UDACITY

PROJECT

Path Planning

A part of the Self-Driving Car Engineer Program

PROJECT REVIEW

CODE REVIEW 5

NOTES

▸ src/main.cpp 4

▾ README.md 1

```
1  # CarND-Path-Planning-Project
2
3  This repository contains C++ code for implementation of Path Planner. This path of way
4
5
6  ## Background
7
8  A critical module in the working of a self driving car system is the path planning mod
9
10 Following parameters serve as an input to the path planning module:
11   1. Map of the environment with start and goal location. This is the global map havi
12   2. Local map of th environment. This map is a subset and a more detailed version of
13   3. Position of the other vehicles, pedestrians, animals, traffic lights, etc. in the
14   4. Current position of the car in the local map. This is derived by the localisatio
15
16 Information from all the inputs is then used to perfom following tasks:
17   1. Prediction - This involves predicting the behavior of car and other elements in t
18   2. Behavior planning - This involves plannning the possible states of the car. For e
19   3. Trajectory planning - This involves determining the trajectory of the car for a t
20
21
22 ## Working of Path Planning Module
23
24 Path planner assumes that the controller module of car is loss less and that it follo
25
```

```
  1. Creating smooth transition path from current location of the few meters ahead tou
  2. Providing discrete waypoints having information on the desired velocity of the ca
  3. Updation of the path in real time based on changes in the environment


## Project Goal

The goal of this project was to design a path planner that is able to create smooth, s


## Project Implementation

Simulation of a circular track was achieved in the [Udacity's self driving car simula

### 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

("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
("sensor_fusion") A 2d vector of cars and then that car's (car's unique ID, car's x po

The final implementation consisted of following major steps:
```

AWESOME

This file is full of resources that I find impressive. The reasoning behind this piece is well appreciated.

```

  ### 1. Creation of smooth trajectory ahead of car

  In this step, C++ [spline tool](http://kluge.in-chemnitz.de/opensource/spline/) was

  a. A point behind current car's location
  b. Current location of car
  c. Point ahead of car by 30m
  d. Point ahead of car by 60m
```

```
  e. Point ahead of car by 90m

  The speed limit for car was 50 MPH. Hence, the path planner followed a safe speed l

  C++ code for this task is implemented from line 375 to line 503 in main.cpp.


  ### 2. Prediction of behavior of other cars on the highway

  In this step, sensor fusion data passed by simulator was used to find cars ahead, i

  a. is_car_ahead - This flag was raised when the self driving car was approaching a
  b. is_car_right - This flag was raised when cars in the lane to the right of self d
  c. is_car_left - This flag was raised when cars in the lane to the left of self dri

  This information was prepared to be consumed by the behavior planner. Behavior of c

  C++ code for this task is implemented from line 283 to line 339 in main.cpp.

  ### 3. Determination of behavior of self driving car

  In this step, the car followed a less complex version of finite state machine havin

  a. Accelerate - Continue in current lane and accelerate reaching speed limit
  b. Decelerate - Slow down in current lane in order to avoid collision with car ahea
  c. Lane change Left - Change lane to left with current speed if not in leftmost lan
  d. Lane change RIght - Change lane to right with current speed if not in rightmost

  This information was prepared to be consumed by the trajectory planner to enhance t


## Project Output

Path planner was used to drive car with a maximum speed of 48 MPH along the highway.

![Car straight motion](https://raw.githubusercontent.com/sohonisaurabh/CarND-Path-Pla

![Car overtake left](https://raw.githubusercontent.com/sohonisaurabh/CarND-Path-Plann

![Car overtake right](https://raw.githubusercontent.com/sohonisaurabh/CarND-Path-Plan

![Car prepare pvertake left](https://raw.githubusercontent.com/sohonisaurabh/CarND-Pa

![Car prepare overtake right](https://raw.githubusercontent.com/sohonisaurabh/CarND-P

The was able to drive for more than 4.32 miles to meet the rubric specification of th

Detailed insight into features of the simulator and implementation is demonstrated in


## Steps for building the project

### Dependencies

* cmake >= 3.5
 * All OSes: [click here for installation instructions](https://cmake.org/install/)
 * Linux and Mac OS, you can also skip to installation of uWebSockets as it installs

* make >= 4.1(mac, Linux), 3.81(Windows)
  * Linux: make is installed by default on most Linux distros
  * Mac: [install Xcode command line tools to get make](https://developer.apple.com/x
```

```
  * Windows: [Click here for installation instructions](http://gnuwin32.sourceforge.n
  * Linux and Mac OS, you can also skip to installation of uWebSockets as it installs

* gcc/g++ >= 5.4
  * Linux: gcc / g++ is installed by default on most Linux distros
  * Mac: same deal as make - [install Xcode command line tools]((https://developer.ap
  * Windows: recommend using [MinGW](http://www.mingw.org/)
  * Linux and Mac OS, you can also skip to installation of uWebSockets as it installs

* [uWebSockets](https://github.com/uWebSockets/uWebSockets)
  * Run either `install-mac.sh` or `install-ubuntu.sh`. This will install cmake, make
  * If you install from source, checkout to commit `e94b6e1`, i.e.
    ```
    git clone https://github.com/uWebSockets/uWebSockets
    cd uWebSockets
    git checkout e94b6e1
    ```
    Some function signatures have changed in v0.14.x.

* Fortran Compiler
  * Mac: `brew install gcc` (might not be required)
  * Linux: `sudo apt-get install gfortran`. Additionally you have also have to instal

* [Ipopt](https://projects.coin-or.org/Ipopt)
  * If challenges to installation are encountered (install script fails).  Please rev
  * Mac: `brew install ipopt`
  * Linux
    * You will need a version of Ipopt 3.12.1 or higher. The version available throug
    * Then call `install_ipopt.sh` with the source directory as the first argument, e
  * Windows: If you can use the Linux subsystem and follow the Linux instructions or

* [CppAD](https://www.coin-or.org/CppAD/)
  * Mac: `brew install cppad`
  * Linux `sudo apt-get install cppad` or equivalent.
  * Windows: If you can use the Linux subsystem and follow the Linux instructions or

* Simulator. You can download these from the [Udacity simulator releases tab](https:/

### Running the project in Ubuntu

  1. Check the dependencies section for installation of gcc, g++, cmake, make, uWebso

  2. Manually build the project and run using:
    a. mkdir build && cd build
    b. cmake ..
    c. make
    d. ./path-planning

  3. Run the Udacity simulator and check the results
```

▸ src/Eigen-3.3/unsupported/Eigen/CXX11/src/Tensor/README.md

▸ src/Eigen-3.3/test/bug1213_main.cpp

▸ src/Eigen-3.3/demos/mandelbrot/README

- src/Eigen-3.3/bench/tensors/README
- src/Eigen-3.3/bench/btl/libs/ublas/main.cpp
- src/Eigen-3.3/bench/btl/libs/tvmet/main.cpp
- src/Eigen-3.3/bench/btl/libs/mtl4/main.cpp
- src/Eigen-3.3/bench/btl/libs/gmm/main.cpp
- src/Eigen-3.3/bench/btl/libs/blaze/main.cpp
- src/Eigen-3.3/bench/btl/libs/STL/main.cpp
- src/Eigen-3.3/bench/btl/libs/BLAS/main.cpp
- src/Eigen-3.3/README.md
- src/Eigen-3.3/demos/opengl/README
- src/Eigen-3.3/demos/mix_eigen_and_c/README
- src/Eigen-3.3/bench/btl/README

RETURN TO PATH

Student FAQ