1. Screenshots of car completing a lap without incident

Given below are two snapshots of the car driving a full circuit without any incident.

2. Approach to Behavior Planning

I revised my approach of trajectory generation and behavior planning on the lines of the udacity walkthrough. The car starts with zero velocity and in each step the velocity is incremented by 0.1 m/s. The ego car starts in the middle lane. It keeps increasing the speed to reach target speed of 22 m/s ~ 50 mph. In case ego gets stuck behind a slow moving vehicle, it looks for opportunity to switch to nearby lanes using a simple set of rules

a) Look into left lane only if the car is not already in the innermost lane and see if there is safe clearance in the left lane to switch. It checks for safety by ensuring that there is no car in front in that lane which is either very close or little far buy moving slower than ego. Also there is no car behind in the target lane and is moving faster than ego. Based on the rules, if the left lane is found to be clear it sets its end goal state of the current trajectory to be the left lane.
b) In case left lane is not free, planner performs the same set of safety checking for right lane switch, unless the ego (car) is already in the outer most lane.
c) Rules (a) and (b) are not applied if the car in front is moving at close to target speed. In such a case ego slows down just a bit to not switch lane and still stay close to target velocity.
d) If no lane change can be done, ego slows down by 0.1 m/s in the current cycle.

This check is performed every cycle. Sensor data from simulator is used to plot the positions of all the cars. In essence my FSM right now is a very simple one with three states – KL (keep Lane), LCL (lane change Left) and LCR (Lane change right).

3. Trajectory Generation

Once target lane and speed is identified by behavior planner, trajectory generator uses the data from previous path, current ego car state and target speed and target lane to generate a smooth set of points for new trajectory.

As suggested, I use spline.h to generate a spline of five points – first two from the end of previous_path. If no previous path points are returned by simulator, two points close to the current ego position are taken. The next three points are generated at distance of 30m, 60m and 90m from the current “s” value of ego. Mid of Target lane based on FSM state is used as end state “d” value. These are then converted into global (X,Y) coordinates using “getXY()” function.

All these points are converted into local coordinate system which is either the current car state or the end of previous trajectory depending on availability of previous trajectory data from simulator. A spline is then fitted on these points.

At this point, we are ready to generate the new trajectory for ego. The length of new trajectory is always 50 points comprising of all the points from previous trajectory pts returned by simulator and balance generated using the above spline.

To generate points from spline, we first take a line segment on x axis which is 30 mts in front of ego. We use above fitted spline to find Y coordinate of this end goal point. After this distance of end point from current car coordinates i.e. (0,0) is calculated. We then divide the line connecting (0,0) to (X,Y) into small segments of size which respect the target velocity of ego as determined by behavior planner. These points are then projected down on X axis to get the x-cords of the new points and spline is used on these x-cord values to get the y-cords.

The points are then converted back into world co-ordinates and appended with previous points data. This the final list of points which is fed back to simulator as new trajectory.

The above approach is largely based on the Udacity walkthrough with some improvements esp. on behavior planner.

4. Areas of improvement

Some areas of improvement which I plan to work on in future.

a) This simple approach works satisfactorily and car is able to switch lanes. However, as explained in lectures, the absence of “prepare lane change” phase and associated behavior to slow down and align to a right spot in the target lane can get car stuck in its current lane at times. It can be improved by building a “prepare lane change” behavior to actively speed down the car to find a right opening in target lane.
b) Current behavior planner is based on strict and hard coded rules instead of using a cost function approach which is a recommended approach for better and stable behavior.
c) We can use sensor data to build a naïve bayes classifier and have a better model of prediction for future state of all the other cars.