UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Path Planning

REVIEW

CODE REVIEW 3

HISTORY

Meets Specifications

Keen Learner,
Congratulations!🎉👏
A great job has been done in the implementation of this path planning project. I watched the simulation for over 6 minutes and it was very stable in auto-adjusting speed and changing lanes when necessary. It was a pleasure reviewing this project. You sure will make a fine SDC Engineer. Keep up with this good work and all the best to you moving forward.

Advanced Learning Tips

Here are a few resources you might find useful for more insight and further learning.

- The path planning problem in depth
- A discussion on What is the difference between path planning and motion planning?
- Introduction to Robotics #4: Path-Planning
- Introduction to robot motion: Robot Motion Planning
- Introduction to robot motion: Path Planning and Collision Avoidance

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.

Nicely done! Indeed code compiles without errors when run with `cmake` and `make`.👍

```
Scanning dependencies of target path_planning
[ 50%] Building CXX object CMakeFiles/path_planning.dir/src/main.cpp.o
[100%] Linking CXX executable path_planning
[100%] Built target path_planning
```

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.

Indeed the top right screen of the simulator shows the current/best miles driven without incidents. The car is able to drive more than 4.32 miles without incidents as required. Nice!

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.

Well done in this section. The ego car did not at any time exceed the speed limits while driving. All precautions were taken to ensure that the car knows when to slow down and when to take up speed. Nice work specifying the highest attainable speed to `49 * 0.44704` mph as seen in main.cpp(line 111) 👍

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

Impressive! The ego car does well to ensure a high degree of safety during simulation. It at no point in the simulation exceeded the acceleration and jerk limits. This was cautiously implemented. 👍

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

Brilliant demonstration in play here. The ego car is so smart that it knows when to slow down especially when behind a slower car as well as knows when to change lanes.

Suggestions

The following links can also provide more insights on this section of the project:

- 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.

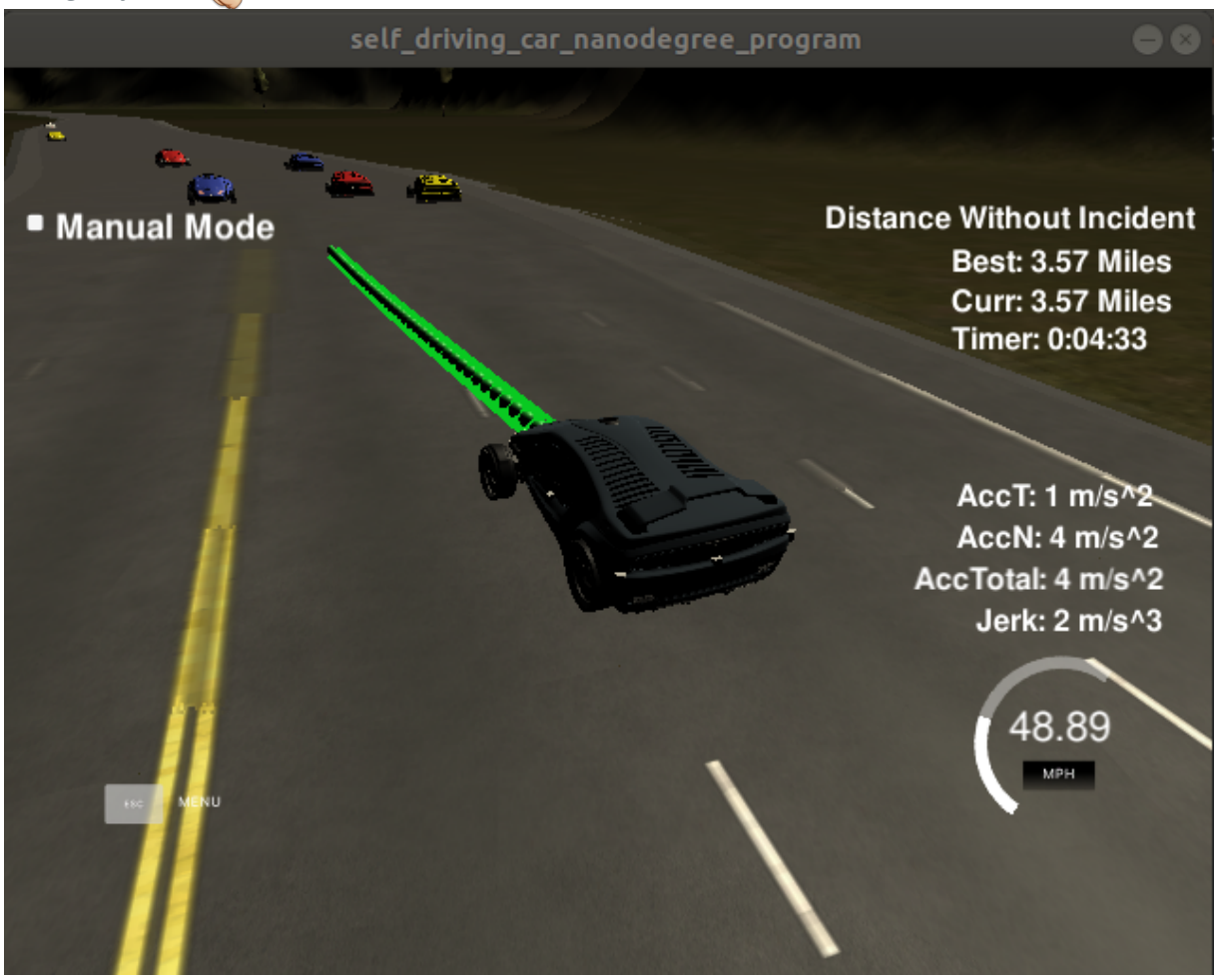
The car doesn't spend more than a 3 second length out side the lane lanes during changing lanes, and every other time the car stays inside one of the 3 lanes on the right hand side of the road.

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

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.

Awesome! The ego car was smart enough to know when to switch lanes, especially when behind a slower car. It always ensured that the new lane it is headed is free of other cars before making the move. This was

intelligently done.👏



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".

Outstanding job discussing with illustrations in the submitted `WRITEUP.md` file, how path generation was done in your implementation. 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

3 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review