POLYTECH NANCY

Lab 7

Altitude control for the Tello mini-drone

Emile Laugel[1]
Hugues Garnier
Mayank Jha

September 2023

[1]5A IA2R SIA Student in 2023-2024

Contents

Altitude control for Tello mini-drone

1.1 Introduction

For this lab, you are tasked with controlling a mini-UAV. An Unmanned Aerial Vehicle (UAV), commonly known as a drone, is an aircraft without any human pilot, crew, or passengers on board.
There are many types of UAVs, but the mini-UAV at disposition is a quadcopter. A quadcopter is a type of helicopter which has 4 rotors. Each one of them produces thrust and torque at the same time. For this system to be able to fly and stabilize itself in midair, the total thrust produced by the 4 rotors should be equal to the gravitational force subjected to the system.

The mini-UAV we will be using is the Tello mini-UAV controlled via Wi-Fi from a Python program.

Figure 1.1: Tello mini-drone from DJI

1.2 A set of videos to watch

Watch all together the first two videos from Brian Douglas that will provide you with the essential knowledge required to control the mini-UAVs:

Drone Simulation and Control, Part 1: Setting Up the Control Problem

Drone Simulation and Control, Part 2: How Do You Get a Drone to Hover?

1.3 Control performance requirements

The performance requirements and time-domain specifications for controlling the altitude of the mini-UAV are described in Table **??**.

Requirements	Assessment criteria	Level
Control the altitude of the mini-UAV	Position setpoint tracking	No steady-state error
	Settling time at 5%	As short as possible
	Percent overshoot	$D_1 = 5\%$

Table 1.1: Performance requirements for altitude control of the Tello mini-UAV.

1.4 Download the files required for the lab

1. Download the zipped file *Lab.zip* from the course website, save and unzip it in a folder.
   It is recommanded that you use Visual Studio Code as an IDE for this lab. But if you feel more comfortable with another IDE, you can use it.
2. Follow these steps to install the DJITelloPy library:
   - Open your terminal or command prompt
   - Run the following commands to make sure you have Python and pip installed on your computer:

     ```
     python --version
     pip --version
     ```

   - Run the following command to install the library and wait for the installation to end:

     ```
     pip install djitellopy
     ```

3. Open the Python program that was inside the zipped file. Look through and try to understand each of its functions.

1.5 Transfer function model identification from step response experiment

The determination of a model is the first crucial step for the design of a feedback control system.

1.5.1 Transfer function model for altitude control

The input and output of the mini-drone are:

- input: percentage of maximum velocity on the Z axis $u(t)$ (related in some way to the thrust generated by the 4 rotors);
- output: altitude (or position along the Z axis) of the mini-drone $y(t)$ in cm.

The maximum velocity percentage-to-altitude transfer function takes the form of a first-order plus pure integrator model

$$\frac{Y(s)}{U(s)} = \frac{Ke^{-\tau s}}{s(1+Ts)} \tag{1.1}$$

where $Y(s) = \mathcal{L}[y(t)]$ and $U(s) = \mathcal{L}[u(t)]$ K is the model steady-state gain, T is the model time-constant and τ is the time delay of the system.

As the velocity on the Z axis $v_z(t)$ is the time-derivative of the altitude, both variables are linked in the Laplace domain by an integrator so that (1.1) can be expressed as:

$$\frac{Y(s)}{U(s)} = \frac{Y(s)}{V_z(s)} \times \frac{V_z(s)}{U(s)} = \frac{1}{s} \times \frac{Ke^{-\tau s}}{1+Ts} \tag{1.2}$$

where $V_z(s) = \mathcal{L}[v_z(t)]$ is the velocity of the mini-UAV on the Z axis.

Identifying a system having a pure integrator is tricky and it is better when the measure is available to identify the response between the velocity on the Z axis and the input (maximum velocity percentage on that axis) model takes the form of a simple first-order system.

The Tello mini-UAV velocity percentage-to-velocity transfer function has therefore the well-known first-order form which parameters can be easily estimated from a step response:

$$\frac{V_z(s)}{U(s)} = \frac{Ke^{-\tau s}}{1+Ts} \tag{1.3}$$

1.5.2 Model identification from step response experiments

Each mini-UAV can only be connected to one computer when it is turned on via Wi-Fi. You can identify your mini-UAV by removing the battery and looking inside the battery case. You should see written on a white paper ***WIFI:*** *TELLO-****XXXXXX*** where the six ***X*** represent the ID of your mini-UAV.

1.5.3 Recording of the step response

This experiment is carried out in manual mode, i.e. there is no feedback control to the altitude. As the system is marginally stable, the input command should be designed with special care. As shown in Figure 1.2, the input command is a series of positive and negative steps around a working operating point. The duration of each step has been chosen so that the drone altitude increases of about 50 cm.

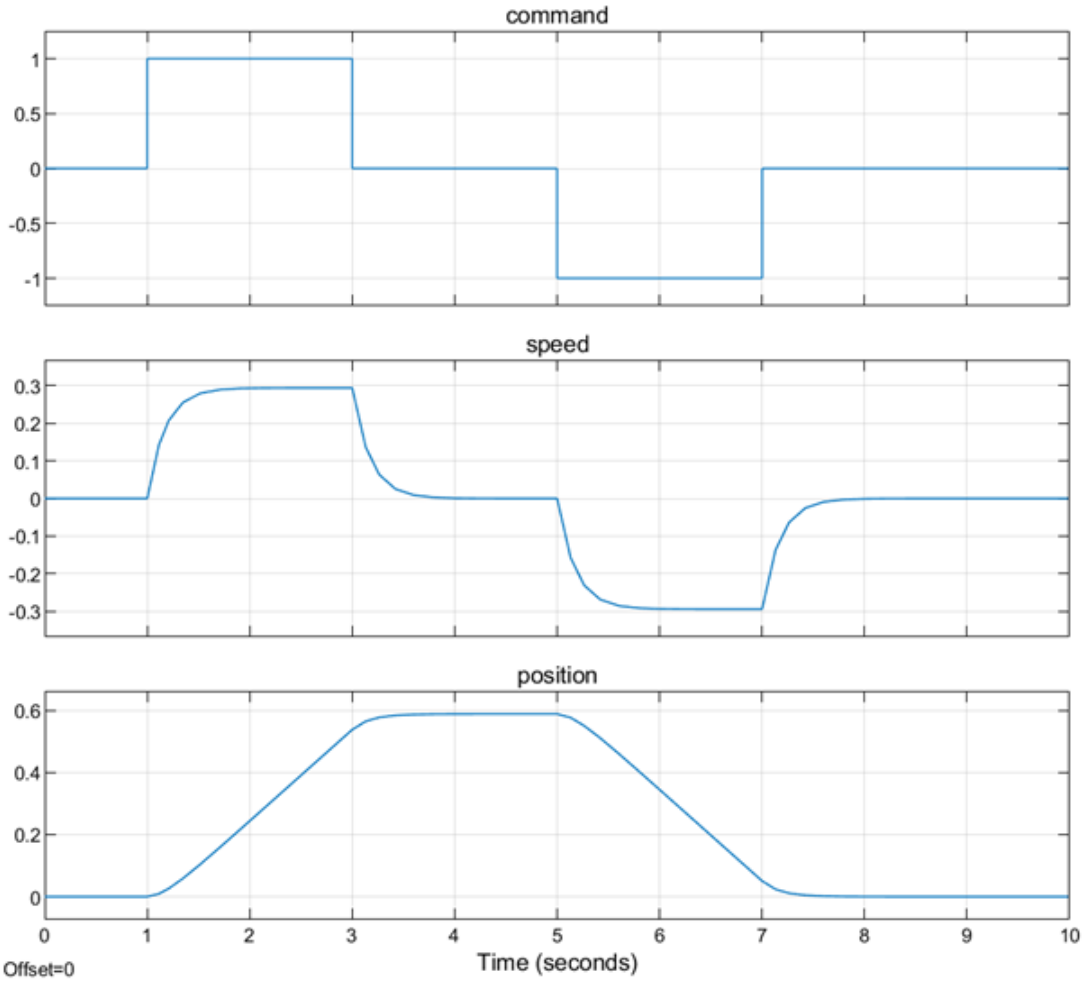

Figure 1.2: Position and velocity responses to a series of step inputs sent to the mini-drone in open-loop

From Figure 1.2, as expected from the marginally stable model, a a positive step to the command input leads to a monotonic increasing altitude. As mentioned above, the following negative step time should be short enough for the mini-UAV to not collide with the ceiling, while being long enough for the velocity response to reach its steady state.

1. Open the `"open_loop.py"` Python program provided in the zipped file you downloaded.
2. When you scroll down from here, you will come across **Open Loop** surrounded by #.

   In this part, we will use a function from DJITelloPy library, called **send_rc_control(X,Y,Z,YAW)**. This function requires 4 input arguments, each representing a specific type of motion control for the mini-UAV.

   - 'X' represents the velocity percentage of the drone in the left and right direction.
   - 'Y' stands for the drone forward or backward velocity percentage.
   - 'Z' controls the drone upward or downward velocity percentage.
   - Lastly, 'YAW' refers to the drone rotational velocity percentage around its center axis.

   Essentially, these input arguments allow us to send specific directional and rotational commands to the drone, controlling its movement in real-time. But as we are going to control the velocity on the Z axis only, the other input arguments will be set to 0. They are the velocities of their own axis in cm/s and range from -100 to 100.

We will define a sequence of step inputs for the mini-UAV using the function **tello.send_rc_control(0,0,Z,0)** where Z is the velocity you choose ranging from -100 to 100.

For the step sequence, your Z should alternate between 80 and -80 with a delay of 2 seconds and it should look like this:

tello.send_rc_control(0,0,80,0)
time.sleep(2)
tello.send_rc_control(0,0,0,0)
time.sleep(2)
tello.send_rc_control(0,0,-80,0)
time.sleep(2)
tello.send_rc_control(0,0,0,0)
time.sleep(2)
Add your step commands below the commented line: `Write your set of step commands below`. Duplicate this set of commands for 3 to 5 times then wait for the mini-UAV to land.
To run the experimental tests, go to the room 335. Place the drone inside the cage.
Run the open-loop test with the drone inside the cage.

1.5.4 Identify your model

After your mini-UAV has landed, a file named **"result.txt"** should be present in the same folder where you have placed your Python program. Additionally, you need to ensure that you do not close or terminate your Python program before the recorded values are plotted.

1. Open and run the **"identify_your_model.mlx"** file. This file uses the PROCSRIVC routine to identify a first-order model with a delay with the data you recorded.
2. Observe the identified values of K, T and τ. Does the estimated values make sense ?
3. What can you say about the fit percentage? What can you say about the comparison between your model to the measured output.
4. Crop your data to the segment where the mini-drone has reached its steady-state value and identify your model once more from this cropped segment. Does it change anything ?
5. Now open and run the **"identify_your_model_corrected.mlx"** file. This file does the same thing as the previous file but fix the values of the time-delay in a range to avoid the incoherence found by the PROCSRIVC routine.
6. Repeat questions 2 to 4 with this newly identified model.
7. Conclude on the difference between the two methods and select your best identified model.

1.6 P and PD control for altitude tracking

For systems represented by a first-order transfer function plus a pure integrator, a simple P controller or a PD controller is enough to get good tracking performances. For now, you are going to implement a simple P controller using the pole placement method.

1.6.1 P controller

For the rest of the lab, we are going to ignore the time-delay as it is close to zero. Therefore, the mini-drone system is now represented by the following transfer function:

$$G(s) = \frac{K}{s(1+Ts)} \tag{1.4}$$

and the following block diagram:

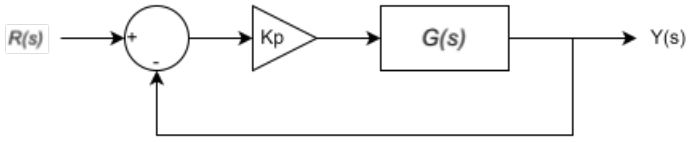

Figure 1.3: Simple P feedback control for altitude tracking of the mini-drone

1. Determine the closed-loop transfer function $F_{CL}(s)$
2. From the requirements specified in Table 1.1, determine the K_p proportional gain value.
3. Open the Simulink file **P_simul_control.slx**. Enter the parameters of your identified model and set the gain of the P controller.
4. Run the file. Are the specification requirements satisfied ? If not adjust the gain of the P controller.
5. . Open the `closed_loop.py` file and set the P controller gain below the commented line: `Enter the P gain value below`.
   Place the TELLO mini-drone inside the cage in room C335 and run the test.
6. Are the specification requirements satisfied in practice ? If not adjust the gain of the P controller.

1.6.2 PD controller

Try to increase the performance of the altitude tracking by implementing a PD controller.

1.7 Troubleshooting

This section provides solutions to calibration issues you might encounter when using the TELLO drone during this lab.

First, if you try to connect to your drone and it takes too long with the computer you are using, try to execute the Python code. If it prints the battery level, then your drone is already connected to the computer.

Then comes the second trouble. If it returns an error after printing the battery level, it either means that your drone battery is too low (under 10 %) or needs to be calibrated.

To calibrate your drone, you first need to download the Tello app on your phone from this following link: `www.dji.com/uk/downloads/djiapp/tello`

Then, when you open the App, it will automatically try to connect your phone to a Tello drone which will open your Wi-Fi interface and show you the available devices. Now, choose your drone. As it is not an internet device, your phone might ask you if you still want to connect to it. Confirm and when you are connected, go back until you see the App again.

Once you are connected to your drone and at the main interface of the App, you should see a gear on the top left side of your screen. Click on it then to "More" and to the "..." on the left side. The first option to appear should be "IMU Status". Click on "Calibrate" on the right side of this option and follow the instructions.

If you try to calibrate your drone and the App indicates that it has been calibrated without the need to place the drone in 6 different positions, close the App. Restart your drone and do the whole process again. If it is still the same, then it means your drone cannot be calibrated. Mark it and use another one.

In this calibration instructions, it is said that you should remove the propellers. There is a metallic and flat stick in the plastic bags inside th drone box. Itt is the Propeller Removal Tool they are talking about. The hollow part is to be used between the propeller and the motor to which it is attached.

Before removing them, mark that there is a pair with marks near the axis. The marked propellers should be at the top left and bottom right of the drone while the unmarked fill the other slots.

Note that this calibration is necessary for the Python code to be uploaded to the drone.