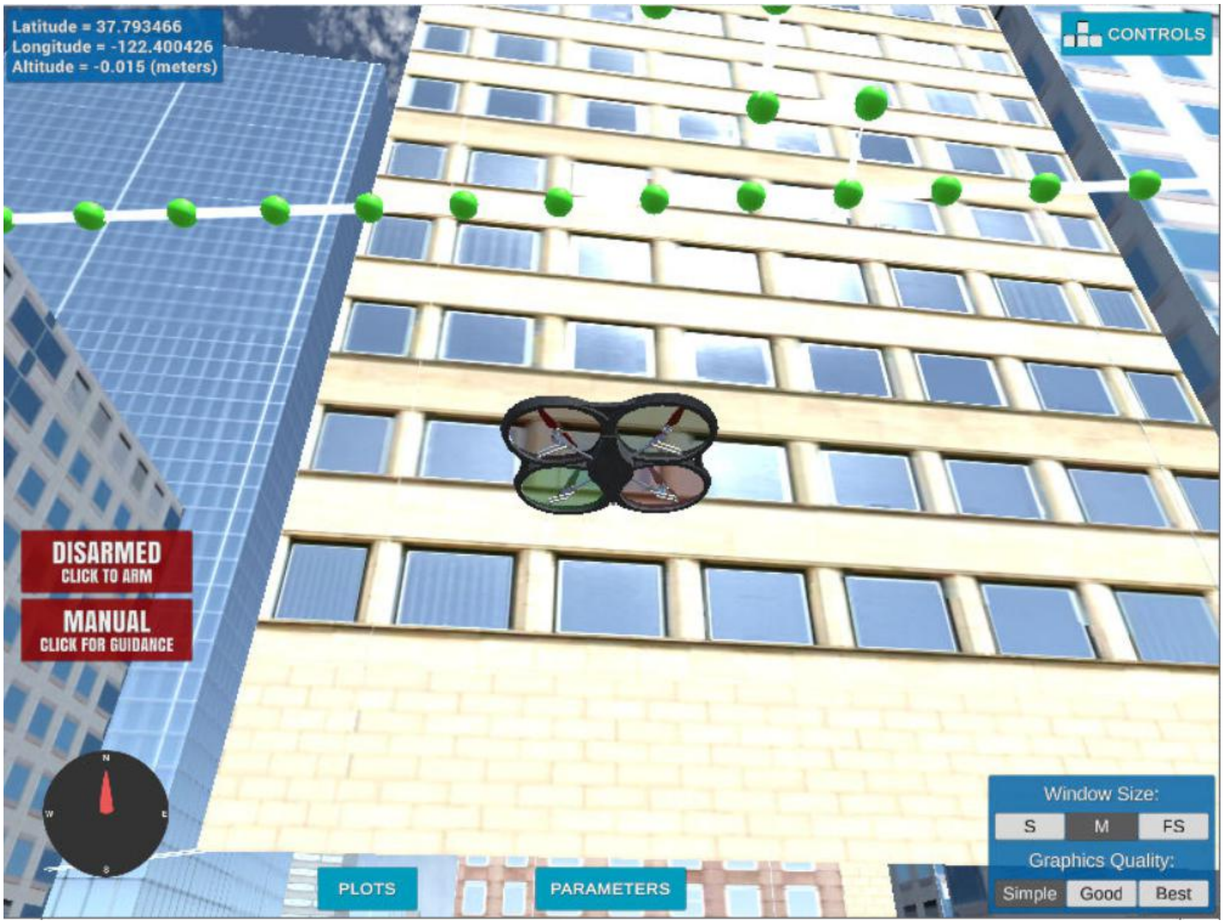
July 10 , 2 0 1 8 B Y S W A R O O P K S

3D Motion Planning



The second project consists of planning and executing a trajectory of a drone in urban environment. This project was built on top of event based strategy utilized on the first project . The software written for planning will communicate with Udacity FCND simulator using Udacidrone API .

Prerequisites for project

To run this project , We have following two options ,

Option 1: Udacity Cloud VM

For this project, you will have the option to complete all of your work in a cloud-based GPU supported virtual machine (VM) right in the Udacity classroom! Graphics rendering in the VM seems to work best for internet connections faster than 10Mbps, so if you're on a good connection we highly recommend checking it out! You can test the speed of your connection by going to fast.com. Then simply follow along with these instructions for getting started and you'll be up and running in no time with your new VM.

Option 2: Work on your local machine

If you would prefer to complete the project on your local machine, then you can skip ahead to the instructions for local setup.

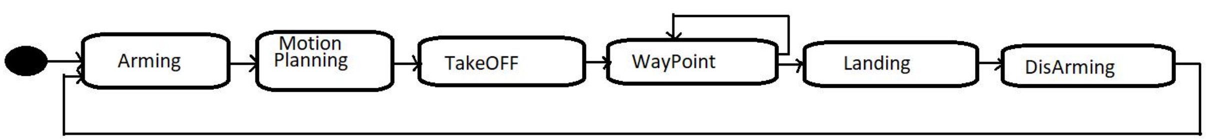
I have choose option 1 to complete project

Project description

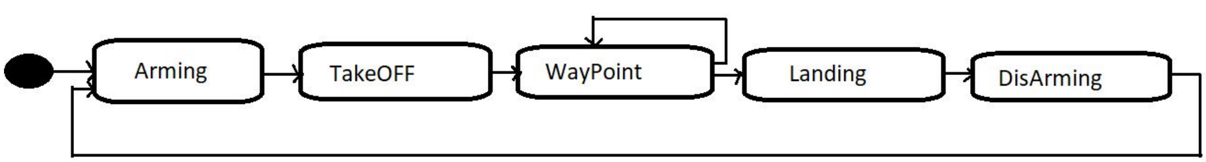
The following are the main code used on the project:

- **motion_planning.py:** This has many basic implementation provided by Udacity . These are some of modification done on basic project :
    - **Set Home position:** The global home location is read from the colliders.csv file
    - **Determine your current local position relative to global home:** global_position is set to [self._longitude, self._latitude, self._altitude] global_to_local(global_position, self.global_home) determines the current local position.
    - **Change start point for planning to current local position:** north_offset and east_offset are the vector to (0,0) of the grid. This vector is subtracted from the the current local position in the map reference frame to get the start position in the grid reference frame.
    - **Set goal location:** The goal location is set from command line arguments (--targ_lat, --targ_lon, --targ_alt).
    - **Purge redundant waypoints:** The calculated path is pruned with a collinearity function to eliminate unnecessary waypoints.

The state machine implemented on motion_planning.py is as shown in diagram:



- **planning_utils.py:** It was also provided by Udacity with basic funtions create_grid, valid_actions, a_star heuristic cost funtion . Only additional funtionality added was collinearity_purge which is used to delete redundant waypoints .
- **backyard_flyer_solution.py:** This is also provided by udacity with all basic funtionalities like ,States, local_position_callback, velocity_callback,arming_transistion, takeoff_transition, waypoint_transition, landing_transition, disarming_transition, manual_transition. The state transition diagram as shown in figure below just with out planning state,



The diagonals movements were implemented by adding them to the Action enum , and assign them a cost of sqrt(2) for A star algorithm. Also , we are making sure new grid doesnt contain obstrucle while considering for any diagonal moment .

Run the Project

This involves following steps :
Run simulator require following commands:

```
cd /home/workspace/linux-64bit/

./FCND-Sim_Linux_64-bit
```

Run project code requires following commands :

In a new terminal, run the following commands to clone the **project repository** into your `/home/workspace` directory:

```
cd /home/workspace

git clone https://github.com/udacity/FCND-Motion-Planning
```

Next, activate the `fcnd` conda environment by typing `source activate fcnd` and then `python` at the prompt. You should now be running Python 3.6 in the `fcnd` conda environment.

```
source activate fcnd

python motion_planning.py --targ_lon -122.400453 --targ_lat 37.793480 --targ_alt 0
```

Conclusion & Pending tasks

Over all , this was good project which pretty much give idea about 3D motion planning for drone . I would like to take extra challenge, consider implementing some of the techniques described in the "Real World Planning" lesson.Also , implementing a vehicle model to take dynamic constraints into account, or implement a replanning method to invoke if you get off course or encounter unexpected obstacles.