Highway Driving

Student: Guido Rezende de Alencastro Graça

Project Goals and Code Pipeline:

This is the report of the "*Highway driving*" project of the Self Driving Vehicles course on Udacity. This project goal is to build a code that is able to drive the autonomous vehicle in a highway, populated with other vehicles, while fulfilling criteria on both safety and efficiency. These criterias are the following, as detailed on the *Project Rubric*:

1. The car must drive according to the speed limit. The car does not drive faster than the speed limit. In addition, the car is not driving much slower than speed limit unless obstructed by traffic.
2. The car must not exceed a total acceleration of 10 m/s^2 and a jerk of 10 m/s^3.
3. The car must not come into contact with any of the other cars on the road.
4. The car should stay in its lane, except for the time between changing lanes. The car should not spend more than a 3 second length outside the lane lanes during changing lanes, and every other time the car stays inside one of the three lanes on the right hand side of the road.
5. The car must be able to smoothly change lanes when it makes sense to do so, such as when behind a slower moving car and an adjacent lane is clear of other traffic.

The simulator keeps track of all the incidents that inflict any of those criterias. Also, the vehicle must be able to run for at least 4.32 miles without ant incident. Figure 1a shows that the vehicle is able to run nicely up to 12 miles. The vehicle is also able to perform lane changes, as illustrated on Figures 1b and 1c.

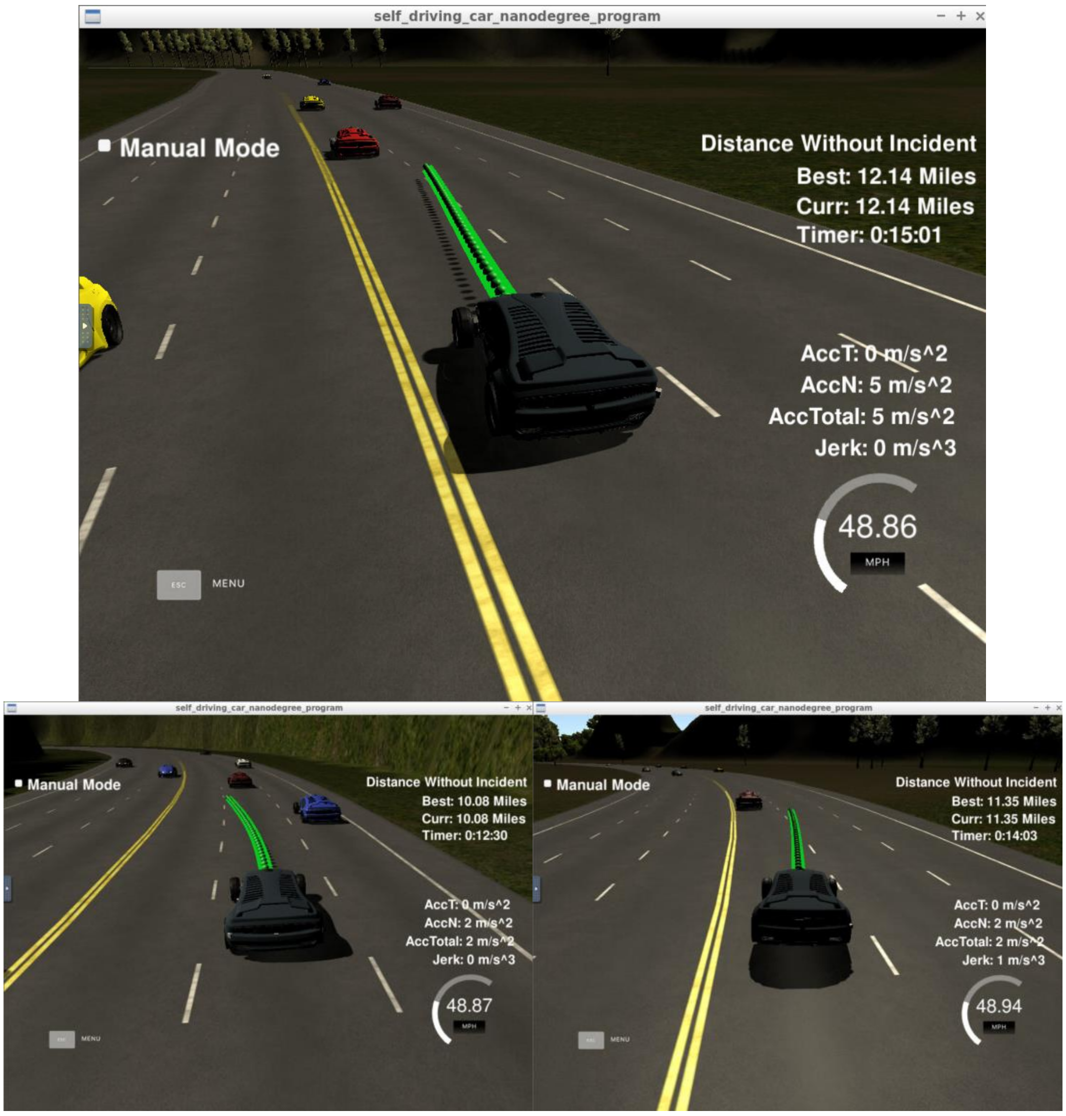

Figure 1: Vehicle on the simulator. a) Running for 15 minutes without incidents. Total distance equals to 12.14 miles; b) left lane change; c) right lane change

Code structure:

The code is inside the main function. The step developed are:

1. Sense the surrounding using sensor fusion (lines 119 to 149). Here, the data from sensor fusion is processed. The surrounding vehicles are analysed and, depending on their positions, certain boolean variables are changed. For instance, if there is a vehicle on the same line but going slower, it triggers a *too_close* boolean variable to change its value. Similary, booleans variables *safe_turn_left* and *safe_turn_right* are also changed when there are cars near the adjacent lanes.
2. Deciding the behavior (lines 152 to 175). Using the Boolean variables from the sensing step, the behavior of the vehicle is planned. If there is a slow vehicle on the lane, then some action is made. First, the code will try to perform a lane change, first to left otherwise to the right. If lane change is unavailable, the vehicle will slow down to match the slow vehicle speed.
3. Creating the path (lines 177 to 289). This step is divided into 2 steps
    a. Creating the spline (lines 177 to 246). Using the spline.h file, a spline is created using five points: The current and previous car position, along with three

points forward. These three points were defined as 30, 60 and 90 meters ahead in the s Frenet Coordinate of the vehicle.

b. Setting the Next Values for the position of the vehicle (lines 248 to 287). In this step, the cars path is calculated, using the current velocity and acceleration, without trespassing speed, acceleration nor jerk restrictions. This is done using the calculated spline and dividing into points spaced closed enough for avoiding trespassing the said limits. The list of points are given to the *next_vals* lists.