MEAM 620 Phase 4 Prelab: Waypoint Trajectory Generator

February 7, 2019

1 Introduction

During the first lab session, you are asked to accomplish two goals:

1. Tune your controller with the real CrazyFlie
2. Fly to a set of waypoints

2 Preparation

In preparation for the lab section, you are asked to implement a trajectory generator that takes a set of way points as input and creates a trajectory for the crazyflie to follow. During the first session, these waypoints will be a standard set of test points. However, during the second session, they will be the waypoints generated from your Dijkstra algorithm. We recommend that you fly simple straight line trajectories between waypoints during both sessions, unless you finish phase 3 early and are very confident with your results.

3 Environment

The simulation environment is almost identical to the one you used for developing the controller. Download the `.zip` file from the project website, unzip it, and copy your `controller.m` into the same directory.

The file `trajectory_generator.m` is where your code goes. The function takes a variable number of arguments, and is first (and only once) called with the path as the fourth argument. This is the point where you generate the trajectory, and store it in a persistent variable. On subsequent calls you will be using the precomputed trajectory and the time argument to calculate the desired state. For more details, see the header of the file.

4 Testing

The simulator has 4 preprogrammed paths in `runsim.m` that you can test. For example to test path 4:

```
runsim(4)
```

Your output should be a graph that looks like the one in Figure 1. Once you can fly all four trajectories in simulation you are ready for the lab! But read on below what to pay attention to.

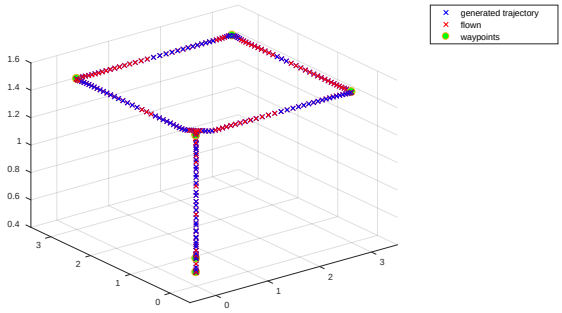

Figure 1: Trajectory for test path 4

5 What to expect in the lab

Considerations for the lab section:

1. Your generated trajectory *must* pass through all waypoints.
2. No need for speed! You will not be graded on the quality of the trajectories, and only the battery limits your flight time to 1-2min.
3. If your controller cannot quite follow the generated trajectory, that is fine, but crashing the quadrotor is not acceptable. So make sure your control inputs are reasonable.
4. You will notice (in particular for trajectory #3) that sometimes the way points are closer together, sometimes further apart. Try to make your generator robust to that.
5. Don't worry if you do stop-and-go all along a straight line with lots of waypoints. It may not look elegant, but it will be sufficient to pass the lab.

6 Looking Forward

During your second lab session, you will be asked to fly the robot through an environment with obstacles. To do so, you will use your Dijkstra implimentation to generate a set of waypoints, then use these waypoints in your trajectory generator. Closely following this path will prevent you from colliding with obstacles during flight