UDACITY

PROJECT

# 3D Motion Planning

A part of the Flying Car Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

## Meets Specifications

Brilliant Student,

Your submission is exceptional. I am a big fan of your code. Please keep the momentum going and you will definitely enjoy the other projects.

### 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.** |

Awesome. The write-up explains how each rubric item was addressed and clearly states where each state was handled in the code.

## 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.**

Nice description of how your planner works. The description references the various files submitted, and how these files address the various parts of the planner.

## 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.**

Awesome. `np.loadtxt` is used to read to the CSV file and self.set_home_position() method is used to set the set the global home.

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

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

✓

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.

✓ 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.

Awesome. The functions a_star(), and valid_actions() and the class `Action()` have been adapted to support diagonal motion(action).

✓ 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.

Well done using collinearity test to test and remove collinear points thereby pruning the path of unnecessary waypoints. The explanation of the code is clear enough for anyone to understand what the function prune_path() in planning_utils.py is doing.

## 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.

Awesome. The vehicle flies from the start to the goal.