Underwater Acoustics

NaviGator UAV	Raspberry Pi 5 Kit	Canakit	Raspberry Pi 5	(128GB Edition) (8GB RAM)	Purchased	$159.99	2024	Companion Computer for UAV
	Neodymium Magnets (Circles)	Zoprima Magnets	Neodymium N52 Powerful Rare Earth Magnets	Dia 1.26x0.06	Purchased	$16.99	2024	Used for UAV tin collection task
	XT30 to JST Adapters (3-pack)	OliYin	3pcs Male XT-30 to Male JST Connector Adapter	Wire Length: 3.93 inch/10cm Wire Gauge: 20awg	Purchased	$15.96	2024	UAV: connectors to power distribution and BEC
	Garmin Lidar	Adafruit	Garmin LIDAR-LiteOpti cal Distance Sensor - V3	Resolution: 1cm Accuracy: +/- 2.5cm Range: 5 cm to 40 meters	Purchased	$129.95	2024	UAV: For precision landing
	Raspberry Pi Camera 3	Adafruit	Raspberry Pi Camera Module 3 Standard - 12MP Autofocus	Sony IMX 708 12-megapixel sensor with I2C-controlled focus actuator	Purchased	$25.00	2024	UAV: For vision
	Raspberry Pi Camera Cable (500mm)	Adafruit	Raspberry Pi 5 FPC Camera Cable - 22-pin 0.5mm to 15-pin 1mm - 500mm long	500mm long designed to work with the Raspberry Pi 5 series	Purchased	$5.00	2024	UAV: To connect camera to RPi5
	Cube Orange+ with Carrier Board	IR-LOCK	Cube Orange+ Standard Set ADS-B	Accelerometer, Gyroscope, Compass, Barometric Pressure Sensor	Purchased	$350	2024	UAV: Flight Controller
	Hero3+	CubePilot	Hero3+ with iStand	u-blox high precision GNSS modules (M8P-2)	Purchased	$225	2024	UAV: GPS
	Drone ID	CubePilot	CubeID	Enable information broadcasting (e.g., identification number	Purchased	$39	2024	UAV: ID required for FAA Registration
	Algorithms	MRAC Controller, RRT Motion Planning						
	Vision	OpenCVOpenCV (Canny Edge Detection, Thresholding, Optical Flow), RCNN (YOLOv7)						
	Acoustics	Scipy, Numpy (Time of Arrival, Least Squares, Fast Fourier Transform)						
	Autonomy	Robot Operating System (ROS) Noetic						
	Open-Source Software	All of our software is open-source						