

Path Planning

1. Purpose :

Purpose of the project is to design a path planner that is able to create smooth, safe paths for the car to follow along a 3 lane highway with traffic. Path planner shall keep the car inside its lane, avoid hitting other cars, and pass slower moving traffic all by using localization, sensor fusion, and map data.

2. Project Inputs :

- a. A Github repo with starter code(Workspace can also be used)
- b. A Simulator which provides the script for the noisy position data, vehicle controls, and noisy observations. The simulator can also display the best particle's sensed positions, along with the corresponding map ID associations.
- c. Files in the Source (src) folder includes –
 - i. main.cpp
 - ii. helpers.h
 - iii. json.hpp
- d. Other files
 - i. The map of the highway is in data/highway_map.txt
- e. Simulator
 - i. Simulator is provided to create highway environment and provide required data
 - ii. Following data is provided from the Simulator to the C++ Program
 1. 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
2. Previous path data given to the Planner
- ["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
3. 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
4. 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.



Figure 1: Simulator – Vehicle status information

3. Project Outputs :

- a. To satisfy project rubric following artifacts are expected –
 - i. build Folder
 - ii. path_planning run file
- b. To satisfy project rubric following specifications to be satisfied –
 - i. Compilation - The code compiles correctly.
 - ii. Valid Trajectories –
 - The car is able to drive at least 4.32 miles without incident.
 - 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
 - iii. Reflection –
 - There is a reflection on how to generate paths.

4. Introduction to Path Planning :

In the real world path planning is required for supporting different feature of Advance/Autonomous Driving Assist systems. The features which needs path planning at slow speed(<30 Kmph) can be Traffic Jam Co-Pilot and Valet Parking. The features which needs path planning at high speed(>60 Kmph) can be Highway Pilot, Highway Lane Change Assist.

As per the autonomous level of the vehicle the path planning becomes very critical. For Level 2 feature since driver is in operational loop so he is attentive about the road conditions and can take over the steering in case of emergency situation. But as we go above Level 3 where driver's involvement is less there the safety criteria has to be robust. The vehicle should take care of the environmental situation to make the decision for lateral and longitudinal movement.

Following is the information about different Driver assistance levels and system involvement from US Transportation Department.

Level	Name	Narrative Definition	DDT - Sustained lateral and longitudinal vehicle motion control	DDT - OEDR	DDT fallback	ODD
Driver performs part or all of the DDT						
0	No Driving Automation	The performance by the <i>driver</i> of the entire DDT, even when enhanced by <i>active safety systems</i> .	<i>Driver</i>	<i>Driver</i>	<i>Driver</i>	<i>n/a</i>
1	Driver Assistance	The <i>sustained</i> and ODD-specific execution by a <i>driving automation system</i> of either the <i>lateral</i> or the <i>longitudinal vehicle motion control</i> subtask of the DDT (but not both simultaneously) with the expectation that the <i>driver</i> performs the remainder of the DDT.	<i>Driver and System</i>	<i>Driver</i>	<i>Driver</i>	<i>Limited</i>
2	Partial Driving Automation	The <i>sustained</i> and ODD-specific execution by a <i>driving automation system</i> of both the <i>lateral</i> or the <i>longitudinal vehicle motion control</i> subtask of the DDT with the expectation that the <i>driver</i> completes the OEDR subtask and supervises the <i>driving automation system</i> .	<i>System</i>	<i>Driver</i>	<i>Driver</i>	<i>Limited</i>
ADS ("System") performs the entire DDT (while engaged)						
3	Conditional Driving Automation	The <i>sustained</i> and ODD-specific performance by an ADS of the entire DDT with the expectation that the DDT fallback-ready user is <i>receptive</i> to ADS-issued <i>requests to intervene</i> , as well as to DDT performance-relevant system failures in other vehicle systems, and will respond appropriately.	<i>System</i>	<i>System</i>	<i>Fallback-ready user (becomes the driver during fallback)</i>	<i>Limited</i>
4	High Driving Automation	The <i>sustained</i> and ODD-specific performance by an ADS of the entire DDT and DDT fallback without any expectation that a user will respond to a <i>request to intervene</i> .	<i>System</i>	<i>System</i>	<i>System</i>	<i>Limited</i>
5	Full Driving Automation	The <i>sustained</i> and unconditional (i.e., not ODD-specific) performance by an ADS of the entire DDT and DDT fallback without any expectation that a user will respond to a <i>request to intervene</i> .	<i>System</i>	<i>System</i>	<i>System</i>	<i>Unlimited</i>

Figure 2: Autonomous Levels

Following are some examples of possible lane change maneuvers -

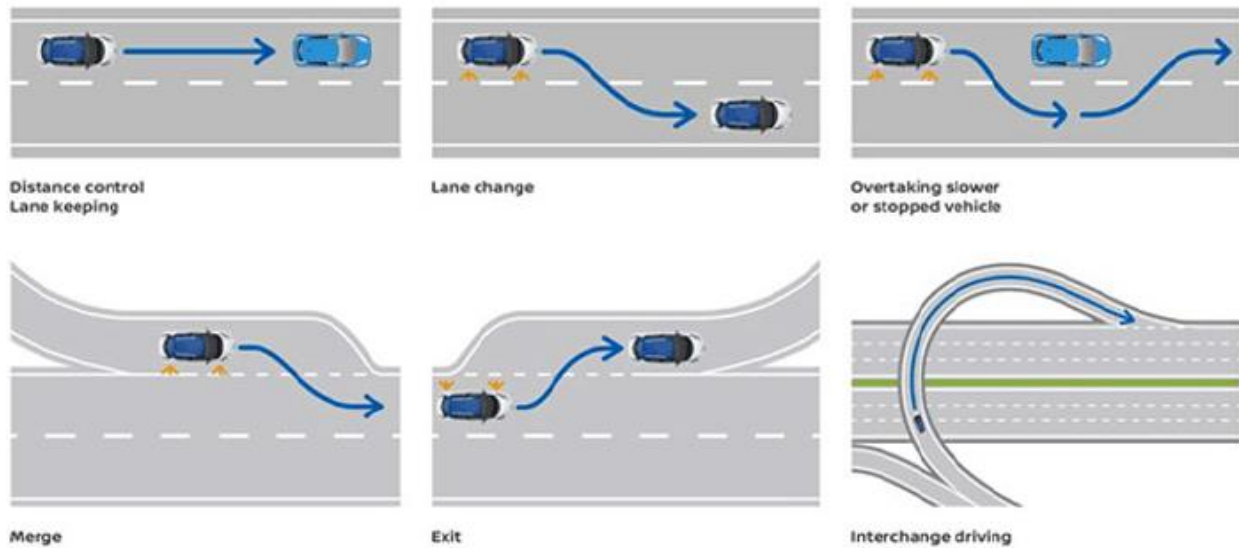


Figure 3: Lane change maneuvers

5. Project Execution :

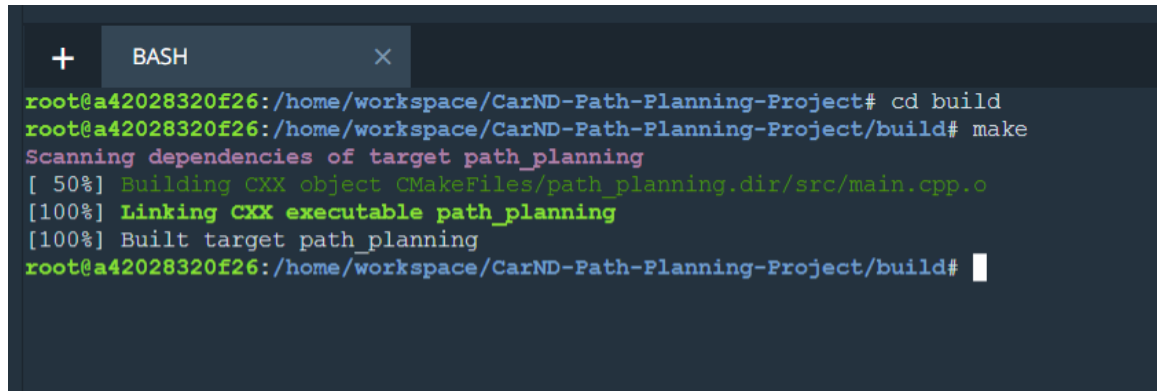
a. Code Build :

- Following is the snapshot for cmake command –

```
root@e8fdf0433046:/home/workspace/CarND-Path-Planning-Project/build# cmake ..
-- The C compiler identification is GNU 7.2.0
-- The CXX compiler identification is GNU 7.2.0
-- Check for working C compiler: /opt/conda/bin/cc
-- Check for working C compiler: /opt/conda/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: /opt/conda/bin/c++
-- Check for working CXX compiler: /opt/conda/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
root@e8fdf0433046:/home/workspace/CarND-Path-Planning-Project/build#
```

Figure 4: cmake command execution

- Following is the snapshot for make command –



```
+ BASH X
root@a42028320f26:/home/workspace/CarND-Path-Planning-Project# cd build
root@a42028320f26:/home/workspace/CarND-Path-Planning-Project/build# make
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
root@a42028320f26:/home/workspace/CarND-Path-Planning-Project/build#
```

Figure 5: make command execution

- Final Folder Structure –

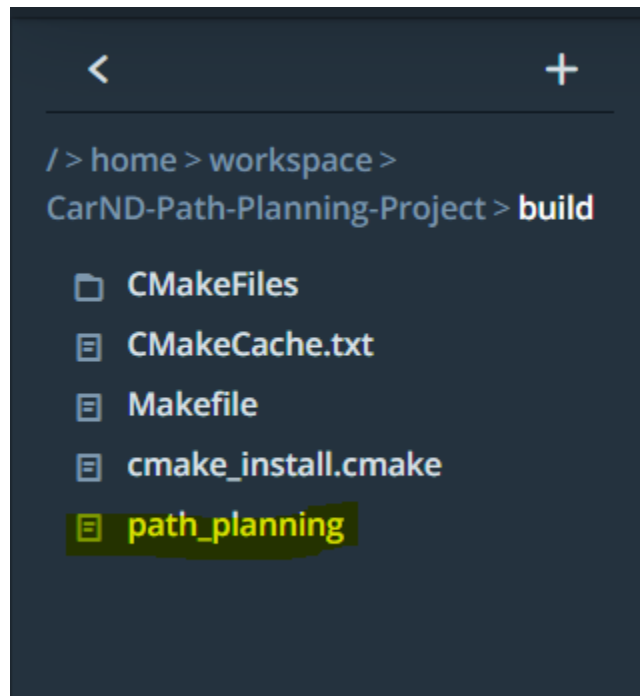


Figure 6: Build Folder Structure

b. Output:

The output video file is uploaded on the workspace at following location –

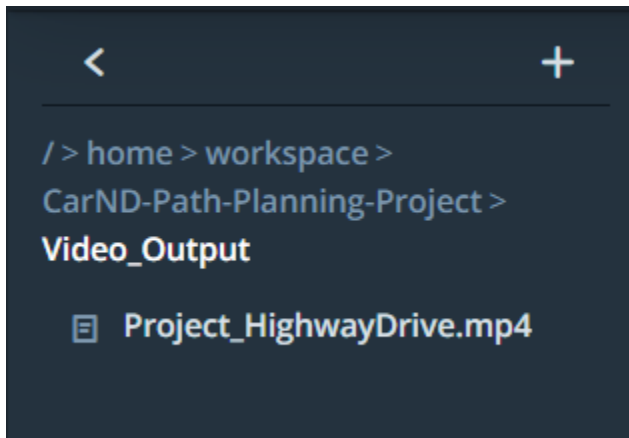


Figure 7: Output Video file

The video file is also available on you tube at [Project HighwayDrive](#).

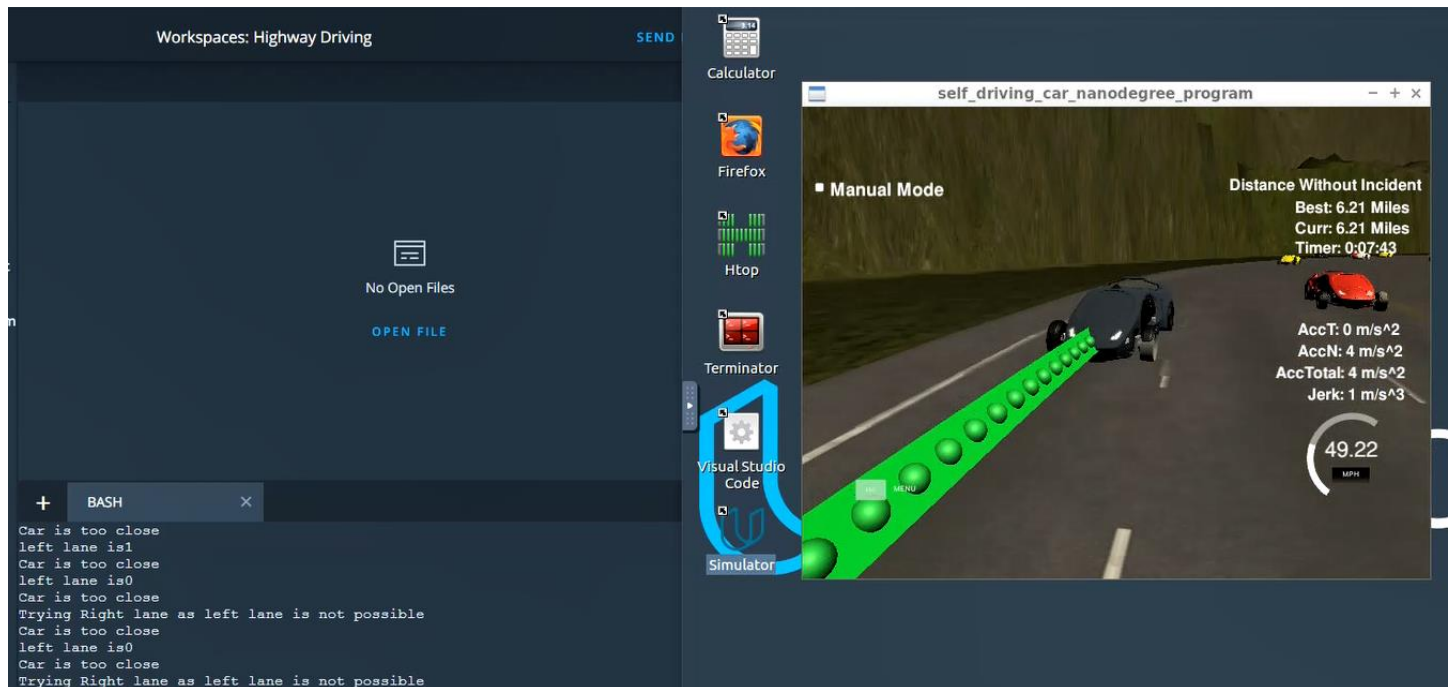


Figure 8: Output Video Snapshot

c. Reflection on path generation:

The simulator sends information consisting of Car's location, velocity, yaw rate, speed, frenet coordinates and sensor fusion data. A spline function (Cubic Spline interpolation in C++) is used as described in the Udacity Q&A video to convert given points in a fairly smooth function.

- **Prediction of other traffic participants**

The sensor fusion data and the simulator is used to generate the behavior prediction of what the other vehicles are likely to do. We change the Ego vehicle behavior on the basis of the Prediction of other car's behavior.

- **Path Planning Behavior**

Naming convention:

Ego car – in the autonomous driving systems or the Intelligent driving the term Ego car is used for the car with reference to which we are planning the motion. Or in simple words we can say the car for which we have developed the control software.

The algorithm for behavior planning is explained below -

1. The Ego car checks 30 m in front for the presence of any other vehicle.
2. If the Ego car senses another vehicle within 30 m in front and it is slow moving then Ego car tries to change the lane.
3. It checks the the left lane first, if it's free and contains no vehicle within 30 m in front and 10 m rear of the Ego car. The rear check is needed while changing lanes because collision can happen with the cars moving point to point parallel.

4. If left lane is not available then it checks to switch to right lane and check if it's free and contains no vehicle within 30 m in front and 10 m rear of the Ego car.
5. If there is no possibility for a lane change then it stays in the current lane without colliding with the car moving in front.

- **Trajectory Generation**

Refer main.cpp : *Lines 194 - 296*

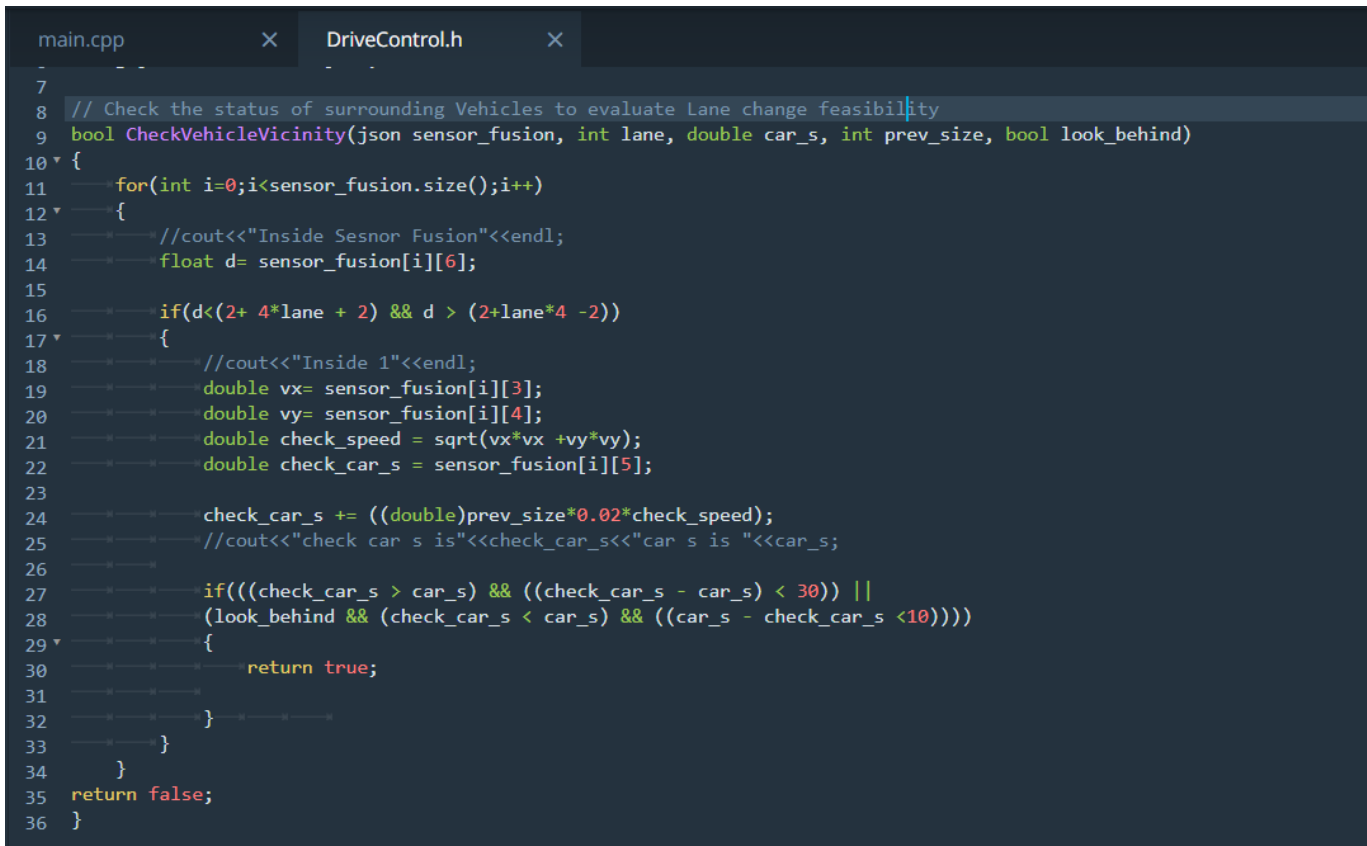
The Ego car calculates the trajectory based on its own speed, the speed of surrounding cars, current lane, intended lane and past points.

To make the trajectory smoother, we add last two points. If there are no previous points then we calculate the previous points from the current yaw and current car coordinates.

Also the 3 points are added in next 30, 60, and 90 meters to the trajectory. All these points are shifted to Ego car reference angle.

▪ Lane Change Logic

The lane change logic is implemented in DriveControl.h file.



```
main.cpp      x      DriveControl.h      x
7
8 // Check the status of surrounding Vehicles to evaluate Lane change feasibility
9 bool CheckVehicleVicinity(json sensor_fusion, int lane, double car_s, int prev_size, bool look_behind)
10 {
11     for(int i=0;i<sensor_fusion.size();i++)
12     {
13         //cout<<"Inside Sesnor Fusion"<<endl;
14         float d= sensor_fusion[i][6];
15
16         if(d<(2+ 4*lane + 2) && d > (2+lane*4 -2))
17         {
18             //cout<<"Inside 1"<<endl;
19             double vx= sensor_fusion[i][3];
20             double vy= sensor_fusion[i][4];
21             double check_speed = sqrt(vx*vx +vy*vy);
22             double check_car_s = sensor_fusion[i][5];
23
24             check_car_s += ((double)prev_size*0.02*check_speed);
25             //cout<<"check car s is"<<check_car_s<<"car s is " <<car_s;
26
27             if(((check_car_s > car_s) && ((check_car_s - car_s) < 30)) ||
28                (look_behind && (check_car_s < car_s) && ((car_s - check_car_s < 10))))
29             {
30                 return true;
31             }
32         }
33     }
34     return false;
35 }
36 }
```

Figure 9: Lane Change Logic

d. Rubric points:

Specification	Description	Result
Compilation		
The code compiles correctly	Code must compile without errors with cmake and make. Given that we've made CMakeLists.txt as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.	PASS The code compiled successfully. The spline.h and DriveControl.h are two additional files added in the folder.
Valid Trajectories		
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. Each incident case is also listed below in more detail.	PASS The EGO car is successfully driven for <u>7 miles</u> without any incident.
The car drives according to the speed limit.	The car doesn't drive faster than the speed limit. Also the car isn't driving much slower than speed limit unless obstructed by traffic.	PASS The EGO car follows the speed limit restriction and is always maintained speed < 50mph.
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 .	PASS The Acceleration and Jerk are followed as per specification.
Car does not have collisions.	The car must not come into contact with any of the other cars on the road.	PASS The Ego car does not have any collision.
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, and every other time the car stays inside one of the 3 lanes on the right hand side of the road.	PASS The Ego car always stays with in the lane. The lane number is provided by the code, the control is handled by the simulator.
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.	PASS In case of slowing moving vehicle in front, Ego car tries to change the lane and it does it successfully.
Reflection		
There is a reflection on how to generate paths.	The code model for generating paths is described in detail. This can be part of the README or a separate doc labeled "Model Documentation".	Subject to Evaluation Added in Section <u>5-c Reflection on path generation.</u>
For Project to Stand Out!		
	Create a path planner that performs optimized lane changing, this means that the car only changes into a lane that improves its forward progress.	Subject to Evaluation In the current implementation, lane change is done to improve Ego car's forward progress.