Udacity – Path Planning Project

Submitted by: Neeraj Gulia

Date: May-11-2018

# Objective

The objective of this project is to run the car in simulator to follow lane lines and to not collide with other cars. Also to switch the lanes safely if there is a slow moving vehicle ahead of our car.

# Rubric Points

## Compilation

### The code compiles correctly.

I used spline.h library and added in the source folder. The code compiled without any errors.

## Valid trajectories

|  |  |
| --- | --- |
| Description | Result |
| The car is able to drive at least 4.32 miles without incident. | Car drove more than 12 miles without any incident |
| The car drives according to the speed limit. | Car never crossed the speed limit |
| Max Acceleration and Jerk are not Exceeded. | Within given limit |
| Car does not have collisions. | No colisions |
| The car stays in its lane, except for the time between changing lanes. | Car stays in the lane |
| The car is able to change lanes | Car successfully changed lanes when it’s safe to change |



Figure 1 Screenshot of the successfull run

## Reflection

### Model description:

The main.cpp file listens to the message sent from the simulator (using web sockets). The simulator sends the data related to car, like Car’s location, velocity, yaw rate, speed and Frenet coordinates (longitudinal and lateral displacement). It also sends the data from the sensor fusion which contains data for other cars in vicinity.

The working of the model can be broken down in following three parts:

#### Prediction

The data from the sensor fusion and simulator is used to generate the predictions about the likely behavior of moving objects. Like if there is a car in front of us, what it will do in future and based on that we will decide the behavior of our car.

#### Behavior Planning

Based on the prediction of the other moving objects we decide the behavior of our car. There can be following behaviors which our car can do based on the other moving cars:

|  |  |
| --- | --- |
| Prediction | Behavior of our car |
| Car ahead us is too close | Slow down our car |
| Car ahead us is driving slow | See if we can change lane safely |
| Car ahead us is slow and there is no car on left lane | Change the lane to left lane (but see first if it is safe to change the lane and we are not in the left most lane) |
| Car ahead us is slow and there is another car in left lane | Change the lane to right (but see first if it is safe to change the lane and we are not in the right most lane) |
| There is no car ahead of us or car is too far away | Increase the speed of the car to approx. speed limit and maintain the lane |

The buffer distance between our car and car ahead is set as 30 meters.

We try to change the lane if there is no vehicle in the side lane in front of us for 30 meters and there is no car too close to us which is coming from behind (there might be a car coming too fast from side lanes while we plan to change the lane).

#### Trajectory Generation

Our car calculates the trajectory based on the speed, other vehicles, current lane, car coordinates and past path points. To make the trajectory smooth we adds the previous path points (which are not yet consumed by the car) to the future path points.

Also we adds three way points in next 30 meters, 60 meters and 90 meters

First, the last two points of the previous trajectory (or the car position if there are no previous trajectory, lines 321 to 345) are used in conjunction three points at a far distance (lines 348 to 350) to initialize the spline calculation (line 370 and 371). To make the work less complicated to the spline calculation based on those points, the coordinates are transformed (shift and rotation) to local car coordinates (lines 361 to 367).

In order to ensure more continuity on the trajectory (in addition to adding the last two point of the pass trajectory to the spline adjustment), the pass trajectory points are copied to the new trajectory (lines 374 to 379). The rest of the points are calculated by evaluating the spline and transforming the output coordinates to not local coordinates (lines 388 to 407). Worth noticing the change in the velocity of the car from line 393 to 398. The speed change is decided on the behavior part of the code, but it is used in that part to increase/decrease speed on every trajectory points instead of doing it for the complete trajectory.

# Final Thoughts

I found that MPC gave much better results as compared to PID. However, it is little difficult to implement but it seems worth the efforts. Now I am looking for some more examples where can implement the MPC.