CarND-MPC-Project

1. In this project I used kinematic equations for the state parameters(x coordinate, y coordinate, psi – car orientation angle, v – velocity, cte – Cross Track Error, psi error). Steering angle change (delta) and acceleration (a) are actuators and Lf – distance from center of gravity of the car to its front.

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f} * \delta_t * dt$$

$$v_{t+1} = v_t + a_t * dt$$

$$cte_{t+1} = f(x_t) - y_t + (v_t * sin(e\psi_t) * dt)$$

$$e\psi_{t+1} = \psi_t - \psi des_t + (\frac{v_t}{L_f} * \delta_t * dt)$$

2. For N (time step length) and dt (elapsed duration between time steps) I eventually chose these values: N = 7 and dt = 0.1 second. In my opinion, with these values my computer can make all needed calculations pretty fast so the car can move without any difficulties.

   Previous values were: N = 10, dt = 0.1. With these values calculations were quite slow.

   Then I temporarily changed the parameters to N = 5 and dt = 0.15. The car couldn't move properly with these values because the number of point was insufficient for a proper prediction of the trajectory.

3. I preprocessed waypoints so they could be represented as part of the car coordinate system: I wrote get_car_coordinates function for this purpose:

```
void get_car_coordinates(int pts_size, double psi,
double px, double py,
vector<double> ptsx, vector<double> ptsy,
VectorXd &ptsx_car, VectorXd &ptsy_car)
{
  for(int i = 0; i < pts_size; i++)
  {
    double dx = ptsx[i] - px;
    double dy = ptsy[i] - py;

    double x_car = dx * cos(psi) - dy * sin(psi);
    double y_car = dx * sin(psi) + dy * cos(psi);

    ptsx_car[i] = x_car;
    ptsy_car[i] = y_car;
   }
}
```

Then I get new coordinates in the main.cppusing get_car_coordinates function (using VectorXd from Eigen library):

```
psi *= -1.0;

int pts_size = ptsx.size();

VectorXd ptsx_car(pts_size);
VectorXd ptsy_car(pts_size);

get_car_coordinates(pts_size, psi, px, py,
                    ptsx, ptsy, ptsx_car, ptsy_car);
```

4. In this project I dealt with the latency of actuations that have a delay of 100 ms by taking into account the actuators from the previous state:

```
a0 = i > 0 ? vars[a_start + i - 1] : a0;
delta0 = i > 0 ? vars[delta_start + i - 1] : delta0;
```

Then I used these values in the update equations.