VORC Competition and Task Descriptions

2020 Virtual Ocean Robotics Challenge

1. Introduction

The purpose of this document is to outline the structure of the Virtual Ocean Robotics Challenge (VORC) for 2020, including the individual tasks which make up the competition. The details of the individual tasks, including their implementation and scoring, are under active development. This document, along with the VORC Technical Guide, are being made available as a means for competitors to provide feedback early in the design of this new competition.

2. VORC Goals and Approach

VORC is a means of supporting engineering development for maritime robotics. The tasks and scoring reflect this intent by emphasizing the foundational capabilities that lead to better autonomous performance. Testing autonomy algorithms in a relevant simulation environment is an efficient and cost-effective approach when compared to the challenges of testing system performance in a physical environment.

The simulation-based competition also rewards robust, repeatable performance by scoring each task over multiple trials where the environmental conditions (e.g., sea state, wind magnitude and direction, lighting, etc.) are varied between trials.

2.1. Submission and Evaluation

Details on the process for submission of software solutions are provided in the VORC Technical Guide, available at the VORC wiki. To summarize the process, evaluation of the VORC tasks will be done in a specified *evaluation environment* composed of computational resources consistent with the System Requirement for Running VORC. Team submissions, consisting of configuration and software, will be executed by the VORC technical team to generate task scoring.

The details of the evaluation environment will be provided to teams, along with working examples and tutorials to allow teams to thoroughly test their submissions within an equivalent evaluation environment. This should ensure that the performance of the team's submitted solutions will be equivalent to performance during development.

2.2. Competition Scoring

VORC scoring is inspired by the low-point system used in sailboat racing. Teams will receive a task rank for each task. The VORC score is the total of the task rank for all tasks with lowest total points ranked first and others ranked accordingly. The final ranking is assigned based on this score, with lowest total points ranked first and others ranked accordingly. For entries that are classified as did not start, did not finish or disqualified for a specific task, the task rank shall be equal to the number of teams competing in that task.

Table 1: Low-Point Scoring

Task Rank	Task Points
First	1
Second	2
Third	3
...	...

Task Ties: Typically ties in task rank are not possible. However, if a tie does occur, points for the place for which the teams have tied and for the place(s) immediately below shall be averaged. For example, if there was a tie for ranks fourth through eighth, all tied participants would receive a rank of 6.

Competition Ties: If there is a competition score tie between two or more teams, each team's task points shall be listed in order of best to worst, and at the first point(s) where there is a difference the tie shall be broken in favor of the team(s) with the best score(s). If a tie remains between two or more teams, they shall be ranked in order of their task points in the last task. Any remaining ties shall be broken by using the tied teams' task points in the next-to-last task and so on until all ties are broken.

To illustrate the scoring strategy, consider the following example:

Table 2: Competition Scoring Example

Task	Team A Points	Team B Points	Team C Points
1	1	2	3
2	2	1	3
3	1	3	3
4	3	2	1
5	2	1	3
Total	8	8	13

In this example, Teams A and B would be tied for first place in the competition and Team C would be in third place. The task points for A and B would be listed in order for each team. For both teams the ordering is 1, 1, 2, 2, 3. To break this tie the scores would be compared for Task 5; Team B has 1 and Team A has 2, so the tie is broken in favor of Team B. The final results would be Team B, first place; Team A, second place; and Team C, third place.

3. Competition Phases

VORC consists of two sequential phases during the last quarter of 2020.

- Phase 1: Hello World
- Phase 2: VORC final

Submissions for Phase 2 will be accepted over a week-long submission window to allow time to correct technical errors in the submission process. The final dates of the phases are posted on the VORC wiki page.

3.1. Phase 1: Hello World

This simple check encourages teams to start early and is a means to identify technical issues with the simulation environment. The goal of this phase is for teams to demonstrate that they have set up the VORC simulation environment locally, on their own computers, and to provide a means for teams to demonstrate prototype solutions to the VORC tasks.

- Preparation:
  - Teams have access to the VORC code, documentation and tutorials to support setting up their local development environment.
  - Teams have access to the competition documents: Competition and Task Descriptions, Technical Guide, etc., available on the VORC Wiki.
  - For technical support, teams are encouraged to submit to the VORC issue tracker.
- Submission:
  - Each team submits a video demonstrating the team's ability to run their own autonomy software within the VORC simulation environment. The purpose of the video is to document team status and progress towards completing the VORC tasks. While not required, it is expected that many teams will be able to demonstrate prototype solutions to a subset of the VORC tasks.
  - It is expected that the VORC simulation and the team's solutions (control and autonomy software) run locally on the team's computers.
  - Teams are encouraged to demonstrate successful solutions to as many of the VORC Tasks as possible.

- Evaluation:
  - Teams must submit a video to be eligible to participate in future phases.
  - Videos will be shared with the community, unless teams request their videos to be private.
- Next Steps:
  - Instructions for final submissions (VORC Technical Guide) and the evaluation environment infrastructure for automatic evaluation is available for teams.
  - Teams continue to develop solutions to the VORC tasks and begin testing solutions within the evaluation environment.

3.2. Phase 2: VORC final

In the final phase of the challenge, teams will be given a **week-long window** to submit their solutions for automatic evaluation within the evaluation environment as described in the VORC Technical Guide. The resulting task scoring will count toward the team's final rankings. Only one submission is permitted; however, during the submission window the VORC technical team will make all reasonable efforts to check that submissions are functioning properly and can be run in the competition environment (see Submission Period details, below).

- Preparation:
  - Teams provided with instructions to create and test within the VORC evaluation environment.
  - Teams are expected to test their solutions in their own evaluation environment, including running the automated scoring as preparation for the dress rehearsal submission.
  - Teams will have detailed documentation on the submission process (VORC Technical Guide) and tutorials on how to submit and score their solutions.
  - Technical support leading up to the submission period continues to be primarily based on submissions to the VORC issue tracker.
- Submission Period:
  - Following the VORC Technical Guide, each team submits their software for automatic evaluation and scoring within the evaluation environment.
  - Real-time technical support is provided during the submission window.
  - The intent of the window is to allow teams to correct errors in the submission format that might prevent the evaluation of their solution. In the event of a submission error, teams will be notified and given the opportunity to correct the error up to the end of the submission window.
  - To allow sufficient time to identify and remove such errors, teams are encouraged to submit toward the beginning of the window.
- Evaluation:
  - After the submission window closes, the VORC technical team will execute each team's solution for all of the tasks. The simulated environmental conditions (e.g., wind, waves, buoy locations, etc.) will be within the environmental envelope described in the VORC Technical Guide.
  - A scoring report will be generated for each team and for each task. The scoring process is detailed in the competition documents as well as the Task Tutorials.
  - Final task performance rankings will be determined based on automated scoring plugins documented below.
- Next Steps
  - Post-processing in preparation for December 2020, including finalizing scoring and creating highlight videos. Winners will be announced 2-3 weeks after the VORC final submissions.

4. Descriptions of Tasks

Each of the VORC tasks is scored with respect to the performance metrics described below. For each task, participating teams are ranked in order of the individual task score. The overall competition score is determined by the sum of the rank ordering for the individual tasks; the lowest score indicates the best performance.

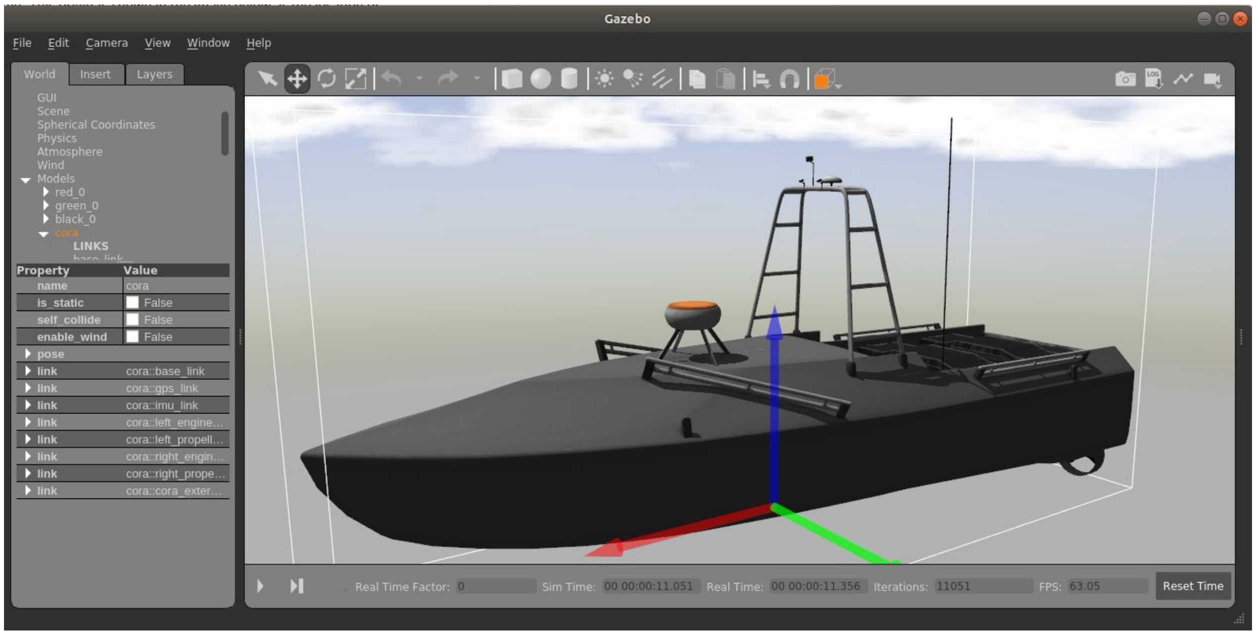
In addition to the descriptions below, the VORC project contains a VORC Tasks 2020 Wiki with working demonstrations for each of the tasks. Teams are encouraged to run these demonstrations to understand how the tasks will be implemented and scored.

General task information (e.g., task name, task status, etc.) are available in real-time through a ROS API. Each run of a specific task evolves through a set of sequential states Initial->Ready->Running->Finished. Participants should review this general task interface and structure as described in the VORC Technical Guide documentation available at the VORC Wiki.

4.1. General Definitions

4.1.1. Simulated Pose Reference

Task evaluation often involves the use of the true, simulated pose (position and attitude) of objects within the environment. For the CoRa, the pose is evaluated at the origin of the *CoRa* model frame, which is coincident with the `base_link` reference frame. This origin is shown in the image below as the location of the red-green-blue axes.

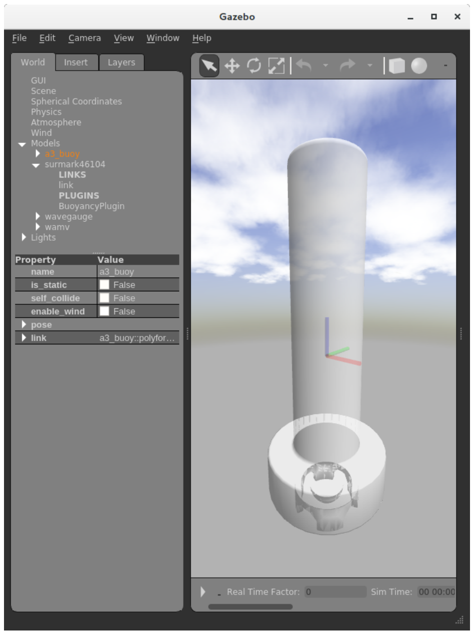


Property	Value
name	cora
is_static	False
self_collide	False
enable_wind	False
pose	
link	cora::base_link
link	cora::gps_link
link	cora::imu_link
link	cora::left_engine…
link	cora::left_propell…
link	cora::right_engin…
link	cora::right_prope…
link	cora::cora_exter…

Real Time Factor: 0 Sim Time: 00 00:00:11.051 Real Time: 00 00:00:11.356 Iterations: 11051 FPS: 63.05

Figure 1: Illustration of origin reference frame location for CoRa.

Similarly, for other objects in the simulation (e.g., buoys, markers, totems, etc.) the true, simulated location is the origin of the link frame associated with the object. The image below illustrates this reference frame for the surmark4601 buoy (Can Buoy).



Property	Value
name	a3_buoy
is_static	False
self_collide	False
enable_wind	False
pose	
link	a3_buoy::polyfor…

Figure 2: Illustration of origin link location for can buoy object.

4.1.2. Task Names

During every task the status of the task is published as a custom ROS task message over a ROS topic as detailed in the VORC Technical Guide. The name field of this task message uniquely specifies the current task, as detailed in the table below.

Table 3: Task Naming

Task	Task Message Name
Station-Keeping	`station_keeping`
Wayfinding	`wayfinding`
Landmark Localization and Characterization	`perception`
Black box search	`gymkhana`

4.2. Level 1: Control and Perception Fundamentals

4.2.1. Task 1: Station-Keeping

Capability:

System should be capable of performing localization by fusing sensor data (e.g., GPS, IMU, etc.) and maintaining USV position and heading in the presence of environmental forcing (e.g., wind and waves).

Implementation:

The performance in this task is quantified by the root-mean-square (RMS) pose error calculated as

$$\mathrm{RMS} = \sqrt{\frac{\sum_i^n \left[(x_i - x_o)^2 + (y_i - y_o)^2 + W(h_i - h_o)^2\right]}{n}}$$

where x_i, y_i, h_i is the true, simulated 2D position and heading at time i, x_o, y_o, h_o is the goal pose, n is the total number of time steps in the simulation and W is a weighting coefficient. The difference in x and y is in units of meters and the difference in heading is in units of radians.

The task is implemented in the following sequential states:

- *Initial*: The simulation is initialized with the USV in a random location within the operating area.
- *Ready*: The goal pose is published to the `/vorc/station_keeping/goal` ROS topic.
- *Running*: Scoring begins. For the purposes of development and debugging, the following items are provided as ROS topics:
  - The instantaneous pose error is published to the `/vorc/station_keeping/pose_error` ROS topic.
  - The current RMS error is published to the `/vorc/station_keeping/rms_error` ROS topic.

  Note: The above publications will not be available to team software during the final, scored runs of the competition.
- *Finished*: Scoring concludes.

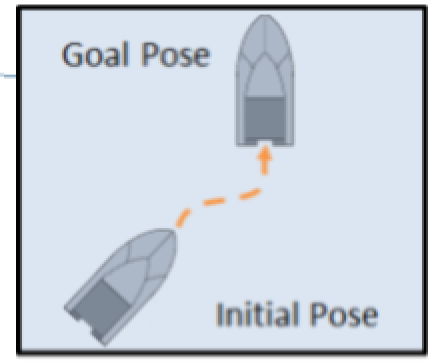


Figure 3: Station Keeping Task

Scoring:

- Run score is the RMS pose error for each run of the task.
- Task score is the mean of the run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest.

The RMS pose error is determined for each run of the task. The overall task score which determines the task ranking for each team is the mean RMS pose error over all scored runs.

Table 4: Station Keeping API

Topic Name	Message Type	Description
/vorc/station_keeping/goal	Geographic_msgs::GeoPoseStamped	The goal pose, consisting of a position in spherical (WGS84) coordinates and a heading, given as a quaternion.
/vorc/station_keeping/pose_error	std_msgs::Float64	A 1 Hz sample of the current 2D pose error metric, which summarizes the current difference between USV and goal in terms of position and heading.
/vorc/station_keeping/rms_error	std_msgs::Float64	The total RMS pose error accumulated over the duration of the run so far.

4.2.2. Task 2: Wayfinding

Capability: System should be capable of command and control of USV to achieve a series of given goal states, specified as a series of locations/heading values.

Implementation:
Teams receive a list of goal states which they are free to visit in any order. The simulation continues until the maximum time is exceeded. For each waypoint, performance is quantified by the minimum pose error achieved for that waypoint over the duration of the task. That is

$$E_{min}(p_j) = \min_i \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + W(h_i - h_j)^2}$$

where x_i, y_i, h_i is the true, simulated 2D position and heading at time i, p_j is the j^{th} waypoint, x_j, y_j, h_j are the position and heading of p_j, and W is a weighting coefficient. The overall performance for a given run is calculated as the mean of the minimum pose errors over all waypoints:

$$\bar{E}_{min} = \frac{\sum_i^N E_{min}(p_j)}{N} \qquad (1)$$

where N is the total number of waypoints.

The task is implemented in the following sequential states:

- *Initial*: The simulation is initialized with the USV in a random location within the operating area.
  - Note that in some run's obstacles may also be initialized in the operating area.
- *Ready*: The full list of goal states (waypoints) is published to the `/vorc/wayfinding/waypoints` ROS topic.
- *Running*: Scoring begins. For the purposes of development and debugging, the following items are provided as ROS topics:
  - The minimum error achieved so far for each waypoint is published to the `/vorc/wayfinding/min_errors` ROS topic.
  - The mean of the current minimum errors for each waypoint is published to the `/vorc/wayfinding/mean_error` ROS topic.

  Note: The above publications will not be available to team software during the final, scored runs of the competition.
- *Finished*: Scoring concludes.

Scoring:

- Run score is the mean of the "minimum pose errors over all waypoints" (1) for a single run of the task.
- Task score is the mean of the run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest.

Table 5: Wayfinding API

Topic Name	Message Type	Description
`/vorc/wayfinding/waypoints`	`Geographic_msgs::GeoPath`	An array of waypoints, each consisting of a position given in spherical (WGS84) coordinates and a heading given as a quaternion.
`/vorc/wayfinding/min_errors`	`Std_msgs::Float64MultiArray`	An array containing the minimum 2D pose error so far achieved between the USV and each waypoint.
`/vorc/wayfinding/mean_error`	`std_msgs::Float64`	The mean of the minimum errors so far achieved for all waypoints.

4.2.3. Task 3: Object Localization and Characterization

Capability:

Using perceptive sensors (cameras, LiDAR, etc.), the system should be capable of identifying, characterizing and localizing RobotX objects including buoys, totems, placards and docks. Perception should be robust with respect to vehicle motion (heave, pitch and roll) and environmental conditions (lighting, camera noise, etc.).

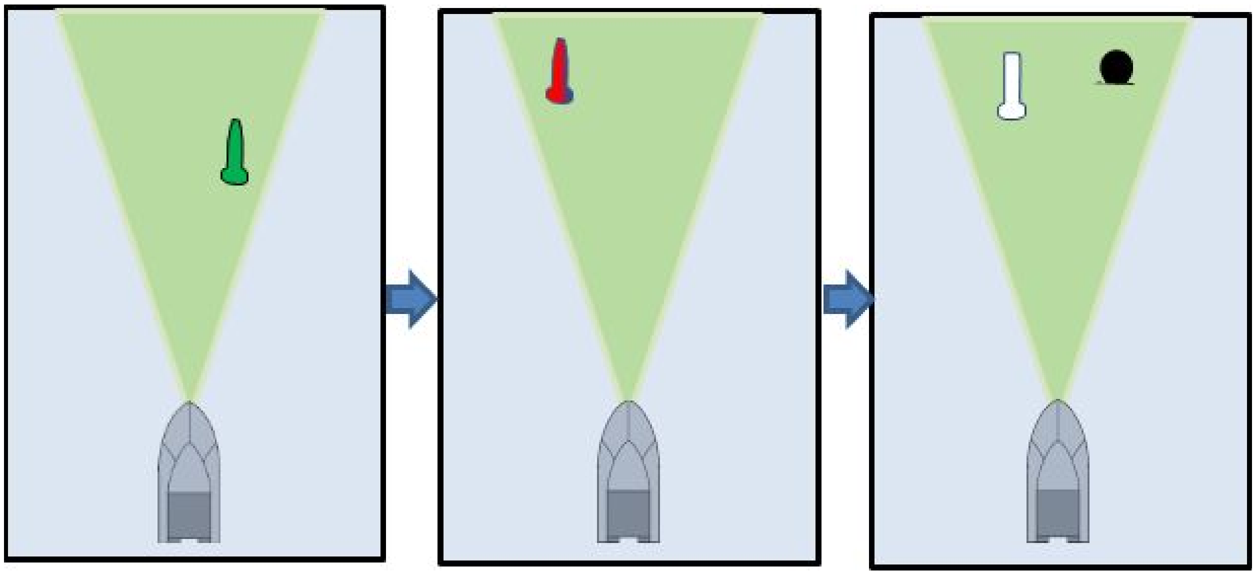


Figure 4: Three individual cases during the task as markers and objects are sequentially added to the field of view.

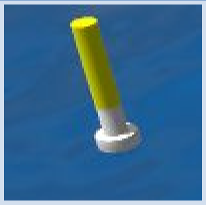
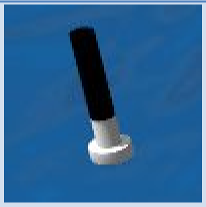
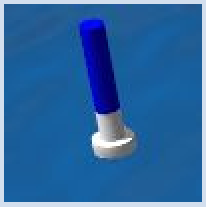
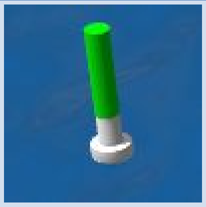
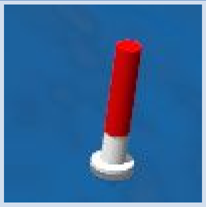
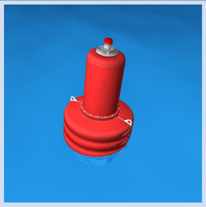
Implementation:

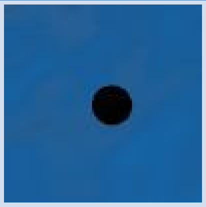
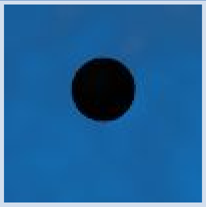
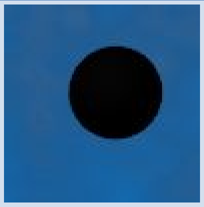
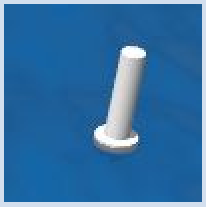
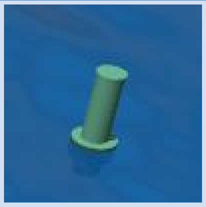
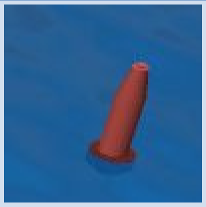
This *task* is made up of multiple *runs*, each under different environmental conditions. A single run is made up of multiple *trials*, where each trial consists of spawning one or more objects within the field of view of the USV and then removing the objects for that trial. Objects for identification and localization may include models such as navigation markers, totems and obstacles. The *trial time* is the elapsed time between the appearance and disappearance of an object. For an identification/localization answer to be scored, the message must be received during the trial time for that object. In competition, the trial time for all trials will be fixed at five seconds.

The task is implemented in the following sequential states:

- *Initial*: The simulation is initiated with the USV in a fixed location. The USV remains fixed for the duration of the tasks in the X (surge), Y (sway) and yaw degrees of freedom, but free to move in Z (heave), pitch and roll.
- *Ready*: No change -- the USV remains constrained in 2D position and heading.
- *Running*: Scoring begins. A series of simulated RobotX objects (Gazebo models) appear within the field of view of the USV. Teams will locate and identify the objects. Localization and identification is reported via the ROS API described below. For each trial, only one publication per object will be accepted - subsequent publications for that trial will be ignored.
- *Finished*: Scoring concludes.

Table 6: List of 3D objects to be considered in Task 3

3D object	Identification String
	yellow_totem
	black_totem
	blue_totem
	green_totem
	red_totem
	buoy_red

3D object	Identification String
	polyform_a3
	polyform_a5
	polyform_a7
	surmark46104
	surmark950400
	surmark950410

Scoring:
Scoring is designed to incentivize both correctly identifying and precisely localizing objects.

Localization error is defined as the horizontal distance between the location reported by the team and the true location in meters.

During each trial, one or more objects are spawned in the field of view of the USV. For each object in the field of view, an error value is added to the total error for the run:

- If the object is not identified, or incorrectly identified, an error value of 10 m is added to the total error.
- If the object is correctly identified and the localization error is greater than or equal to 2 m, an error of 2 m is added to the total error.
- If the object is correctly identified and the localization error is less than 2 m, the localization error value is added to the total error.

Scoring for the task includes the following:

- Run score is the mean error per object, calculated as total error divided by the total number of objects in the run.
- Task score is the mean of run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest

Table 7: Landmark localization and characterization API

Topic Name	Message Type	Description
`/vorc/perception/landmark`	`Geographic_msgs::GeoPoseStamped`	Teams report their estimated location (latitude and longitude) of a detected object. The identification of the object is reported using the message header `frame_id` string (see Table 6 for enumeration of object identifications).

4.3. Level 2: Integrating Control and Perception

4.3.1. Task 4: Black box gymkhana

Capability:

System should be capable of combining all level 1 tasks into a gymkhana. The arena is divided into two areas: the channel and the obstacle field. In the channel, teams should create and execute a motion plan to traverse a navigation course specified by red and green markers. After crossing the channel, the USV enters into the obstacle area. Here, the goal is to search and locate an underwater black box containing an acoustic pinger without hitting any obstacles. The obstacles are black buoys of different sizes. Once the black box is localized, USV will have to maintain its position as close to the black box as possible (on the surface of the water) until the end of the task. The solution should be robust with respect to environmental forcing and obstacles within the channel.

Channel implementation:

The navigation channel is defined as a series of gates, where each gate is a pair of colored buoys. The entrance gate is a white-red pair, the exit gate is a blue-red pair and the other gates are green-red pairs. While the layout of a particular navigation channel will vary, all competition navigation channel runs will satisfy the following constraints:

- The width of a gate, measured as the distance between the two buoys making up the gate, will be between 15-25 m.
- The distance between gates, measured as the distance between the centroid of the two gate markers, will be greater than the maximum of the widths of the two closest gates.
- The number of gates in an individual run can be between 2 and 5.
- The maximum run time (T) for each run is T=D/V, where V is a nominal speed of 0.25 m/s and D is the total piecewise linear distance between gates.

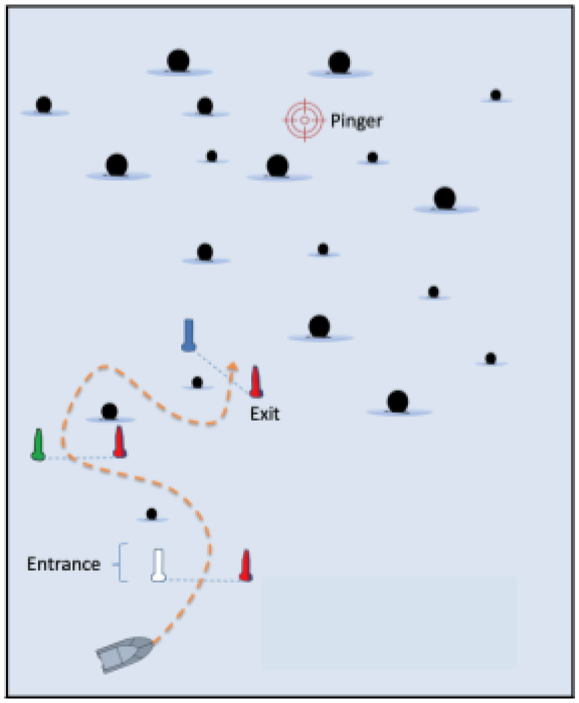


Figure 5: Black Box Gymkhana

Obstacle area implementation:

The obstacle area is an open water zone with an unknown number of black buoys of different sizes representing obstacles. The black box is located in this area at a maximum depth of 5 meters.

Each run of the task will progress through the following stages:

- *Initial*: The simulation is initiated with the USV in a random location, in the vicinity of the entrance gate (white-red buoys).
- *Ready*: The USV is free to move in all degrees of freedom.
- *Running*: The USV may begin traversing gates. The USV is considered to have passed through the gate if the reference point of the USV crosses the line connecting the reference points of the two buoys making up that gate. The direction of travel is specified as “Red, Right, Returning”. Gates must be traversed in the appropriate direction.
- Once the USV crosses the exit gate, it will enter the obstacle area.

- In the obstacle area, the USV should use its acoustic sensor to localize the pinger within the black box.
- The run lasts until the predetermined maximum time.
- *Finished*: Scoring ends.

Scoring:
This task's run score will be calculated according to the same criteria as task 1, except in this task the goal pose is the vertical projection from the black box to the surface of the water. In addition, the following rules also apply:

- The USV must cross the navigation channel exit gate to get a score. Otherwise, the score will default to the maximum error.
- Despite the above condition, the calculation of the error distance to the goal still begins as soon as the task enters the *Running* state. This is intended to incentivize speedy navigation of the channel.
- There's an additional 10 cm penalty for each collision with an object on the course.
- Task score is the mean of the run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest.

API:
There is no task-specific ROS API for this task. Development and debugging information, which teams may find helpful, is printed to standard output by the `navigation_scoring_plugin`, e.g., "New gate crossed!" or "Transited the gate in the wrong direction. Gate invalidated!