| **Criteria** | **Meets Specifications** |
| --- | --- |
| There is a reflection on how to generate paths. | The code model for generating paths is described in detail. This can be part of the README or a separate doc labeled "Model Documentation". |

I would split my path planning algorithm into 3 parts:

1, **Prediction** the position of all the other cars on the road heading the same way as the ego car at time “t” (defined by horizon or based on remaining prediction from previous planning cycle)

I store this information and pass to the behavioral planning process model as input.

Basically the new values at time “t” get calculated by the following equitation:

s = s + v\*t\*0.02 + a\*t\*t/2 (in this project I calculate with 0 acceleration)

v = v + a\*t (as acceleration is 0, v=v, constant speed)

2, **Behavioral planning**

This is the second phase of the path planning. I run it only if there is a car close (30 m) in front of the ego car and makes the ego to slow down (CarInMinDistInLane function).

Then I look for alternative lanes to proceed quicker (SearchForLane function). It returns the action the car should take, like KL – Keep Lane, LCL – Lane Change Left, LCR – Lane Change Right. Currently I implemented only these 3 possibilities.

Within SearchForLane I do the following:

* looping over all the lanes
* if the lane is not the current lane I calculate cost for:
  + if the lane is free so the car could change to that lane (use CarInMinDistInLane function to check within the range of +20 ahead, - 10 behind)
  + lane change greater than 1 lane gets higher cost
  + too slow speed also has higher cost (< 20)
* trying to stay closer to the max speed limit, so in any lane I add cost for higher deviation from the maximum
* I check also in all lanes if there is a car within the range and further calculate the cost:
  + reward lanes where the closest car is faster, so it is worth to change and go faster
  + lanes get higher cost if the closest car is closer to the ego car (comparing “s” values)
* I calculate cost per lane and at the end pick the one with the lowest value. This is the new target lane for trajectory planning

3, **Trajectory generation**

the output of the behavioral planning algorithm is the input for trajectory creation. The car is going to follow the path projected by the trajectory.

X, Y values for the car to follow are calculated in GenerateTrajectory method.

For trajectory generation we need map waypoints, the ego car current localization data, prediction by behavioral planning, remaining trajectory point from previous planning cycle.

If there is sufficient remaining trajectory points from previous planning run, I use it, so the car can continue its path smoothly, there will be no sudden change.

I extend those points with new ones.

I create first 3 reference point 30 m spaced which serve for calculating polynomial coefficients. I use “spline” for that.

Then with the help of the polynomial I compute a more detailed path for the car to follow between the reference points.

As the car makes the distance between 2 points within 20 ms, I need to involve in the calculation the car’s speed and possible change rate of speed to avoid too high acceleration and jerk.

At the end, I convert the point calculated in the Car’s coordinate to map coordinate.

This trajectory then is passed to the simulator.

With the above logic, I could successfully and safely drive 4.32 miles with the simulator.

I think I got a good understanding how the path planning process flow works and I know that with applying further cost functions or generating multiple possible states through the behavioral planning and then let the trajectory generator calculate trajectory for all of them and decide on the best one, I could further improve the algorithm.

Prepare lane change left and right has not been implemented as I could see the car changes lane as soon as there is no traffic in next lane and a car is slowing too slowly ahead of it.

Definitely there is some room for improvement! Really interesting and challenging topic, I liked it!