Introduction

In this project your goal is to safely navigate around a virtual highway with other traffic that is driving +-10 MPH of the 50 MPH speed limit. You will be provided the car's localization and sensor fusion data, there is also a sparse map list of waypoints around the highway. The car should try to go as close as possible to the 50 MPH speed limit, which means passing slower traffic when possible, note that other cars will try to change lanes too. 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.

Details

The car uses a perfect controller and will visit every (x,y) point it recieves in the list every .02 seconds. 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 recieves should not have a total acceleration that goes over 10 m/s^2, also the jerk should not go over 50 m/s^3. (NOTE: As this is BETA, these requirements might change. Also currently jerk is over a .02 second interval, it would probably be better to average total acceleration over 1 second and measure jerk from that.

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 you have used so you can have a smooth transition.

previous_path_x, and previous_path_y can be helpful for this transition since they show the last points given to the simulator controller with the processed points already removed. You would either return a path that extends this previous path or make sure to create a new path that has a smooth transition with this last path.

Rubric

Compilation

The code compiles without errors or warnings.

Valid Trajectories

The car is able to drive at least 4.32 miles without incident..

The car drives according to the speed limit.

Max Acceleration and Jerk are not Exceeded.

Car does not have collisions.

The car stays in its lane, except for the time between changing lanes.

The car is able to change lanes.

Reflection

In the following subsections, I will first describe the path generation method in more detail:

all the codes are given in main.cpp file.

1. In lines (57-58) I initialized the lane and velocity of the ego car.

2. Codes in lines (106-139) describe the prediction part. I used the fusion data to predict:

(1) if there are cars stay in front of the ego car and very close to ego car at the same lane blocking the traffic.

(2) if there are cars stay very close to ego car and at the left lane of the ego car that make left lane change not safe.

(3) if there are cars stay very close to ego car and at the right lane of the ego car that make right lane change not safe.

I set: a car is considered "dangerous" when its distance to ego car is less than 40 meters in front or behind us. (Firstly I set the distance = 30, but one collision occurs, then I set a greater one, it works well.)

3. Codes in lines (143-155) describe the behavior planning part. By using the result of prediction part, I decided here if the ego car slow down, left lane change or right lane change and the value of goal lane number and goal velocity.

4. Codes in lines (157-177) describe the cost function part. To get the best behavior planning with lower cost, I set a cost function considering the cost of lane change and cost of acceleration.

5. Codes in lines (181-252) describe the trajectory generation part. Depends on the chosen behavior planning I generated the trajectory. The last two points of the previous trajectory, or the car position if there are no previous trajectory, are used with three points at a far distance to initialize the spline calculation. I used the information from the previous path ensures that there is a smooth transition from cycle to cycle.