

CarND-Path-Planning-Project

Self-Driving Car Engineer Nanodegree Program

Project Goal

The goal of this project was to safely navigate around a virtual highway with other traffic that is driving ± 10 MPH of the 50 MPH speed limit. The inputs are:

- the car's localization;
- sensor fusion data (location and velocity data from other cars);
- a sparse map list of waypoints which describe the highway's path

The car should try to go as close as possible to the 50 MPH speed limit, which means passing slower traffic when possible. The car should avoid hitting other cars at all cost as well as driving inside of the marked road lanes at all times, unless going from one lane to another. The car should be able to make one complete loop around the 6946m highway. Since the car is trying to go 50 MPH, it should take a little over 5 minutes to complete 1 loop. Also the car should not experience total acceleration over 10 m/s^2 and jerk that is greater than 10 m/s^3 .

Considerations

1. The car uses a perfect controller and will visit every (x,y) point it receives in the list every .02 seconds (50 Hz). The units for the (x,y) points are in meters and the spacing of the points determines the speed of the car. The vector going from a point to the next point in the list dictates the angle of the car. Acceleration both in the tangential and normal directions is measured along with the jerk, the rate of change of total acceleration. The (x,y) point paths that the planner receives should not have a total acceleration that goes over 10 m/s^2 , also the jerk should not go over 50 m/s^3 .
2. There will be some latency between the simulator running and the path planner returning a path, with optimized code usually its not very long maybe just 1-3 time steps. During this delay the simulator will continue using points that it was last given, because of this its a good idea to store the last points used to achieve a smooth transition. `previous_path_x`, and `previous_path_y` are helpful for this transition since they show the last points given to the simulator controller with the processed points already removed.

Implementation

Making sense of other cars

One of the first things to do on a new iteration is to process information about what the car is sensing on the road. In particular, to make sense of the other cars driving in the same direction as the ego vehicle. A `Car` class was implemented to make this an easier task work on. Each time a new car is spotted, it's added to an internally kept `Car` library. This way we can keep track of data about other cars from past iterations, if we wish. Each

Decision making

The problem at hand didn't require complex decision making to hit all criteria. We want to move as fast as possible in a legal (observing speed limit), safe (avoiding collisions and keeping within lane limits) and comfortable manner (not accelerating or jerking hard).

With this in mind, the maximum speed that the car will ever drive (henceforth referred to as the `GOAL_SPEED`) is 49.5 MPH, just under the legal limit of 50 MPH.

Trajectory generation

A really helpful resource for doing this project and creating smooth trajectories was using <http://kluge.in-chemnitz.de/opensource/spline/>, the spline function is in a single header file is really easy to use.

Results

Over 6.10 miles, 7 minutes and 44 seconds and 4709 readings, the average speed was 47.53 MPH