TÉCNICO SOLAR BOAT

Técnico Solar Boat

Universidade de Lisboa - Instituto Superior Técnico

Technical Report - Njord Challenge

August 2021

Index

1 Introduction

Técnico Solar Boat is a university project, made up of a group of students from the most varied engineering courses of Instituto Superior Técnico. We have been working on the development of manned competition vessels powered exclusively by solar energy in the last five years. Recently we have also developed a new boat powered by Hydrogen. We compete with both prototypes in the Monaco Energy Boat Challenge that takes place in Monaco every summer. One of our main objectives is to promote renewable energies, sustainability, and electric mobility and demonstrating the potential of this sources in order for a greener future. Currently we are a unique project in the Iberian Peninsula, already counting on two fully working prototypes, SR01 (São Rafael 01) and SR02, with which we participated in three years of international competitions. In 2019 SR02 was awarded the 2nd place in our class.

After the development of three succesful solar boats and one hydrogen-powered boat, we were looking for ways to innovate in our project, and having a functional autonomous boat has been in our sights for quite some time. Achieving full autonomy would definitely be a step in the right direction.

Njord Challenge seemed like a great opportunity to start getting familiar with everything that's needed to achieve this goal. We've all learned a lot, since none of us had experience with working in autonomous systems. We would like to thank the Njord Challenge Team for their efforts in the organization of this competition.

With the purpose of fulfilling Njord's Challenge tasks, we design a system consisting of PID and MPC controllers and AI powered vision, all connected through ROS, which will be explained in greater detail in the pages below.

Firstly, it is important to explain how we approached the task we had in hand. Our initial idea was to use AI and vision to identify obstacles, fuse its' information with the lidar scanner, and utilize this data in our controllers. Therefore, our task division consisted of getting familiar with ROS, learning about AI and develop our own controller.

However, due to our lack of experience and tight deadlines, we weren't able to fully connect the AI with our ROS system, by the time of delivery of this report. Our solution as of now, consists in gathering all the relevant information regarding pose, twist and lidar scan, and providing it to our controller in order to get the boat moving. All of this will be further explained below.

Nevertheless, we are still hoping to be able to display our AI functioning by the time the competition is held.

2 ROS

In our application, ROS played a main role in connecting all the information, and communicating with the simulator. Learning how to function with it was one of the most important tasks of the project.

Our main goal was to ensure that our controller received all the important information for it to be able to plan a path and make the vessel navigate through it.

Firstly, we needed to convert the 3d lidar pointcloud into a 2d laserscan, which would be very useful to utilize in the path planning, since it filters the majority of points, and narrows it down to the relevant ones. For this, we installed the "pointcloud_to_laserscan" package, and created a node with all the important parameters for the conversion.

Then, we discovered a tool that was really helpful for the development of our controller - the pid package. This package creates a series of topics that are relevant for the controller, and will be further discussed below.

Having gathered all the relevant information, it was time to assembly everything onto the node responsible for the controller - the "control" node. This node subscribes to the simulator given topics "/nav/twist", "/nav/pose", to the one created by us "/scan" (which contains the converted laserscan information) and finally, the PID package topics "/forward_velocity_pid/state", "/lateral_velocity_pid/state", "/yaw_rate_pid/state", "/forward_velocity_pid/control_effort", "/lateral_velocity_pid/control_effort" and "/yaw_rate_pid/control_effort".

With all this information, this node procceeds to do all the required calculations in order to determine the needed input for the force controller. Finally, the node publishes this information into the "/force_control" topic, which then communicates with the simulator on Unity.

3 Vision

Vision assumes a very important role in autonomous vessels, once it provides the system with information that is vital to the correct functioning of the vessel. The points obtained with Lidar aren't enough to understand the surroundings, specially when dealing with different sea markers that have similar shapes, but different colors, and give information about what path to take. The challenge it's not just about avoiding obstacles, it's about having a clear understanding of the surroundings and using that information to define a path, and that is why Vision is so important.

The first obstacle we encountered was deciding which library and deep neural network architecture to use. Regarding this problem, we decided to use tensorflow 2.0 and an already built architecture, SSD ResNet101 V1 FPN 640x640 (RetinaNet101). In order to decide, we had to choose the best compromise between speed and mAP (mean average precision), and we had a lot of options to choose from, since there are a lot of already built models.

The second obstacle was to retract the data necessary to train our model, as that is a big part of developing an efficient AI. For that purpose, we ran some scenarios in the simulator with different sea markers and boats, then converted the video to frames, and then used those same image frames to label every single obstacle, by drawing boxes around. In order to label the images, we used CVAT (Computer Vision Annotation Tool), as it allows to have multiple people labeling the images, making the process easier and faster.

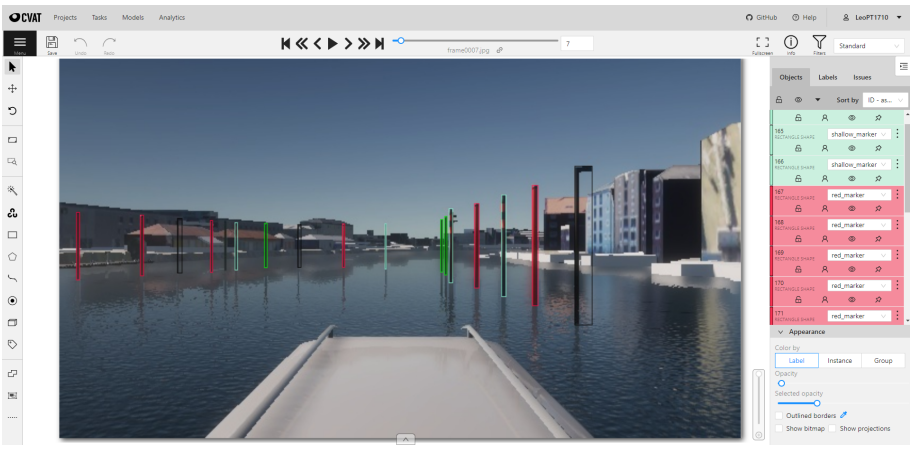

Figure 1: Labeling images from the simulator using CVAT

The reason behind having so many markers on the scenario is related to the fact that on early tests, we used only a few images with a few markers and the model wasn't able to perform well, so we decided to use a larger dataset with more markers, in an attempt to reach a model that would perform with a good precision.

The third obstacle was to find a way to train the model we chose, using the dataset we collected. As college students, our computers didn't have good enough graphic cards and RAM to train the model locally, so we opted to use Google Colaboratory, as it's free and allowed us to train our models without any problem.

4 Control

The control task was split into 2 loops: the inner loop, which controls the vessel's velocity and an outer loop, which controls the pose.

The inner loop consists of 3 PIDS, having the following inputs and outputs:

- Forward velocity PID: controls the longitudinal velocity of the vessel using the x force;
- Lateral velocity PID: controls the lateral velocity of the vessel using the y force;
- Yaw rate PID: controls the yaw rate of the vessel, using the moment in the z axis, n.

The first task was to tune the PID's, which proved to be harder than expected. As of now, the response is not satisfactory enough, however we are confident that with enough dedication we will be able to tune the PID's properly.

The trajectory controller has 2 objectives: arrive at the desired coordinates and avoid any obstacles. Among all the control laws, we found that MPC would be the most suitable for the job, given that it is able to predict future trajectories. This way, it is possible to attribute a high cost if the predicted trajectory hits an obstacle.

There are several ways of defining zones where the predicted trajectory should not enter, among which we have chosen to provide the points from the LIDAR sensor in a 360^{o} field of view, separated by one degree and having the MPC trajectories' not hit any of these circles. The main advantage of this procedure is that it simplifies the integration of sensor data into the MPC. For instance, had we defined circles or ellipsis around LIDAR points, and then feeding them to the MPC, additional work would be required to define these shapes properly.

The optimization algorithm chosen was the PSO algorithm. There are more refined algorithms built to tackle the MPC problem, such as Casadi or Acado, but we had surprisingly better results with PSO. Additionally, given PSO's nonlinear properties, any condition can be included in the trajectory generated by the particles, whereas in the case of quadratic programming or interior point methods one has to be more careful.

The MPC results where quite satisfactory in MATLAB, where we initially tuned the controller, however, we haven't tested it yet in the Unity platform.

5 Conclusion

To sum up, participating in the Njord Challenge has been a great opportunity to learn what it takes to develop software for an autonomous boat. Thanks to the simulator, we were able to test and optimize our ROS application, and perfect our controllers. Also, it enabled us to tackle the immense world of vision, and what it takes to create a AI for obstacle detection.

We're aware that there's still a long margin to improve and we just have scratched the surface of autonomous navigation. We look forward to continue working and discovering the amazing potential of this project.