Path Planning Project

Self Driving Car Nano Degree

Bryan Bui-Tuong

March, 2018

## GOALS

In this project, the goal is to design a path planner that is able to create smooth, safe paths for the car to follow along a 3 lane highway with traffic. A successful path planner will be able to keep inside its lane, avoid hitting other cars, and pass slower moving traffic all by using localization, sensor fusion, and map data.

## REQUIREMENTS

1. Driving > 4.32 miles without incident
2. Follows speed limit
3. Max acceleration and jerk not exceeded (10m/s^2 and 10m/s^3 respectively)
4. No collisions
5. Stays in lane except during lane change
6. Able to change lanes when car is behind a slower vehicle and adjacent lanes are clear of traffic

## BREAKDOWN

This project was accomplished by breaking down the problem into various phases:

1. Prediction
2. Behavior Planning
3. Trajectory Planning



Figure - Path Planning Steps

# Prediction + Behavior Planner

Our Behavior planner was modified from the planner used in the lecture using the Road, Vehicle and Cost classes. During each cycle, we generate a target speed and lane through our behavior planner. This is accomplished by the following steps:

1. Update localized and sensor fusion data in our “map”
2. Generate predictions of where each obstacle will be
3. Generate potential next states (see Figure 2)
4. Calculate cost for each state which is done by the following steps:
   1. In each potential next state, generate trajectory using predictions from Step 2.
   2. Calculate cost of trajectory
5. Return next state with the lowest cost



Figure - Finite State Machine

The cost that I used to calculate the best next state was lane inefficiency. The lane inefficiency equation is:

Vt = Target Speed.

This took into consideration the speed of the vehicle ahead of our car. If it was further away, its speed would be penalized less than if it were closer. All of our math was done in Frenet coordinates (Figure 3) to simplify things, then converted into X/Y when calculating actual trajectory.



Figure - Frenet Coordinates

# Trajectory Planner

Given a target lane and speed, we can now set a trajectory that is safe, comfortable, and feasible. To get our trajectory, we used the Spline library which we would use to help generate the appropriate waypoints. I approached this using the following steps:

1. Determine our velocity by adding target acceleration to our current velocity, preventing the velocity from going above our speed limit.
2. If we are in any of the “Prepare Lane Change” states, check if there are vehicles next to us.
3. Set the target lane if it’s safe to do so
4. Generate smooth and comfortable trajectory
   1. Create a list of widely spaced (x,y) waypoints, evenly spaced at 30m
   2. Use the previous path's endpoint as a starting reference
      1. Use the EGO’s starting point if previous path is nearly empty
   3. In Frenet add evenly 30m spaced points ahead of the starting reference
   4. Shift and then a rotation to move to the origin (ego/car's reference point). This is because when using spline, certain orientations of the car relative to map could give us undefined or infinite points. By rotating we prevent this problem from happening.
   5. Set x,y points for spline
   6. Define the actual x,y points for the spline
      1. Start with all of the previous path points from last time (this helps give a smooth transition)
      2. Calculate how to break up spline points so that we travel at our desired reference velocity
      3. Rotate back to normal after rotating it earlier: Rotate and then shift from Car's reference to map reference.

## Lessons Learned

* It was really important to keep understand what the sensor fusion data was providing, as it showed vehicles going in the opposite direction as well as the EGO’s lane.
* The prediction needed to be very accurate, or everything else will be thrown off.
* Generating trajectory for cost analysis in the Behavior Planning can be as simple or as complex as you want it to be. It was slightly different from the lecture in that we wanted to look at the speed of the vehicle ahead of us and that not all vehicles will travel at the same speed.



Figure - Completed Lap

## Future Improvements

Things that I would improve upon

* Behavior Planning
  + Adding more cost functions
    - Look at all lanes to determine if making multiple steps is better than staying in same lane. Eg: When there are two cars going near the same speed in your lane and the lane next to you, check the third lane to see if it makes sense to start making your way over there.
    - Safety Cost
    - Cost for changing lanes too much
* Trajectory Planning
  + Put into its own module
  + Add more safety checks
  + Implement Polynomial Trajectory Solver