UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom DISCUSS ON STUDENT HUB

Path Planning

REVIEW

CODE REVIEW 1

HISTORY

Meets Specifications

Keen Learner,
Congratulations on completing this project.🎉
This was really a great submission and I applaud you for the work. You have come a long way and are getting towards the end of the Nanodegree. Keep up the hard work showcased so far and good luck to you.

More In-depth Knowledge

Here are a few links where you can find useful material for further learning on the subject.

- Robot Motion Planning
- Robotics Path Planning
- Path Planning module in Robotics
- Difference between path planning and motion planning
- Excellent tutorial on a robot path planning
- Path Planning and Collision Avoidance
- Safe Motion Planning for Autonomous Driving
- Local and Global Path Generation for Autonomous Vehicles Using Splines

Compilation

Code must compile without errors with `cmake` and make.

Given that we've made `CMakeLists.txt` as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.

On compiling your project with `cmake && make`, there were no errors. Good job avoiding all syntax errors.👍

```
[100%] Linking CXX executable path_planning
[100%] Built target path_planning
```

More In-depth Knowledge

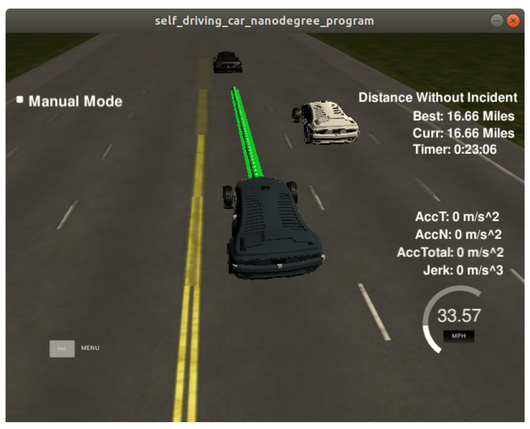
To know more about CMAKE and MAKE, please check some resources below:

- CMake Script Debugger provided by the VisualGDB tool
- Difference between Makefile and cmake to compile the code
- Cmake FAQS.
- Using make and writing Makefiles.
- Youtube set of tutorials on using make and writing Makefile.
- MakeFiles.

Valid Trajectories

The top right screen of the simulator shows the current/best miles driven without incident. Incidents include exceeding acceleration/jerk/speed, collision, and driving outside of the lanes. Each incident case is also listed below in more detail.

The top right screen of the simulator shows the current/best miles driven and the car was able to cover more than 4.32 miles without incidents like exceeding acceleration/jerk/speed limits, collisions, and driving outside of the lanes, as required. Well done!👍



The car doesn't drive faster than the speed limit. Also the car isn't driving much slower than speed limit unless obstructed by traffic.

Nice job! The ego car never exceeded the speed limit. It drives smoothly and knows when to slow down depending on the traffic scenarios.

The car does not exceed a total acceleration of 10 m/s^2 and a jerk of 10 m/s^3.

There were no reports of the acceleration and jerk limits being exceeded during the simulation. You handled this well by considering both the lane and velocity of the car especially during lane changing. Well done!

The car must not come into contact with any of the other cars on the road.

Splendid!. The ego car really knows when to change lane and it does take safety precautions before implementing decisions. It reduces speed accordingly when behind a slow-moving car and if all lanes were occupied.

More In-depth Knowledge

Here are a few links with useful information on collision avoidance.

- Path Planning for Collision Avoidance Maneuver
- Optimal Trajectory Planning for Glass-Handing Robot Based on Execution Time Acceleration and Jerk
- This discussion on StackExchange can be of interest Which trajectory planning algorithm for minimizing jerk.

The car doesn't spend more than a 3 second length out side the lane lanes during changing lanes, and every other time the car stays inside one of the 3 lanes on the right hand side of the road.

Bravo! The ego car never spends more than 3 seconds while changing lanes. This shows all caution was taken to this effect as seen in main.cpp.👏

The car is able to smoothly change lanes when it makes sense to do so, such as when behind a slower moving car and an adjacent lane is clear of other traffic.

Impressive! The car was able to change lanes smoothly when behind a slower vehicle, the car was capable of smartly deciding whether the lane it was going to take was cleared off. This avoided collision with other vehicles on the new lane and possibly endangering passengers.👍

cl.png

Reflection

The code model for generating paths is described in detail. This can be part of the README or a separate doc labeled "Model Documentation".

Good job discussing in the submitted `README.pdf`, not only how path generation was done in your implementation but also the entire project in detail. You made a good choice by using cubic spline interpolation. The single `spline.h` file makes trajectory generation much easier in this project.

DOWNLOAD PROJECT

1 CODE REVIEW COMMENTS ❯

RETURN TO PATH

Rate this review