Path Planning

ECE 493 T21

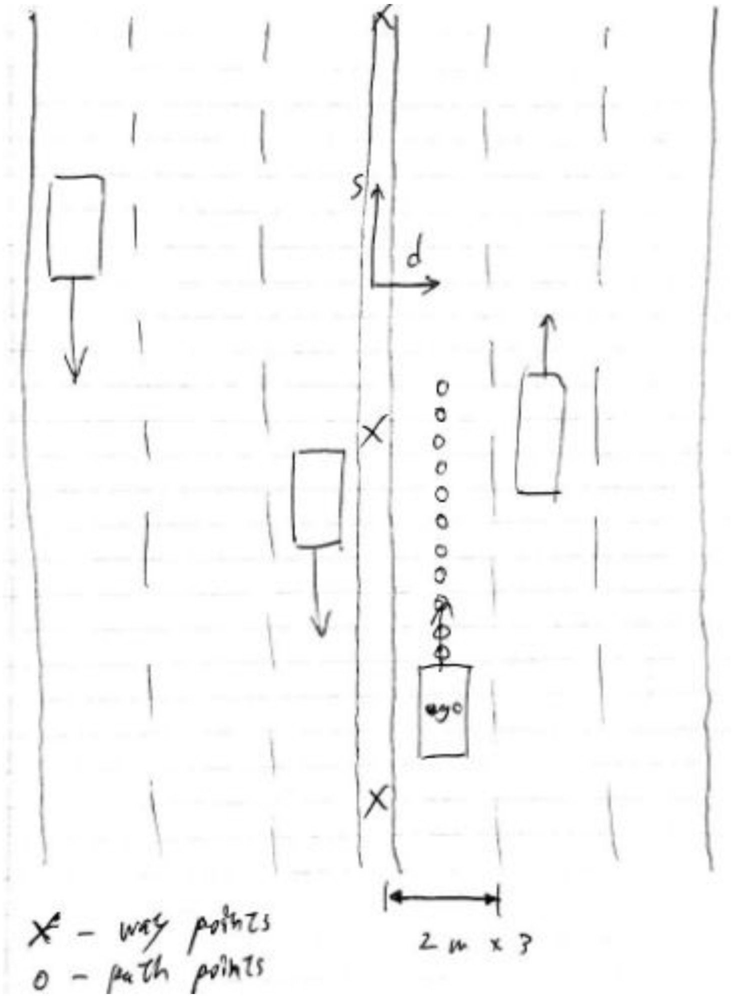
Winter 2020

Overview

- 6-lane track, 3 lanes each direction, 6946m long
- 50 mph speed limit
- Total acceleration limit: 10 m/s^2
- Jerk limit: 10 m/s^3
- Be able to follow a lane smoothly
- Change lane smoothly when vehicle in front is too slow
- Avoid collisions at all cost
- As close to speed limit as possible
- Assignment repo: https://github.com/udacity/CarND-Path-Planning-Project

Setup

- Follow the instructions in the project repo
- Need to install uWebSockets before you can build and run the project. First, `sudo chmod u+x install-ubuntu.sh`, then `./install-ubuntu.sh`. Do the mac version if you have mac. For windows, try install from source (see instruction in repo)
- Go to the directory where you downloaded the simulator, do `sudo chmod u+x <simulator-name>` as well
- Replace the `main.cpp` file in `src` with the one posted on Learn
- Put `spline.h` into the `src` folder
- Build and run the project
- Start simulator



Road Layout

- s and d: Frenet coordinate
- Each lane is 2m wide
- Way points: points along road center, given to you, use them as reference points to generate path points
- Path points: equally spaced out points which the ego vehicle will follow, 0.02s each

Rubric

Trajectory generation (2.5 points)

- <u>Option 1 (2.5 points)</u> : watch Udacity’s Q&A video, answer the following questions:
    - 1) how is lane following achieved? (0.5 points)
    - 2) how to use spline to generate a smooth trajectory? (1 point)
    - 3) how to avoid collision with the car in front? (0.5 points)
    - 4) how to avoid cold start? (0.5 points)
    - Can elaborate on 2), but be <u>very concise</u> (2 sentences max each) for 1) 3) 4)
- <u>Option 2 (3.5 points)</u>: implement jerk minimized trajectory <u>successfully</u> and explain your approach

Rubric

Behaviour Planning (2.5 points)

- During **2 miles** of driving, achieve the following:
- 1) Perform lane shift when front vehicle is too slow, done at least once (1 point)
- 2) No collisions (0.5 points)
- 3) (Count(Exceed(speed lim || acc lim || jerk lim)) <= 1) && (complete 2 miles within 3 min 30 sec) (0.5 points) automatically lose this 0.5 if you exceed any limits for more than 2 sec continuously.
- 4) Explain your approach concisely (0.5 points)
- Optional challenge (1 point): never exceed any limits
- Optional challenge (1 point): optimize lane shift so that you can complete 2 miles within 2 min 45 sec (without violating the aforementioned limits of course…)
- Note: bonus marks available only if you achieve 1) 2) 3) 4) successfully
- Highly recommended: to detect potential bugs, run your car for at least 5 miles when you test your code.

Helper Functions

- Functions in helpers.h are helper functions provided to you
- Please don’t explain them in your write-up
- Just understand how to use them is enough

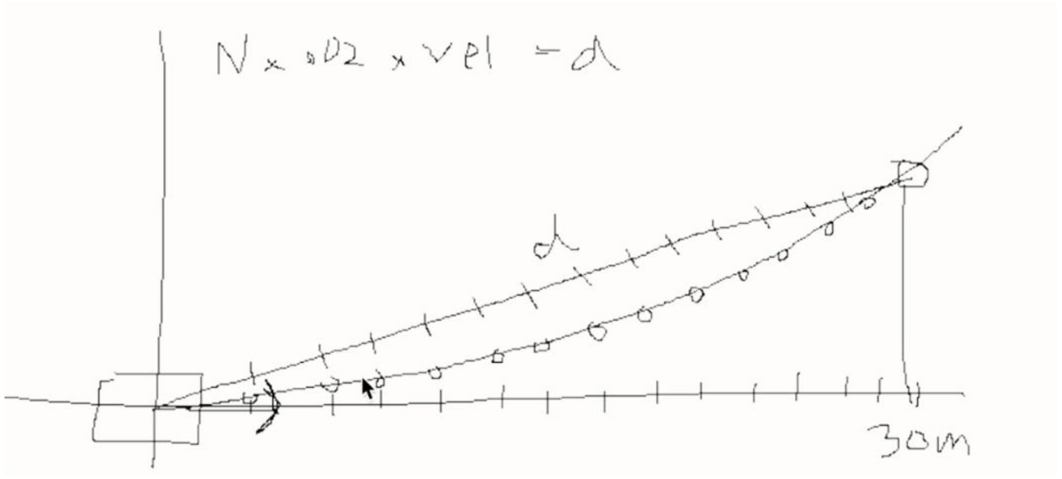
Starter Code

- The code for Task 1 (option 1) is provided to you in “main.cpp” on Learn
- Enables the ego vehicle to: 1) follow a lane smoothly and 2) avoid collision with vehicle in front in the same lane
- But feel free to modify it as you see proper

Task 1: Trajectory Generation (Option 1)

- Use spline to generate path
- Assignment Q&A video:
  https://www.youtube.com/watch?time_continue=369&v=7sI3VHFPP0w&feature=emb_logo
- 12:00-19:00 Lane following
- 19:00-40:00 Generate a smooth path for the ego to follow (i.e. a set of path points)
- 40:00-48:30 Collision avoidance
- 48:30-53:00 Avoid cold start
- 53:00-58:00 How lane change happens
- Note: code from the help video will be provided

How does Spline Work?



- Fit a spline to several sparse waypoints
- Distance horizon: 30m ahead of the starting position of the car along its original heading
- Use original heading as x direction here
- Obtain the y position of the end point (the big circle) corresponding to the distance horizon using spline function
- Compute d using trigonometry
- Compute N using d (m), velocity (m/sec), and 0.02 sec
- Divide 30m along x direction into N parts, find all corresponding y values on the spline
- Now you have the path points to follow in order to traverse the spline
- Note: local coordinate system here

Task 1: Trajectory Generation (Option 2)

- Use the **Jerk Minimized Trajectory** (JMT) approach
- See the third link in the last slide for details
- **1 bonus** mark if you implement JMT and explain your approach clearly in the writeup

Task 2: Behaviour Planning

- Finite state machine: design several functions to help the ego decide when to switch lane, and which lane to switch to etc.
- Cost functions: evaluate the cost of the possible states, say cost of stay in the same lane, cost of switching lane etc.
- Make decisions and use the trajectory planner to execute the decisions
- Lots of freedom for this task, you can be very creative here

Tips

- When running the simulator, choose lower resolution to make sure it runs fast enough
- Need to install uWebSockets before you can build and run the project. First, `sudo chmod u+x install-ubuntu.sh`, then `./install-ubuntu.sh`. Do the mac version if you have mac. For windows, try install from source (see instruction in repo)

Helpful Blog Posts

https://towardsdatascience.com/teaching-cars-to-drive-highway-path-planning-109c49f9f86c

https://medium.com/@mithi/reflections-on-designing-a-virtual-highway-path-planner-part-1-3-937259164650

https://medium.com/@mithi/reflections-on-designing-a-virtual-highway-path-planner-part-2-3-392bc6cf11e7

https://medium.com/@mithi/reflections-on-designing-a-virtual-highway-path-planner-part-3-3-a36bf629d239