Udacity Self Driving Car Term 2 – MPC Project Writeup

Vehicle Model

The model that is used in this assignment is kinematic model with vehicle x, y coordinate system with orientation psi and velocity variables at time t to predict new states of vehicle at time t1. One of the important thing is to minimize the error between new trajectory path and vehicle actual path. The method we learned from the class is to adjust the actuators to minimize the errors. I used cross track error and orientation error with dynamic variables steering and throttle to control the vehicle. These are the constraints that I built in my implementation.
The equations that I used to implement the new state are defined in the Lesson 18, Lecture 5 "Global Kinematic Model". I also use particular value of Length from front to CoG (Lf) that tuned and given by Udacity class which is 2.67.

Time step Length and Frequency

As I was implementing to come with time step and frequency, I recalled the lesson 19 lecture 5 from the class. It is suggested a good approach which I need to decide the T then tune dt and N appropriately. Recall (T = N * dt)
I needed to pick a T first so, I decided to choose T as between 1.5 to 2.5 seconds. dt has to be in milliseconds with decimal number of N. I defined to dt to be between 200 ms and 100 ms. N then comes down to between 15 and 25. I thought I have a good amount of combination in these values.
It was painful as I was tuning these parameters, I found that out the model is not totally just depending on these parameters but the other upper and lower limits of variables in solver as well.
Then also found out that the larger dt the less frequent actuations happen so it is harder to predict reference trajectory. I later settled with 100 ms for dt with 1.5 second of T then 15 for N. dt is in seconds which is 0.1 for 100 milliseconds. These values seem to be stable enough but still having problems with the vehicle not being on the track and I decided to tweak in the cost and solver variables.

Polynomial Fitting and MPC Processing

As I learned from the class exercise to fit the waypoints using third order polynomial function. The third order polynomial function can fit well the curve roads than lower order polynomial functions. I wonder if I fit the waypoints using fourth order polynomial function then would it be smoother than current third order solution which I will try out after this class. I also wonder whether I would suffer performance.

As simulator gives the points in global coordinate system being y-axis is pointed north and x-axis is point east. But we have to deal with the vehicle coordinate system. There are ways to deal with it and I decided to pick trigonometry method over matrix transformation. Although I did not test it in which method will have better performance, I just choose simplicity over complex although one might argue, matrix is simpler than trigonometry. I cited the stackexchange on this issue. (https://gamedev.stackexchange.com/questions/79765/how-do-i-convert-from-the-global-coordinate-space-to-a-local-space).
I then fitted the points with polyfit() provided by Udacity with 3rd order and find the cte using polyeval() at the x = 0. And orientation error is the desired orientation subtracted from the current orientation and we know current orientation is in our state. And we can calculate desired orientation as tangential angle of the polynomial function evaluated at xt.

Model Predict Control with Latency

I chose latency of the MPC as the same as dt which is 0.1 second. And I used the following equations to take consideration of latency before I feed these into solver.

x[t+1] = x[t] + v[t] * cos(psi[t]) * dt
y[t+1] = y[t] + v[t] * sin(psi[t]) * dt
psi[t+1] = psi[t] + v[t] / Lf * delat[t] * dt
v[t+1] = v[t] + a[t] * dt
cte[t+1] = f(x[t]) - y[t] + v[t] * sin(epsi[t]) * dt
epsi[t+1] = psi[t] - psi_desired[t] + v[t] * delta[t] / Lf * dt

I started to tune the parameters inside the solver by adjusting constraints multipliers. That was a challenging as well to make sure the car is on the track. I finally got to minimize the use of actuators by picking the factors of 50 and 60 for delta and acceleration respectively. Then adjusted the lower and upper bound of delta to be between -20 and 20 degree and acceleration to be between -1.0 and 1.0. These values seem to be working very well. But I can't seem to speed up the car over 30 mph. That is another area that I will try to make sure to tune parameters so I can drive faster.

Reflections

This project is the single most challenging project among term 2 projects. I have learned so much on this term 2 as well as from term 1. I hope to learn how it all come into the real car and wish the term 3 will cover on how these pieces work together. After all, the reason I am taking this Nano program is to learn how I can apply these knowledge that I learned to real self-driving cars to help building SDC for the people who cannot drive.