WriteUp.md

Path Planning

WriteUp for Path Planning project in Term 3.

Compilation

The code compiles correctly

The code is able to run using cmake.

Valid Trajectories

The car is able to drive at least 4.32 miles without incidents.

The car will (most often) run the track without incidents. Before submission I ran a track for 26.2 Miles (~40 min) without any incidents. The solution will however eventually run into problems.

The solution complices with the following.

- The car drives according to the speed limit
- Max Acceleration and Jerk are not Exceeded
- Car does not have collisions
- The car stays in its lane, except for the time between changing lanes
- The car is able to change lanes (more on this in the next heading).

More fun is to set the top_speed to 80 or 100.

Reflections

Reflection on how to generate paths

Multiple procedures are required to move smoothly through traffic.

Behavior Planning

First some optimal behavior must be determined... More specifically, we need to find a suitable speed and lane position for the vehicle to drive fast and safe through traffic. The procedure for this is to generate a set of potential trajectories. The submitted solution generates a set of 15 trajectories - five different speeds for each of the three lanes. The initial filtering removes speeds below 0, speeds above 49.5 mph and invalid "lanes" - lanes that are either in the oppersite direction or outside the road... For valid speeds and lane positions, the interpolated path is calculated in the fresnet coordinate system. By predicting the position of other vehicles on the road - using a simple constant velocity model - it is possible to determine when or if a given trajectory will collide with other vehicles.

An optimal solution for the given set of trajectories is defined as simply the longest path without collision - this concept is taken from mikkelkh. We have now determined the "optimal" speed and lane position. The next goal is to make actuation of this path smooth (low jerk). This is described in the following section.

Trajectory Generation

Two tricks - similar to those presented in the project walkthrough - are used to generate smooth trajectories. First, the speed of the vehicle is restricted to only increasing or descrease by a specific factor for each iteration. This ensures that the max acceleration and jerk is not exeeceded (in the d-direction of the fresnet coordinate system). Secondly, smooth lane changes - acceleration and jerk in the s-direction of the fresnet coordinate systems - are performed using spline interpolation. Unlike the walkthrough, I initially used tracks described in the fresnet coordinate system as in the behavior planning module... This was unsuccessful as the fresnet-based trajectories would simply follow the coarse waypoints, which will result in large jerk and accelerations. Eventually, I used the approach presented in the walkthrough, where spline interpolation is performed on the "normalized" waypoints. Finally, it is also important to reuse previously generated trajectory points to avoid sudden movenments and glitch-like movements.