Shell Eco-marathon competition car – 2022 – Backbone
Subsystem: AUTONOMOUS SIMULATOR

DNV Fuel Fighter 2021-2022

Teams channel Autonomous **Subsystem responsible** Joey Cheng, James Premraj

Involved Joey Cheng, James Premraj

31.07.2022

Contents

1 Objective space **3**
1.1 Description . . . 3
1.2 Goals . . . 3
1.3 Risk . . . 3

2 Knowledge space **3**
2.1 Former knowledge . . . 3
2.2 Research . . . 3

3 Solution space **4**
3.1 Solution description . . . 4
3.2 Issues and improvement needed . . . 5
3.3 Future work . . . 6
3.4 Personal reflection . . . 6

4 Path planning **7**
4.1 Filtering . . . 7

1 Objective space

1.1 Description

The Autonomous Simulator is a built environment in Gazebo, which is an open-source 3D robotics simulator. The environment consists of the Fuel Fighter car with its numerous sensors and its control system, and a world. The world could be a race track, a parking lot or an empty space, depending on what the other sub-subsystems prefer and what they want to test. The purpose of the Autonomous Simulator is to test the various subsystems in the Autonomous group, such that the car will drive as desired in the annual Autonomous competition.

1.2 Goals

The goal of this subsystem is to provide a simulator for testing the different subsystems. Furthermore, the subsystem strives to create an environment which is as similar to the real world as possible. This includes, but is not limited to, the dynamics of the car, the steering, the sensors and the world environment. Thus, the Autonomous group has the opportunity to test systems without the risk of financial loss, damage of the car and injuries of test personnel.

1.3 Risk

The main concern is that the car's behaviour in the simulator differs too much from how the actual car will behave in the real world. As the simulator is a controlled environment, it is impossible to replicate the exact behaviour of the car. Significant differences could lead to serious consequences if the subsystems would rely too heavily on a poor simulator.

2 Knowledge space

2.1 Former knowledge

The simulator uses the latest ROS distibution called ROS noetic. A more extensive description of the simulator can be found in "Simulator and control system" from 2021 by Tor Børve Rasmussen.

2.2 Research

A significant part of the fall semester was used to thoroughly understand ROS and Gazebo. Here is a list of sources we would recommmend that you familiarize yourself with:

- http://wiki.ros.org/ROS/Tutorials

- https://classic.gazebosim.org/tutorials
- A Youtube playlist called "ROS In 5 minutes
- https://answers.ros.org/questions/

Furthermore, most of the implementation is done in a URDF file, thus it is important to understand the structure and syntax of these kinds of files.

If you are unfamiliar with using the terminal (Linux), it is highly recommended that you watch a few Linux tutorials on Youtube.

3 Solution space

3.1 Solution description

This year's solution is built on the 2021 system created by Tor Børve Rasmussen. The 2021 simulator consisted of an Audi R8 from the audibot package, a Lidar sensor, and a few walls in an empty world. The car is controlled using the bicycle model, with a commanded velocity and an angular velocity as inputs. As the Lidar and control of the car was quite good, it was decided that we would rather look on adding other elements to further improve the system. A more suitable environment was implemented, in addition to the implementation of several other sensors. Moreover, the car was improved such that it better resembled the actual car, both physically and visually.

A racetrack was made for this year's system. The chosen racetrack is from the Github repository aws-robotics. The object models were used to create different racetracks with different widths and differing numbers of twists and turns [1].

The car is visually different, since the Audi mesh is exchanged with the mesh of the Fuel Fighter car from 2021. In regards to the dynamics of the car, some of the parameters were changed with the data we received from a previous member from the design/mechanical group.

Front track width	1220.783mm
Rear track width	1112.783mm
Wheelbase	1915.242mm
Wheel radius	281.852mm
Wheel thickness	105.00mm
Wheel mass	1.924E5
Bodyheight	1023.936mm

Table 1: Data given from mechanical/design

A few estimations were made in regards to the data. The body mass was for example set to 60kg.

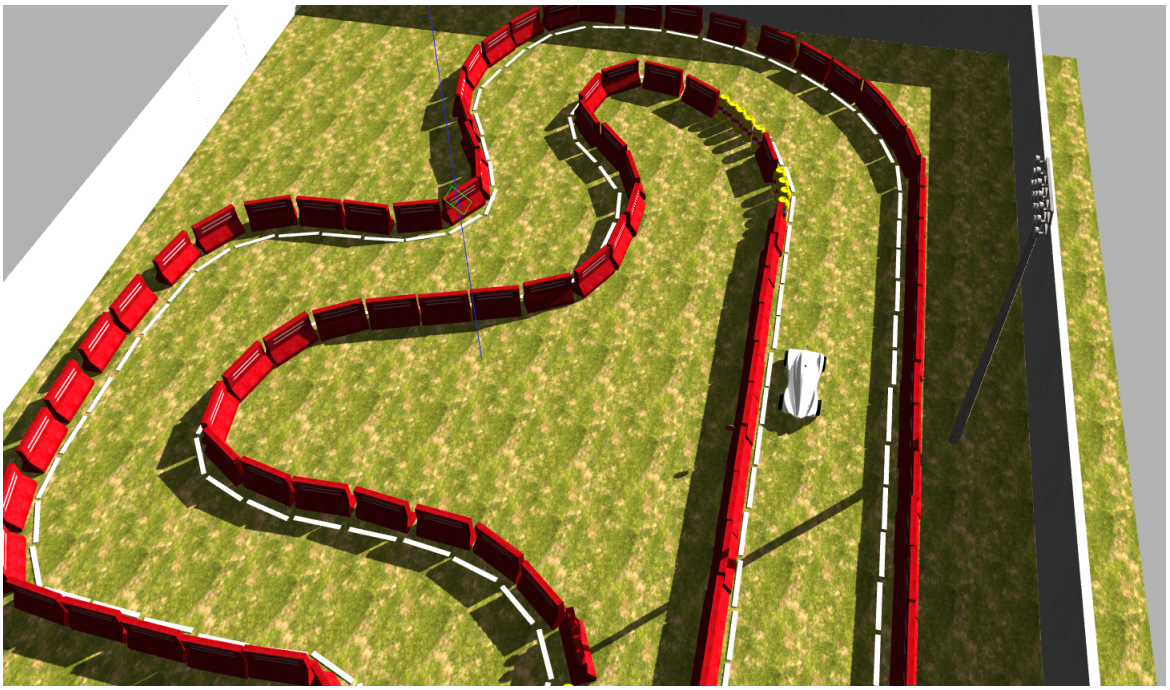
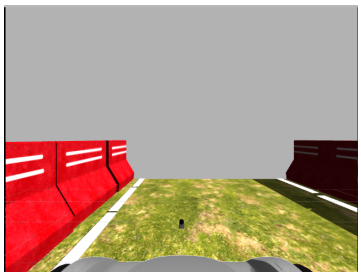


Figure 1: One of the racetracks

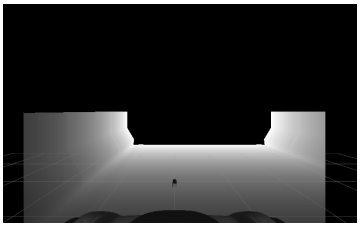
A few of the sensors we implemented for this year's car was a laser scan and a depth camera. Both sensors were mainly used by the Camera sub-system to test their implementations and changes to their systems. Enabling these sensors in Rviz will affect the real time factor drastically. Thus, it is advised that you only enable them if needed.

3.2 Issues and improvement needed

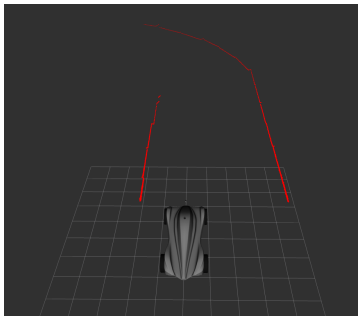
The simulator is not perfect, as it has some issues and bugs. In the environments where the white lines on the racetrack are included, the lines will not be rotated in their correct orientations when starting the simulator. This error doesn't appear every time, and restarting the simulator will solve this problem.



(a) Color view of the depth camera



(b) Depth view of the depth camera



(c) Laser scan

Figure 2: Three simple graphs

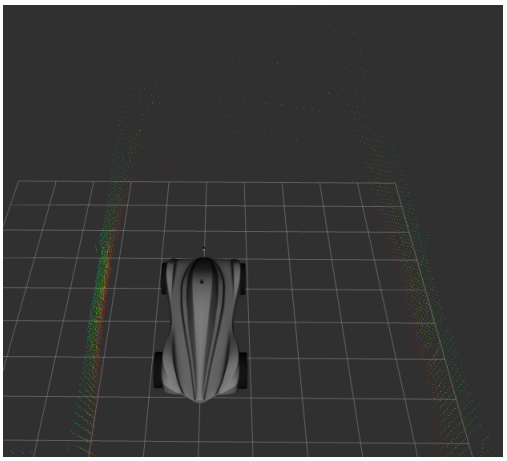


Figure 3: Point cloud from lidar

The real time factor is also noticeably high when running the sensors. This is an issue if you want to test the sensor while driving the car, as driving the car while it is lagging can be quite frustrating.

The breaking of the car is also rather weak. We don't know if the break torque in the simulator is too small compared to the actual car, or if the break torque of the car is indeed that small. Thus, a better estimation of the breaking force should be made. It is noteworthy that if the break torque is too high and the car is driving at a high velocity, a sudden breaking will result in the the car flying.

3.3 Future work

The tuning parameters of the PID controllers could be improved to get a smoother acceleration and breaking. Moreover, the parameters of the car could be further investigated in order to implement a more accurate car dynamic in Gazebo.

In the future, the people working on the simulator will have to communicate with the other sub-systems to get inputs on what needs improving. The main goal of the simulator is after all to test the other sub-systems in the Autonomous group. As the other sub-systems change and improve, the simulator will have to adapt to fit their needs.

Consequently, we would suggest that the next year's student(s) working on the simulator should have a clear overview of what the other sub-systems would want to test, and how they want to use the simulator throughout the year. This will give you a clear guideline on what needs to be improved.

3.4 Personal reflection

Working with the Autonomous simulator has been a great learning experience. At times, the learning curve has been quite steep. Thus, looking back, it is very clear that we have learned

a lot about ROS, Gazebo, Linux and how it is to work in a technical student organization. There has been a lot of debugging and frustration, but also a lot of fun working with the group and learning from each other.

4 Path planning

There was also made an attempt on implementing the global planner from the navigation stack. The user would select a goal in Rviz, which is a point in a particular costmap, and the global planner would create the shortest path from the car's position. At the moment, the issue is that the map is a predefined image of a costmap, thus it is easy to set a goal. However, in the Autonomous competition, we would have a map that updates itself regularly. Consequently, it would be hard to set a specific goal at every given time.

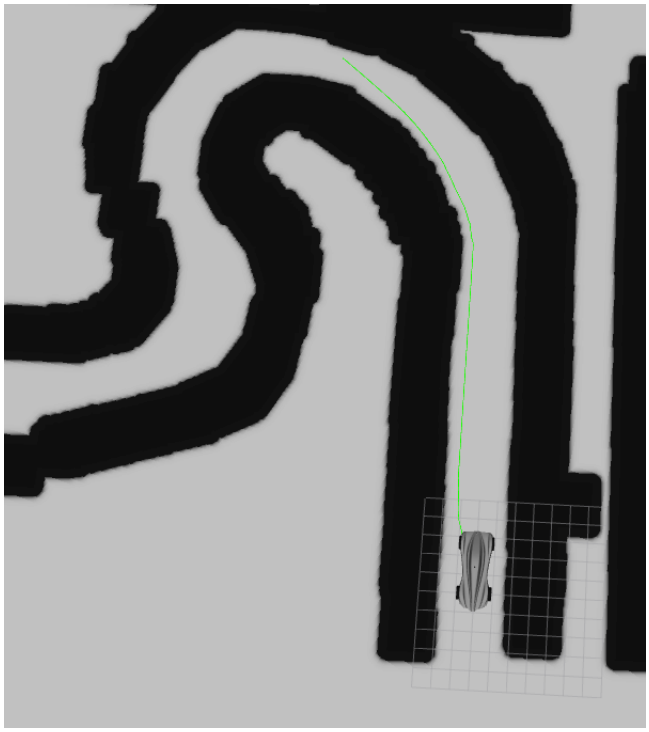


Figure 4: Global Planner

4.1 Filtering

Furthermore, we also worked on filtering the Slam and Lidar map from the Lidar/Slam subgroup. We did'nt end up using this solution. However, it was very rewarding to play around with the filtration of the maps. Check out this website, Morphological Transformations, if you would be interested in learning more about this.

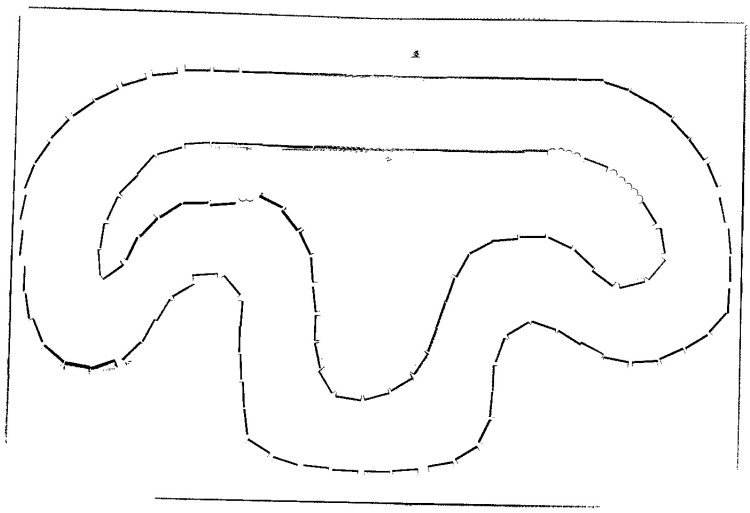


Figure 5: Unfiltered Lidar map

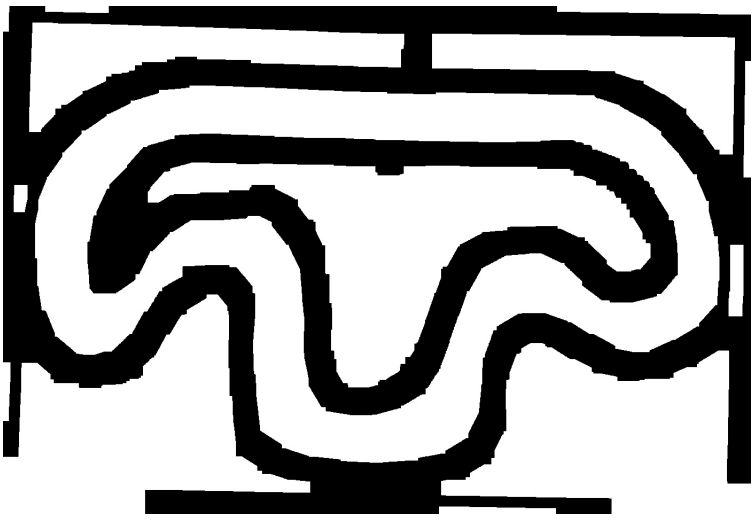


Figure 6: Filtered Lidar map using OpenCV

References

[1] ahtsan. *aws-robomaker-racetrack-world*. https://github.com/aws-robotics/aws-robomaker-racetrack-world.