UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

3D Motion Planning

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Greetings Learner
Congratulations on passing your project! 🎉 Your project was very impressive and shows your outstanding efforts! We very much appreciate the time and interest you have put into this project for passing it. This application is very intelligently developed and you should be proud that you have been able to do so! I direly hope that you keep up the spirit and continue giving this amazing performance! Wishing you all the best.

Further resources:

- A Real-Time 3D Path Planning Solution
- Advanced Robotic Systems 3D Dynamic Motion Planning

Writeup

The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.

Good job student! You have provided an amazing write up explaining each rubric point in a very detailed manner with images. 👏🏻

Explain the Starter Code

The goal here is to understand the starter code. We've provided you with a functional yet super basic path planning implementation and in this step, your task is to explain how it works! Have a look at the code, particularly in the `plan_path()` method and functions provided in `planning_utils.py` and describe what's going on there. This need not be a lengthy essay, just a concise description of the functionality of the starter code.

Bravo! You have properly explained the starter code which shows you have properly understood it as well! 💯

Implementing Your Path Planning Algorithm

Here you should read the first line of the csv file, extract lat0 and lon0 as floating point values and use the `self.set_home_position()` method to set global home.

Well done! You have successfully read the first line of the csv file and passed home position to the constructor of motion planning and set it in planning phase.

```
    def get_lon_lat(file):
        f=open(file, newline='')
        reader = csv.reader(f)
        data = next(reader)
        f.close()
        lat0 = float(data[0].split()[1])
        lon0 = float(data[1].split()[1])

        return [lon0, lat0]

    if __name__ == "__main__":
        # TODO: read lat0, lon0 from colliders into floating point values
        obstacles_file = 'colliders.csv'
        drone_home_g = get_lon_lat(obstacles_file)
        drone_home_g.append(0)
```

Here as long as you successfully determine your local position relative to global home you'll be all set.

This specification was met successfully too! Good job!

```
        # TODO: retrieve current global position
        global_position = self.global_position

        # TODO: convert to current local position using global_to_local()
        global_home = self.global_home
        grid_start_l = global_to_local(global_position, global_home)
```

This is another step in adding flexibility to the start location. As long as it works you're good to go!

Simulation works well! 👍

```
    def core_plan_path(motion_p, global_home, grid_start_l, grid_goal, target_altitud
e):
        # TODO: convert start position to current position rather than map center
        grid_start = motion_p.get_grid_coord(grid_start_l, target_altitude)
```

This step is to add flexibility to the desired goal location. Should be able to choose any (lat, lon) within the map and have it rendered to a goal location on the grid.

Nice work! An option to define any goal location was added.

```
    def core_plan_path(motion_p, global_home, grid_start_l, grid_goal, target_altitud
e):
        ....

        # TODO: adapt to set goal as latitude / longitude position and convert
        grid_goal_l = global_to_local(grid_goal, global_home)
        grid_goal = motion_p.get_grid_coord(grid_goal_l, target_altitude)
```

Minimal requirement here is to modify the code in `planning_utils()` to update the A* implementation to

Minimal requirement here is to modify the code in `planning_utils()` to update the A* implementation to include diagonal motions on the grid that have a cost of sqrt(2), but more creative solutions are welcome. In your writeup, explain the code you used to accomplish this step.

Excellent job here! You have used a great technique for this part and explained it very well. Hats off to you! 👏🏼

For this step you can use a collinearity test or ray tracing method like Bresenham. The idea is simply to prune your path of unnecessary waypoints. In your writeup, explain the code you used to accomplish this step.

You have implemented this correctly and the code that was used to implement this was mentioned as well.

```
def prune_path(path)
```

Executing the flight

At the moment there is some mismatch between the colliders map and actual buildings in the scene. To ensure success build in a 5+ m safety margin around obstacles. Try some different goal locations. Also try starting from a different point in the city. Your reviewer will also try some random locations so be sure to test your solution! There is no firm constraint or requirement on how accurately you land exactly on the goal location. Just so long as your planner functions as expected.

Great work! This specification met the requirements as well. 💯

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START