Programming a Real Self-Driving Car

Meets Specifications

Congratulations Udacian on passing the Capstone project! 🎓

This is a very great attempt as I can see you put a lot of effort to accomplish this project. Throughout the whole simulation, not a single unusual behavior occurred which shows that you have very well implemented the waypoint updater, drive-by-wire node, and twist controller.

I also liked that your code is logical and well-structured which makes the project easy to follow. I enjoyed reviewing your project and I am sure that you also enjoyed working on the project. 🎉

Cheers, and Keep up the Good Work!

Further Reading

You might also check out these resources for further research about DBW, Path planning and Traffic light detection:

- How Drive-by-wire Technology Works
- Path Planning for Collision Avoidance Maneuver
- YOLO ROS Node for Traffic Light Detection

Running the Code

Running `catkin_make`, `source devel/setup.sh` and `roslaunch launch/styx.launch` within the `ros` directory results in no errors and allows the program to connect to the simulator.

Awesome, your project builds successfully with `catkin_make` without any errors on my local machine. For more information on this build tool, you may refer to the following:

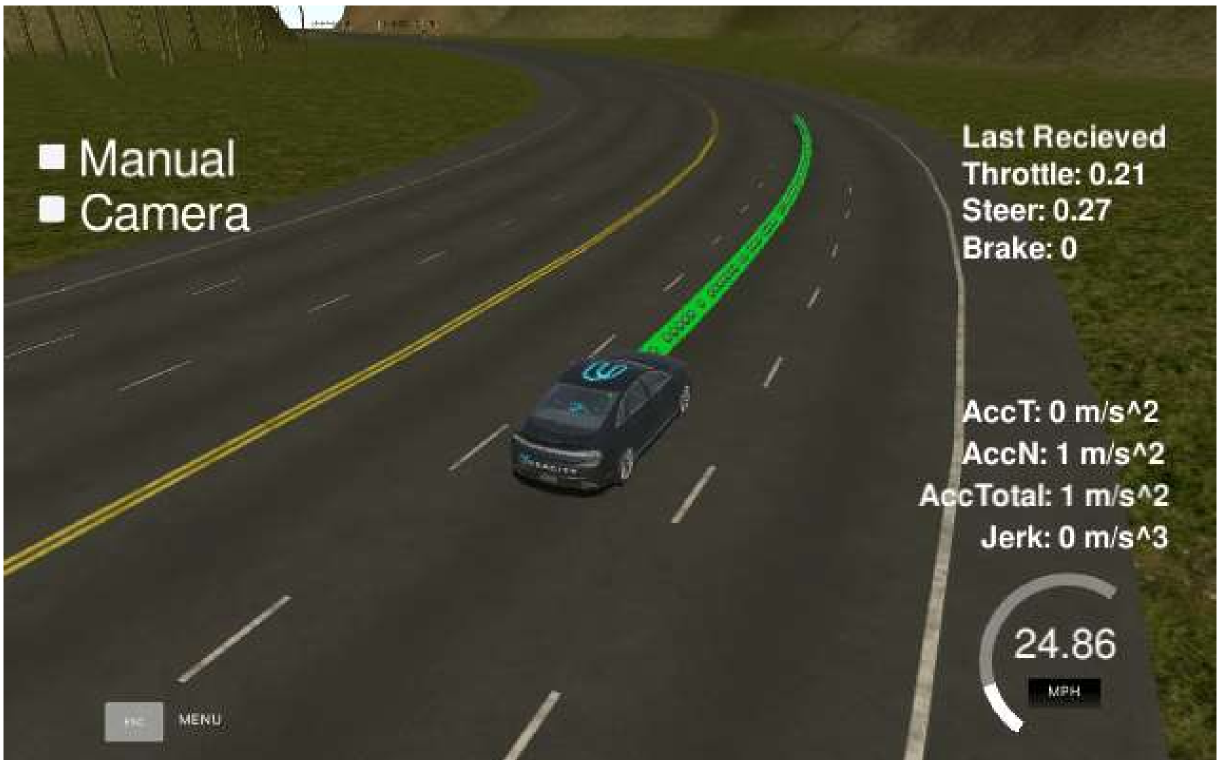
- What is catkin_make
- ROS cheat-sheet

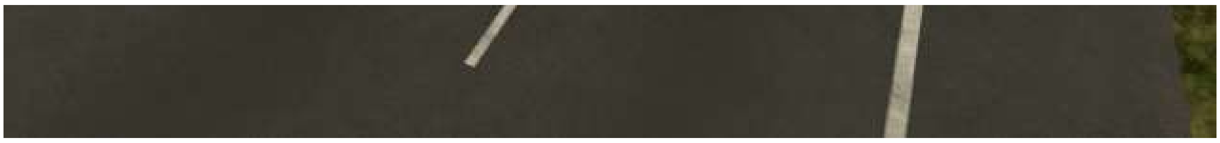
Control and Planning

Waypoints should be published to `/final_waypoints` to plan the vehicle's path around the track. No unnecessary moves (excessive lane changes, unnecessary turning, unprompted stops) should occur.

As in the Path Planning project, acceleration should not exceed 10 m/s^2 and jerk should not exceed 10 m/s^3.

Be sure to limit the top speed of the vehicle to the km/h velocity set by the velocity rosparam in `waypoint_loader`.





The car drove smoothly by not violating the total acceleration of 10 m/s^2 and max jerk of 10 m/s^3 and the top speed was below the maximum velocity defined. During the whole simulation, I did not notice any unnecessary moves. Nice work. 👏

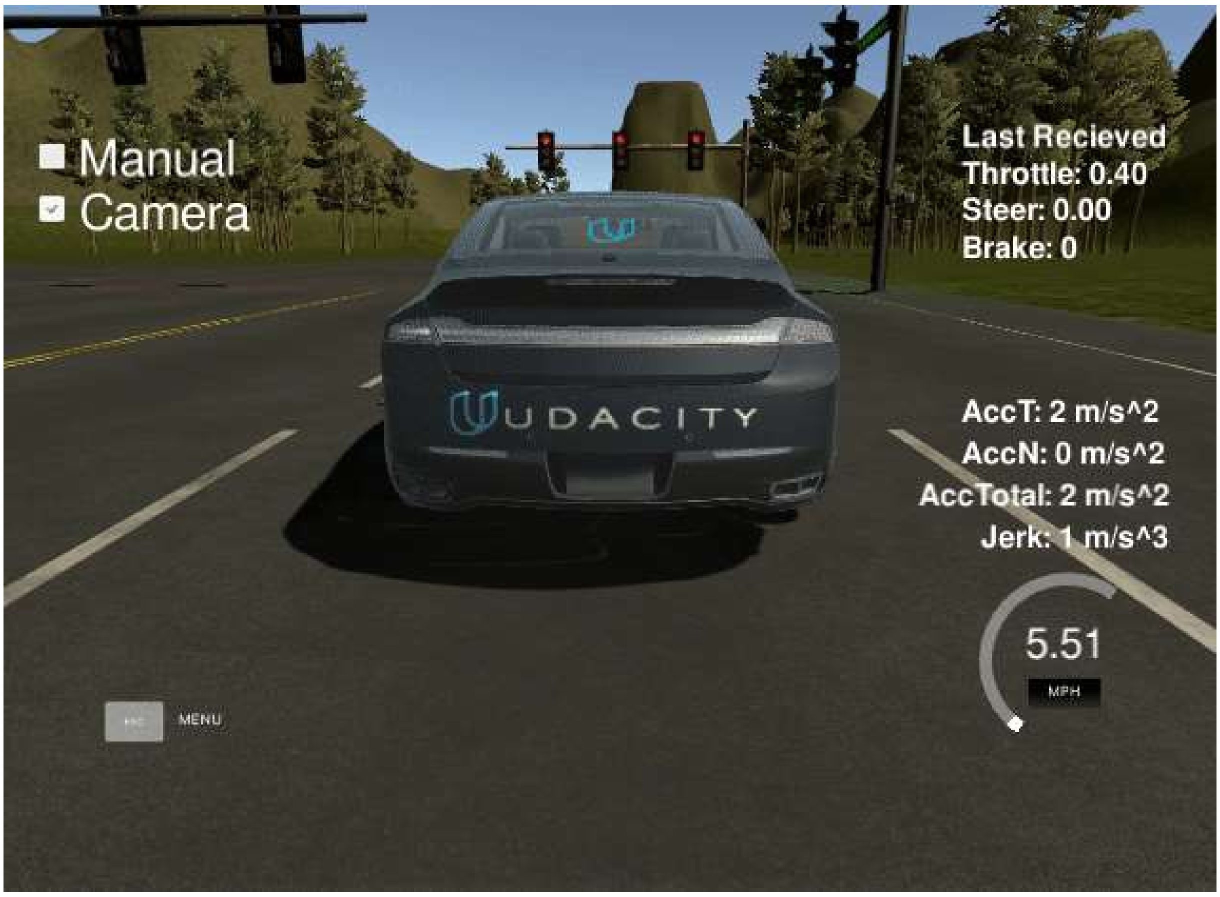
`dbw_node.py` has been implemented to calculate and provide appropriate throttle, brake, and steering commands.

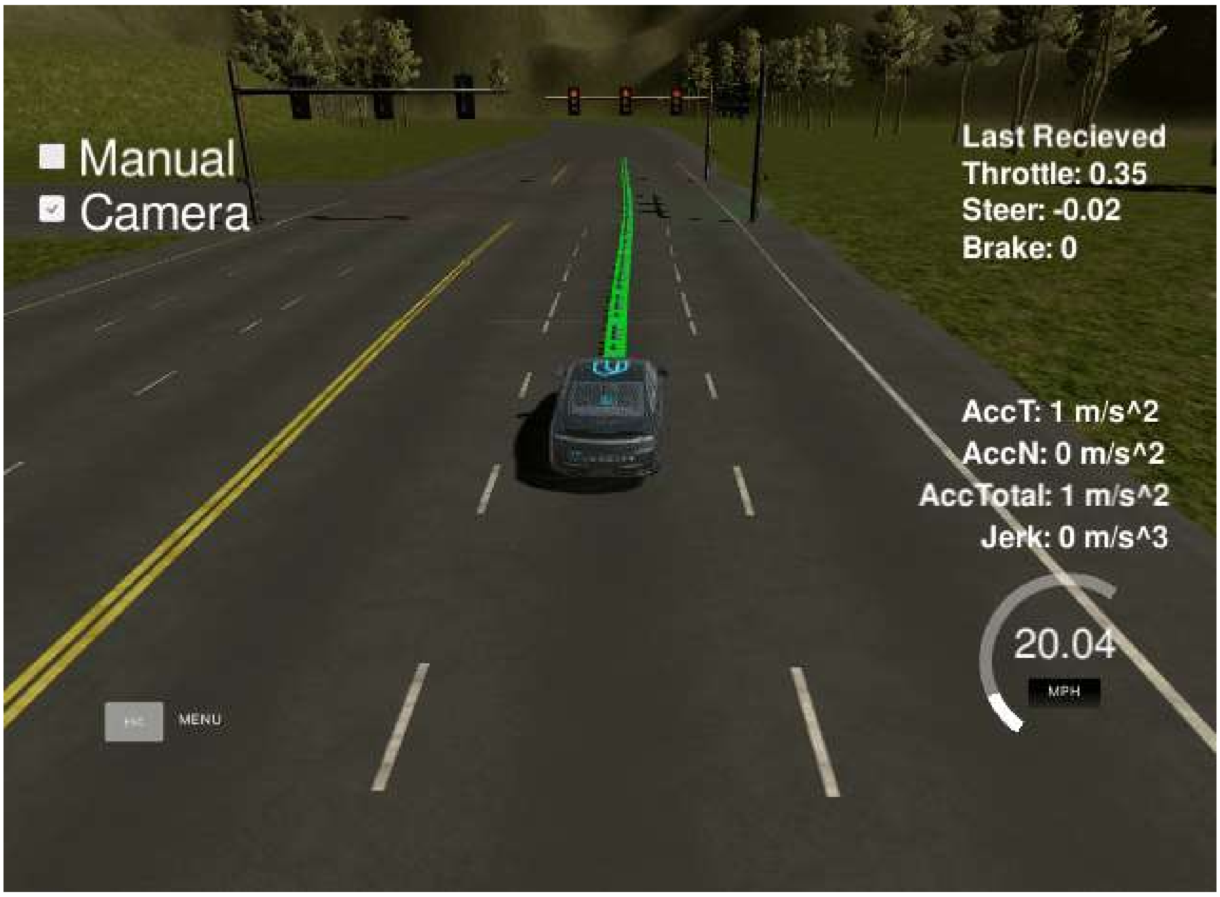
The commands are published to `/vehicle/throttle_cmd` , `/vehicle/brake_cmd` and `/vehicle/steering_cmd` , as applicable.

The vehicle appears to have appropriate throttle, brake and steering commands. I went through your whole code and have added my comments in the code-review section.

Successful Navigation

The vehicle is able to complete more than one full loop of the track without running off road or any other navigational issues (incorrect turns, random stops, teleportation, etc.).





I watched the simulation for several minutes and the car drove through a full lap of the highway without any incident as required. With the camera off, the vehicle passed the traffic lights irrespective of their color but when the camera was turned on, It was able to stop slowly and safely at the red light and was able to accelerate when the light turned green. Kudos!!

Additional discussion

You may implement your own implementation of the traffic light detection using the TensorFlow object detection API. This article provides an excellent tutorial to train the model step-by-step which is also simple to implement.

下载项目

5 代码审阅评注 ›

返回 PATH

给这次审阅打分

开始