Path Planning Project

Goals

In this project the goal is to safely navigate around a virtual highway with other traffic that is driving +-10 MPH of the 50 MPH speed limit. We will be provided the car's localization and sensor fusion data, there is also a sparse map list of waypoints around the highway.

The car should try to go as close as possible to the 50 MPH speed limit, which means passing slower traffic when possible, note that other cars will try to change lanes too.

The car should avoid hitting other cars at all cost as well as driving inside of the marked road lanes at all the times, unless going from one lane to another.

The car should be able to make one complete loop around the 6946m highway. Since the car is trying to go 50 MPH, it should take a little over 5 minutes to complete 1 loop.

Also, the car should not experience total acceleration over 10 m/s^2 and jerk that is greater than 10 m/s^3.

Compilation

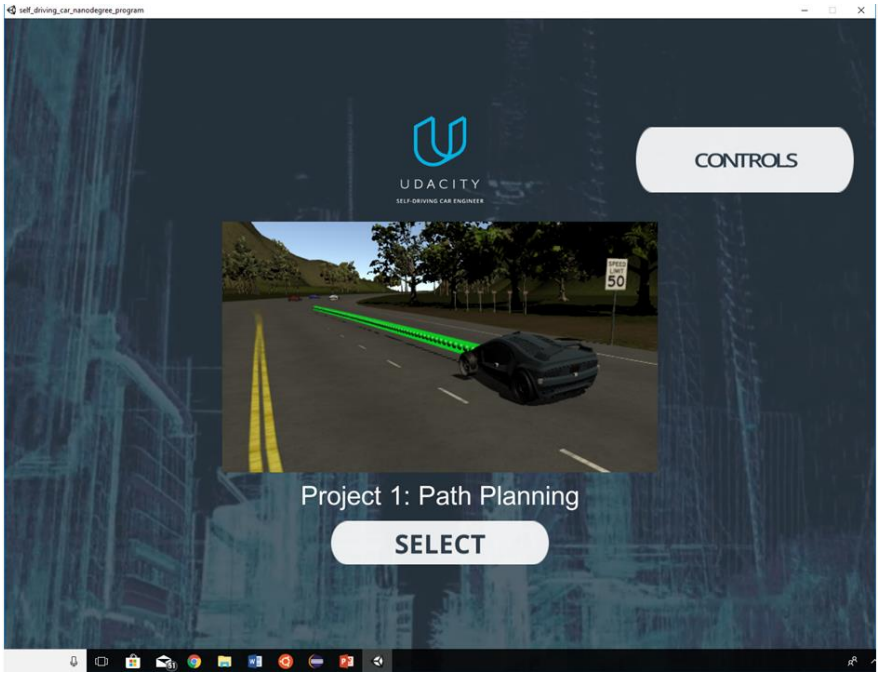
The programs that have been written to accomplish the project is: src/main.cpp (to create smooth trajectories, a single header file spline.h is added into the folder /src).

This project made a 'build' directory in the project folder, and start compiling by doing the following:

```
cmake .. && make
```

The C++ code will be compiled without errors. After the C++ code is compiled as command above, you can input following command:

```
./path_planning
```



Now the path planner is running and listening to port 4567 for messages from the simulator. Next step is to open Udacity's simulator in latest term 3 version (see picture above). In “Project 1: Path Planning”, click the "SELECT" button, then the car starts driving.

Valid Trajectories

In my continuous observation on the simulator in 11:21 time duration for 8.64 miles driving (two loops of the track), no incident happened (see the picture below). Following is the 6 criteria to see whether the valid trajectories is made by our project.

(A) The car is able to drive at least 4.32 miles without incident

The top right screen of the simulator shows the current/best miles driven without incident. Incidents include exceeding acceleration/jerk/speed, collision, and driving outside of the lanes. – Two loops of the track finished running without incident.

(B) The car drives according to the speed limit

The car doesn't drive faster than the speed limit: 50 MPH. Also, the car isn't driving much slower than speed limit unless obstructed by traffic. – Driving in 8.64 mile during 11:21, the average speed (by calculating) is about 45.7 MPH for the two loops.

(C) Max Acceleration and Jerk are not Exceeded

The car does not exceed a total acceleration of 10 m/s^2 and a jerk of 10 m/s^3

(D) Car does not have collisions

In my continuous observation on the simulator in 11:21 time duration, the car doesn’t come into contact with any of the other cars on the road.

(E) The car stays in its lane, except for the time between changing lanes

The car doesn't spend more than a 3 second length outside the lane lines during changing lanes, and every other time the car stays inside one of the 3 lanes on the right-hand side of the road.

(F) The car is able to change lanes

The car is 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.

In four of the pictures above, the pictures showed as:

(1) Picture on the top-left corner shows the car is going to change to the right lane, from middle lane to right lane.
(2) Picture on the top-right corner shows the car is going to change to the right lane, from left lane to middle lane.
(3) Picture on the bottom-left corner shows the car is going to change to the left lane, from middle lane to left lane.
(4) Picture on the bottom-right corner shows that the car is just slow down and keep the lane because both side is with the cars for making a lane change not safe.

Reflection

The code model in main.cpp for generating paths is described in detail as following.

1. **Prediction -- Utilize the sensor fusion data to check each lane's availability**

In main.cpp from line 255 to line 306, the code went through all the detected vehicle's information, status, and check the availabilities of each lanes.

The object "sensor_fusion" is a 2d vector of cars. The meaning of seven related values are: [car's unique ID, car's x position in map coordinates, car's y position in map coordinates, car's x velocity in m/s, car's y velocity in m/s, car's s position in frenet coordinates, car's d position in frenet coordinates].

In the 'for Loop' from line 264 to line 306, the code deals with the telemetry and sensor fusion data, and want to know whether:

(a) there is a car in front of us blocking the traffic (line 275 to line 281), or

(b) there is a car to the left of us making a lane change not safe (line 283 to line 293), or

(c) there is a car to the right of us making a lane change not safe (line 295 to line 305).

2. **Behavior Planning -- Based on the lanes' availability to plan the next state of the car**

In main.cpp from line 308 to line 322, based on the lane's availability which is obtained from the sensor fusion. Choose one of the following states for the car.

(a) Change to left lane (line 310 to line 312)

(b) Change to right lane (line 313 to line 315)

(c) Keep lane (speed up at "line 320 to line 322", slow down at "line 316 to 318", or keep speed)

3. **Trajectory Generation -- Use anchor points to interpolate a spline curve**

In main.cpp from line 324 to line 439, this code does the calculation of the trajectory based on the speed and lane output from the behavior, car coordinates and past path points.

First, the last two points (line 351 to line 367) of the previous trajectory (or the car position if there are no previous trajectory, lines 339 to line 350) are used in conjunction three points at a far distance (lines 369 to 372) to initialize the spline calculation (line 393 and 397).

To make the work less complicated to the spline calculation based on those points, the coordinates are transformed (shift and rotation) to local car coordinates (lines 382 to 390).

In order to ensure more continuity on the trajectory (in addition to adding the last two points of the pass trajectory to the spline adjustment), the pass trajectory points are copied to the new trajectory (lines 399 to 409). The rest of the points are calculated by evaluating the spline and transforming the output coordinates to non-local coordinates (lines 420 to 439).

Conclusion

In this project, the autonomous car is able to complete loop around the 6946m highway for twice without collisions with other vehicles and uncomfortable jerk, and its follows the road speed limits with gentle acceleration. Besides, the car is able to take over the slow vehicles when there are suitable opportunities, e.g. enough spaces to both leading and following vehicles.

The lane changes algorithm is quite simple. To further improve the path planner, we can build a finite state machine to move between actions, and build a cost function to determine the best action.