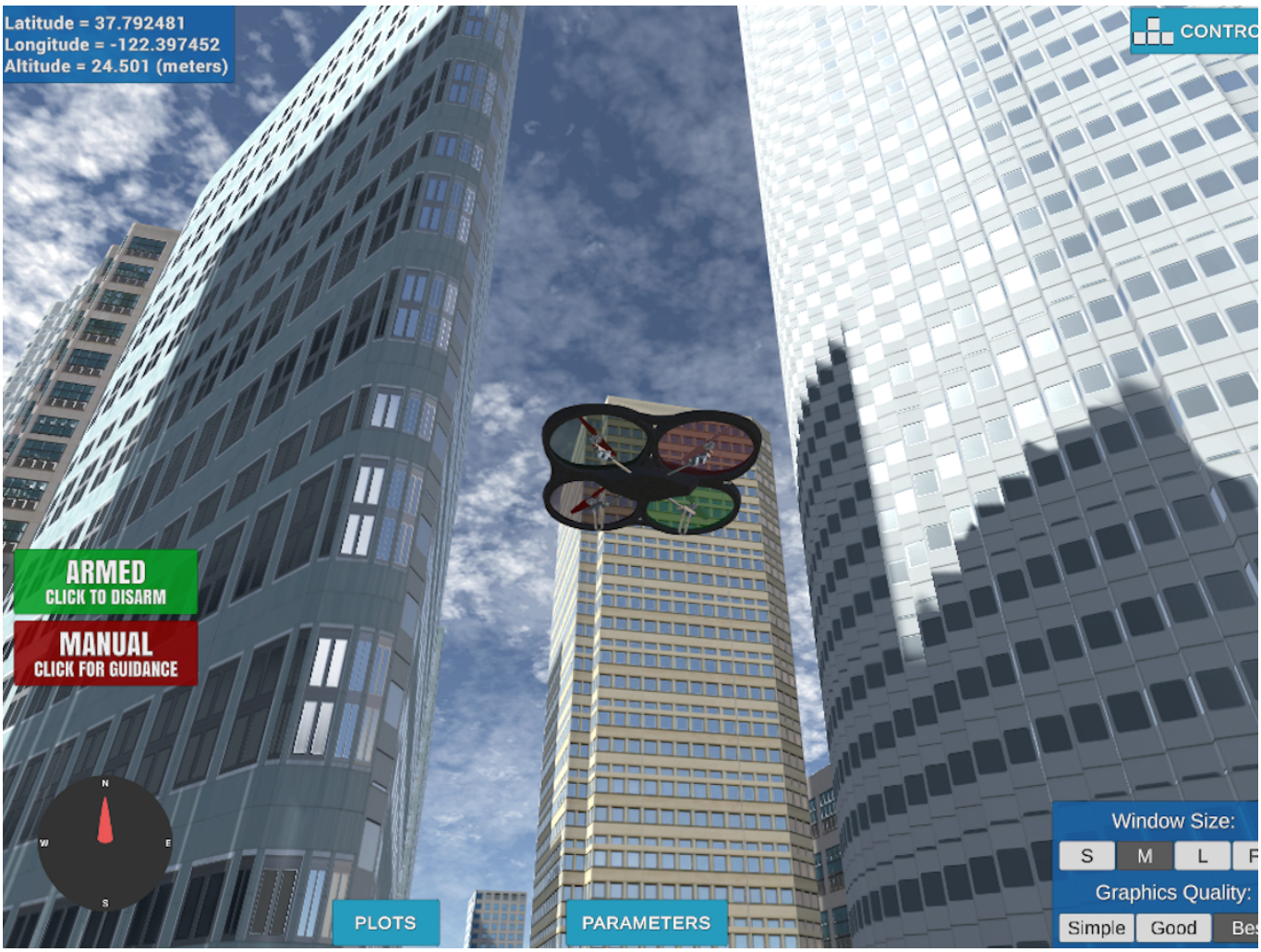
Project: 3D Motion Planning



Required Steps for a Passing Submission:

1. Load the 2.5D map in the colliders.csv file describing the environment.
2. Discretize the environment into a grid or graph representation.
3. Define the start and goal locations.
4. Perform a search using A* or other search algorithm.
5. Use a collinearity test or ray tracing method (like Bresenham) to remove unnecessary waypoints.
6. Return waypoints in local ECEF coordinates (format for `self.all_waypoints` is [N, E, altitude, heading], where the drone's start location corresponds to [0, 0, 0, 0].
7. Write it up.
8. Congratulations! Your Done!

Rubric (https://review.udacity.com/#!/rubrics/1534/view) Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf.

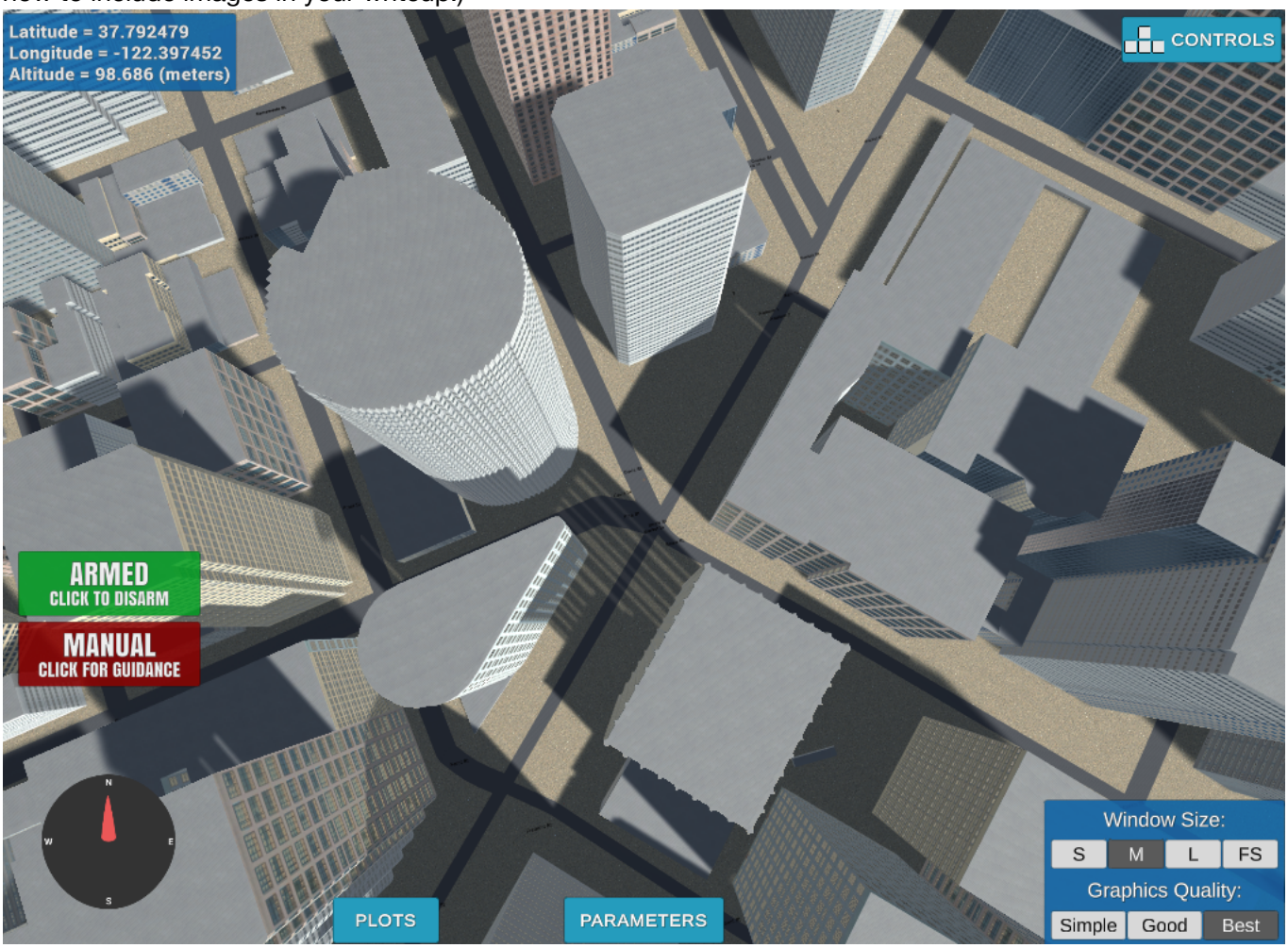
You're reading it! Below I describe how I addressed each rubric point and where in my code each point is handled.

Explain the Starter Code

1. Explain the functionality of what's provided in `motion_planning.py` and `planning_utils.py`

These scripts contain a basic planning implementation that includes...

And here's a lovely image of my results (ok this image has nothing to do with it, but it's a nice example of how to include images in your writeup!)



Here's	A	Snappy	Table
1	`highlight`	**bold**	7.41
2	a	b	c
3	*italic*	text	403
4	2	3	abcd

Implementing Your Path Planning Algorithm

1. Set your global home position

Here students 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. Explain briefly how you accomplished this in your code.

And here is a lovely picture of our downtown San Francisco environment from above!

2. Set your current local position

The starter code assumed the drone takes off from map center, but the drone need to be able to takeoff from anywhere.

I retreived the drone's current position in geodetic coordinates from `self.global_position`, and the global home position set from last step from `self.global_home`, then used the utility function `global_to_local()` to convert the current global position to local position.

I did this in line 132 (motion_planning.py#L132) of `motion_planning.py`.

3. Set grid start position from local position

The starter code hardcoded the map center as the start point for planning. To further enhance the flexibility to the start location, I changed this to be the current local position in line 142 to 144 (motion_planning.py#L142-L144) of `motion_planning.py`.

4. Set grid goal position from geodetic coords

The starter code hardcoded the goal position as some location 10 m north and 10 m east of map center. To add flexibility to the desired goal location, I modified the code in line 148 to 151 (motion_planning.py#L148-L151) of `motion_planning.py` to accept arbitrary goal postion on the grid given any geodetic coordinates. By defaults I set the coordinates to (longitude, latitude, altitude).

To assign the goal position, use command line arguments `goal_lon` for longitude, `goal_lat` for latitude, and `goal_alt` for altitude. For example:

```
python motion_planning.py --goal_lat 37.792945 --goal_lon -122.397513 --go
al_alt 26
```

5. Modify A *to include diagonal motion (or replace A* altogether)

I updated the A* implementation to include diagnoal motions on the grid that have a cost of sqrt(2). With diagnoal motions included, the jerky movement disappeared and the trajectories planned for the same goal changed. I did this in lines 58 to 61 (planning_utils.py#L58-L61), 91 to 98 (planning_utils.py#L91-L98) of `planning_utils.py`.

Here's a comparison between paths with and without diagonal motion:

6. Cull waypoints

To prune the path of unnecessary waypoints, I implemented collinearity test in lines 167 to 188 (planning_utils.py#L167-L188) of `planning_utils.py` and applied it to the path obtained from A* search.

Here's a comparison between paths before and after removing the unnecessary waypoints:

Execute the flight

1. Does it work?

It works!

Double check that you've met specifications for each of the rubric (https://review.udacity.com/#!/rubrics/1534/view) points.

Extra Challenges: Real World Planning

For an extra challenge, consider implementing some of the techniques described in the "Real World Planning" lesson. You could try implementing a vehicle model to take dynamic constraints into account, or