CarND: Path Planning Project

Goals

In this project your goal is to safely navigate around a virtual highway with other traffic that is driving +-10 MPH of the 50 MPH speed limit. The car's localization and sensor fusion data are provided, 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 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.

Each waypoint in the list contains [x,y,s,dx,dy] values. x and y are the waypoint's map coordinate position, the s value is the distance along the road to get to that waypoint in meters, the dx and dy values define the unit normal vector pointing outward of the highway loop.

The highway's waypoints loop around so the frenet s value, distance along the road, goes from 0 to 6945.554.

Dependencies

The project has the following dependencies (from Udacity's seed project):

- cmake >= 3.5
- make >= 4.1
- gcc/g++ >= 5.4
- libuv 1.12.0
- Udacity's simulator (https://github.com/udacity/self-driving-car-sim/releases/tag/T3_v1.2).

Basic Build Instructions

1. Clone this repo.
2. Make a build directory: mkdir build && cd build
3. Compile: cmake .. && make
4. Run it: ./path_planning.

Here is the data provided from the Simulator to the C++ Program

Main car's localization Data (No Noise)

["x"] The car's x position in map coordinates

["y"] The car's y position in map coordinates

["s"] The car's s position in frenet coordinates

["d"] The car's d position in frenet coordinates

["yaw"] The car's yaw angle in the map

["speed"] The car's speed in MPH

Previous path data given to the Planner

//Note: Return the previous list but with processed points removed, can be a nice tool to show how far along the path has processed since last time.

["previous_path_x"] The previous list of x points previously given to the simulator

["previous_path_y"] The previous list of y points previously given to the simulator

Previous path's end s and d values

["end_path_s"] The previous list's last point's frenet s value

["end_path_d"] The previous list's last point's frenet d value

Sensor Fusion Data, a list of all other car's attributes on the same side of the road. (No Noise)

["sensor_fusion"] A 2d vector of cars and then that car's [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.

Code explanation

The entire code is based on the seed project provided by udacity.
The core of the implementation is in the src/main.cpp file from line 100 to line 307
The code could be break down in different functional parts:

Prediction (from line 100 to line 161)

This part of the code deal with the telemetry and sensor fusion data.
Every element in the sensor fusion data represent a detected car in the highway.
For every car we extract the lane, the s position, and the velocity in order to define if there is:

- A car ahead of us, within 30 m;
- A car on the left lane in range +- 30 m;
- A car on the right lane in range +- 30 m.

Behavior Planning (from line 162 to line 199)

This part, given the output of the prediction step, is in charge of defining the behavior the car has to follow.

The algorithm always tries to stay in the central lane with the maximum speed allowed.

If another car is ahead, within 30 m from us, there are different choices available:

- If the left lane exists and is free, the target lane to follow becomes the left lane;
- If the left lane is busy or does not exist and, the right lane exists and is free, the target lane to follow becomes the right lane;
- If neither left nor right lane are available, the car slows down until it reaches the velocity of the car ahead.

Trajectory generation (from line 200 to line 307)

This code does the calculation of the trajectory based on the speed and lane output from the behavior planning, car coordinates and past path points.
In order to make the math easier, the coordinates are first shifted and rotated to the local car coordinates.
Then, the last two points of the previous trajectory and three points ahead at 30 m distance each other are used to initialize the spline calculation.
The rest of the points are evaluated by the spline and then the output coordinates are transformed to global coordinates.

Rubric points (https://review.udacity.com/#!/rubrics/1971/view)

Compilation

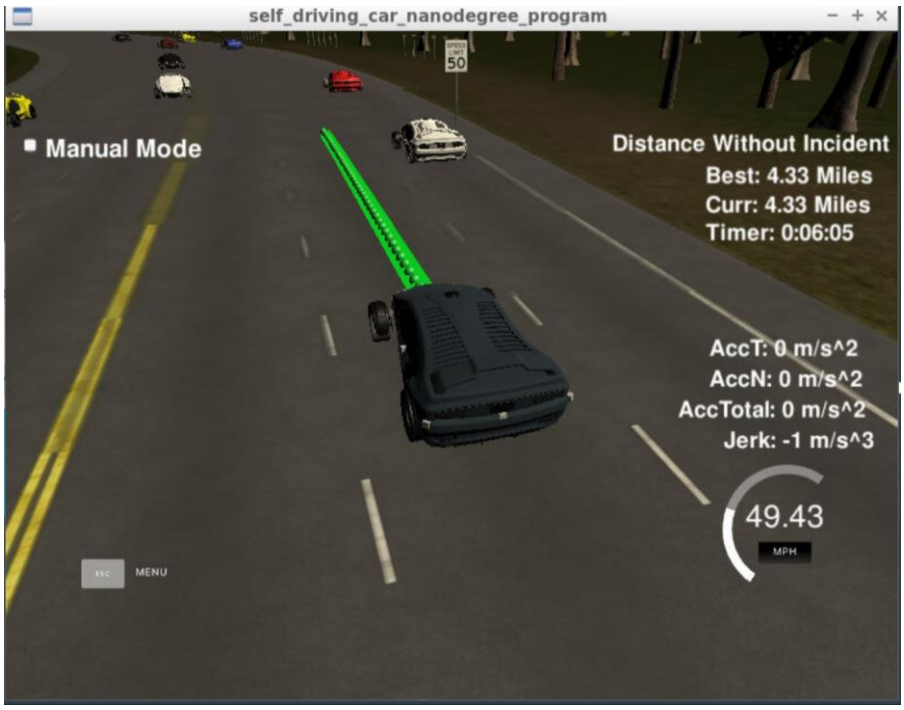
The code compiles correctly.

```
root@4c43cee83b30:/home/workspace/CarND-Path-Planning-Project/build# cmake .. && make
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/workspace/CarND-Path-Planning-Project/build
Scanning dependencies of target path_planning
[ 50%] Building CXX object CMakeFiles/path_planning.dir/src/main.cpp.o
[100%] Linking CXX executable path_planning
[100%] Built target path_planning
```

Valid trajectories

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

The simulation completes a loop without any incident. Here there is an image.



The car drives according to the speed limit.

The car always stays under the 50-mph limit.

Max Acceleration and Jerk are not Exceeded.

Acceleration and Jerk are always under control and do not exceed the limits

Car does not have collisions.

No collisions happen during the entire simulation.

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 lanes during changing lanes.

The car is able to change lanes

If the car ahead is slower and one of the side lanes is free, the car smoothly change lane.

Further improvements

Some improvements might be applied in the project especially in the behavior planning part.
In fact, when defining whether another car is in the left/right lane and blocks the lane change, we only check it the car is in range +- 30 m from us without considering the speed.
A more sophisticated algorithm might consider also the velocity so that, if a car is on the left/right lane below us within 30 m, if its velocity is lower than ours, it is safe to change lane.
In the same way, If a car is in the left/right lane ahead us within 30 m, if its velocity is higher than ours, it is safe to change lane.