AE483 Lab Manual: Lab #3
Flight Control

T. Bretl

October 6, 2019

1 Goal

Your goal in the next three weeks will be to design, implement, and test a controller that makes the drone hover at a desired position with zero yaw angle. You will need to work steadily and to be organized in order to achieve this goal. We recommend the following schedule:

- Complete all of Section 2 in lab during Week 1.
- Complete all of Section 3 **before** coming to lab in Week 2.
- Complete all of Section 4 in lab during Week 2.
- Use your remaining time in Weeks 2 and 3 to complete all of Section 5—this will *not* be easy!

It *is* possible for you to complete Sections 2 and 3—and, perhaps, to make a good start on Section 4—all during Week 1. We strongly encourage you to try. It will make things so much more fun.

You will be responsible for writing a report that describes your approach and your results that will be submitted as a group no later than 11:59PM on Friday, November 1, 2019. Details will be posted to slack.

2 Framework

Do the following:

- Follow the instructions in Appendix F to enable sending "commands" from the ground-station to the drone. Doing so will give your on-board code access to MOCAP data and to the desired position and yaw angle.
- Replace your on-board `lab.c` file with the following template:

  https://uofi.box.com/s/pa3dpofd2hfem9lxl7c96l088mh175u7

- Delete these lines in `main.c` of the on-board code (they use the red LED for something we do not need — we will use the red LED for a different purpose):

```
1 //blink red led if no GPS lock available
2 led_cnt++;
3 if((GPS_Data.status&0xFF)==0x03)
4 {
5     LED(0,OFF);
```

```
}
else
{
    if(led_cnt==150)
    {
      LED(0,ON);
    }
    else if(led_cnt==200)
    {
      led_cnt=0;
      LED(0,OFF);
    }
}
```

- Also delete these lines in `main.c` of the on-board code (they use the green LED for something we do not need — we will use the green LED for a different purpose):

```
if(led_state)
{
    led_state=0;
    LED(1,OFF);
}
else
{
    LED(1,ON);
    led_state=1;
}
```

- Follow the instructions in Appendix D so that you log only the following variables:
  - off-board time
  - on-board counter
  - position (x, y, z) and orientation (yaw, pitch, roll) from MOCAP
  - desired position (x, y, z) and orientation (yaw)

Show your TA that both your on-board and off-board code runs. Show your TA, in particular, that the red lights flash on your drone when the on-board code runs, and that the green lights flash on your drone *only* when the off-board code runs *and* when the ground-station receives MOCAP data.

3 Control Design and Implementation in Simulation

Go through the full process of control design and implementation to make the drone hover at a desired position with zero yaw angle in simulation. You are welcome to begin with the template provided in class:

https://uofi.box.com/s/wr28hxfqn5q2bb9rtzuisy50sfwcev1e

In addition to using your own parameter values and your own control design, you should modify this template to bound the squared spin rates. You should also compute and log the motor commands that would be generated by your controller and tune your gains so that these motor commands

remain (for the most part) in the range 1 to 200. Show your TA a movie of your drone hovering — starting with some non-zero error — and also a plot of the motor commands that would have been applied as a function of time.

4 Control Design and Implementation in Hardware

Starting from the `lab.c` template (see Section 2), implement your controller with your on-board code. We will discuss strategies for doing so in class. Work with a TA to fly for the first time with your controller, when you are ready. **Be sure to record video of all flights** (e.g., with a smartphone) as well as to collect and store data for future analysis. Be ready for some fun debugging your code. Tune your gains and repeat your hardware experiments until you get reasonable performance.

5 Analysis and Improvements

Do a sufficient number of hardware experiments—at minimum, ten in which the desired position is constant in time, and one in which the desired position is a continuous function of time—to characterize both the accuracy of your model (e.g., by comparing predictions made by your model in simulation to results in hardware experiments) and the performance of your controller (e.g., by finding the mean steady-state error or the bandwidth of the closed-loop system). You will likely find that the controller does not work very well at first—in particular, it will likely show a significant amount of steady-state error. Your challenge will be to eliminate this error as best you can—your report will be evaluated, in part, on how well you do this. We will discuss good ways (and bad ways) to improve your controller in class. You should also talk about this with your TA.