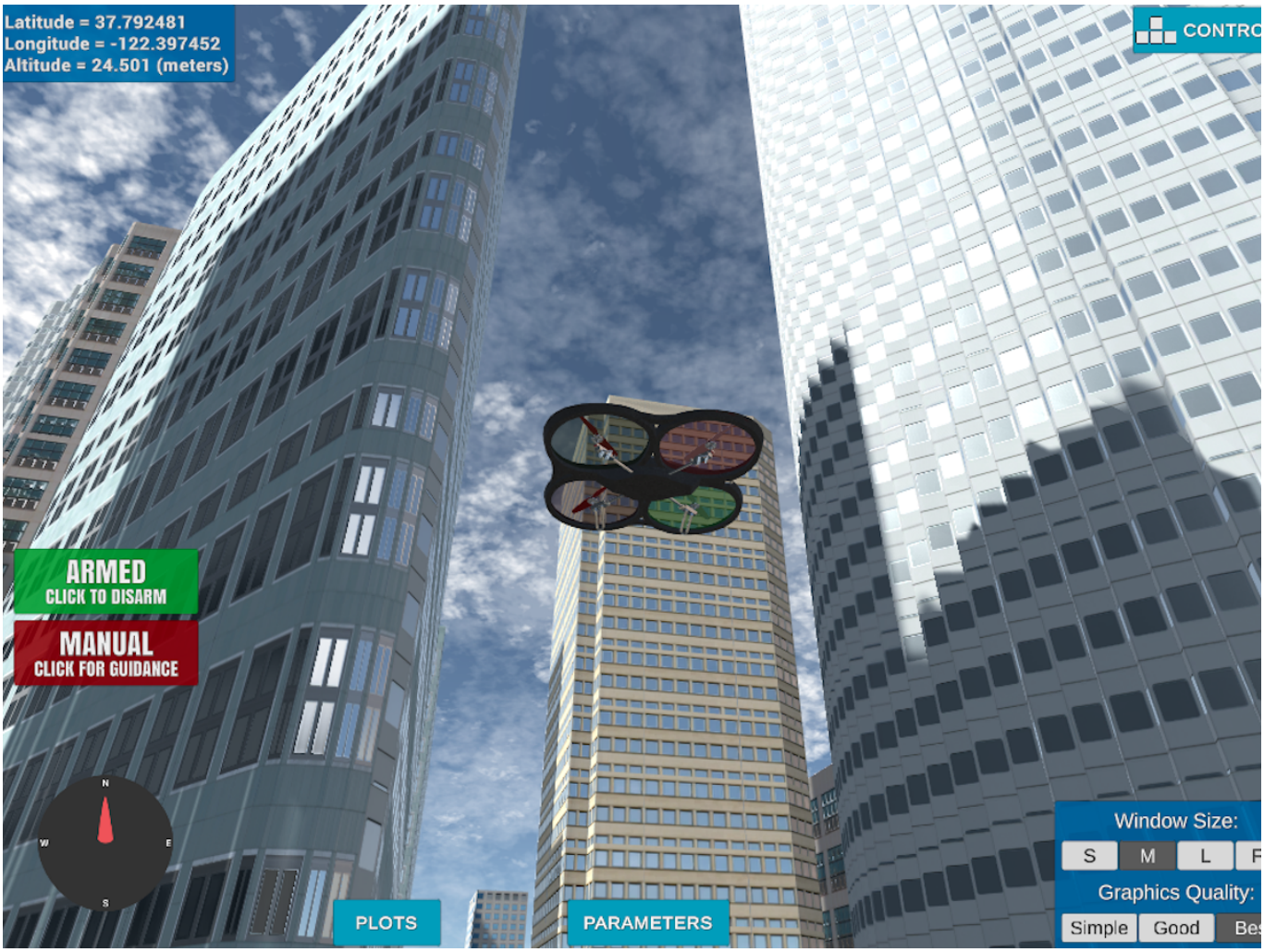
FCND - 3D Motion Planning



SETUP AND RUNNING

- to run : >python motion_planning.py [options]
- command line options:
- **--mode [voronoi | grid | spiral]**
  - select planning mode
  - default is grid if --mode is not specified
- **--plot**
  - if specified, a plot of the grid and path is generated
  - note: generating the plot takes a long time and can result in a network timeout. if that occurs just use --plot to get the plot then rerun without it to see the flight. The plot is saved to disk in the current directory.
- programmatic options that are set in the main function and can be modified
  - **drone.set_goal(lat/lon)** to change the goal location
  - **drone.waypoint_tolerance** : M meters to control when to perform a waypoint transition
  - **drone.loiter_seconds** : a value > 0 to have the drone loiter at waypoints 5 or more seconds seems to be needed to compensate for the overshoot

FILE ORGANIZATION

- backyard_flyer_silution.py : Original backyard flyer implementation
- motion_planning-origina.py Copy of original motion_planning.py starter implementation
- **motion_planning.py** : Contains the updated main code for the drone including grid,voronoi and spiral planners
- planning_utils.py : As provided with some small modifications that are marked with comments
- **project_utils.py** : New file with various utilities in support of modified motion_planning.py

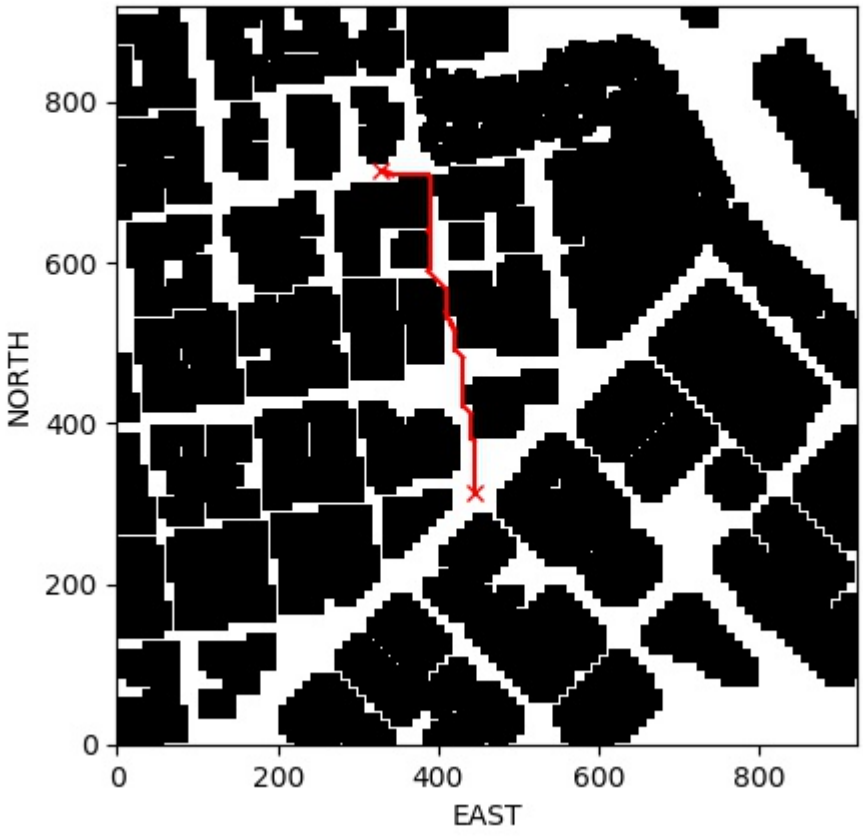
RUBRIC NOTES

- global and local positions are set using lat/lon from colliders.csv
- local start position is set to current position in each of the planner modes
- the goal is set in the **main** function (see above)
- The Action class in planning_utils was modified to produce diagonal moves
- The grid and voronoi planners prune the path. see prune_path() in project_utils.py
- If there is no path between the start and goal, the program will terminate

GRID SEARCH APPROACH

The grid search mode is build on the provided example, just adding the required features to have a variable start and goal position. It uses the standard a_star grid search function. This mode takes a while to generate the path so some progress information is printed while the a_star algorithm runs.

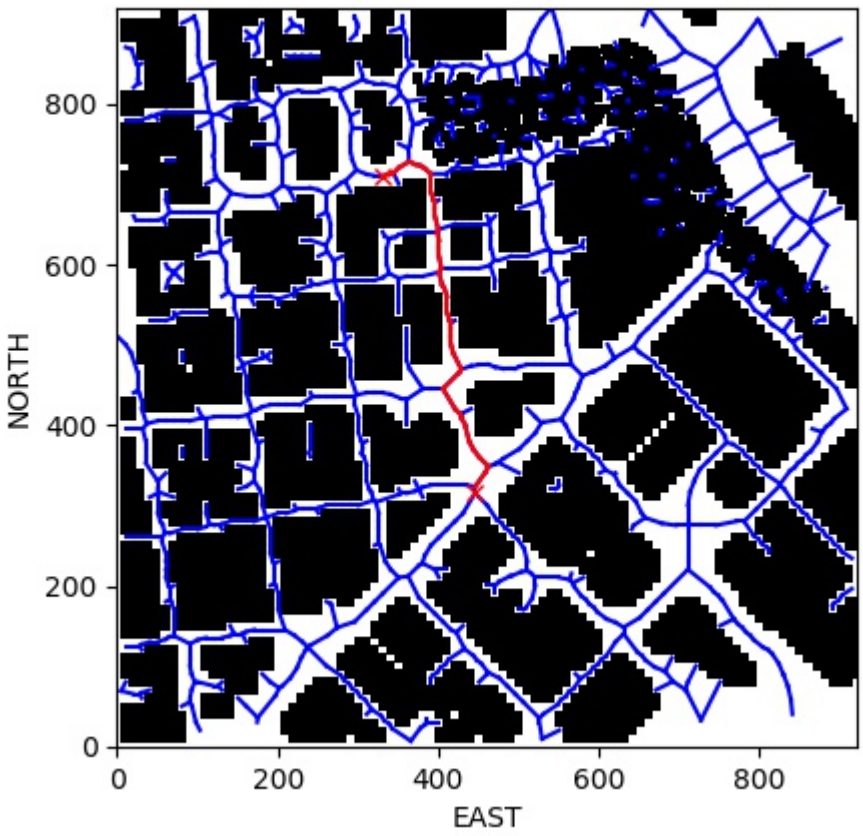
Example Grid Search



VORONOI GRAPH APPROACH

The voronoi mode uses a Voronoi graph and modified a_star graph search to find the path. It runs much quicker than the grid search, so it does not print a_star progress information.

Example Voronoi Search



SPIRAL APPROACH

The spiral plan computes coordinates for a circle in increments of PI/8 then iterates the circle a specfied number of times, ascending with a step in Z of 0.2 meters per iteration. Then it descends back along the same path and then returns to the initial start position. This implementation does not attempt to avoid obstacles so the drone must be position in an area clear of obstructions. The computation is parameterized int he plan_spiral() function so that the radius, number of waypoints per circle, Z step and number of circles can be modified.

DIAGNOSTICS

This is an example of the diagnostic printout when the simulation runs:

```
Logs/TLog.txt
mode = GRID_SEARCH                                          (search mode)
Logs/NavLog.txt
starting connection
arming transition
Searching for a path ...
global home [-122.3974533   37.7924804    0.       ]         (compute
```

```
positions)
global_position [-122.3962654   37.794106     0.219    ]      "
local position [180.66429138 105.60977936  -0.22151029]       "
current_position [181.03458043 103.42442738  -0.219     ]     "
North offset = -316, east offset = -445                       (grid offsets)
Local Start and Goal:  (497, 548) (716, 329)                  (start and
goal position)
Printing Astar Progress (every 1000 iterations)               (a_star
progress)
0 0 (497, 548)                                                (iteration,
cost, position)
1000 334 (573, 53
...
Found a path.
Path 380:407.580736 : Pruned Path 30                          (path info)
Sending waypoints to simulator ...
ET: 29.746826                                                 (elapsed time
to plan)
takeoff transition [0. 0. 5.]
```

ENHANCEMENTS

Deadband

A parameter is added to the MotionPlanner object called self.waypoint_tolerance. This sets the raidius for detecting arrival at a waypoint.

Heading Commands

- The Voronoi mode has heading control implemented. It was not added to the grid search mode to provide a comparison of the results. The drone should point towards the next waypoint while flying.

Loiter

- A parameter is added to the MotionPlanner object called self.loiter. If this parameter is greater than 0, the drone will loiter at each waypoint for the specified number of seconds. A value of 5 or greater gives best results. Less than 5 and the drone just stutters and then continues. This makes a flight a lot longer so don't enable it unless you need it.

Plotting

At the end of each run the TLog.txt files is read and the data is stored in a pickle file (log.pkl) for later plotting, without having to connect to the drone.

TESTING

- The program was tested on both Windows and Linux. There were more frequent network timeouts and aborts when running on Windows. No problems were found running on Linux (once I fixed my own bugs).

Observations

- The drone flies a bit too fast to adhere to the planned path. Sharp turns exhibited a lot of overshoot. The default fly-to-the-waypoint behavior is not ideal for generating a smooth flight. The overshoot can cause contact with obstacles. One fix would have the controller slow the drone down on approach to a waypoint and then accelerate once it passes the waypoint. The decel/accel could be scaled to the magnitude of the turn direction. Or maybe ramp the north and east velocity terms in the direction of the next waypoint.
- The drone successfully starts wherever it is located in the map

Grid Search

- The grid search mode flies very close to the obstacles, as expected since it is trying to find the shortest path. With the resulting overshoot the vehicle comes very close to colliding with the obstacles. On the positive side, the grid search generates a path that is amenable to pruning so that the number of waypoints is significantly reduced, resulting in a bit smoother flight (if it doesn't hit a wall).

Voronoi

- The voronoi mode generates a path with more obstacle clearance than the grid search, as expected. This eliminates most near-miss with obstacle incidents, with the exception of some sharp turns when obstacles are close together. The disadvantage of this mode is that the path doesn't generate as many collinear waypoints, so that pruning doesn't help much and the flight path appears jerkier.

SPIRAL

- The spiral mode flies as expected. The altitude steps because the Z position is an integer. It appeared that 8 waypoints per circle looked best. Less than that and it wasn't a circle, more than that (e.g. 16) was too busy.