Chapter 6: Co-simulation Between Autoware and MATLAB with a Simulink Model

In this chapter, a detailed guide will be provided which is intended to demonstrate co-simulation between Autoware and MATLAB & Simulink. A Simulink model consisting of subsystems for twist filter, vel pose connect, and plot of instantaneous ego vehicles' position along with MATLAB file for wf simulator were run together in conjunction with Autoware to demonstrate interconnectivity between MATLAB and Autoware.

It should be noted that Autoware's version 1.8.0 has been utilized for this task in order to avoid compatibility issues. Furthermore, Autoware Toolbox was specifically implemented for this version.

The following steps outline the complete process of running a Simulink Model in conjunction with Autoware. For complete and detailed overview of Autoware installation, please visit the following link:

- https://gitlab.com/autowarefoundation/autoware.ai/autoware/wikis/Source-Build

In order to avoid installation error, after step 3 under "For version 1.11.1 or older" (namely install dependencies using rosdep) in the link above, please run the following additional command provided below (in red) which is missing from the documentation on Autoware's website.

```
$ rosdep update
$ git submodule update --init –recursive
$ rosdep install -y --from-paths src --ignore-src --rosdistro $ROS_DISTRO
```

The rest of the process is the same as in the link above.

Furthermore, ***computing.yaml*** file must be modified slightly to include the two of the additional computing modules that are not available in the newer versions such as 1.8.0 or 1.12.0. These two modules are named as "way_planner" and "dp_planner". This ***computing.yaml*** file is found in Autoware folder, and then the parts related to "way_planner" and "dp_planner" must be uncommented so that they can appear in Autoware GUI.

Step 1: Run Autoware version 1.8.0 using the following commands:

```
$ cd autoware/ros
$ source devel/setup.bash
$ ./run
```

Step 2: After the Autoware Runtime manager window pops up on the screen, under the “Setup” tab, Vehicle Model will be selected. The path to the file for vehicle model can also be seen in the following screenshot. Vehicle Model button can be clicked to select it.

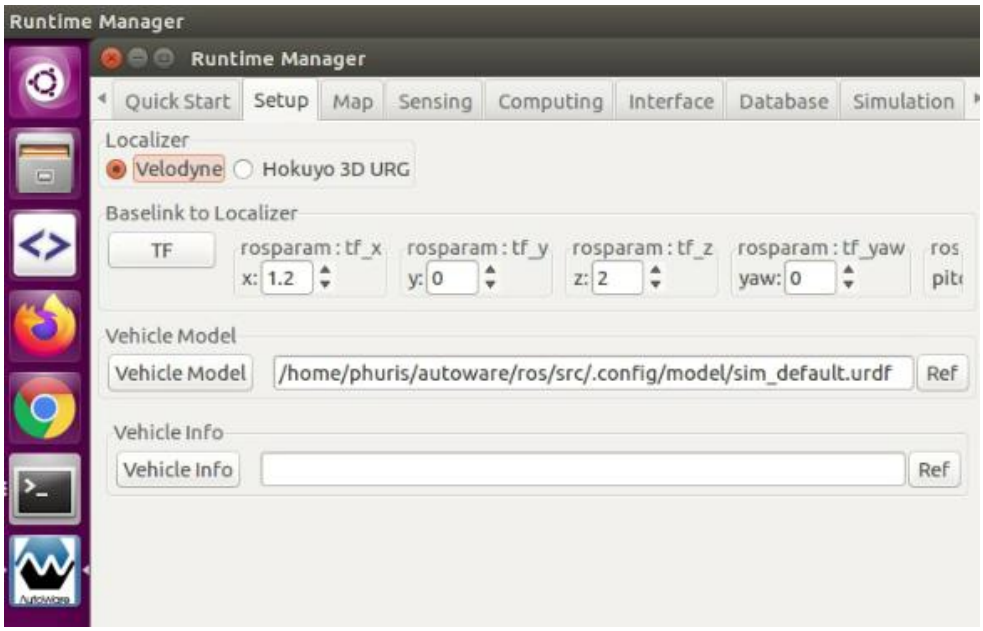

Figure 1: A screenshot of the Setup Tab to select Vehicle Model

Step 3: Next step is to setup the “Map” tab. The file paths used for Vector Map and TF can be seen on the screenshot that follows.

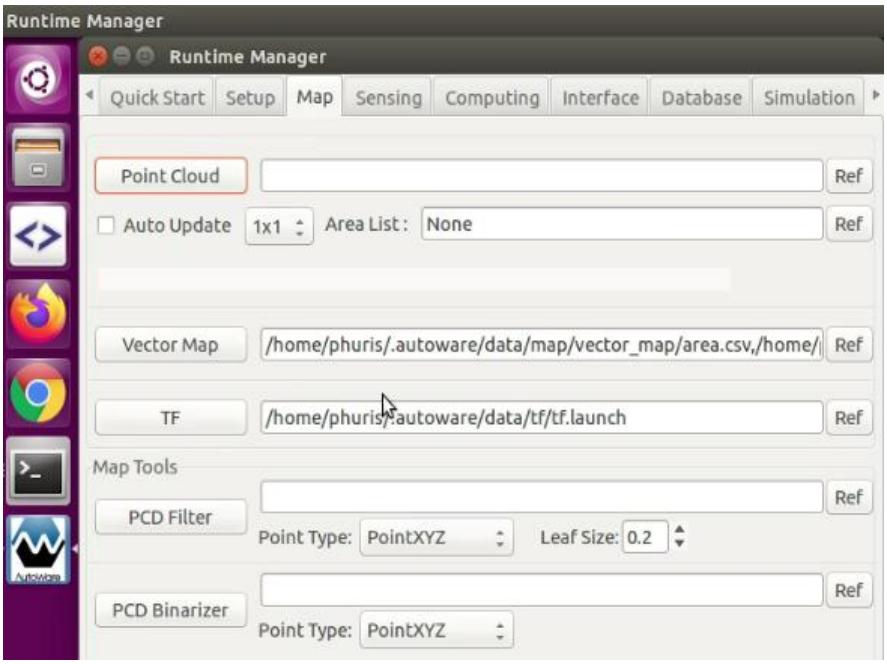

Figure 2: A screenshot of Map tab

Step 4a: In the computing tab, we will first click on the *op_global_planner* app and perform following modifications so that the relevant options are selected.

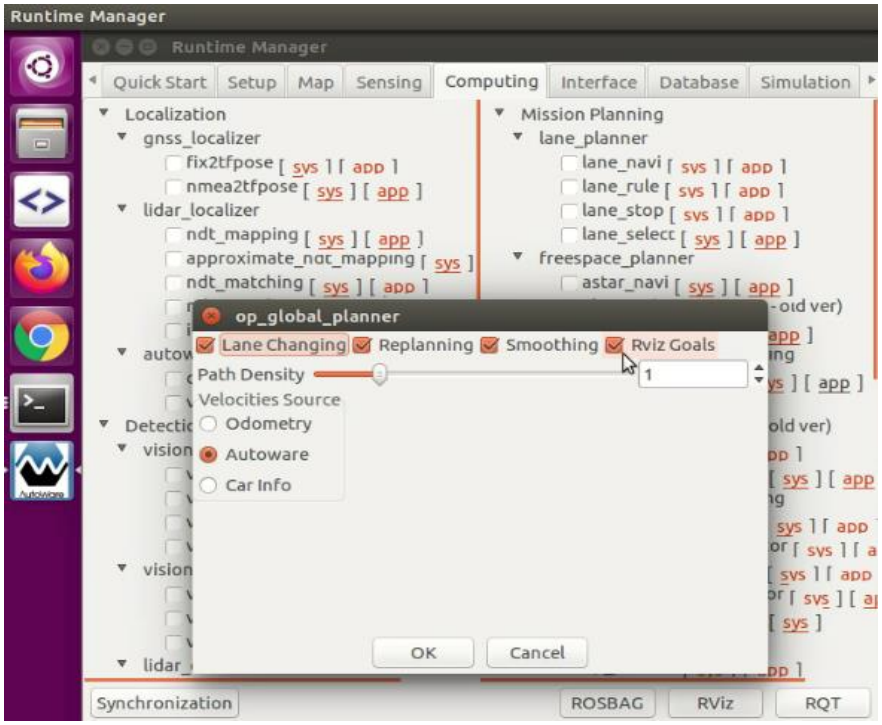

Figure 3: Selections for op_global_planner app

Step 4b: Now that the *op_global_planner* has been checked, we will open Rviz and set *2D Pose Estimate* and *2D Nav Goal* from the toolbar as shown below.

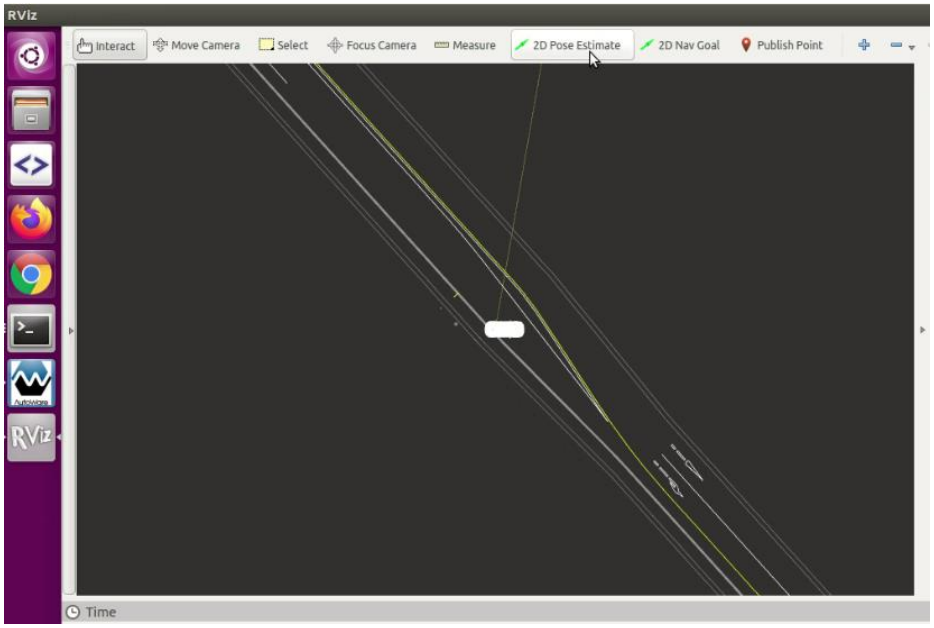

Figure 4: Initial RViz setup showing 2D Pose Estimate and 2D Nav Goal

By setting up 2D Pose Estimate and 2D Nav Goal, a three-lane path in blue will appear on the map in RViz.

Step 4c: Please now select the *obstacle_avoid* app and *velcoity_set* app, and make sure that the relevant options are selected similar to the photos below.

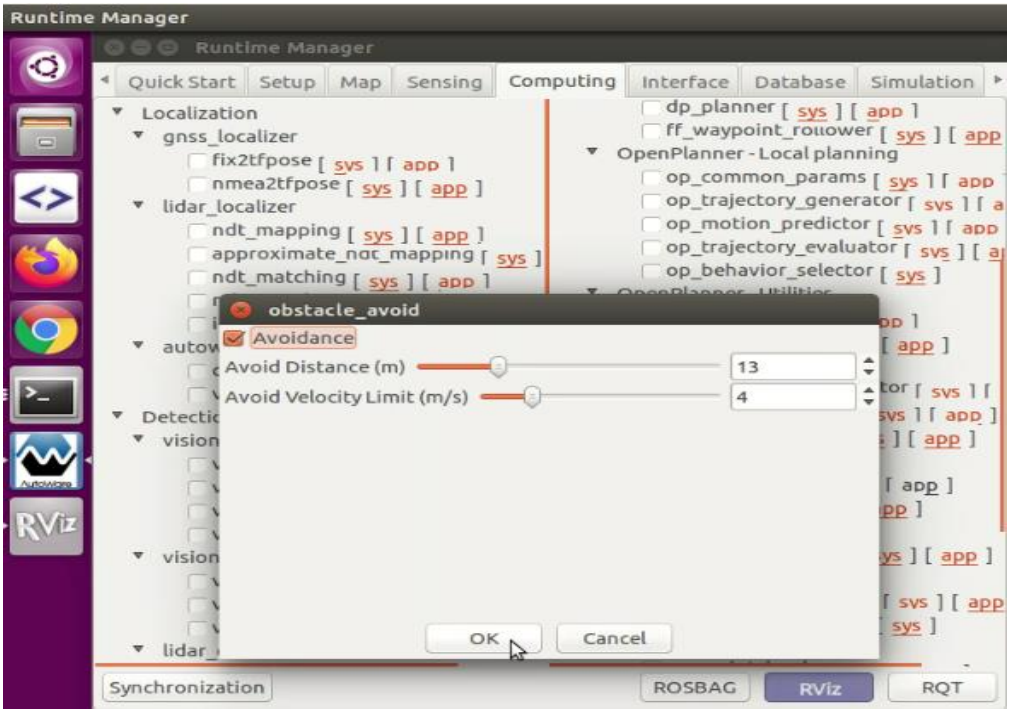

Figure 5: Obstacle_avoid app

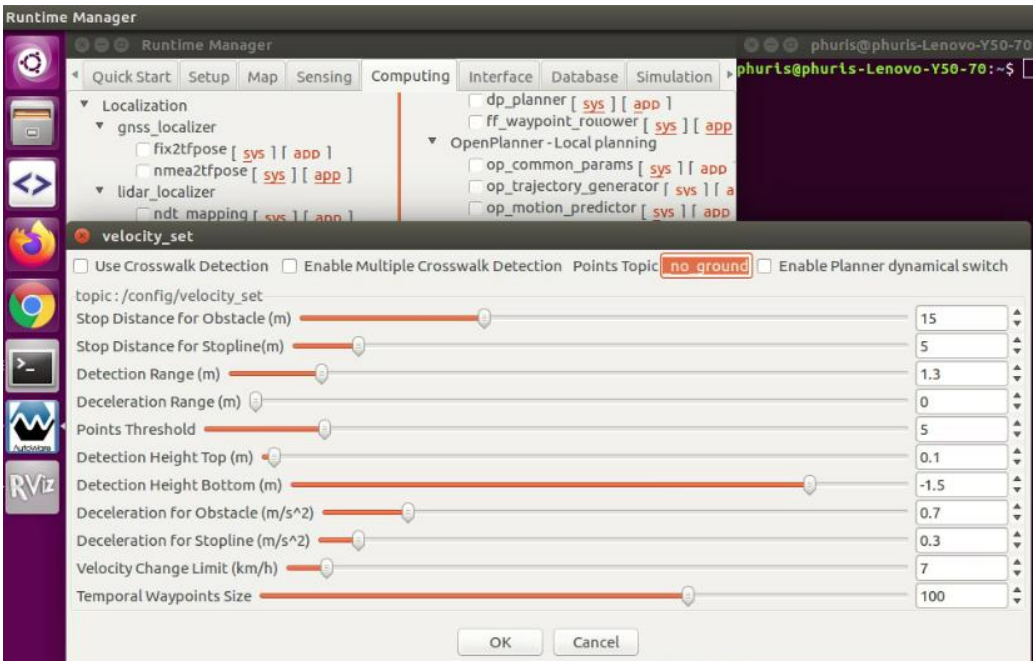

Figure 6: velocity_set app

Step 5: Now that the previously mentioned three apps have already been selected, we will run MATLAB in the background.

Step 6: Please open the MATLAB file named as *Matlab_Autoware_CoSimulation_starterScript.m* and run it. Now, the Simulink will be opened.

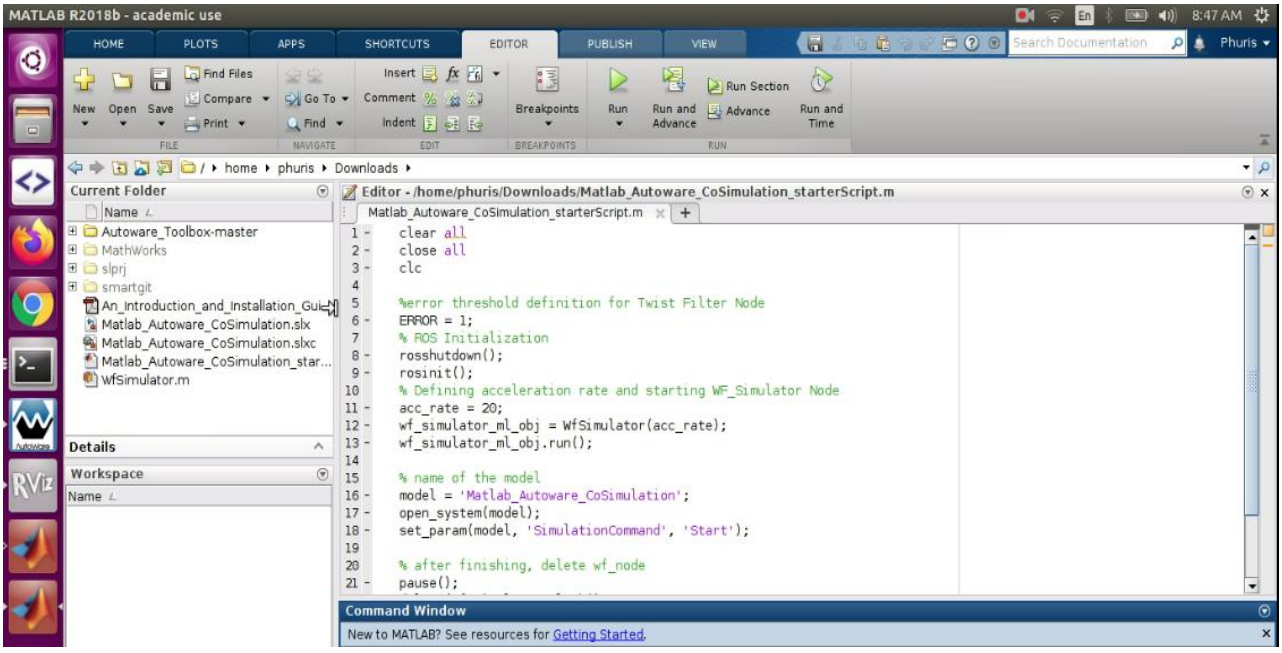

Figure 7: Matlab_Autoware_CoSimulation_starterScript.

We will notice that the Simulink is now being run.

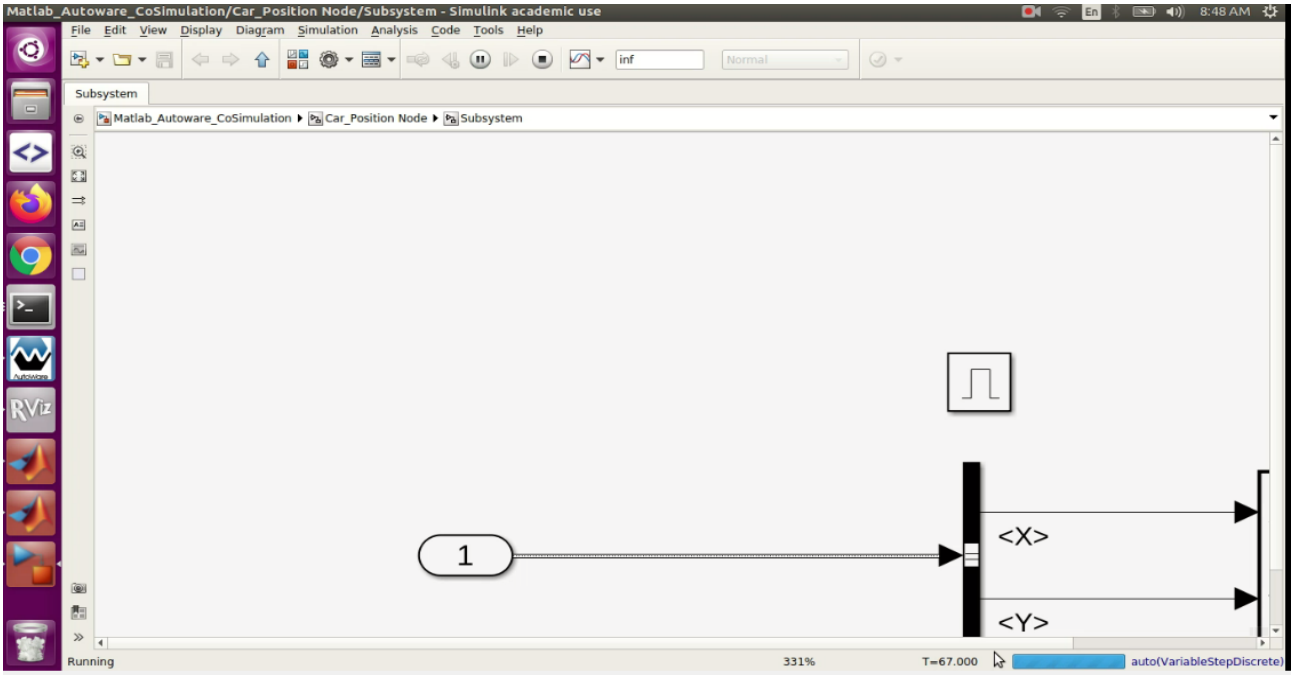

Figure 8: Simulink Model being run

Step 7: While the Simulink is being run, we will go to RViz and visualize our car by setting up 2D Pose Estimate once again, and we will notice that a black car (ego vehicle) appears on the map as shown below.

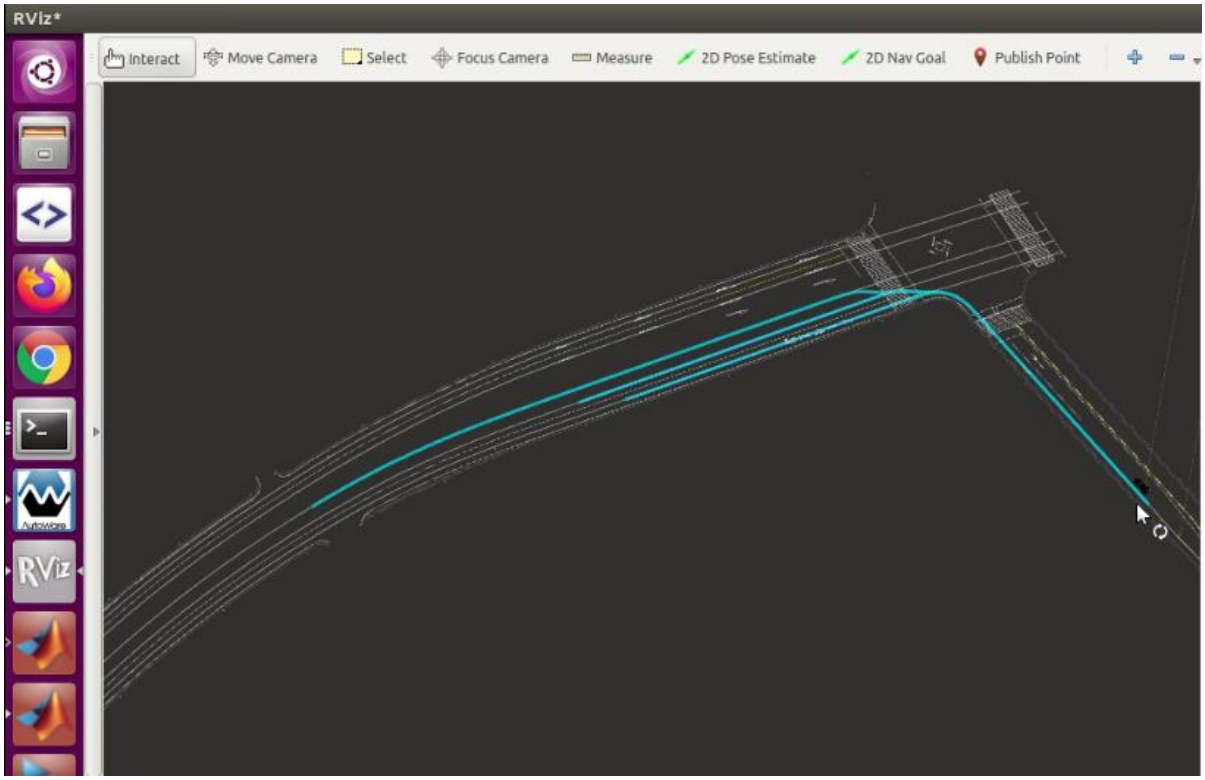

Figure 9: Visualizing our Car (Ego Vehicle)

Step 8: After visualizing our car, we will select the *pure_pursuit* app from *way point follower* under computing tab and make sure that it looks like the photo below and that all the options are selected as follows.

Figure 10: Pure_Pursuit app

Step 9: Now we will select three *lane_planner* apps under computing tab namely *lane_rule, lane_stop,* and *lane_select.*

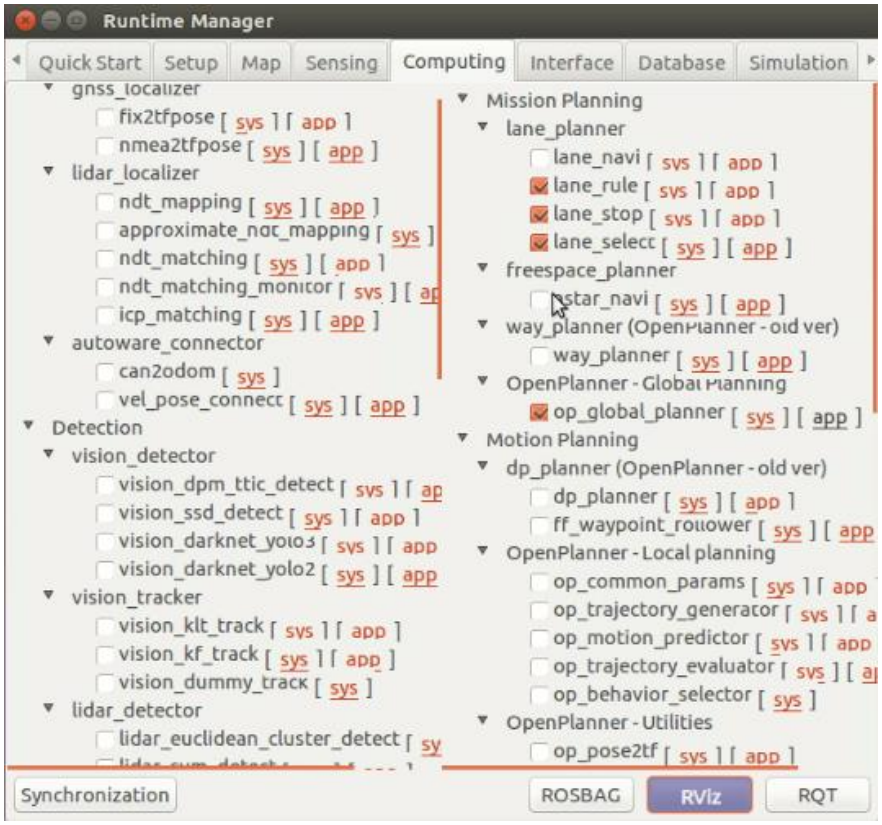

Figure 11: Selecting lane_planner apps

Step 10: We then select *way_planner* app.

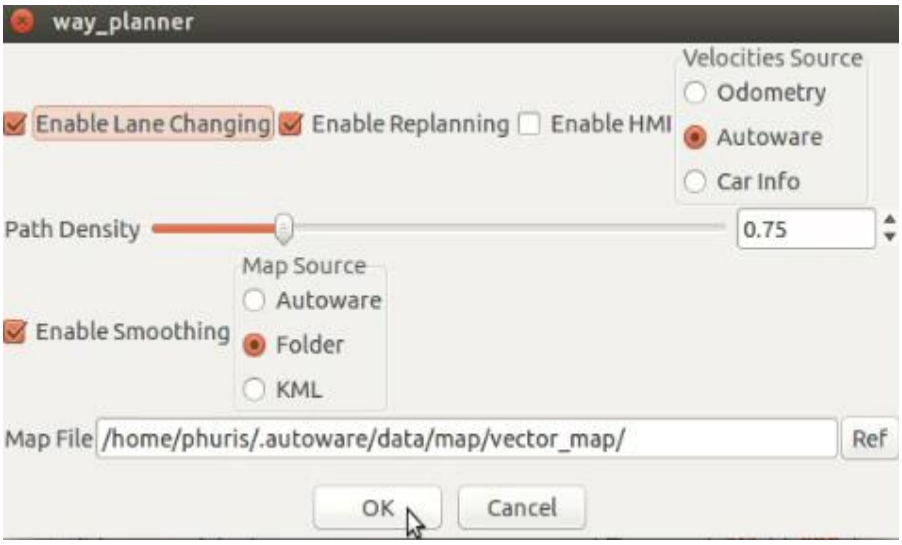

Figure 12: way_planner app

Step 11: Please now select *dp_planner* app as follows.

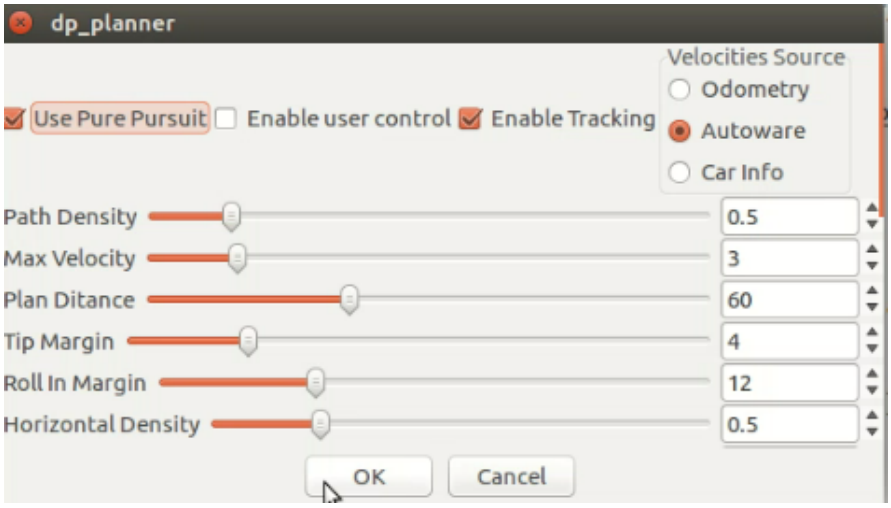

Figure 13: dp_planner app

Step 12: Now, we will notice the movement of the car on the map in RViz, and the same time we can view the position of this ego vehicle on the plot as shown below. Next step would be to put an obstacle in the path of the vehicle to make sure that the car avoids the obstacle and takes the best possible path.

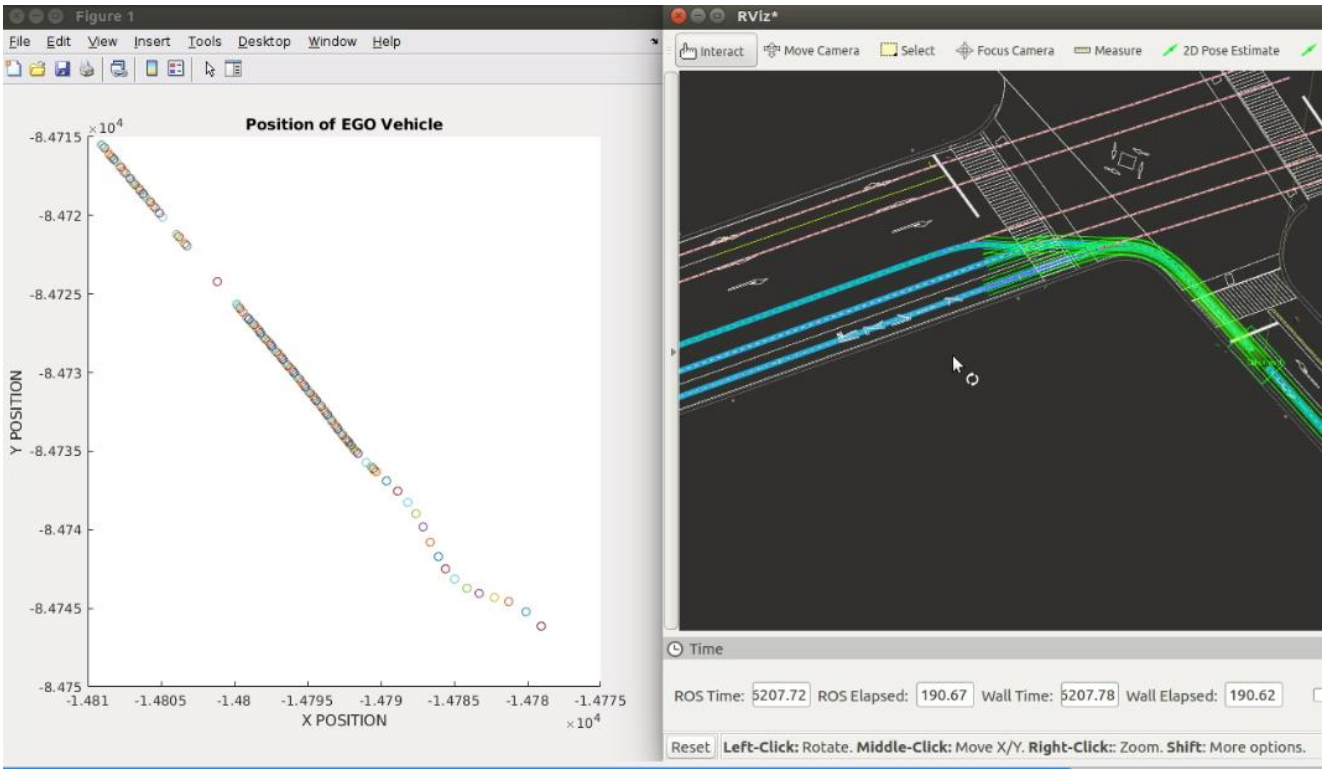

Figure 14: The car's movement visualized on RViz and MATLAB Simulink simultaneously

We can put an obstacle on the path by selecting “Publish Point” from the toolbar in RViz. We can notice that the car changes its path after few seconds, and this can also be seen from the Position of Ego vehicle plot.

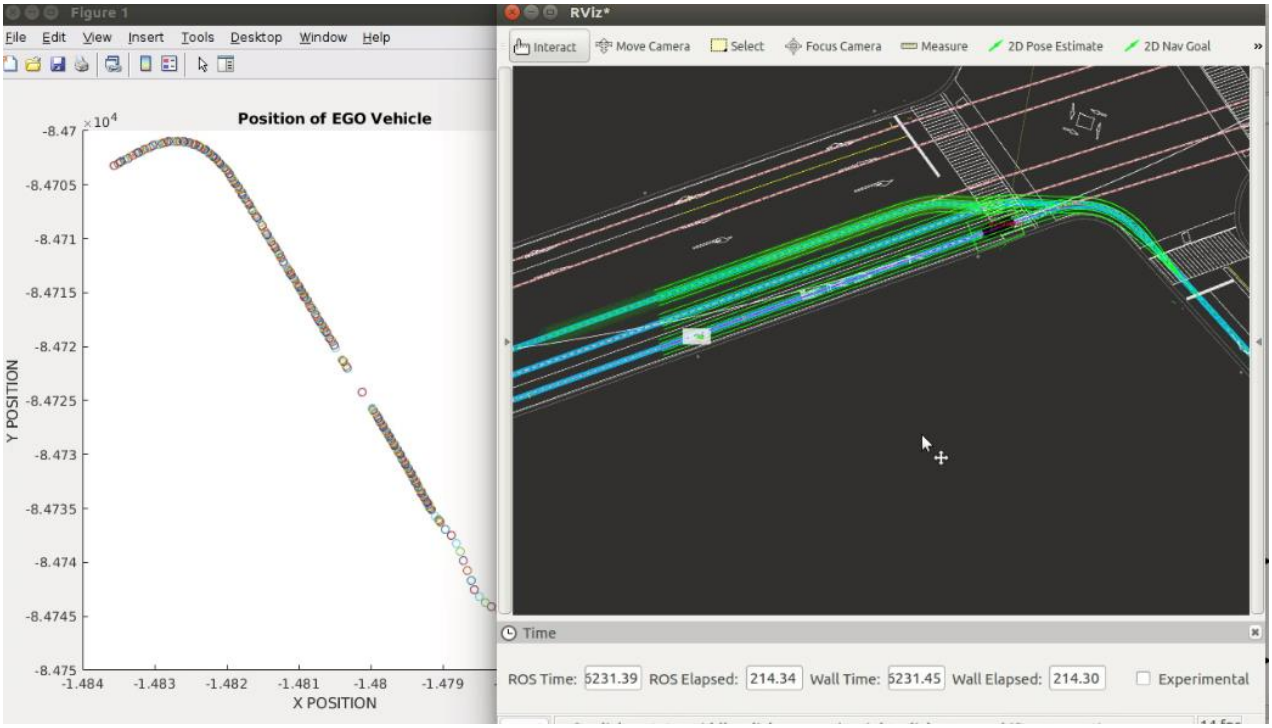

Figure 15: Putting an obstacle (in white) on the path

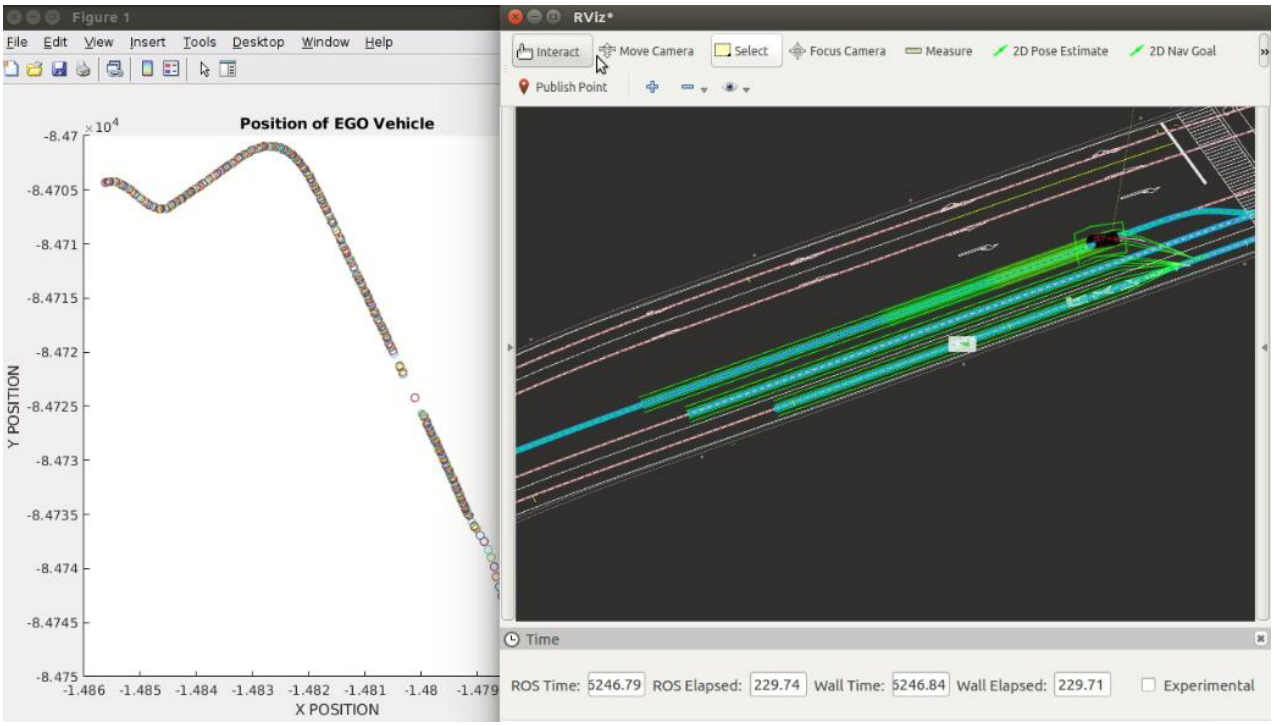

Figure 16: The car avoids the obstacle

We then put another obstacle on the path, and again the car avoids it as can be seen below on the plot of Position of ego vehicle.

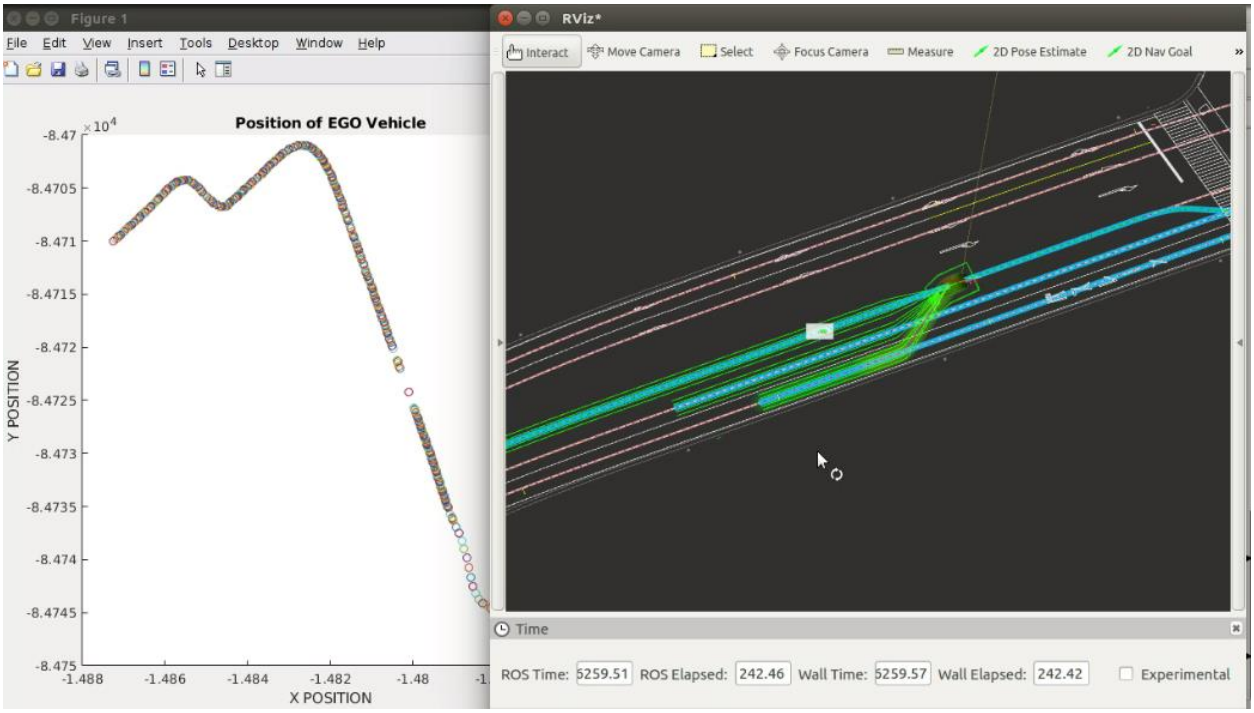

Figure 17: Obstacle avoidance