CMSC 828T
Vision, Planning and Control in Aerial Robotics
Project 1 Phase 2 (P1Ph2):
Dynamic Modeling, Control and Simulation
Due on 11:59:59PM on September 21st, 2017

Prof. Yiannis Aloimonos, Nitin J. Sanket,
Kanishka Ganguly, Snehesh Shrestha

September 13, 2017

1 Introduction

In this project, we introduce the modeling of quadrotors used in the Drone Lab at UMD in the Department of Computer Science under Prof. Yiannis Aloimonos. These drones are used in the lab for research purposes. The modeling, control and integrating off-the-shelf components approach to build quadrotors is based on the research work being done in the Drone Lab at UMD and Mellinger's thesis [1].

2 Quadrotor Platform

The quadrotor platform constructed for the course is based on the very popular 250 racing quad frame. The frame is in the `X` configuration, making it ideal for the control equations you have learned in the course. The size of the quadrotor 250, means the diagonal distance between the motors is 250mm. The total weight of the platform is around 500 grams. The power train of the quadrotor consists of 4 $\times$ 2400KV motors driving 5 inches propellers which provides a combined thrust of about 4kgs. The quadrotor has a multitude of sensors which includes cameras, range finders and an IMU. The hardware consists of 3 layers:

- A *Betaflight* flight controller which controls the power train of the quadrotor. Under the 'level flight' mode, an control input is translated into desired roll, pitch, and yaw angles.
- A microcontroller which handles the low level communication, input and output to and from the peripherals.

Figure 1: Quadrotor 250: DroneLab @UMD

- A single board computer running Ubuntu 16.04 is the brain of the platform. It uses Robot Operating System (ROS) as the middleware to communicate and manage various software modules in the quadrotor.

3 Background Literature

Please refer to the lecture slides/videos and D. Mellinger's thesis [1] for this phase of the project as listed below:

- Lecture Videos
  - Quadrotor Dynamics Video
  - Quadrotor Controls Video
- Lecture Slides
  - Quadrotor Dynamics Slides
  - Quadrotor Controls Slides
- Trajectory Generation and Control for Quadrotors, D. Mellinger, Chapters 2.1 and 2.2 [1]

4 Tasks

You will simulate the quadrotor dynamics and control using the matlab simulator provided as part of this project [1]. The simulator relies on the numerical solver `ode45`, details about this solver can be easily found online. Your tasks include:

4.1 Trajectory Generator

You will first implement two trajectory generators that generate a circle and a diamond trajectories. In order to specify the trajectory in the simulator, you will need to change the variable `trajhandle`. `trajhandle` is the name of a function that takes in current time `t` and current quadrotor number `qn` and returns a struct of desired state as a function of time. The struct `desired_state` has 5 fields, which are `pos, vel, acc, yaw` and `yawdot`. For a proper trajectory, you need to specify `pos, vel` and `acc`, whereas `yaw` and `yawdot` can be left as 0. This is because we are assuming that we don't change our yaw angle.

- `circle.m`
  A helix in the xy plane of radius 5m centered about the point `(0;0;0)` starting at the point `(5;0;0)`. The z-coordinate should start at `0` and end at `2:5`. The helix should go counter-clockwise.

- `diamond.m`
  A Quadrilateral of a given size with corners at for example `(0;0;0), (0; p2; p2), (0; 0; 2p2)`, and `(0; -p2; p2)` when projected into the yz plane, and an x coordinate starting at 0 and ending at 1, where p2 could be some arbitrary integer value, for example, you can assume p2 is 1. The quadrotor should start at `(0;0;0)` and end at `(1;0;0)`.

4.2 Controller

You will then implement a controller in file `controller.m` that makes sure that your quadrotor follows the desired trajectory. This controller will be used again in the following phases. The controller takes in a cell struct `qd`, current time `t`, current quadrotor number `qn` and quadrotor parameters params and outputs thrust `F`, moments `M`, desired thrust, roll, pitch and yaw `trpy` and derivatives of desired roll, pitch and yaw `drpy`.

`qd` is a cell array contains structs for each quadrotor. This means `qdfqng` contains all the information (position, velocity, euler angles and etc.) about quadrotor number `qn`.

Pose information can be accessed via `qdfqng.pos` for example. The reason for using `qd` here is to be compatible with Drone flight control software that you will use in the final phase of this project to fly the quadrotor.

4.3 Simulation

Lastly, you need to fill in the appropriate variables for `trajhandle` and `controlhandle` in the script `runsim.m` for simulating your result.

5 Simulator

The quadrotor simulator comes with the student code. Before implementing your own functions, you should first try running `runsim.m` in your matlab. If you see a quadrotor falling from position `(0;0;0)` then the simulator works on your computer and you may continue with other tasks. This is because the outputs of `controller.m` are all zeros, thus no thrust generated.

When you have the basic version of your trajectory generator and controller done, you will be able to see the quadrotor flying in the space with proper roll, pitch, and yaw, leaving trails of desired and actual position behind. The desired position is color-coded blue and the actual position is red. After the simulation finishes, two plots of position and velocity will be generated to give you an overview of how well your trajectory generator and controller are doing. Note that you will not be graded on these plots but by the autograder.

5.1 Submission

When you are finished, submit your code using the CMSC 828T website.

Your submission should contain:

- A `README.txt` file that includes a short summary of what you did, lessons learned, any key hurdles you faced and how you overcame them. Please also include text detailing anything extra we should be aware of about your submission.
- A `LateDays.txt` file with the number of late days you wish to use, from whatever is remaining. The file should contain only a single digit number less than 4 as that is the maximum number of late days you will receive.
- Files `drone250x.m`, `controller.m`, `diamond.m`, `circle.m`, as well as any other Matlab files you need to run your code.

Zip everything into `code.zip` and use the Submit Page on the website to make your submissions. Based on the load on the server, the report might be immediate or might take up to 24 hours. You have limited submissions so please do not use the submit server as a debugging tool. It is to validate your final submissions and to make any additional modifications and optimizations.

You will be graded on successful completion of the code and how quickly and accurately your quadrotor follows the circle and diamond paths and one other trajectory which will not be released to you and will be our secret evaluation tests. There is no hard time cutoff on this phase of the project. However, as mentioned earlier, your score will also be a function of how quickly and accurately it follows the desired paths.

6 Acknowledgements

The project has been adapted from the University of Pennsylvania MEAM 620 course.

7 Collaboration Policy

You can discuss with any number of people. But the solution you turn in MUST be your own. Plagiarism is strictly prohibited. Plagiarism checker will be used to check your submission. Please make sure to **cite** any references from papers, websites, or any other student's work you might have referred.

References

[1] Daniel Mellinger. *Trajectory generation and control for quadrotors*. University of Pennsylvania, 2012.