
Gazebo USV Simulation

Release 0.1

Alexandre Amory

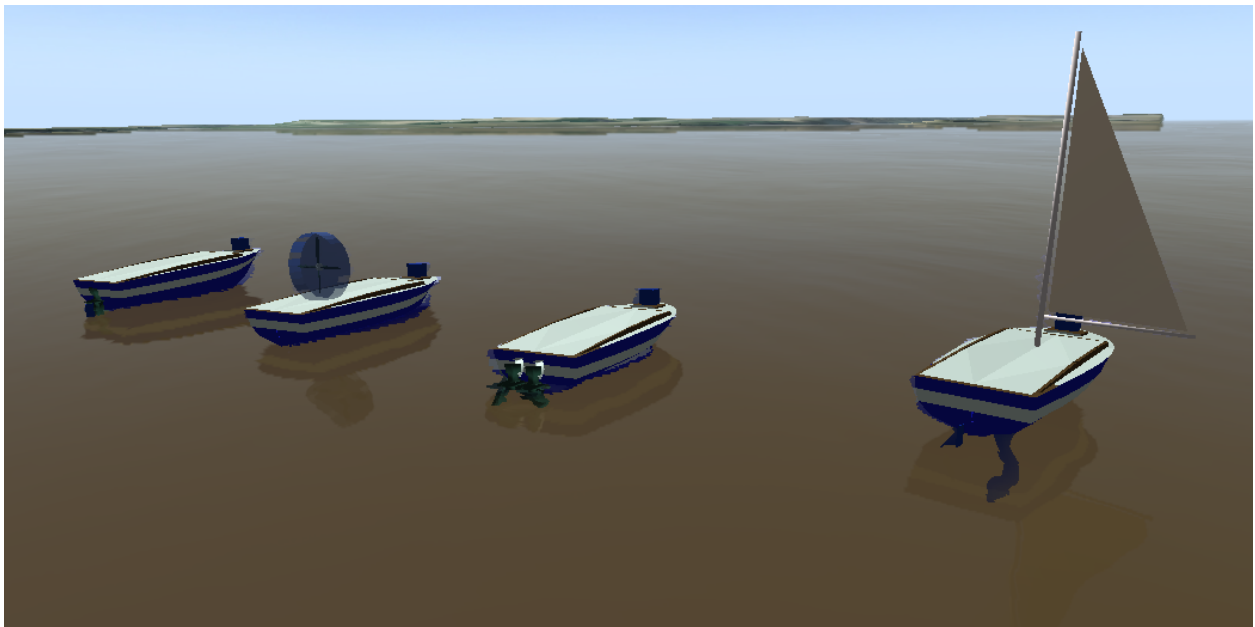
Mar 23, 2018

Contents

| | | |
|----------|--|-----------|
| 1 | About the Boats | 1 |
| 2 | Autonomous Systems Laboratoty - PUCRS | 3 |
| 3 | Summary | 5 |
| 3.1 | Getting Started | 5 |
| 3.1.1 | Depedencies | 5 |
| 3.1.2 | Compilation | 5 |
| 3.1.3 | Testing | 5 |
| 3.2 | Environmental Disturbances | 5 |
| 3.3 | Supported Boats | 6 |
| 3.4 | Evaluation Scenarios | 7 |
| 3.5 | Environments | 7 |
| 3.6 | Control Strategies | 8 |
| 3.7 | Contributors | 9 |
| 4 | Disclaimer | 11 |
| 5 | Feedback | 13 |

CHAPTER 1

About the Boats



talk a little bit about the boats and the purpose of this document.

show some youtube videos of the boats in action.

CHAPTER 2

Autonomous Systems Laboratoty - PUCRS

talk about LSA, link to the webpage

3.1 Getting Started

3.1.1 Dependencies

describe OS dependencies, Gazebo, other ROS packages, other OS packages

3.1.2 Compilation

describe how to compile the simulator

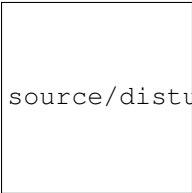
3.1.3 Testing

describe how to launch an example scenario

3.2 Environmental Disturbances

Describe how the disturbances work. describe water current and wind modules. describe the buoyance.

Figures: show the environment, the water speed view, and the wind speed view




source/disturbances/./images/env.png



source/disturbances/./images/water.png



source/disturbances/./images/wind.png



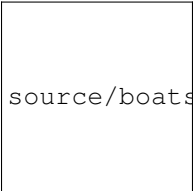
source/disturbances/./images/buoyance.png

Warning: @ To be done by Marcelo

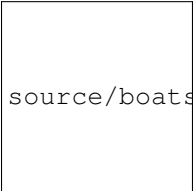
3.3 Supported Boats

Describe the available boats and how they were built.

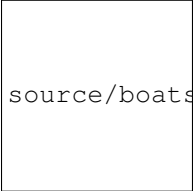
Figures: show each boat




source/boats/./images/airboat.png



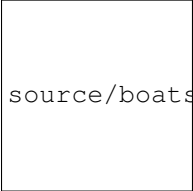
source/boats/./images/rudder.png



source/boats/./images/diff.png



source/boats/./images/sailing.png



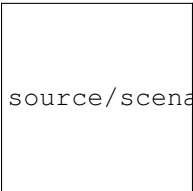
source/boats/./images/wam.png

Warning: @ To be done by Marcelo

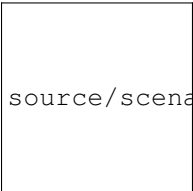
3.4 Evaluation Scenarios

Scenarios are variations of a environment which were built to evaluate the boats performance. List the existing evaluation scenarios.

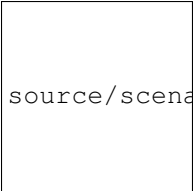
Figures: show the scenarios



source/scenarios/./images/straight.png



source/scenarios/./images/zigzag.png



source/scenarios/./images/circular.png

Warning: @ To be done by Marcelo

3.5 Environments

An environment describes a complete simulation scene. Cite the available environments.

Figures: show the Porto Alegre environmet



| |
|---|
| Warning: @ To be done by Marcelo |
|---|

3.6 Control Strategies

The robots of this simulator have a build in movement control that we use to test some scenarios. The main purpose of this control strategy is to get the vehicle from point A to point B, point A being the current position and point B being the desired or target position. The control strategy implemented is one of the simplest found in the literature, called heading control.

This heading control strategy consists of performing consecutive corrections of the robots heading by choosing appropriate actuators position/velocity. The idea of this control is as follows: if at each step the robot is pointing to the target position and if there's some forward velocity (in the robots frame) then it will eventually reach the desired position. Thus, what the controller does is receive the information necessary to adjust the robots heading, namely the current and desired position, then calculates the proper position of the actuator to reduce the error generated by subtracting the current and desired heading. The behavior generated by only using this raw erro to adjust the heading is pretty slow. To reduce this problem and set a desired behavior we use a PI strategy on the error. Another part of the control must be responsible for activating the thrusters power (or at least the actuator responsible for propulsion) properly.

A version of the control strategy describe above is currently running in all platforms of usv_sim. We chose this strategy due to its simplicity which makes it quick and easy to implement. The main difference between the implemented control strategies on the platforms is how the actuation is done. For example, in the Airboat the robots heading is changed according to the orientation of the fan. The Rudder Boat and Sailboat use the rudder to change their heading. The Differential Boat set different thruster power on the left and right thrusters to achieve heading change. The details of each control are shown below.

All of the controllers bellow use the same strategy to find the heading to be followed (pre-processing). First the controller gets information about odometry (current position) and target position, then uses these information to find the heading to be followed. After that, finds the error between the current heading and the desired heading and uses information of this error (sign and absolute value) to find the appropriate action to correct it and bring the platform to the desired heading. For each platform:

Heading control of the Airboat: Uses the heading error to find suitable fan orientation. Publishes the fan angle to /airboat/jointSetpoint.

Forward propulsion control of the Airboat: We implemented this behavior as a set of conditions. If the distance to target is greater than 10 meters the propulsion (fan power) is 100%. If the distance is between 2 and 10 meters then the propulsion is 50%. If the distance is less than 2 meters then the propulsion is set to 0%.

Heading control of the Rudder Boat and Sailboat: These platforms use the same rudder control, which consists of using the heading error to find suitable rudder angles. The rudder angle is published in `/boat_rudder/jointSetpoint` for the Rudder Boat and `/sailboat/jointSetpoint` for the Sailboat.

Forward propulsion control of the Rudder Boat: Similar to the Airboat.

Forward propulsion control of the Sailboat: Open and closes the sail according to the wind direction on the sailboats frame: head to wind (in irons) -> close sail (0 degrees). rear to wind (running) -> open sail (90 degrees). wind between these positions -> sail position chosen linearly between 0 and 90 degrees.

Heading control of the Differential Boat: Uses the heading error to find suitable velocities for both thrusters (left and right). The differential boat rotates according to the power of both thrusters. So, for example, to rotate clockwise (on its axis) a differential boat must use opposite power values on each thruster, for example, 10% on the right and -10% on the left thruster. So the controller check the error sign, choose a suitable direction of rotation and, according to the absolute value of the error, chose the thruster velocity.

Forward propulsion control of the Differential Boat: Uses a similar set of condition describe on the Airboat forward control. To get forward velocity in the Differential Boat it must set the same power in both thrusters.

| Heading Control | File |
|-------------------|--|
| Airboat | <code>usv_sim_lsa/src/usv_base_ctrl/scripts/airboat_control_heading.py</code> |
| Differential Boat | <code>usv_sim_lsa/src/usv_base_ctrl/scripts/diff_control_heading.py</code> |
| Rudder Boat | <code>usv_sim_lsa/src/usv_base_ctrl/scripts/rudder_control_heading.py</code> |
| Sailboat | <code>usv_sim_lsa/src/usv_base_ctrl/scripts/sailboat_control_heading.py</code> |

The videos below show the control strategies described above woking at different levels of disturbance (water current and wind).

3.7 Contributors

The list of contributors to this document.

- [@Alexandre Amory](#)
- [@Davi Henrique](#)
- [@Luiz Marcos Gonçalves](#)
- [@Marcelo Paravisi](#)
- [@Vitor Jorge](#)

CHAPTER 4

Disclaimer

The purpose of this document is for the use of LSA group only, but we open it in case it can be usefull for someone else. Everything you find here is without absolutly no warranty and I'm not responsible for any inconveniences or issues that might occurs.

CHAPTER 5

Feedback

Don't hesitate to ask about some additional info or next guides and also if you find some mistakes, please let me know. This can be done by submitting an issue or a push request on github.