

[< Back to Flying Car Nanodegree](#)


3D Motion Planning

审阅

代码审阅 3

HISTORY

Meets Specifications

Congratulations 🎉 on successfully completing this project. You certainly did a good job by implementing path planning algorithms well in a 3D environment. 

Check these extra readings on :

- [Drone programming](#) basics by Intel Aero team
- An [Evolution based](#) path planning algorithm, is a good paper to read in free time.

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.

✓ Your report has all the components stated in the rubric.

Completeness: You have covered all the required rubric points thoroughly

Appearance: Your write-up is very well written.

Explain the Starter Code

The goal here is to understand the starter code. We've provided you with a functional yet super basic

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.

✓ `plan_path()` method and function are explained well. *Good Job*

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.

✓ Your `self.set_home_position()` is well fitted with the csv file data. 🍷

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

✓ Local position is determined successfully along with the global home.

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

✓ Necessary modification done and works well.

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.

✓ Flexibility is observed with the desired goal location too and satisfies the rubric criteria. 👍

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.

✓ Implementation for A* is done right along with setting delta and cost parameter values.

Additionally, checkout these links for an interesting visualization for A* algorithm.

- <https://qiao.github.io/PathFinding.js/visual/>
- <https://brilliant.org/wiki/a-star-search/>

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.

✓ Again nice work by implementing collinearity. You can also use bresenham planning algorithm. It's quite efficient.

- Read this [paper](#) for more insights on `bresenham` Algorithm

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.

✓ There seems to be no issue that was in the case of mismatch, and your solution works well on different random locations too. Well done, the drone performs well and executes a smooth flight. 100

 下载项目

3 代码审阅评注



返回 PATH

给这次审阅打分