Drone landing report

Hao Liu

April 2022

1 Introduction

This is a report about the drone landing experiment based on Tello EDU drone. In this report, I'll explain the coding logic, the progress and the problems remained to be solved for further tests.

In this project, Python is chosen as the coding language and we use 5 packages which are djitellopy (the package designed for Tello EDU, which has been explained in detail in the Tello Documentation), cv2 (for image processing), numpy (for math processing), pygame (for keyboard control) and time (to record operation time).

2 Camera and figure analysis

In this project, we are using the bottom camera of the drone to detect the position of the landing platform. Since the bottom camera is a gray scale camera, we can only obtain gray scale of the object, which means we can easily use cv2.findContours to find contours of objects with color different from the environment. However, based on $https : //www.computervision.zone/courses/drone-programming-course/$, if we create a mirror clip for the front camera, an RGB camera, we can simply design a filter so that the camera can only identify objects with certain colors. All in all, we can simply use the camera to identify the position of the landing platform relative to the center of the drone. To prove this, we designed a very simple proportion controller for the drone and it can simply navigate itself to a still landing platform and hovering itself above.

3 Control

Recalling the partial observation case in the Literature Review of this project, shown in Fig 1, which is just the case of this experiment, we cannot directly measure the speed of the platform. Thus we need to use that observer to estimate the speed of the platform so that we can track a moving platform.

We created a global reference frame based on the starting position and direction of the drone. $\bar{y}_m$ is the observed position of the platform, which is calculated based on the camera reading. Other procedures are just the same as discussed in the Literature Review. Meanwhile, just as mentioned in the literature review, we tried to use the tracking policy discussed in [1]. The control model is simplified because we no longer need to use the attitude controller to

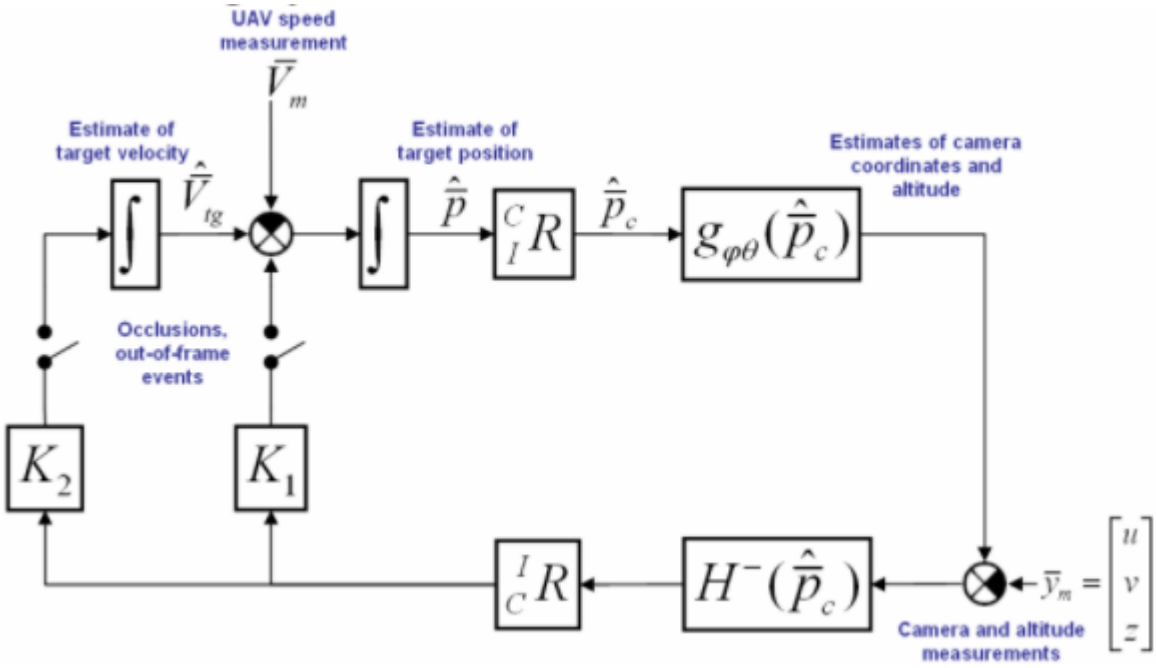

Figure 1: Observation logic

calculate for rotor control. We can only control the drone with send_rc_control(lr,fb,ud,yv) (lr and fb indicates the amount of joysticks being pushed to change the longitudinal and horizontal moving speed). Here we have an assumption on lr and fb. We assume that we can find a linear relationship between lr, fb and the desired instant acceleration. Thus, Eqn (26) and (27) in the Literature Review becomes:

$$fb = K(-(1/T_1)V_{Q_x} + (1/T_1)u_1) \quad (1)$$

$$lr = K(-(1/T_2)V_{Q_y} + (1/T_2)u_2) \quad (2)$$

All other control logic are the same as is discussed in the third part of Literature Review.

By the way, the time step used to update the state of everything is obtained by using time.time(). We record the time spent every iteration and use it as time step.

4 Problems Remaining

Unfortunately, due to time limit, the project is still not finished yet. The main issue is the relationship between pixel reading from the camera and the cm/s reading from the drone. The unit difference made it very difficult to find an appropriate parameter to have the system converged. Up till now, I haven't reached a good observation of the platform state. The speed of the platform diverges quickly after the platform is detected, which had never happened during the simulation. At the same time, the built-in coordinate of the drone is very weird which might lead to some potential mess up in the coordinate transfer. All of these are very hard to be debugged and some more time needs to be spent on it.

References

[1] H. Voos and H. Bou-Ammar, "Nonlinear tracking and landing controller for quadrotor aerial robots," in *2010 IEEE International Conference on Control Applications*, 2010, pp. 2136–2141.