VRX Competition and Task Descriptions

2019 Virtual RobotX Competition *www.RobotX.org*

1. Introduction

The purpose of this document is to outline the structure of the Virtual RobotX (VRX) competition for 2019, 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 preliminary VRX Technical Guide, are being made available as a means for competitors to provide feedback early in the design of this new competition.

2. VRX Goals and Approach

The VRX competition is a means of supporting engineering development for the Maritime RobotX Challenge. Participation in this competition should raise the level of vehicle performance at the Maritime RobotX Challenge. 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 VRX Technical Guide, available at the VRX Documentation webpage. To summarize the process, evaluation of the VRX tasks will be done in a specified *evaluation environment* composed of computational resources consistent with the System Requirement for Running VRX. Team submissions, consisting of configuration and software, will be executed by the VRX 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 team's submitted solutions will be equivalent to performance during development.

2.2. Competition Scoring

VRX scoring is inspired by the low-point system used in sailboat racing. The VRX competition score is the total of the task points for all tasks with lowest total points ranked first and others ranked accordingly. The task score for each task is determined by the rank order of performance on that task as follows:

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 added together and divided equally.

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

The VRX competition consists of three sequential phases during the last quarter of 2019.

- Phase 1: Hello World
- Phase 2: Dress Rehearsal
- Phase 3: VRX Challenge

Final due dates are posted on the VRX 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 VRX simulation environment locally, on their own computers, and to provide a means for teams to demonstrate prototype solutions to the VRX tasks.

- Preparation:
  - Teams have access to the VRX 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 VRX Wiki and the RoboNation Forum
  - For technical support, teams are encouraged to submit to the VRX issue tracker.
- Submission:
  - Each team submits a video demonstrating the team's ability to run their own autonomy software within the VRX simulation environment. The purpose of the video is to document team status and progress towards completing the VRX challenge tasks. While not required, it is expected that many teams will be able to demonstrate prototype solutions to a subset of the VRX tasks.
  - It is expected that the VRX 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 VRX 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 (VRX Technical Guide) and the evaluation environment infrastructure for automatic evaluation is available for teams.
  - Teams continue to develop solutions to the VRX tasks and begin testing solutions within the evaluation environment.

3.2. Phase 2: Dress Rehearsal

To successfully complete this phase teams should have prototype solutions for the VRX tasks and should be able to submit their solutions for automatic evaluation within the evaluation environment as described in the VRX Technical Guide.

- Preparation:
  - Teams provided with instructions to create and test with within the VRX 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 (VRX Technical Guide) and tutorials on how to submit and score their solutions.
  - Technical support continues to be primarily based on submission to the VRX issue tracker.
- Submission:
  - Each team submits a solution as described in the VRX Technical Guide.
  - Teams are encouraged to submit standalone documentation of their solutions such as video demonstrations of their solutions. These submissions are optional.
- Evaluation:
  - To be eligible for Phase 3, teams must submit their solution for automatic evaluation as described in the VRX Technical Guide.
  - After the submission deadline, the VRX technical team will execute each team's solution for all of the VRX tasks. Execution is done on the same evaluation environment used for the final stage of VRX (see Phase 3: VRX Challenge, below).
  - A scoring report is generated for each team and for each task. The scoring process is detailed in the competition documents as well as the Task Tutorials.
  - A "leaderboard" is shared with all the competitors summarizing scores from the Dress Rehearsal.
- Next Steps:
  - Teams continue to develop and test their solutions. Teams will be supplied with software, instructions and computational specification to do their own automatic evaluation, emulating the final VRX scoring.

3.3. Phase 3: VRX Challenge

This final phase of the competition is where the task scoring will count toward the team's final rankings.

- Preparation:
  - Teams continue to develop and test their autonomous solutions based on results from the Dress Rehearsal.
  - Real-time technical support is provided for the last days/hours before final submission.
- Submission:
  - Following the VRX Technical Guide, each team submits their software for automatic evaluation and scoring within the evaluation environment.

- Evaluation:

  - The VRX team executes each team's solution for all the VRX tasks. The simulated environmental conditions (e.g., wind, waves, buoy locations, etc.) will be different from the Dress Rehearsal, but within the same environmental envelope described in the VRX Technical Guide.
  - Final task performance rankings will be determined based on automated scoring plugins documented below.
- Next Steps
  - Post-processing in preparation for December 2019, including finalizing scoring and creating of highlight videos. Winners and prizes will be announced 2-3 weeks after the VRX Challenge submissions.

4. Descriptions of Tasks

Each of the VRX 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.

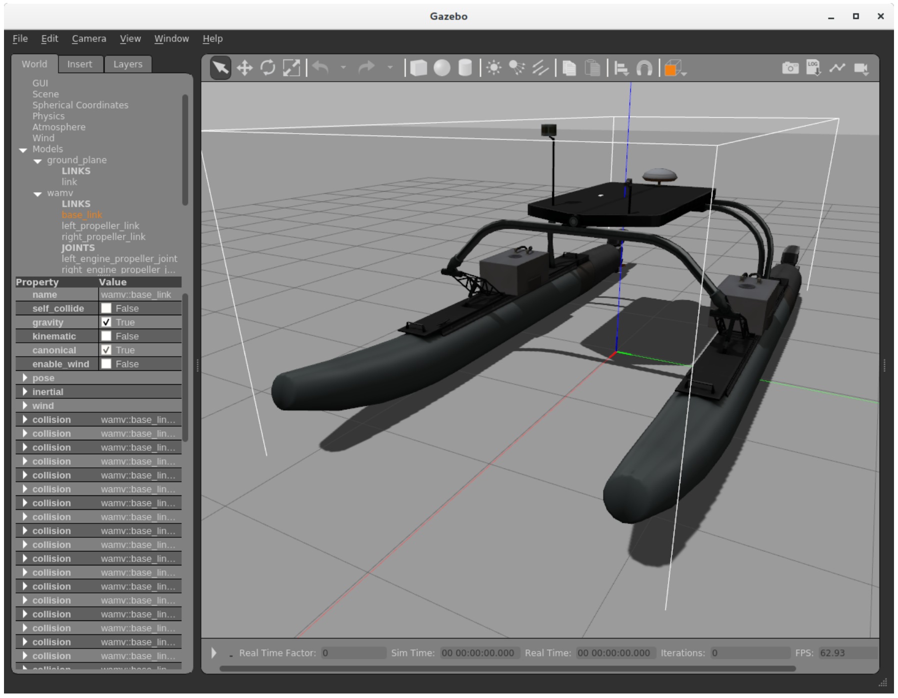
In addition to the descriptions below, the VRX project contains a VRX Tasks 2019 Wiki with working demonstrations for each of the tasks. Teams are encouraged to run these demonstrations to understand how the tasks will be implemented and scored.

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 VRX Technical Guide documentation available at the VRX Documentation 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 WAM-V, the pose is evaluated at the origin of the `wamv` 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.

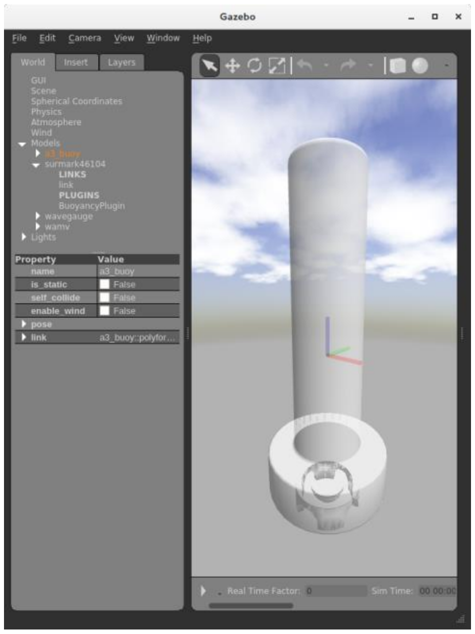


Property	Value
name	wamv::base_link
self_collide	☐ False
gravity	✓ True
kinematic	☐ False
canonical	✓ True
enable_wind	☐ False
pose	
inertial	
wind	
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...

```
Real Time Factor: 0   Sim Time: 00 00:00:00.000   Real Time: 00 00:00:00.000   Iterations: 0   FPS: 62.93
```

Figure 1: Illustration of origin reference frame location for WAM-V.

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...

```
Real Time Factor: 0   Sim Time: 00 00:00
```

Figure 2: Illustration of origin link location for can buy 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 VRX 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`
Traverse Navigation Channel	`navigation_course`
Dock	`scan`
Scan and Dock	`scan_and_dock`

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 `/vrx/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 `/vrx/station_keeping/pose_error` ROS topic.
  - The current RMS error is published to the `/vrx/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.

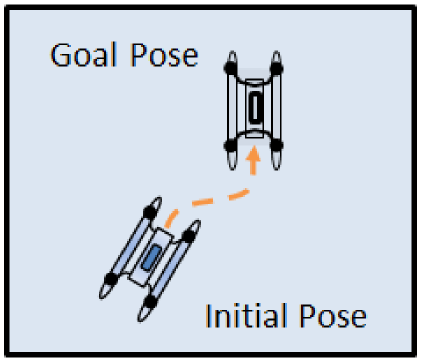


Figure 3: Station Keeping Task

Scoring:
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
/vrx/station_keeping/goal	Geographic_msgs::GeoPoseStamped	The goal pose, consisting of a position in spherical (WGS84) coordinates and a heading, given as a quaternion.
/vrx/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.
/vrx/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:

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

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 `/vrx/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 `/vrx/wayfinding/min_errors` ROS topic.
    - The mean of the current minimum errors for each waypoint is published to the `/vrx/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:
The error across all runs is the average of the final mean errors achieved for each individual run. The final task score is determined by rank ordering participants based on overall state error.

Table 5: Wayfinding API

Topic Name	Message Type	Description
`/vrx/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.
`/vrx/wayfinding/min_errors`	`Std_msgs::Float64MultiArray`	An array containing the minimum 2D pose error so far achieved between the USV and each waypoint.
`/vrx/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.), 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.).

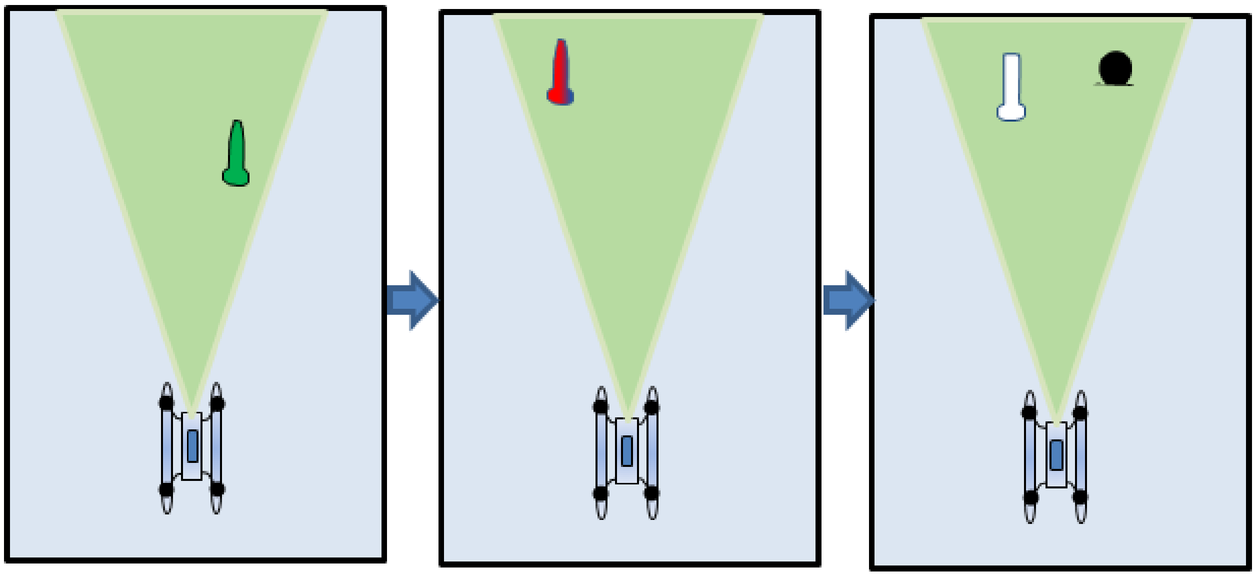


Figure 4: Three individual cases during the task as markers and objects are sequentially added to the field of view.

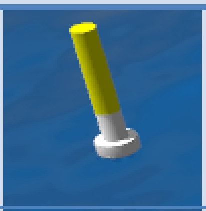
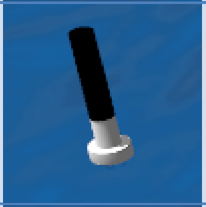
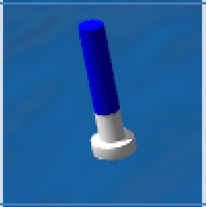
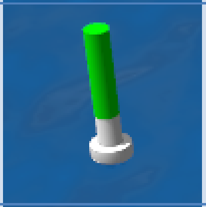
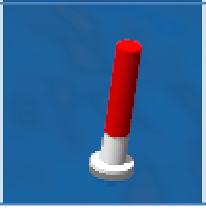
Implementation:

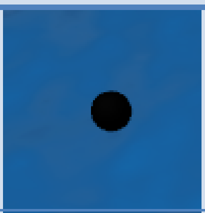
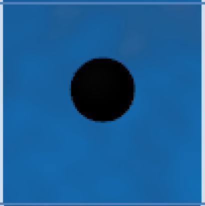
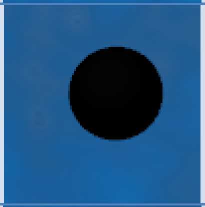
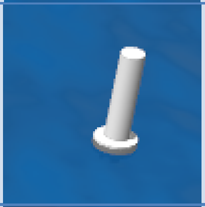
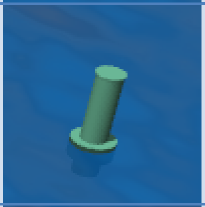
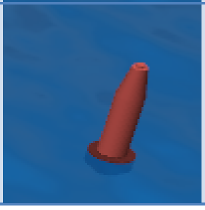
This *task* is made up of multiple *runs*, each under different environmental conditions. A single run is made up of multiple *trials*, where each trial consists of spawning one or more objects within the field of view of the USV and then removing the objects for that trial. Objects for identification 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

3D object	Identification String
	polyform_a3
	polyform_a5
	polyform_a7
	surmark46104
	surmark950400
	surmark950410

Scoring:
Two scores make up the overall task score.

1. The identification score is determined by rank ordering teams based on the total number of correct object identifications.
2. The localization score is determined by rank ordering teams based on position error between their estimated 2D position and the true, simulated 2D position of the object.

The overall task score is the rank ordering of the combined identification score and localization score. In the event of a tie, the higher rank goes to the team with the smallest total position error.

Table 7: Landmark localization and characterization API

Topic Name	Message Type	Description
`/vrx/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 3 for enumeration of object identifications).

4.3. Level 2: Integrating Control and Perception

4.3.1. Task 4: Traverse Navigation Channel

Capability:

System should be capable of creating and executing a motion plan to traverse a navigation channel specified by red and green markers. The solution should be robust with respect to environmental forcing and obstacles within the 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. Obstacles may be included within and around the navigation channel.

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 5-15 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 10.
- 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.

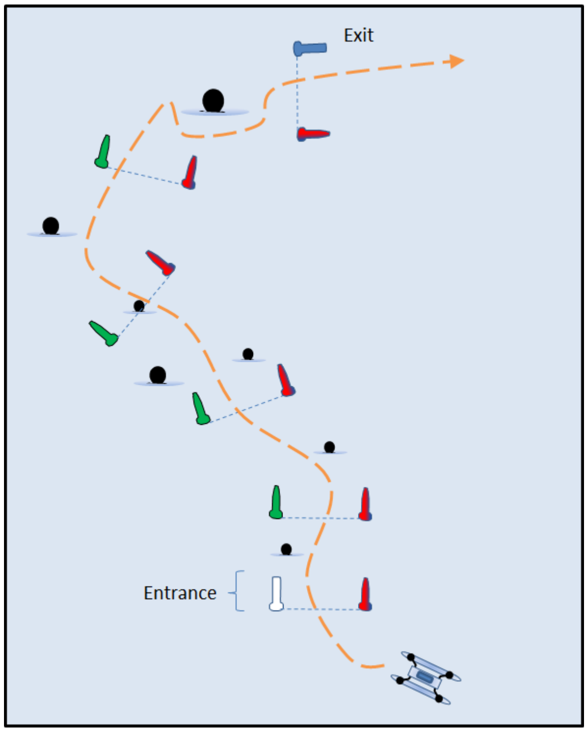


Figure 5: Navigation Task

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. The run lasts until the predetermined maximum time or the USV passes through both the entrance and the exit gates.
- *Finished*: Scoring ends and elapsed time for the run is recorded.

Scoring:

- Individual Run Score
  - To receive a run time the USV must traverse all gates in order, starting with the entrance gate and finishing with the exit gate.
  - Run time is the elapsed time between traversing the entrance gate and the exit gate.
  - A time penalty (e.g., 10 s) is added to the run for each collision with an object on the course.

  - The run score is the rank order of participants based on run time. Participants not successfully traversing all gates in order will be tied for last rank.

The total task score is the sum of the individual trial scores over all the simulation runs.

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!

4.3.2. Task 5 (Dock) and Task 6 (Scan-and-Dock)

Capability:
Given multiple docking bays (similar to the arrangement in the RobotX Challenge) the USV should be capable of deciding on the appropriate bay for docking, executing a safe and controlled docking maneuver and then exiting the dock.

These two tasks make use of virtual models of the "symbols" using the 2016 and 2018 RobotX competitions[1]. The AMS must demonstrate the ability to successfully dock in bays identified by a symbol. The symbols may be red, green, or blue in color and may be in the shape of a circle, triangle, or cruciform (cross) on a white background.

There are two related tasks associated with this capability.

Task 5 Dock:
The correct docking bay is determined by the color and shape of the placard. The correct color and shape is provided via the ROS API. A maximum simulation time is specified.

- *Initial*: The simulation is initiated with the USV in a random location in the vicinity of the dock. Each docking bay has an associated placard with a unique color and shape. The goal is to dock in the docking bay with the correct specified color and shape. The goal color and shape is published by Gazebo on the `/vrx/scan_dock/placard_symbol` ROS topic, indicating the correct docking bay for that run.
- *Ready*: The USV is free to move in all degrees of freedom.
- *Running*: The USV must dock in the correct docking bay. Successful docking consists of having the USV fully enter the dock, stay within the dock area for 10 s and then exit the dock. The USV may only dock once per simulation run.
- *Finished*: The simulation ends when the USV has successfully docked and exited or the maximum simulation time is exceeded.

Task 6 Scan-and-Dock:
This is identical to Implementation A except in this case the correct placard color and shape is not provided directly to the team via the ROS API. Instead, the team must read the color sequence from the scan-the-code buoy. The team may then report this color sequence via the ROS API using the `/vrx/scan_dock/color_sequence` service. The color sequence uniquely determines the placard color and shape indicating the correct docking bay, so once the system has correctly perceived the color sequence from the scan-the-code buoy, the USV should attempt to dock in the correct bay. The mapping between color sequence and placard color/shape is constant. For example, the three-element color sequence determines the placard color and shape as follows:

- The first color in the sequence determines the placard color: Red, Blue or Green.
- The last color in the sequence determines the placard shape:
  - Red = Circle
  - Green = Triangle

[1] Task descriptions for both 2016 and 2018 RobotX are available at https://www.robotx.org

    - Blue = Cruciform/Cross
- The middle color in the sequence can be any color that does not cause the same color to appear twice in a row in the sequence.

Scoring:
Points are accumulated for the following:

- Only the first successful docking event is scored per simulation run.
    - Docking (entering dock, staying within dock for a minimum time and exiting) in any dock (e.g., 10 points)
    - Docking in the correct dock, as specified by the placard color and shape (e.g., 10 additional points). A docking attempt is considered successful if the USV fully enters a docking bay between two of the adjacent pontoons and keeps that position for the minimum time.
- (Only Scan-and-Dock) Correctly reporting the color sequence from the scan-the-code buoy (e.g., 10 points).

The total task score is the sum of the points for all simulation runs. Any ties are broken based on the total elapsed time for all simulation runs.

Table 8: Dock and Scan-and-Dock Task API

Service Name	Message Type	Description
/vrx/scan_dock/color_sequence	vrx_gazebo::ColorSequence	The sequence of three colors perceived. Allowed values are "red", "green", "blue" and "yellow".
Topic Name	**Message Type**	**Description**
/vrx/scan_dock/placard_symbol	std_msgs::String	Contains the color and shape of a symbol with the format <COLOR>_<SHAPE>. Possible combinations are: red_circle red_triangle red_cross green_circle green_triangle green_cross blue_circle blue_triangle blue_cross