Udacity Self Driving Car Term 3 Project 1
Path Planning

Introduction

This is by far one of the most difficult and time consuming projects in this Self Driving Car Nano Degree Program. This is a difficult problem in a way that I could not design the software before implementing it. Most of the times spent on trying to learn the data coming in from the simulator as well as how can I move my car without colliding with the others as well as not jerking the speed of the car and driving within the speed.

The lessons learned in the prior to this project gives tools and techniques to use in this project helps me to prepare to work on this project and also the reading of the slack channel helps a lot in practical ways. With these, I followed the simplest approach to solve this path planning problem.

Car Data, Map Data, Previous Path Data and Sensor Fusion Data

At first, I read in the map data given by the project starter "data" folder into mapData_t data structure which is defined in utility.h then pass in the map data to onMessage method. I store the car data from the simulator in data structure called carData_t also is defined in utility.h. From the simulator, I store x and y values of previous path and sensor fusion values of the other cars to be used in my findNextPath() function which is the main function for the finding the path points to move my car on the High Way with 3 lanes under 50 mph or within the traffic without collision, within the jerk limit and the whole HWY route which is 4.32 miles long.

I broke down total of two main parts in my implementation as: Generating Path Points (Path Planning) and Learning others' car behaviors and acting on it (Behavior Planning) as follows.

Path Points

With testing out the data that are in from simulator and feeding the test points to move the car to have feel for it to design this path planning software, I ended coding without designing the software. It was pretty hard that way since I started this project a little late. So, I ended having procedural approach rather than object oriented approach.

I first decided to have some points along the highway for my car to move, I use spline tool suggested by project instruction to create points (line 112, main.cpp). Since there map coordinates and car coordinates are different, I declared and defined those helper functions in the utility.h and utility.cpp. I made a decision on maintaining velocity of the car in spline points to have smooth velocity for my car not to have jerks (line 123-138, main.cpp). Then generate the trajectories of my car for the start of the driving using lane spline and velocity spline created above (line 154-179, main.cpp) with some house keeping codes.

I handle the trajectories generation using previous path information for generating the path points after the car is already started running (line 183 – 287, main.cpp). I save these generated path points for feeding back to the simulator.

Sensor Fusion Data with Planning Others' Behaviors

This part is even harder for me with learning other cars how they drive using sensor fusion data from simulator as well as how my car path points and speed will mesh with the other cars without having collision (that means, changing lane, slow down the speed, and speeding up).

I place the other cars from the sensor fusion data to the respective lanes (we have total of 3 lanes). That is implemented (line 295 – 318, main.cpp). And figure out the closest cars from back of my car and in front of my car (line 321 – 343, main.cpp).

There are three cost weights that I use in this implementation inspired by Mohan Karthik (fellow udacity student). From his article, I have followed the same path as him to select the lane my car to be on. Basically, I have three cost weights for deciding which lane my car should be on: Lane Change Cost, Distance Cost, and Velocity Cost (line 354 – 383, main.cpp).

With that, I loop through to three lanes my car could be on to check with distance gaps from back of my car, and from in front of my car and speed limitation for my car to safely change the lane if it needs to be. It was implemented from (line 387 – 455, main.cpp).

What's Next

The first thing come to my mind is that almost 400 lines of code for a function has trouble me a lot. Setting debug breaks, printing out and all the other things just hard for me to find where is it having issue. Global variables do not help also. Refactor is the first action item that I have.

The second thing is that can I handle real world Freeway with exit, entrance, interchanges, HOV lane and the speed limit is different along the same Freeway with my implementation. I don't think I can. What about local street driving with cars, traffic lights, stop sign, intersections, in and out of commercial places and a lot of people on the street. I definitely don't think I can handle it. This implementation is just merely handle the project satisfactions. It makes me think to work and spend more time on to this issues that we learned in term 3 of Udacity Self Driving Car Program.

I feel humble and proud at the same time as I am a part of this self-driving cars community which is trying to help elders, children, disables, and all of the social communities for better.

Acknowledgement

I would like to show appreciation to all the slackers on Self-Driving Car slack channel, especially John Chen and Mohan Kathik. Without them, I am not sure where will I be with this project. I also would like to thank all Udacity's instructors for these three terms. I have learned a lot.

As I stated in the beginning of the document, this project was by far one of the most challenging projects if not the most hardest in this course. With full time job and being a full-time dad and the project due in the middle of sizzling hot summer (at lease in Los Angeles), it added more pressure.