Project 07 Highway Driving

Introduction

The goal of this project is to build a path planner that creates smooth, safe trajectories for the car to follow. The highway track has other vehicles, all going different speeds, but approximately obeying the 50 MPH speed limit.

Rubric Points

Compilation:
Code compiles without errors with cmake and make.

Valid Trajectories:

- The car is able to drive at least 4.32 miles without incident.
- 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.
- The car does not exceed max acceleration and jerk of 10 m/s^2 and a jerk of 10 m/s^3 respectively
- 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 when needed.

Reflection

There are three main parts to the implementation of trajectory planning:

- Checking to see if there are other vehicles nearby that are going in the same direction.
- Based on the nearby vehicle check, a decision is made if the car needs to change lanes, stay in the same lane or reduce the speed of the vehicle.
- Waypoints are generated to give a path indicating potential future points of the car and are calculated with an aim of following maximum velocity, acceleration and jerk numbers.

Waypoints are classified further into anchor points and spline points. There are 5 anchor points. 3 of them are points which are a set 30m, 60m, and 90m away from the current position. 2 of them are taken from the path previously travelled. This helps in continuity and avoids exceeding velocity/acceleration/jerk limits between calculations. Spline points are points which are interpolated between anchor points. These adjust how far the vehicle will travel in between time intervals along the curve without exceeding set limits. For calculating the waypoints, the points are transformed from the inertial coordinates to the car coordinates for ease of calculation and later transformed back into the inertial coordinates to pass it to the simulator.

Details about calculations after suggested edits:-

- Details about the other cars were obtained from the sensor fusion data that we got from the simulator. It consisted of the location of the car in cartesian and frenet coordinates. Based on this data the speed of the other cars was found out. (Lines: 126 - 137)

- 30m was the distance considered as the "collision possible" distance. For this logic to work, the positions of our and the other cars were projected ahead to the end of the current calculation cycle. If the car in front / left /right was within 30m of our car then a collision flag was set (indicated by variables car_in_front, car_in_left_lane, car_in_right_lane). This action of projecting ahead and then checking assisted in making safer and smoother trajectories during the current calculation cycle.(Line: 140- 167)

- When the "collision possible" flag for the current lane was true, it checked for the "collision possible" flags for left and right in that order. If they were found to be false, then the lane variable was changed to the consequent value. If the left and right flags were also true, it meant that there were cars in other lanes such that it could cause collisions in case of lane change events. In this case the speed was reduced setting an acceleration value that did not exceed any limits. (Lines: 170 - 187)

- One of the other goals was to always drive in the center lane. Hence, if our car was not in the center lane and the collision flag for the required side was false then the lane was changed to center. (Lines: 189 - 204)