Writeup – HIGHWAY DRIVING

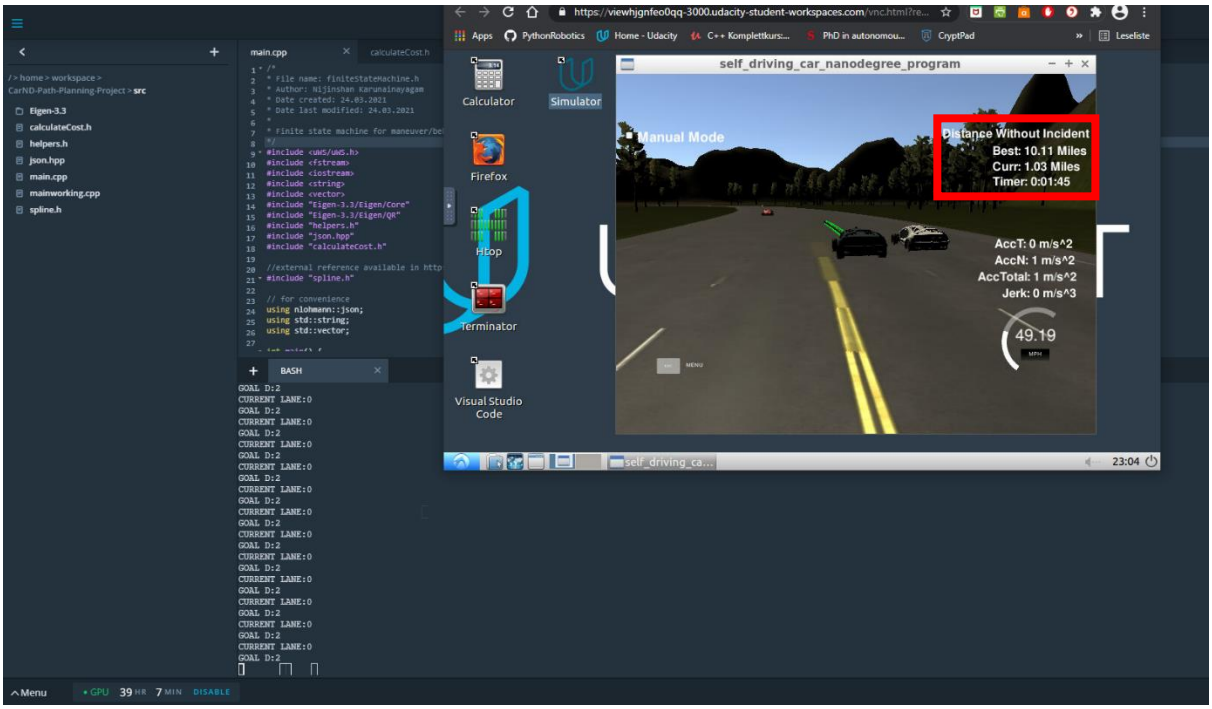
COMPILATION

The code compiles without errors. It can be started by by typing “cmake .. && make” in the terminal within the build folder.

VALID TRAJECTORIES

DISTANCE WITHOUT INCIDENTS

The car is able to drive at least 4.32 miles without incidents (at least in my tests). Because of the randomness of the traffic participants and their impact to the performance of the path planner, my own test requirement was to check how far the Ego Vehicle can drive without incidents. Finally, the Vehicle was able to drive about 10 miles without any incidents (see picture below), which is quite acceptable.

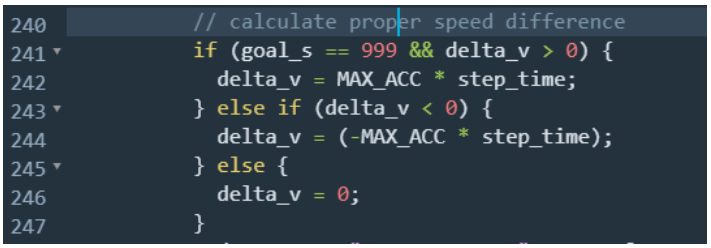


SPEED LIMIT REQUIREMENT

```
352             // Set ego's velocity according to lead vehicle and speed limit
353             ref_vel += delta_v;
354             if(ref_vel > MAX_VEL){
355               ref_vel = MAX_VEL;
356             }
```

In this code sequence, it is provided, that the vehicle is not exceeding the speed limit (*MAX_VEL)*. Additionally, the speed difference between the ego and the lead vehicle is considered to adapt ego’s velocity to the lead vehicle’s velocity.

MAX ACCELERATION AND JERK



The fulfillment of the acceleration and jerk requirements is achieved by considering a smaller maximal acceleration of 9 m/ss in the calculation of the speed difference between ego and target.

COLLISION AVOIDANCE

To avoid collisions, first the ego's environment is checked by considering all objects in the senor fusion data. During this process, the distance and velocity of the closest front and rear vehicle of the same lane, the left lane, and the right lane (from ego's point of view) is determined. With these values it is possible to evaluate a collision-free action inside a finite state machine, which represents the behavioral planner. According to ego's environment, some costs for different actions are calculated (*calculateCost.h*). Thus, the state machine switches between "CRUISE" (no lead vehicle in front; drive with a speed of 50 mph) "LANE_CHANGE_LEFT" (change to left lane) and "LANE_CHANGE_RIGHT" (change to right lane). Finally, a goal lane and a goal d-coordinate are determined, depending on the current state.

PATH PLANNING

To execute a smooth lane change, a spline is calculated. The calculation of the spline is adapted from Mr. Tino Kluge (*spline.h*), which is a simple cubic spline interpolation. For detailed information about that process, see the comments in the code.