Udacity Self-Driving Car Nanodegree
Term 3, Project 1: Path Planning

Mark Helms, November 26, 2018

Goal/Summary

The goal in this project is to autonomously control a vehicle in a highway-driving simulator using path planning and knowledge about the map, location and speed of the vehicle and other nearby vehicles. The vehicle must make a single lap around the track without leaving the road or colliding with other vehicles, all while trying to remain close to the speed limit without ever exceeding it. The path should include speed and lane changes. The simulator will move the car perfectly between waypoints generated by the path planner at 20 millisecond intervals. As often as possible, the simulator will ask for the latest path.

Approach

Step 1: Smoothing the Map

The map coordinates are given in Frenet and Cartesian coordinate systems. Frenet describes the position of a car along a road as (s,d), where s is the distance the car has traveled and d is the deviation from the center lane. In this case, d has a value of 0 at the center line and each lane has a width of 4 meters. There are three lanes. This means the d value of the center of the middle lane is 6 meters.

One issue with the map coordinates is that the distance between map points if very large. Interpolation between them therefore becomes necessary. Linear interpolation results in discontinuities, so a spline interpolation is used, as suggested by Udacity. The map waypoints (x,y) are converted into (r, theta) points, where r = sqrt(x^2 + y^2) and theta is the arctan(y/x). The map waypoint vector for s is then used along with r and theta to create two splines, which can be used immediately for a smooth representation of the entire map. Note that additional points need to be added at the beginning and end of the point vectors fed into the spline calculation in order to close the curve smoothly.

Step 2: Picking a Target Lane and Target Speed
Code: isLaneOpen, getClosestCar, getSensorSpeed, getSensorDistance, onMessage lines 474-549

As the car is driving, it may be desirable for it to change into a faster lane to avoid traffic. A process is repeated for each lane:

- Detect if the lane is open, by looking for vehicles with a Frenet s value close to ours within a margin of safety (one example of how easy working with Frenet coordinates can be). If the lane is occupied, ignore it.
- Find the closest car in front of us in that lane.
- Record the closest car's distance and speed. If the closest car is far enough, we may consider it to be non-existent.
- If the speed is the greatest speed we have available, set our target speed to that speed and our target lane to that lane. If the resulting target lane is more than two lane changes away, however, we set our target speed and lane to that of the intermediate lane.

In addition, priority is given to being in the middle lane, due to the option of being able to change lanes into any lane in the future. If a lane change is already in progress, any suggestion about a new lane is ignored, although suggestions about a new speed in our lane are honored and result in a nearly entirely new trajectory.

Step 3: Drawing a Trajectory
Code: Generating a lane change spline: line 589, Creating an s progression: line 617

If a lane change or significant speed change is detected, an almost entirely new trajectory is started, otherwise most of the old trajectory is reused.

In order for the car to transition smoothly, the trajectory must be planned to avoid sudden changes in acceleration, velocity, heading and location. The target speed and lane dictate the new path entirely, along with the previous points that are reused. Instantaneous acceleration and velocity are calculated and stored so that past points are easily accessible. The number of points used are always dictated by the length of a 2.5 second lane change (always 125 points).

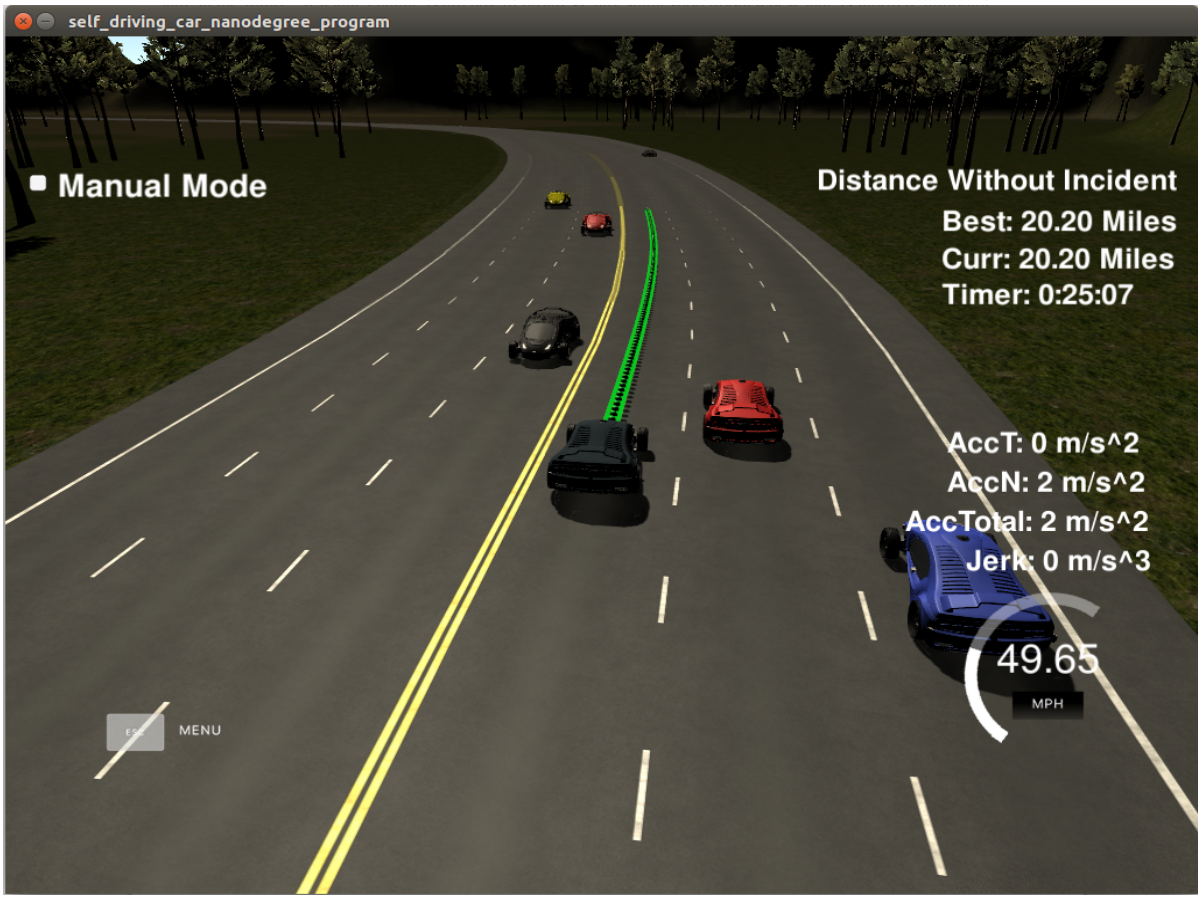
The process for managing velocity is:

- Adjust acceleration (limited by maximum jerk allowed) if velocity is not at the desired level
- Adjust velocity with the acceleration * the waypoint transition time (always 20 milliseconds).
- Adjust position along the curve with the velocity * the waypoint transition time (always 20 milliseconds).
    - Position is the s value of the Frenet coordinates.

If a lane change is necessary, a spline is used to calculate transition points from the current lane to the target lane. This simply becomes the d value of the Frenet coordinates.

Now that a (s,d) exist for each waypoint of our trajectory, we can get the smoothed (x,y) coordinates using the spline representation of the map and then converting the resultant (r,theta) using some trigonometry (x=r*cos(theta),y=r*sin(theta)).

The problem has now been solved! See a screenshot of a successful 20 mile run below.



Discussion

Unfortunately, no use of A* was implemented, and so the strategy coded is not optimal. One example of this is that the car will never slow down in order to get over to a faster lane. The algorithm here is a greedy one in that if it gets stuck, it will tailgate the car in front of it until the lane next to it opens up. A* could be used by generated a predicted map for each timestamp in the future.