Akshay Khole
Udacity SDCND Term 2
MPC Project

The Model:

We use the global kinematic model for predicting our next state and actuator inputs. The model is simple and ignores certain parameters like tire forces, gravity, mass of the vehicle. This approach helps us build a model that is simple to compute in real time as well and give an output that can be used to drive the vehicle safely.

The model consists of a state having following parameters -

1. X position
2. Y position
3. Car's orientation (PSI)
4. Velocity of car (V)
5. Cross track error (how far vehicle is from actual trajectory) (CTE)
6. Error in PSI (difference in car's orientation trajectory's orientation) (e_PSI)

The steps for implementing the model are as follows:

1. Get waypoints of the actual path
2. Use waypoints to generate a curve / trajectory that fits the waypoints using the polyfit function
3. Now using the curve and our own X values, we can find corresponding Y values using the polyeval function
4. We define number of future time steps and frequency of steps i.e. N and dt.
5. We can now find actuator values of steering angle (delta) and acceleration / throttle for each of the time steps
6. We fetch the 1st of these actuator values and move the car by giving an input of delta and throttle.
7. Since the car has now moved, the future trajectory has changed and we must compute a new curve with respect to the new state and repeat this process again.

The trajectory is represented by a 3rd degree polynomial since it can fit most roads.

The general equations for the model used in the project are as follows:

$$x_{t1} = x_t + V_t * \cos(PSI) * dt$$
$$y_{t1} = y_t + V_t * \sin(PSI) * dt$$
$$V_{t1} = Vt + A * dt$$
$$PSI_{t1} = PSI_t + (V_t / Lf) * delta * dt$$
$$CTE_{t1} = CTE_t + V_t * \sin(e\_PSI_t) * dt$$
$$e\_PSI_{t1} = e\_PSI_t + (V_t / Lf) * delta * dt$$

L_f = Set as distance between front of car and center of car's gravity. We use recommended value of 2.67.

Here x and y are positions, V = velocity, PSI = car's orientation, A = Acceleration / Throttle, CTE = cross track error

Timestep Length and Elapsed Duration (N & dt):

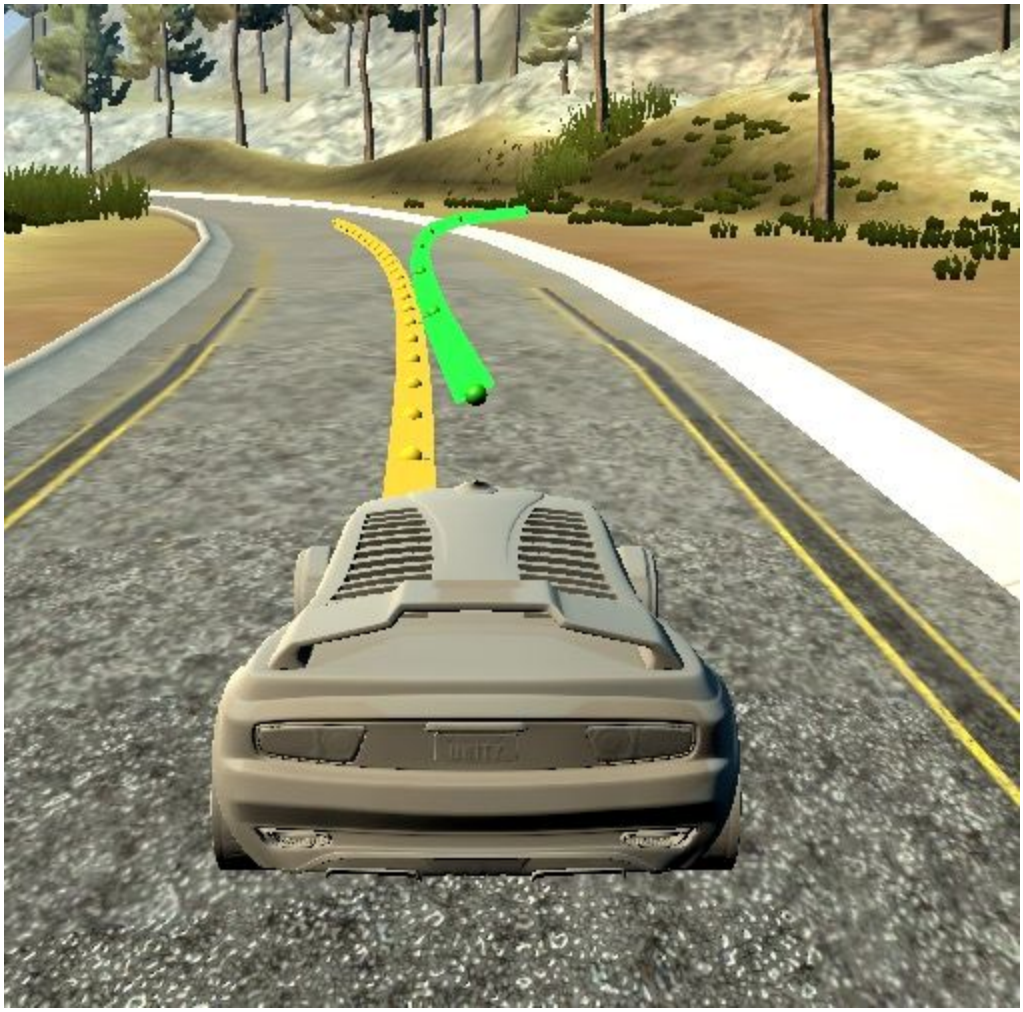
To get the ideal value of N and dt, I used the trial and error method as the first approach. I tried the following values with speed set to 50 -

1. N = 5, dt = 0.05
   This curve was too small and car would not have enough data to go ahead and would eventually go off track.

2. N = 20, dt = 0.5
   This curve was too large and car would drive very slow at turns.

3. N = 12, dt = 0.2
   The car drives well.

4. N = 8, dt = 0.5
   The car drives fine but has a very weird trajectory, probably because it does not have enough future information.



5. N = 12, dt = 0.5
   Car stops too often and does not drive smoothly. This may be because we are not calculating the trajectory often enough.
6. N = 12, dt = 0.2, speed increased to 65.
   Car drives well and trajectory is also good. The car reaches a max speed of 62 on straight road. I selected these values for the project.

Polynomial Fitting and MPC Preprocessing:

The waypoints are preprocessed to bring the x and y with respect to the origin and car's orientation is adjusted to align it with with a horizontal line over mapping the waypoints. This helps us get a

better polynomial fit.

Model Predictive Control with Latency:

Since in real life there may be a delay between giving an actuator command and the actuator attaining that position, we add a delay of 100 milliseconds between us predicting the command and actually having the vehicle set to the predicted throttle and steering angle. This is done by putting our thread to sleep for 100 milliseconds in our simulation.