RobotX 2026 Technical Design Report

Shrey Patel, Thomas Devlin, Aaron Wu, Samuel Ibidapo, Jorge Ortiz, Sean Fish, Kieran Abbott, Paul Dubrelle, Kavin Prakash, Soham Goel, Nathan Xie, Carl Johnson
Marine Robotics Group at Georgia Tech
Atlanta, Georgia

***Abstract*—The Georgia Tech Marine Robotics Group addresses the Maritime RobotX Challenge through a deliberately focused system of systems (SoS) strategy built around one Unmanned Surface Vehicle (USV) and one Unmanned Aerial Vehicle (UAV). Robust execution of Safe Passage and Coordinated Logistics tasks was prioritized. The resulting design centers on shared situational awareness and composable autonomy. The USV serves as the primary mission-management platform while the UAV extends its sensing reach, mobility, and task-execution capabilities. A common spatial and object representation enables cross-domain information sharing, while reusable Behavior Tree-based autonomy supports task execution, interruption, and recomposition as mission conditions change.**

This report presents the competition strategy, the systems engineering process, the subsystem design and integration decisions, and the testing methodology used to develop this architecture. The systems were progressively verified through simulation, bench testing, and integrated field trials. Integrated field trials demonstrated bidirectional vehicle-state exchange and cross-domain perception transfer under representative operating conditions.



Fig. 1: USV during field testing.

I. Competition Strategy

Maritime environments present severe operational challenges for autonomous systems due to dynamic obstacles, environmental uncertainty, and degraded perception [1]–[3]. Real-world missions such as environmental monitoring, infrastructure inspection, post-disaster response, and search-and-rescue inherently exceed the capabilities of any single vehicle type [4]–[7]. Multi-domain architectures address these challenges by exploiting platform asymmetries across aerial, surface, and underwater assets, transforming individual trade-offs in speed, endurance, payload capacity, and vantage point into complementary operational capabilities [8]. For example, Bartlett et al. [9] deploy an agile UAV for rapid reconnaissance while a USV provides heavier, higher-fidelity sensing. The 2026 Maritime RobotX Challenge captures these field realities by incorporating dynamic task parameters, unfamiliar targets, and mandatory cross-domain collaboration. In a controlled competition environment, these evolving elements serve as a practical proxy for the incomplete prior knowledge and environmental variability encountered by field robotic systems.

This Technical Design Report details the Marine Robotics Group's engineering implementation for RobotX 2026. The remainder of this section defines the team's competition approach, platform scope, and task-selection rationale. Section II details the design strategy and technical architecture across mechanical, electrical, perception, navigation, and autonomy subsystems. Section III presents the testing philosophy, operational procedures, and empirical field results validating system performance.

A. Competition Approach

The RobotX challenge presents a diverse set of operational tasks spanning air, surface, and subsurface domains. Approaching each task as an isolated engineering problem quickly fragments development effort and produces narrow point solutions with little reusable value across the rest of the course. Instead, the team's competition strategy focuses on foundational capabilities that support the maximum number of tasks, guided by two core principles: *shared situational awareness* and *composable autonomy*.

Situational awareness in robotics is essential to safety and effective decision making [10]. The maritime domain further exacerbates these issues with dynamic obstacles and challenging perception conditions. In dynamic, unstructured scenarios such as disaster response, unpredictable environmental dynamics may degrade situational awareness to the point that vehicles cannot safely operate. Given that the competition emulates these scenarios, our approach to both task execution and vehicle development explicitly accounts for information uncertainty and, to the extent possible, leverages the benefits of including multiple mobile sensing platforms.

A quick observation from past RoboNation events is that the days on site are hectic. While hardware malfunctions contribute, more often it is the result of the massive volume of software development compressed into only a few days. While not ideal, this need to adapt in the field is a reality of the competition environment. With that reality in mind, a



Fig. 2: UAV during field testing.

portion of our competition strategy lies in preparation. Our focus in software development is on implementing primitive behaviors which can be relied upon during the week of competition. Without access to a full course, coming prepared with autonomy primitives is expected to yield dividends as they are assembled at competition.

B. System Scope and Task Selection

For RobotX 2026, the Georgia Tech team is bringing a single USV and a single UAV, both shown in figures 1 and 2. A brief inspection of the competition tasks shows that the missing UUV platform would be deployed on only a portion of a single task. By restricting to two domains, the team's development efforts are more focused. Furthermore, the USV and UAV platforms have competition heritage.

When the WAM-V was removed from the RobotX competition, the team focused on preparing a USV for RoboBoat so that critical subsystems could be tested in an operational environment [11]. Core components of the safety system, in particular, were shown to be compliant at RoboBoat 2026. While the UAV has not seen competition, it is an iteration on the vehicle used at RobotX 2024 [12]. By employing as many existing subsystems as possible, the team's development efforts were further focused onto the subsystems unique to the 2026 ruleset. With these two vehicles the planned tasks are:

1) *Safe Passage:* ***Disruptive***
   This task embodies the objectives for our competition strategy. It is explicitly formatted to require shared situational awareness between vehicles and involves autonomy primitives which would be reused across the competition. Therefore, the intent is to complete it at the highest level.
2) *Infrastructure Survey and Repair:* ***None***
   This task centers on underwater operations. For this reason, there is no plan to attempt it in earnest. Locating the start with the UAV will be attempted as it is a simple reuse of existing primitives.
3) *Coordinated Logistics:* ***Advanced***
   This task involves the most hardware complexity of any tasks; however team efforts were focused on the development of the UAV tin release mechanism and USV water sprayer. The advanced tier is chosen to avoid the risks of the disruptive tier. Namely that landing is a high-risk behavior and, while tin release does not require landing, tin recovery does.
4) *Dynamic Incident Response:* ***Disruptive***
   This task adds complexity to all other tasks. However, the selected autonomy architecture discussed in II-F1 is amenable to the stop-start behaviors required. Beyond tasking interruption, compliance with no-go zones or object avoidance can be implemented with primitives developed during other tasks.

Collectively, these task selections represent a deliberate trade between capability breadth and system reliability. Adding a UUV or pursuing the disruptive tier of Coordinated Logistics would expand the available competition capabilities, but would also introduce additional hardware, autonomy, integration, and testing requirements. Instead, development effort is concentrated on a smaller set of reusable capabilities and established platforms, allowing a greater portion of the limited development and field-testing time to be devoted to system integration and refinement.

Task ordering is a critical consideration during full competition runs. When deciding task ordering, the primary constraint is UAV endurance, which is shorter than both the duration of a competition time slot and the endurance of the USV. Unfortunately, both the disruptive *Safe Passage* and advanced *Coordinated Logistics* require synchronous operation with the USV. A second criterion to consider is the USV speed, which is lower than that of the UAV. If tasks are completed such that the USV path length is minimized, then the UAV flight time can also be minimized. Therefore, the task ordering strategy will depend on the placement of tasks on the competition field. In fact, the first autonomous behavior will be to estimate the locations of each task with the long-range sensors discussed in II-B3.

While this report discusses competition strategy before design strategy, it should be noted that they are developed in tandem. Establishing a competitive focus on situational awareness and composable autonomy early in the competition season led to the development of sensing subsystems and autonomy primitives. In turn, the available vehicle capabilities gave rise to the actual task level competition strategy. The following section details the vehicle designs and the decision-making process from which they arose.

II. Design Strategy

The Marine Robotics Group adopts a design strategy aligned with formal systems engineering methodologies to minimize operational risk while ensuring high task-completion reliability. Development begins with functional decomposition: competition objective are translated into to concrete functional requirements. These requirements are used to identify potential hardware and software subsystems aligned with operational and development-timeline constraints.

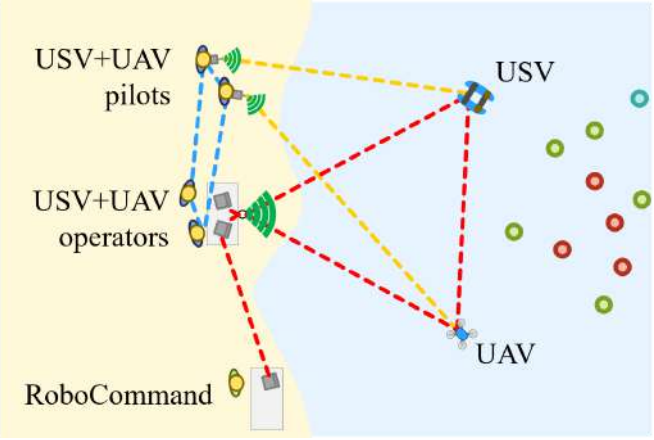


Fig. 3: Operational Concept of vehicle deployment with information flows. Link classification: 900 MHz inter-vehicle and telemetry mesh (red dashed), 2.4 GHz diagnostic and ROS logging link (blue dashed), and dedicated 2.4 GHz RC manual/kill link (yellow dashed).

A. System-of-Systems Architecture

The most important system requirements are those derived directly from activities required to complete tasks. For RobotX 2026, these activities can be described at a high level by which systems are involved in a task. For instance, *Safe Passage* requires the operation of a UAV and USV simultaneously. Before developing the requirements for these subsystems, some consideration must be made for how they interact. This approach treats the competition environment as a system-of-systems where multiple actors communicate or otherwise affect one another to create complex behavior. Figure 3 details the six operational nodes within this SoS: the UAV, USV, UAV pilot, USV pilot, Operator Control Station (OCS) operators, and RoboCommand.

In this architecture, the primary exchange between nodes is information: regulatory scoring downlinks route from vehicles through the OCS to RoboCommand, while direct vehicle-to-vehicle links coordinate the USV and UAV. Again, considering *Safe Passage*, a requirement that the UAV must be able to pass buoy state information to the USV can be derived. Similar requirements on the inter-vehicle transfer of information can be derived from activities throughout the competition. Coordinating vehicles through these transfers gives rise to distinct interface, authority, and bandwidth engineering challenges.

1) Multi-Vehicle Coordination: Cross-domain multi-robot systems greatly extend operational capability, but realizing this advantage requires resolving differences in sensor perspective, spatial reference frames, and decision-making authority under limited bandwidth. For *Safe Passage*, a buoy observed from the UAV must be represented by a location and light state that the USV can interpret for route planning. Coordination therefore requires a common UTM reference frame, consistent obstacle semantics (Section II-D), shared message formats and communication links (Section II-A2), and navigation interfaces translating received data into motion (Section II-E).

To enforce clear authority boundaries, the Marine Robotics Group assigns primary mission orchestration to the USV, while the UAV executes assigned behaviors via an independent Behavior Tree (Section II-F). Strategic goals are dispatched as declarative ROS 2 action intents, but tactical execution—local trajectory generation, obstacle avoidance, and flight safety—remains sovereign to each platform.

Strict interface contracts govern inter-vehicle exchanges: raw sensor streams and tight control loops are barred from crossing vehicle boundaries. Instead, the UAV transmits only compact, georeferenced UTM target tracks with spatial covariances. System safety is maintained through layered autonomy: if an E-stop triggers or communication drops beyond 2.0 s, the multi-agent contract severs. The UAV enters an autonomous station-keeping hover while the USV continues single-agent navigation via onboard LiDAR, preventing localized faults from propagating across platforms.

2) Comms Network and Ground Station: To satisfy these boundary constraints, the communications network reconciles latency, range, and bandwidth tradeoffs across three operational traffic tiers:

- *Mission Coordination:*
  Carries deterministic, mission-critical peer messages between the USV and UAV, such as fused target estimates, search-area bounds, and BehaviorTree state triggers required for autonomous cooperation over a dynamic 900MHz mesh network
- *Operator Awareness*:
  Delivers continuous system health telemetry, actuator states, and mission phase progress to the ground station, while bridging run commands and dynamic incident tasks between the competition RoboCommand server and the autonomy stack
- *Development and Diagnostic Data:*
  Accommodates high-bandwidth payloads (point cloud crops, compressed video, ROS topics) over a 2.4 GHz star network for operator debugging without starving critical command paths.

Routing is isolated by subnetting devices across each platform and the OCS, mapping high-throughput diagnostic telemetry to the 2.4 GHz link while reserving the long-range 900 MHz mesh for critical inter-vehicle coordination and relay capabilities.

B. Surface Vehicle

This subsection describes the USV hardware. Additional details are provided for hardware developed specifically to serve the competition strategy.

1) Base Platform: The surface vehicle is derived from [13] and extends its capabilities to suit the needs of the RobotX competition. The base vehicle is itself an extension on Blue Robotics' BlueBoat USV. To preserve the platform's hydrodynamics and reliability, the team minimized changes to the original hardware. Rather than mounting external payload enclosures that increase drag and introduce sealing risks,

additional compute and electronics hardware were engineered to fit in the stock hull electronics bay.

For completeness, a summary of the capabilities of the base vehicle is provided here. Additional details are available in [13] and [11]. The USV implements a frontseat-backseat architecture to divide computing responsibilities. The frontseat is a Raspberry Pi with a BlueRobotics Navigator hat. These are responsible for localization, actuator control, manual control, and some safety (beyond the RoboNation requirements). The backseat is an Nvidia jetson Orin Nano and is responsible for more complex software such as perception, mapping, planning, and autonomous decision making. Baseline navigation relies on the OEM GNSS receiver and internal IMU. Local perception is provided by two forward-facing Luxonis OAK-D Pro W PoE stereo cameras mounted to the port and starboard bow crossbars, paired with a centrally mounted Livox Mid-360 LiDAR. While this fixed suite provides sufficient field of view for proximal collision avoidance, its effective sensing range is bounded. To satisfy the demands of the 2026 competition tasks, the baseline platform is augmented with three competition-specific subsystems: an integrated safety and E-stop circuit, a long-range stabilized sensor gimbal (Section II-B3), and an actuated water shooter (Section II-B4).

2) Safety Subsystem: The safety subsystem uses multiple layers of redundancy, including a dedicated manual control link, hardware kill circuit, and software failsafe, so that no single failure can leave the vehicle able to move without the operator's knowledge. The subsystem is integrated via the Peripheral, Radio, and E-Stop (PRESto) circuit board. An earlier version was validated with the team's RoboBoat [11] entry and has since been improved to meet new competition requirements and improve operational reliability.

The principal safety change for RobotX is the manual control path. On the RoboBoat vehicle, teleoperation used a game pad routed through the OCS, placing manual control on the same network as telemetry. The higher data volume of the RobotX vehicles, including LiDAR and camera feeds, can induce significant network latency. This, together with the ruleset requirement for a dedicated remote controller, led to a dedicated RC link connecting manual control to the frontseat navigation system and a kill channel to PRESto.

The hardware kill circuit is carried over and improved from the RoboBoat vehicle. An onboard E-stop button and a trigger relay controlled by the RC kill channel are connected in series to open power relays that remove power from actuators on the PRESto kill harness, including thruster Electronic Speed Controllers (ESCs). Cutting ESC power forces the ESCs to reinitialize when power returns, so the thrusters remain idle until deliberately re-armed. The relays are held closed only while the circuit is energized, so an electrical fault, dead transmitter, or lost RC link results in a stop. PRESto also includes a status relay to report the kill circuit state.

PRESto's interfaces were redesigned to reduce harness complexity and improve connector reliability following unintentional disconnections in the RoboBoat competition environment. Connected to both the main computer and kill harness, PRESto also reports vehicle operation status via the light tower. Its direct integration with the kill circuit ensures the red light directly reflects system safety status regardless of autonomy software state. The tower displays colors in accordance with RobotX USV requirements, while operation mode is communicated to the backseat computer as lightweight protobuf messages.

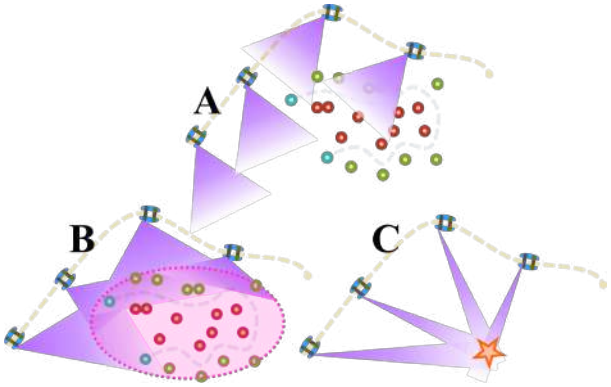


Fig. 4: Gimbal CONOPS: relative angle scanning (A), region of interest scanning (B), and object of interest scanning (C).

A software failsafe complements the hardware kill system. When an E-stop signal is received, the autonomy stack halts command traffic, cancels active behaviors, and suppresses outgoing commands to other systems and vehicles. This prevents a stopped or malfunctioning USV from triggering actions on other systems. The hardware kill circuit remains the authority for disabling onboard actuators, while the software layer ensures a safe, known state across systems for post-failure recovery.

3) Stabilized Sensor Gimbal: A fundamental limitation of any fixed sensor suite is that the information available to those sensors is linked to the vehicle position and heading. In the case of BlueBoat platform this issue is complicated by the fact that the vehicle is underactuated. In the case of underactuated vehicles the vehicle's position and heading cannot simultaneously be controlled. With only fixed sensors, tasks where maneuvering is constricted are likely to also constrict sensor pointing. Furthermore, even in very mild sea states small USVs, such as those in the RobotX competition, experience significant pitch and roll disturbances. While these issues can be resolved with some clever path planning and noise rejection, a more robust solution is the inclusion of a steerable sensor suite. Such a suite would enable the three novel modes of operation illustrated in figure 4.

The gimbal subsystem consists of a 3-axis gimbal, gimbal control boards, Livox AVIA LiDAR, and an OAK-1 camera. The assembled subsystem mounts to the starboard quarter of the vehicle, as shown in figure 5. The gimbal structure is a custom design which prioritizes high stiffness, low mass, and the ability to position the payload center of mass. This collection of characteristics yields a system which is easily steered, rejects resonance, and is statically stable. This final



Fig. 5: Gimbal subsystem mounted during field testing.

feature is critical as it reduces the necessary current for system stabilization. The gimbal is controlled by a BaseCam CAN_MCU brushless gimbal controller paired with separated motor drivers. The CAN_MCU controller provides a firmware interface for controller tuning, gimbal geometry specification, and communication protocol configuration. While some firmware configuration is required to tune the controllers for the specific structure and payload, the use of off-the-shelf boards reduces overall engineering effort. The CAN_MCU board also provides a serial interface for MAVLink gimbal protocol v2, reducing effort for software integration.

The choice of gimbal sensors is driven by the capability gaps of the fixed sensor suite. While the fixed Livox Mid360 advertises a 40 m range, in practice it is reliable up to 10 m. The Livox AVIA advertises a significantly increased range at the cost of a reduced field of view. By including the AVIA on the steerable platform, the overall sensing range of the USV is improved. While the OAK-1 does not provide a novel sensing modality (the fixed cameras include an RGB sensor), it is steerable and, critically, aligned with the AVIA allowing for the association of color data with long-range LiDAR data.

4) Water Shooter: The water shooter subsystem requirements are derived from both the features of the task as well as the capabilities of the existing USV. From the dimensions of the dock and target, a conservative requirement is that the shooter should be sized to continuously spray water onto a target 2.0 m away at a height of 1.5 m. From the vehicle side constraints on mass, dimensions, power consumption, and degrees of freedom can be derived. Selecting the degrees of freedom is most important. As previously discussed, the USV is underactuated and, when within the docking structure will be unable to maneuver beyond avoiding the dock itself. Therefore we design the system with two degrees of freedom - yaw steering and pressure control.

The water shooter system is illustrated in figure 6 and is split into the pump and servo components. The pump is a 12V water pump mounted low on the hull. The pump mount includes a shroud which serves to both reduce hydrodynamic

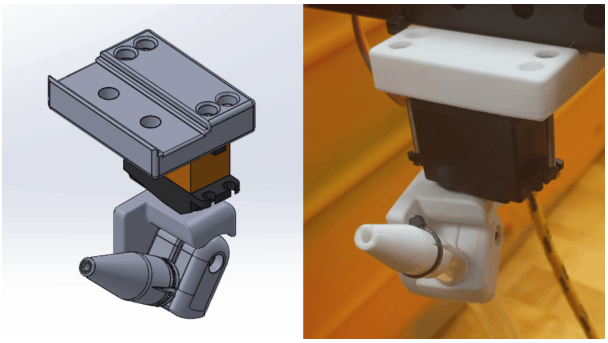


Fig. 6: The water shooter subsystem.

drag and filter debris in the water. The pump speed is regulated by a brushed motor controller which allows the range of the water stream to be varied. The servo subsystem mounts to the fore crossbar and points the water shooter nozzle to $\pm 90^\circ$ off of the longitudinal axis. The pump is rated to IP68 and suitable for submersion. The servo is a standard hobby servo and was tested for 30 cm of continuous submersion. The nozzle is 3D-printed with settings selected to reduce internal roughness and encourage laminar flow. The nozzle was sized through iterative testing to produce both a stream with long range and a sufficient flow rate to reject disturbances from wind.

C. Aerial Vehicle

This subsection describes the UAV hardware. Additional details are provided for hardware developed specifically to serve the competition strategy.

1) Base Platform: The UAV serves as the aerial extension of [13], complementing the surface vehicle with an elevated sensing perspective and payload delivery capability suited for the RobotX competition. The platform is a modified Holybro X650 quadrotor [14], equipped with T-Motor MN4014 330 KV motors and 17×5.8-inch carbon-fiber propellers. The assembled vehicle weighs approximately 6 kg with an overall flight endurance of 20 minutes, supported by a custom battery hot-swap system that connects four 6S batteries in parallel. This configuration increases the available battery capacity while maintaining a nominal voltage of 22.2 V and allows individual batteries to be replaced between flights without interrupting power to the onboard electronics. This reduces battery swap time during field testing and software development.

The onboard electronics mirror the USV architecture wherever practical to maximize hardware and software reuse across the two platforms. An Nvidia Jetson Orin Nano serves as the companion computer and handles perception and mission-level computation while a Pixhawk 6X running ArduPilot version 4.6 serves as the flight controller that performs state estimation and low level flight control. The power system regulates the battery bus to 12V for the companion computer and to 5V for the flight controller, networking hardware and payload release mechanism. A single downward-facing OAK-D camera provides visual sensing. Its compatibility with the USV's dual



Fig. 7: Payload delivery mechanism.

OAK-D PoE cameras allows camera interfaces and perception software to be adapted across the two vehicles.

2) Sensing and Localization: Like in the USV, GPS and IMU measurements support localization. The aerial vehicle additionally uses a radar altimeter for surface-relative altitude estimation. The selection of radar over the previously used LiDAR rangefinder was driven by the need for reliable altitude measurements over water. Near-infrared LiDAR returns are unreliable over water as the rays are reflected away from the sensor and even absorbed by the water in certain conditions.

3) Payload Delivery: The payload delivery subsystem requirements are derived from the activities required by *Coordinated Logistics* advanced tier. The UAV must deliver a preloaded tin of any color to the destination identified by the USV. At this level, the mechanism must support delivery, but not recovery.

Several delivery methods were considered, including electromagnetic retention, mechanical grippers, and servo-driven ejectors. Electromagnets offer a mechanically simple means of retaining and releasing the steel tins; however, they consume power during operation and would reduce UAV endurance. Mechanical grippers can be designed to consume power only when changing state, but are mechanically complex. Servo-driven ejectors consume power only during ejection and can be mechanically simple. For this reason, this approach is used on the UAV for payload delivery.

The tin delivery system, shown in figure 7 is a compact 3D-printed assembly mounted to the belly of the UAV. The system holds two tins, allowing for multiple delivery attempts. The tins are mounted side-by-side and held in place with permanent magnets. A single servo, commanded as a MAVROS actuator, can be used to eject either tin. The assembly was designed to minimize mechanical complexity and subsystem weight.

4) Safety and Failsafes: UAV safety follows some of the design from the USV; however it differs significantly in that failsafe behaviors involve landing instead of disabling hardware. Similar to the USV, pilot selection of manual, autonomous, and kill modes is handled over a dedicated RC link. ArduPilot, running on the Pixhawk, handles most of the behaviors that PRESto manages for the USV. ArduPilot failsafe settings are configured to safely handle events such as loss of RC link, loss of telemetry link, and low battery events. Similarly, ArduPilot is used to configure geofence behaviors.

D. Perception and World Modeling

The perception systems of the vehicles are designed to prioritize reusability and interoperability between sensing modalities and vehicles. Our approach to perception separates object detection by sensing modality. Uncertain information from visual and spatial sensing is fused at the detection level. Finally, fused detections which meet a minimum uncertainty criterion are added to a shared map.

1) Visual Perception: The visual perception approach builds off previous deployments of supervised learning-based detection. The perception pipeline was first evaluated in simulation using simulated objects and was subsequently applied to recorded real-world data using the same detection framework. This approach, wherein a model is trained on labeled data prior to deployment, has proven to be powerful and computationally lightweight given sufficient training data.

Both the USV and UAV carry Luxonis OAK-D stereo cameras with integrated Vision Processing Units (VPUs) capable of running inference directly on-device. Each system is also equipped with a Jetson Orin Nano, enabling additional image processing, most notably of larger neural network models. With this capable hardware, well-trained neural networks can detect objects in near real time. In previous competitions this involved collecting data from the course, labeling it, training a YOLOv11 model, and deploying it to the vehicles [15].

This approach is vulnerable to the domain shift between development and competition environments and therefore requires significant time be allocated at competition to collect and label data from both vehicles. This year, development efforts investigated methods to reduce the course time used for adapting the perception subsystem to the competition.

The first alternative investigated was a keypoint heatmap regression model [16]. Testing was conducted in simulation, where the known coordinates of objects within the Gazebo environment were used to establish ground-truth object locations. These coordinates were projected into the camera image and compared against the center points predicted by the model, allowing localization performance to be quantified using pixel error between the predicted and ground-truth centers. The heatmap regression model frequently produced predictions offset from the true object center. YOLO demonstrated better overall performance on the same simulated dataset, including lower center-point localization error and more reliable classification of different objects. These results supported the continued use of YOLO but failed to resolve the primary limitation of training time.

Another approach to the training time problem was the introduction of a few-shot classification system utilizing a vision foundation model, in this case MobileCLIP v2 [17]. In this approach, the vision foundation model converts cropped images of objects of interest into an embedding representing their features. Large vision foundation models are pretrained

on internet scale datasets, allowing them to serve as generalizing backbones. This backbone is harnessed by passing a few exemplar images of objects of each class through the model, then saving their encodings. During deployment, new images are encoded and their class is inferred based on their distance to each exemplar embedding set. While this method still falls squarely into the category of supervised learning, it requires substantially fewer labeled images.

The YOLO and few-shot approaches represent opposite points in a trade space considering in-situ data annotation time and computational cost. With effective use of the computational resources available, these two approaches are expected to complement each other. For example, YOLO can provide detections with low latency across broad classes, while the few-shot approach can discern between elements in the same broad class with specific details.

Detections from either source ultimately take the format of an image bounding box associated with a class. Using a pinhole camera model, bounding box centers are converted to detection headings and forwarded to the tracking algorithm. In the case of the UAV, multiple detection headings are fused onboard into a detection position before forwarding to tracking. This avoids overuse of network bandwidth.

2) Spatial Perception: LiDAR sensors on the USV provide direct measurements of obstacles in space. Position and range information from these sensors has relatively low uncertainty and is critical to include in the overall perception approach.

The spatial detection approach for RobotX 2026 remains largely unchanged from the existing approach in [12] and [11]. Pointclouds from all LiDAR sources are synchronized and combined. The combined data is clustered using the DBSCAN algorithm [18] where cluster criteria are tuned to exclude shore detections and small object noise. Clusters are forwarded to the tracking algorithm.

3) Detection and Tracking: The tracking system turns object detections from the sensors into full probabilistic estimates of persistent objects referred to as tracks. Detections, depending on their origin, contain both uncertain and incomplete object descriptions. Tracks have three values associated with them: existence probability score, class belief, and a spatial state (position and velocity).

The tracker consumes three distinct types of measurement, each with a unique noise model mapping to the data they provide. Detection bearings, provided by vision systems, constrain objects to a ray, but lack range data. Pointcloud cluster centroids, provided by LiDAR, describe an object position but contain little information about its class. Finally, position-resolved detections can be derived from pre-fused measurements. These are generated when cameras with overlapping fields of view simultaneously detect an object. The UAV also sends detections of this class as it performs onboard fusion to minimize network traffic. The tracking system can incorporate additional data streams with the definition of a new measurement function and noise model.

As new detections arrive, the Joint Integrated Probabilistic Data Association (JIPDA) algorithm is applied to attempt to associate them with existing tracks [19]. The probability that each measurement originates from existing tracks is computed. These probabilities are then used to update the existence probability score and spatial state of each track. In the case that no track was likely to have generated a measurement, that measurement is then used to create a new track. A track is created based on existence probability alone, regardless of classification information.

Incoming measurements may also contain object-class information, which is fused separately from the spatial and existence estimates. Class belief is, by design, maintained separately from other track data, as requiring consistent classification data to maintain or generate a track might result in potential obstacle avoidance hazards. Class detections are combined by maintaining a Dirichlet posterior over the possible object classes. Each associated detection updates this posterior in proportion to its association probability, so uncertain measurements contribute correspondingly weaker class evidence. Classification confidence therefore accumulates over repeated observations rather than being determined by any single detection.

Between measurements, existing tracks are propagated forward using an unscented Kalman filter and a constant velocity model. For static objects, such as buoys, a static prior is provided to the UKF. The use of a static prior, implemented as a zero or near-zero velocity assumption, ensures that noisy measurements of static objects are not attributed to some phantom motion, but are instead absorbed as position uncertainty. Incoming detections are compared against the predicted position of current tracks. This allows for objects to persist in the map even when new data does not arrive, whether due to occlusion, field of view limitations, or sparse LiDAR scans.

Tracks which receive no new data may be stale or faulty; however a missed detection provides evidence against a track only when that track was expected to be observable. The expectation that a new detection arrives is modeled per sensor and accounts for the field of view, range, and potential occlusion by other existing tracks. If a track is expected to receive new data, but does not, its existence probability deteriorates. This approach, in contrast to constant decay approaches, retains objects which are unlikely or impossible to observe.

4) Obstacle and Semantic Map: Two separate maps are derived from the tracking system, the obstacle map and the semantic map. The obstacle map is used solely for obstacle avoidance. It is populated by all existing tracks which exceed an existence probability threshold. These tracks become $Obstacle$ objects with radius corresponding to the class radius and inflated by the track spatial uncertainty. This list of $Obstacle$(s) forms an $ObstacleField$ which is provided to the path planner for navigation corrections. Task-prohibited regions/objects are also merged into the same obstacle field, giving the path planner a single description of all non-traversable space.

The semantic map is used for autonomous decision making. Tracks which exceed both an existence probability threshold

and class distribution entropy threshold are added to the semantic map. Objects added to the semantic map are persistent, so these thresholds are typically high. Information which is available a priori, such as the location of shore landing pads, can be directly added to the semantic map, bypassing the tracking system.

E. Navigation and Control

The navigation system is designed with the goal of cross-vehicle usage as both vehicles use ArduPilot as their main autopilot software and can be controlled through MAVLink/MAVROS commands.

1) Path Planning & Obstacle Avoidance: Both vehicles have to avoid certain regions, either due to tracked obstacles or due to task restrictions. Path planning helps ensure that these constraints are not violated and is an integral component of the navigation system.

This subsystem takes goal poses as inputs from the autonomy subsystem and provides sequences of waypoints to low level controllers. Paths are initialized as a set of waypoints in a straight line to the goal pose. If following these waypoints would intersect any obstacles or forbidden regions, an A* planner is triggered to identify a feasible path.

2) Vehicle Control: The UAV and USV use a setpoint-based control method, where the autopilot uses a tuned velocity controller to drive the vehicle through a set of target positions. As the UAV does not contend with obstacles, this control mode is used for nearly all tasks.

Because the USV is underactuated and sensitive to environmental disturbances, it is prone to drift off-course. For this reason, the USV also has the option to use Integral Line-of-sight (ILOS) guidance. In this mode, the guidance algorithm directly interfaces with the autopilot velocity controller. The line-of-sight algorithm provides stronger disturbance rejection and allows the vehicle to follow smoothed paths [20], [21].

F. Autonomy

The strategic focus on composable behaviors is most closely reflected in the autonomous decision-making software. Task behaviors should be built from reliable primitives and must be reconfigurable between runs. These two constraints primarily led us to adopt behavior trees (BTs) for the decision-making software this year, away from finite-state machines (FSMs) [22].

1) Behavior Tree Architecture: FSMs encode their control flow in the transitions between states, so each state becomes dependent on the states that enter and exit it. Behavior trees instead encode their control flow in the structure of the tree itself. Evaluation is driven by periodic "ticks". At each tick, a node is executed by its parent and returns one of three statuses: SUCCESS, RUNNING, or FAILURE. A small set of control nodes then decide what runs next: a *Sequence* runs its children in order until one fails, and a *Fallback* tries its children in order until one succeeds [22], [23]. Primitive behaviors report only their own outcome without knowledge of previous or following events. They are analogous to functions returning results to their callers. Additionally, nodes in the tree share a common memory, called the blackboard.

The behavior tree structure means the mission logic can be read directly from the shape of the tree, primitives can be reused unchanged across tasks, and the consequence of any modification is local to the subtree where it is made.

Because modifications have only local consequences, the engineering effort to develop autonomy and trace the effects of changes is reduced [24]. This is quantified by comparing the *editing complexity* of BTs to FSMs. Adding or removing a node from a behavior tree requires a single edit to its parent's list of children, a constant $\mathcal{O}(1)$ effort. The same modification in a fault-tolerant FSM requires $\mathcal{O}(n)$ edits in the number of states, since every transition into or out of the affected state must be updated [25], [26]. Hence, modifying such a FSM requires significantly higher engineering effort. The same holds for verification.

The top-level tree shown in figure 8 can be read from left to right and is organized by priority. The first, and highest priority layer is the safety gate, which monitors the E-stop and manual override state. When triggered, it halts every active behavior and aborts the mission. Below it is the pause layer which triggers when RoboCommand transmits an assistance request (*Dynamic Incident Response*). This halts the running task, saves its task and subtask progress to the blackboard, and switches control to the task 4 subtree. Below this is the mission layer which includes subtrees for each task. Task subtrees follow a common template:

- **Declare task identity and progress**: used for communicating vehicle status to RoboCommand.
- **Skip completed sub tasks**: allows the vehicle to resume at the correct point of a "paused" task after incident responses.
- **Control handoff on completion**: used to control the list and sequence of tasks performed.

Tasks are composed of a shared library of autonomy primitives, including go to pose, obstacle-aware go to pose, search patterns, service calls, and RoboCommand requests. This lets the team modify or assemble new tasks largely from existing, field-tested primitive behaviors rather than writing from scratch.

Previously, BT were implemented with `pytrees` [27], a python-based library, primarily due to the ease of use and the small learning curve for team members. This helped compress development for the short timelines. While practical for quick development and learning, `pytrees` was deemed less practical for RobotX because it significantly limits the tooling and capability of behavior trees. In `pytrees`, the tree structure is built alongside its primitive behavior implementations. This means every change to the mission logic is a code change. Separate XML file support to design the tree structure free of code changes, like the current autonomy, is only recently released, so it is still in development and is highly prone to bugs and failures.

In the interest of developing more robust and versatile autonomy BehaviorTree.CPP was used for RobotX 2026 de-

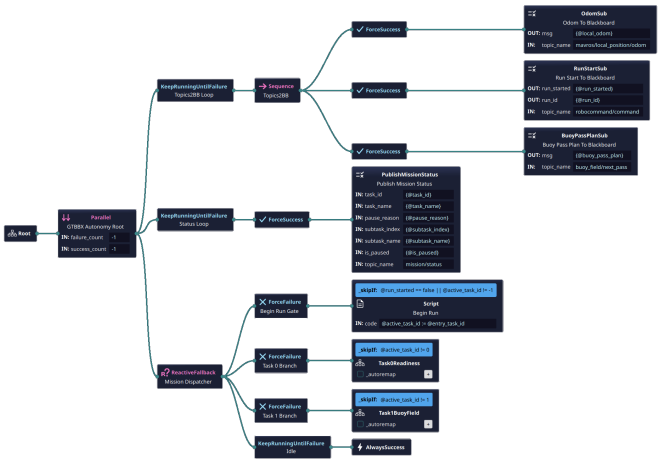


Fig. 8: Top-level Behavior Tree

velopment [28]. It defines the tree structure in XML, separately from the code implementing each primitive behavior. The companion tool, Groot2 [29], allows the tree to be edited graphically and monitored live in a human-readable format. This reduces the debug time during field tests because operators can monitor active behaviors and identify failure sources during execution. This change substantially increases information gained about our systems from each test. BehaviorTree.CPP is also the task engine of the ROS 2 navigation stack, Nav2 [30], [31], so its integration with ROS 2 actions, which our behaviors use to physically command the vehicles, is mature and actively maintained.

The cost of migration is porting our existing python tree and a steeper learning curve for team members. These costs were decisive under 2-month RoboBoat timeline, but are acceptable within the longer RobotX development window and intelligent software development plans of the team, where they are recovered through faster debugging on simulations and water.

2) Dynamic Re-Tasking: The benefits of BTs over FSMs is most apparent in *Dynamic Incident Response*, where the vehicle tasking is interrupted, then resumed. For an FSM with n mission states, this requires that every state include transitions both to and from a pause state, resulting in $2n$ transition edits. As previously discussed our behavior tree architecture places a pause layer above mission execution. In the event of an incident declaration, the running mission is halted. Once cleared from the RoboCommand, the mission executor subtree is ticked again. For our current mission tree of $n = 4$ task-level subtrees, an FSM needs $2n = 8$ transition edits, compared with a single child subtree insertion.

3) UAV Autonomy: The UAV mirrors the software architecture of the USV, including an independent UAV behavior tree. The UAV behavior tree matches the safety and dynamic response layering of the USV. Notably, the UAV safety layer also includes checks for software readiness, in addition to safety features required by the ruleset. Because the safety gate must be passed before takeoff, this ensures that all software is ready before flight and helps avoid spending flight time waiting for software to boot.

The UAV includes an additional layer above the mission executor which listens for tasking from the USV. While both vehicles are autonomous, this layer enforces a hierarchy wherein the less maneuverable USV can direct the UAV as needed. Tasks subtrees follow the same template as in the USV, making them well suited to retasking from both the USV and RoboCommand.

III. Testing Strategy

A. Testing Philosophy

Field testing is one of the team's most constrained development resources. Access to suitable on-water facilities requires significant travel and coordination, limiting field testing to a few hours approximately every two weeks. This also reduces the available personnel during tests, meaning that a small subset of the broader development team must deploy, operate, and debug both UAV and USV systems. These limitations motivate two principles that guide the team's testing process: test efficiency and cross-functional readiness.

Test efficiency aims to reserve field time for observations and interactions that cannot be reproduced effectively in simulation or in lab. Cross-functional readiness addresses a different constraint: the team cannot depend on the availability of a particular developer to operate or troubleshoot a subsystem. Although members retain primary technical responsibilities, field testing is used to distribute operational knowledge across the team.

B. Test Anatomy

The team follows a strict testing lifecycle which promotes both test efficiency and cross-functional readiness. Testing starts with the development of hardware or software. Software is tested on digital twins, while hardware, sensors, and control systems are verified at the subsystem level. After completing isolated testing, the integrated validation of that subsystem is assigned to the next scheduled field test.

Field tests are driven by testing plans which are treated as formal documents. Testing plans enumerate test objectives, sequencing, procedures, and data. They also include packing lists and debugging steps. Testing plans are written the week before a planned field test and peer reviewed. The day before a field test, the testing plan is rehearsed as a dry test in lab. Dry test rehearsals traverse the entire testing plan and validate that testing plans are complete. If a dry test cannot be successfully conducted, the field test is scrubbed. This ensures the limited "water-test" time is used efficiently.

During field tests, the test coordinator directs activities according to the testing plan. Team members are rotated through testing roles including piloting, OCS operator, and shore team. This reduces dependence on individual specialists and prepares the competition team to diagnose and recover from unexpected problems in the field. Data is collected as manual test logs as well as ROS 2 bag recordings. After field tests, a short debrief reviews results and identified failures,

which inform design revisions and objectives for subsequent tests.

C. Testing Outcomes

Following the aforementioned test philosophy and anatomy, the Marine Robotics Group staged experimental campaigns to systematically de-risk operational modes. Initial USV field tests were conducted at Sweetwater Creek State Park, whereas UAV-USV operations were executed on a private freshwater lake in order to satisfy FAA airspace restrictions. These tests served to validate UAV subsystem integration, USV subsystem integration, and SoS coordination.

Testing was structured on a bi-weekly field deployment cadence, balancing on-water empirical validation with adequate post-test log analysis and subsystem development.

1) UAV Outcomes: Following a review and verification of the initial UAV design, the UAV testing campaign began with flights conducted on the Georgia Tech campus. Initial testing focused on characterizing platform capabilities and evaluated the flight endurance, battery performance, and radio communication range. A series of flight maneuvers was also performed to automatically tune the flight controller gains and establish a stable flight behavior that is robust to wind disturbance. Finally, these tests collected documentation required for RobotX Proof of Readiness qualification. Beyond standard maneuvers, this involved rigorous testing of all safety subsystems.

The first demonstrations of coordination between the UAV and USV also occurred during the on campus testing campaign. The USV was placed on the ground in the flight area. The USV shared its location while being manually pulled across the flight area. The UAV tracked the shared position, demonstrating a foundation for coordinated operation between the two platforms.

After transitioning to above-water testing, the UAV perception systems were validated by mapping buoy fields. While exact buoy positions were not available, the UAV recreated the field layout and scale, indicating successful perception.

2) USV Outcomes: USV testing progressed from subsystem verification to integrated autonomous navigation, emphasizing perception, obstacle avoidance, and RobotX-specific hardware. Initial field tests demonstrated semantic buoy detection and spatial detection of unclassified obstacles; recorded field data was subsequently replayed to evaluate object tracking. Integrated tests then demonstrated path planning and execution around detected obstacles while navigating toward commanded waypoints. In the RobotX Proof of Readiness demonstration, this architecture was extended to semantic navigation: a behavior tree searched for and identified red-green buoy gates and autonomously navigated the USV through two gate pairs without collision.

The stabilized sensor gimbal was tested under manual disturbances and vessel motion. During on-water operation, it rejected pitch and roll disturbances from waves, maneuvering, and manual excitation while maintaining stabilization through approximately $\pm 180^\circ$ yaw and $\pm 45^\circ$ pitch, verifying the pointing capability required by the sensing modes in II-B3.

The water shooter was developed through incremental component and integration testing. Nozzle geometry was iterated until the required range was achieved, while the submerged aiming servo was exercised through sinusoidal pointing cycles and retested afterward. These tests verified the range, steering, and environmental requirements for *Coordinated Logistics*.

3) System-of-Systems Outcomes: System-of-systems testing focused on the information flows required for coordinated UAV-USV operation. During integrated field tests, the UAV detected and localized competition objects while station-keeping above the manually operated USV, transformed detections into the shared reference frame, and transmitted them to the USV. Information flow was also demonstrated in reverse, with the UAV using transmitted USV position data for station-keeping and tracking. These tests demonstrated bidirectional state exchange and UAV-USV perception transfer. Autonomous USV response to UAV detections remains a planned integration milestone for *Safe Passage*.

The ground-station communication path was tested using a mock RoboCommand server connected to the OCS. The system successfully transmitted the required heartbeat and completed a run declaration, verifying the basic vehicle-OCS-RoboCommand path. More complex tasking and dynamic retasking remain to be validated.

Integrated testing also exposed a network limitation: high-bandwidth sensor streaming saturated the wireless link and produced unusable communication latency. The network was subsequently redesigned from separate OCS links for each vehicle to a shared radio connected through a WiFi access point, and test procedures were updated to coordinate high-bandwidth streaming. This failure informed the communications architecture in II-A2.

IV. Acknowledgments

The Georgia Tech Marine Robotics Group gratefully acknowledges the financial and material support of the sponsors listed in Appendix B, which supported the development and testing of the RobotX 2026 system. In particular, we recognize ASTM International and Tocaro Blue for their testing sponsorship, Vectornav and Theia Technologies for precision navigation instrumentation and optical hardware, WaterLinked and Ainstein for acoustic and radar sensing, and NVIDIA for edge compute resources.We further thank the Georgia Tech Student Government Association for its significant continuing support of student activities and RoboNation for its generous donation of the USV platform used in the competition. Special thanks to Mahika Maini, Auryn Yamamura, Nicholas Trujillo, Olivia Taylor, Alice Chan, and Nicholas Lai for their key contributions to hardware and software development of the vehicles. Finally, we thank the Aerospace Systems Design Laboratory for the technical guidance and mentorship provided by its research engineers and graduate students throughout system development and testing.

REFERENCES

[1] S. Royo and M. Ballesta-Garcia, "An overview of lidar imaging systems for autonomous vehicles," *Applied Sciences*, vol. 9, no. 19, 2019.

[2] A. S. Alamoush and A. I. Ölçer, "Maritime autonomous surface ships: Architecture for autonomous navigation systems," *Journal of Marine Science and Engineering*, vol. 13, no. 1, 2025.

[3] N. AbuJabal, M. Baziyad, R. Fareh, B. Brahmi, T. Rabie, and M. Bettayeb, "A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots," *Sensors*, vol. 24, no. 24, 2024.

[4] J. Yan, Y. Jiang, X. Wang, J. Yu, X. Yang, C. Chen, and X. Guan, "Ubiquitous sensing of marine activities via the cooperation of autonomous underwater vehicles: J. yan et al.," *Scientific Reports*, vol. 16, no. 1, p. 2430, 2026.

[5] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, p. 3266, 2023.

[6] D. Callyhurn, D. Hurreeram, V. Oree, M. Gooroochurn, A. Ragen, Y. Kadell, and S. Ramjunun, "Design and development of a multi-functional robotic sea navigation, monitoring, and intervention system," in *2026 The 17th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT)*, pp. 184–189, IEEE, 2026.

[7] Y. Dong, M. U. Din, F. Lagala, H. Kuang, J. Sun, S. Yang, I. Hussain, and S. He, "Drone carrier: An integrated unmanned surface vehicle for autonomous inspection and intervention in gnss-denied maritime environment," *Journal of Ocean Engineering and Science*, 2026.

[8] S. Savitz, "Hybrid CoE Working Paper 34: Uncrewed maritime vessels: Shaping naval power in hybrid threat operations."

[9] B. Bartlett, M. Santos, M. Moreno, S. Dalai, C. Fahy, A. Dow, H. Gonzalez, P. Trslic, J. Riordan, E. Omerdic, D. Toal, and G. Dooly, "Cooperative robotic path planning for comprehensive bridge inspection with lidar technology: Navigating unknown structures," in *OCEANS 2024 - Singapore*, pp. 1–7, 2024.

[10] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995.

[11] "RoboBoat 2026: Technical Design Report Georgia Tech Marine Robotics Group."

[12] A. Wu, A. Dey, A. Chan, A. Martinez, A. Hussein, A. Ganjoo, B. Kasharaj, C. Tio, D. Mittelman, J. Gistren, J. Yun, J. Madrigal, J. Kim, J. O. Solano, K. P. Pande, K. Rajesh, K. Patherya, K. Sahoo, M. Woodward, M. Turton, N. Lai, N. Sathisha, N. Ponna, P. Mishra, R. Nguyen, R. Godha, R. Luo, R. Shibuya, S. Ibidapo, S. Fiz, S. Fish, S. Breton, S. Coutinho, T. Devlin, V. Deenadayalan, W. McConnell, W. Sonnleitner, Y. Chen, and Y. Guo, "Design of a Robust Autonomous Maritime System for RobotX 2024 and Beyond."

[13] T. Devlin, W. Sonnleitner, A. Chan, J. L. O. Solano, A. Wu, S. T. Fish, S. Ibidapo, C. Johnson, and D. N. Mavris, "Implementation of a Reconfigurable Unmanned Surface Vehicle Testbed for Multi-Agent Field Robotics," in *OCEANS 2025 - Great Lakes*, (Chicago, IL, USA), pp. 1–8, IEEE, Sept. 2025.

[14] Holybro, "X650 Development Kit." https://holybro.com/products/x650-development-kit. Accessed: September 22, 2026.

[15] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," 2024.

[16] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 529–545, 2018.

[17] F. Faghri, P. K. A. Vasu, C. Koc, V. Shankar, A. Toshev, O. Tuzel, and H. Pouransari, "Mobileclip2: Improving multi-modal reinforced training," 2025.

[18] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231, 1996.

[19] D. Musicki and R. Evans, "Joint integrated probabilistic data association - jipda," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, vol. 2, pp. 1120–1125 vol.2, 2002.

[20] W. Caharija, K. Y. Pettersen, M. Bibuli, P. Calado, E. Zereik, J. Braga, J. T. Gravdahl, A. J. Sørensen, M. Milovanović, and G. Bruzzone, "Integral Line-of-Sight Guidance and Control of Underactuated Marine Vehicles: Theory, Simulations, and Experiments," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1623–1642, 2016.

[21] E. Borhaug, A. Pavlov, and K. Y. Pettersen, "Integral LOS control for path following of underactuated marine surface vessels in the presence of constant ocean currents," in *2008 47th IEEE Conference on Decision and Control (CDC)*, pp. 4984–4991, IEEE, 2008.

[22] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Chapman & Hall/CRC Artificial Intelligence and Robotics Series, Chapman and Hall/CRC, 2018.

[23] M. Colledanchise and P. Ögren, "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees," *IEEE Transactions on Robotics*, vol. 33, pp. 372–389, Apr. 2017.

[24] C. I. Sprague, Ö. Özkahraman, A. Munafo, R. Marlow, A. Phillips, and P. Ögren, "Improving the Modularity of AUV Control Systems using Behaviour Trees," 2018.

[25] M. Iovino, J. Förster, P. Falco, J. J. Chung, R. Siegwart, and C. Smith, "On the Programming Effort Required to Generate Behavior Trees and Finite State Machines for Robotic Applications," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, (London, United Kingdom), pp. 5807–5813, IEEE, May 2023.

[26] M. Iovino, J. Förster, P. Falco, J. J. Chung, R. Siegwart, and C. Smith, "Comparison Between Behavior Trees and Finite State Machines," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 21098–21117, 2025.

[27] D. Stonier, N. Usmani, and M. Staniaszek, "Py trees ros library documentation," 2026.

[28] D. Faconti and M. Colledanchise, "Behaviortree.cpp library documentation," 2026.

[29] "Groot2: IDE to Create and Debug Behavior Trees."

[30] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A Navigation System," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[31] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses in the Wild," *Science Robotics*, vol. 7, p. eabm6074, May 2022.

APPENDIX A: TEST PLAN AND RESULTS

This appendix summarizes the team's testing approach and results across the algorithm, component, subsystem, and integrated-system levels. It documents the test objectives, configurations, evaluation criteria, observed outcomes, and any design or competition-strategy changes resulting from testing.

TABLE I: USV Test Plan Overview

Testing Level	Date	Objective	Test Scope	Environment / Setup	Evaluation
Physical vehicle	04/04/26	Verify physical vehicle: Verify that the boat hardware is buoyant and works in manual mode	Go through a set of buoys, go forward some distance, turn around, and go through the same gate	Set of buoys (try with both tall and small).	stbd motor keeps spinning after zero
Component	04/04/26	Verify individual hardware and software components before system integration. - Testing of the behavior tree, semantic map detections - Verify transform fixes	Sensors, actuators, communications hardware, safety systems, task-specific hardware, and software modules. - See if we can visualize topics using foxglove	Laboratory and bench-testing environments.	Depth map and matching buoy anchors
Physical vehicle	04/08/26	Verify physical vehicle works in manual mode	Go through two sets of tall buoys	Set of buoys (try with both tall and small).	Stbd motor keeps spinning after zero Orange boat motors keep spinning
Integrated System	04/04/26	Behavior tree + Perception + Mapping	Testing of the behavior tree, semantic map detections	Integrated field trials in representative maritime environments.	Depth map and matching buoy anchors Actual buoy anchors
Physical vehicle	04/23/26	Verify the boat is in a "competition ready" state	Go through two sets of tall/short buoys	Set of buoys (try with both tall and small).	
Integrated System	04/25/26 05/07/26	Verify that we can interface with the boat using QGroundControl Testing of the behavior tree, semantic map detections	Go through two sets of tall/short buoys	Set of buoys (try with both tall and small).	Success

TABLE II: UAV Pre-Deployment Tests

Test Scope	Description / Objectives	Methodology	Results
Battery Hot-Swap	Verification of uninterrupted power delivery during individual battery replacement.	Use of a bench setup with the hot-swap board, battery packs, adjustable DC power supply and a representative electrical load; replaced individual packs while monitoring the output.	Verified that the output voltage remained within the operating range of the connected electronics, with different input voltages to simulate powering with batteries of different charge states.
Power System	Verification of the 12V and 5V power distribution lines supplying the onboard electronics.	Completed wiring of the DC-DC regulators and verified that output voltages matched the specifications.	Final evaluation done by a mock bench assembly and wiring of the electrical components, confirming stable output voltage and proper operation under load.
Radio Communication	Characterization of the RC radio range and verification of the UAV-USV data exchange using ROS2 with Zenoh middleware.	Tested RC RSSI reception at increasing distances and exchanged vehicle data between the UAV and USV. Also evaluated alternative mounting locations for the RC receiver and inter-vehicle communication antenna.	Range testing indicated degraded communication performance when both links operated in the 900 MHz band, prompting a switch to 2.4 GHz for the RC link while retaining 900 MHZ for inter-vehicle communication.
Payload Drop-Off	Verification of payload retention and command release by the drop-off subsystem	Loaded the two payloads and actuated the release mechanism using ArduPilot's servo actuation endpoint.	Evaluated secure payload retention before actuation and successful release upon command.

TABLE III: UAV + USV Test Plan Overview

Testing Level	Date	Objective	Test Scope	Environment / Setup	Evaluation
Physical vehicles	06/06/26	Verify Drone radio issue fix Verify Drone + Boat communication in field with GPS tracking	Run radio connection test Run UAV arming test Setup Boat Test fly UAV using RC controller manually to verify antenna and gps Commence UAV + Boat topic bagging Commence UAV+USV tracking mission	Grass Field	Mavros gps setpointrel not set in
Components	06/16/26	- Test USV Readiness - Estop - Task 4	Setup Boat Activate Communication, perception, and semantic map Launch behavior tree for USV Readiness	2 sets of small red and green buoys about 2m apart and 18m from each other 1. Go back to the USV Readiness starting location 2. Perception launch 3. Autonomy 4. Response message	- Controller mapping issues: Deadzone for thrusters, Need to add arming to RC, Need to remap throttle to support backwards - Battery currently ballasted forward, nose dip - Relays bounce - Camera restarts the service - Hardware or driver issue? - Visual indicator for disarm - Disambiguate kill from arm - publish orientation marker to rviz for visual debug
Safety Features Test	06/20/26	Verify RTL fix for UAV+USV tracking Verify Manual Override from Guided/Auto to Loiter Manual Flight Over Landing Pad and Bag Data	Setup Boat Commence UAV + USV topic bagging Commence UAV + USV tracking mission Repeat tracking mission: flip the LOITER switch to take manual control of the UAV. Fly manually in a square loop and proceed to land	Grass Field	disconnected when landing: the drone still disconnects from antenna during manual and autonomous landing
Integrated Systems	07/11/26	Testing GoToPose Filming USV readiness video	Setup Boat Light Tower and Action Server Launch Test GoToPose algorithm Activate Communication, perception, and semantic map Autonomy Launch for USV Readiness	2 sets of small red and green buoys about 2m apart and 18m from each othe	Partial success
Comms Lake Test	07/31/26	UAV + USV GPS tracking	Setup Boat Run radio connection test Run UAV arming test Run UAV sensor health test Test 1:Commence UAV + USV topic bagging Launch the tracking node: Verify that the drone tracks the boat accordingly Repeat the same process (boat driven by RC) Flip the LOITER switch to verify node kills as expected Manually land the drone in the helipad Test 2: Obstacle Avoidance LIDAR Obstacle Detector Verify Obstacle debug topics	Two Robobuoys with 2m gap (expected gate width) Another extra buoy roughly in the vicinity, not very close to the gate	Couldn't kill bag node before landing RTL Everything else, success

Continued on next page

Table III (continued)

Testing Level	Date	Objective	Test Scope	Environment / Setup	Evaluation
Integrated System	08/15/26	Manual UAV readiness flight patterns Autonomous UAV readiness flight patterns	UAV topic bagging Run square, hourglass, and pirouette patterns autonomously Repeat all three patterns manually	Grass Field	- Autonomous takeoff successful - Drone disconnects from Heltec during both RTL and manual-override landings
Integrated Systems	08/22/26	Manual geofence and altitude test Autonomous geofence and altitude test	UAV topic bagging Run geofence test (10 m takeoff, 2 m/s) Fence set to 20 m max altitude, 30 m radius, RTL/LAND action	Grass Field	Success
Integrated Systems	09/13/26	Test gimbal and UAV perception Test BT.CPP autonomy triggered by RoboCommand Collect data bags for model training	Setup Boat Align UAV + USV in a common UTM frame UAV boat tracking at 15 m with topic bagging Obstacle avoidance and pure pursuit tuning	Lake	UAV tracking and bagging completed; USV autonomy bringup not tested Radio latency extremely high reaching $\sim$10000ms with 2 900MHz Heltec radios, making USV unusable Diagnosed as incorrect radio configuration on a radio device resulting in amplified latency when streaming high-bandwidth data (videos)
Integrated Systems	09/14/26	Water shooter servo wet test	Characterization of the amount of waterproofing needed for the servo.	Bucket of water Servo tester Servo	Verified that the servo worked well and further waterproofing was not required.
Integrated Systems	09/26/26	Data Collection (USV + UAV) UTM frame USV Autonomy (Bringup + Partial Task 1) UAV mapping UAV Perception	Setup Boat: Odom/Map TF connection, semantic map entries UAV + USV UTM Coordination Test Perception (semantics) Test PursueUSV algoritm Test BT.CPP-based auto Launch Autonomy Test Declaration and Auto trigger	2 Robobuoys 5 medium buoys Drop-off platform	USV mapping did not populate buoy semantic object even though yolo detections existed

APPENDIX B: SPONSORS

Platinum Sponsors	ASTM International, Tocaro Blue
Gold Sponsors	Vectornav, Theia Technologies
Silver Sponsors	WaterLinked, Ainstein, TDK Corporation
Friends of MRG	Blue Trail Engineering, Southern FiberWorx, NVIDIA