Inter IIT Tech Meet 11.0

PLUTO DRONE SWARM CHALLENGE
Task 1 and 2 report

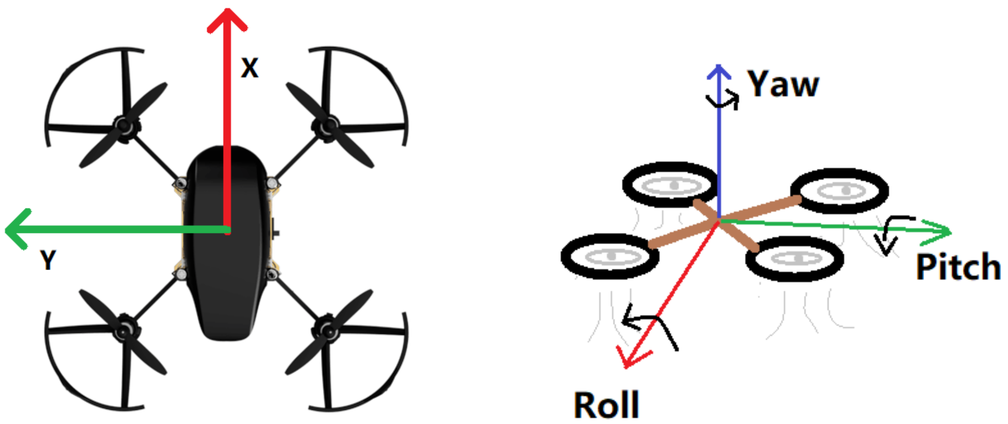
Submitted by - : Team 27

, FEBRUARY 2023

Contents

1 INTRODUCTION

A quadcopter has four propellers (as per the name), hence by controlling the speed of all these propellers, it is possible to direct the resultant thrust vector as per our desire. The Pluto drone already has attitude control, which takes care of the drone orientation. Hence it is needed to command the desired attitude to achieve certain position or trajectory. The figure below describes the defined axis with respect to drone frame, and the sense of rotation about each of them.



(a) Defined body frame (image source:Drona aviation)

(b) Rotational degrees of freedom

Figure 1

Hence in order to translate in positive x-direction, the quadcopter needs to rotate anticlockwise about y-axis, which is called pitching. Similarly for translating towards y-direction, rolling about x-axis is required. There is one more degree of freedom, rotation about the z-axis, which helps in changing the heading of quadcopter. This is called yaw.

2 TASK 1

Purpose:- To fly the drone by sending MSP (Multiwii Serial Protocol) packets via Wifi communication.

The flight controller board has an ESP 8266 wifi module, which allows communication between the drone and a PC. A TCP connection is established between the PC and the drone using python's Socket library.

2.1 The Socket library

The socket library is a collection of low-level programming interfaces for creating network connections and exchanging data between systems over a network. It allows programmers to create and manage client-server applications that send and receive data through a network. The socket library provides a standard set of functions and protocols for communication, making it easier to develop applications that run on different operating systems and network configurations. Additionally, the socket library is widely used for various types of network applications, including web servers, instant messaging services, and file transfer protocols.

2.2 Data packects for communication

A set of data packets are predefined for the pluto drone that allows to send actuation commands and receive raw/processed sensor data. The packets are structured in Multiwii Serial Protocol (MSP) format. There are several packet types, out of which **MSP_SET_RAW_RC** and **MSP_SET_COMMAND** are the 'in' packets, i.e. from computer to the drone.

MSP_SET_RAW_RC is enough to fly the drone using an external controller, a human or a computer algorithm, as it allows to set the roll, pitch, yaw and throttle values to a desired point. Also various modes can be selected like altitude hold, auto pilot etc. **MSP_SET_COMMAND** allows to perform some inbuilt actions like takeoff, land and flips using the integrated inertial sensor.

For the first task, a python wrapper is created which includes functions for arm, disarm, takeoff, land and setting the desired values for roll/pitch/yaw/throttle. Instead of separate functions for all, a single function call sends the desired values for all the actions. This allows the user to set simultaneous values (for example roll and pitch can occur together), which is helpful to apply automatic control for the drone further.

3 TASK 2 - part 1

Purpose:- To make the drone hover at a particular height using the position obtained using ArUco marker.

In order to control the drone position, we need a feedback system that will tell its (atleast) current position with a good update rate. From the position data, velocity can be obtained by discrete differentiation. Update rate is important as the given system has low inertia, hence is very agile. Existing methods include GPS, laser based distance sensors, camera based pose estimation etc. Inertial systems are also used but have limited reliability because of the induced drift due to the integration involved. GPS cannot be used for indoor flights and has slow update rate. Light and sound based distance sensors are fast and can be used indoors, but are trickier to work with as the chance of false estimation due to moving objects or another interfering light/sound source can cause problems.

(a) GPS module

(b) Distance sensors

Figure 2: **Pose estimation techniques** (image source: google)

Camera based pose estimation techniques also have certain pros and cons. They are more robust as the data under observation is comparatively much larger in size (image is a 2X2 matrix hence contain much larger information). However due to the large data size, data acquisition and processing takes more time and energy. Here we are going with camera based pose estimation with an external camera, i.e. camera is fixed outside the quadcopter. Also the data processing and control is done on an external PC and data sent via wireless communication using MSP packets as described in Task 1.

3.1 Aruco marker

Now to locate the drone in the image coming from camera feed, ArUco tag are used. These markers uses black and white square blocks of specific sizes to encode certain information given by the user (ID). The number of square blocks also vary according to the size of information to be encoded. We have used a 4X4 ArUco tag for the detection purpose.

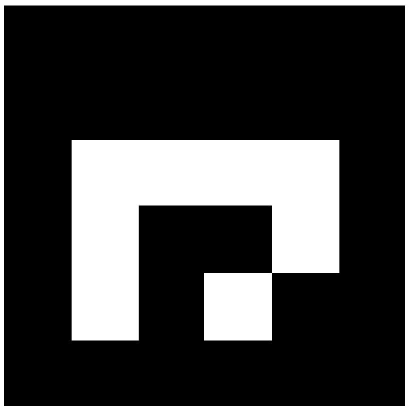


Figure 3: ArUco tag used for the task

Basically the detection of ArUco markers depend upon corner detection algorithms, as the marker has very high contrast (assuming the print quality is good enough). But to extract the relative position from the detected marker, the camera needs to be **calibrated**.

3.2 Camera calibration and Pose estimation (source: gitHub)

Camera calibration is the process of determining the intrinsic and extrinsic parameters of a camera. These parameters describe the camera's internal characteristics, such as its focal length, and its position and orientation in the world. Intrinsic parameters describe the camera's internal optical and mechanical characteristics and include the focal length, principal point, and radial distortion coefficients. These parameters affect the scale, aspect ratio, and radial distortion of the captured images. The camera lens creates image distortion.

Extrinsic parameters describe the position and orientation of the camera in the world and include the rotation and translation vectors. These parameters are important for determining the position of objects in 3D space.

After proper calibration, relative position can be obtained from the camera centered coordinate frame.

In order to move from one position to the other, it is needed to first know the current position and the goal position relative to it. Then the drone can start moving towards the goal position until the relative distance (error) becomes zero. However if the world is ideal, the drone can start suddenly with a non zero velocity, travel with the same velocity and suddenly stop as soon as the goal is reached. But that means infinite acceleration at the starting and ending points, which is not possible in real world. Hence the actuation commands of the drone needs to be a function of the current error (between starting and ending points).

3.2.1 Challenges faced

- Pure translation along z-axis with respect to the camera frame results in slight change in the x,y coordinates as well due to the pin hole model of camera (expanding Field Of View (FOV)).
- Low resolution form the camera gave a good update rate (for the complete algorithm), but could not detect the markers robustly, whereas high resolution detected better but made the update rate slower.
- The black ink (with which marker was printed) is pretty shiny and causes hindrance in detection.
- The focal length and field of view of the camera also posed challenges.

3.3 PID control

PID stands for Proportional Integral Derivative control. Here proportional control part simply **scales the error**, which is feeded to the actuator. But there is mostly some delay involved in reading the sensor data and producing the actuator commands, this is called **phase delay**. Due to it having only proportional control will cause oscillations, which may even grow over time and make the system unstable.

Hence the system needs to have some damping to atleast contain the oscillation, which can later be fine tuned to achieve a good response. Generally systems have damping due friction and fluid drags, but the effect may vary. **For example in our drone, lateral movement (roll, pitch) has very low drag when the drone is natural hovering position. Also there is a plenty of phase lag caused by marker detection and pose estimation part.** Hence, a derivative term is needed in the controller design to create the necessary damping. This will also be used to track velocity while doing trajectory tracking.

The integral term is used in removing steady state error caused due to deviation in estimation of parameters like mass, PWM input to thrust output mapping, inertia etc. However it is kept low as there is potential to induce oscillation. The integrator part also has saturation limits to keep the control outputs within limit.

To put it together proportional term takes care of present, derivative term sets the future and integrator laments on the past mistakes!

Mathematically the controller in continuous time domain can be expressed as:-

$$u(t) = Kp * e(t) + Ki * \int e(t)dt + Kd * de(t)/dt \tag{1}$$

Where, Kp is the proportional gain
Kd is the derivative gain
Ki is the integral gain
e(t) is the error at time t
u(t) is the controller output

3.4 Discretization and filtering

However to implement it in computer we need discretized model. There are several approximation methods for the approximations of the integral and derivative terms. Also pure differentiation is avoided to reduce the effect of noise. In our case we have pre-filtered the position data using a **5 point moving average filter**. The reason for choosing FIR (Finite Impulse Response) filter over IIR (Infinite Impulse Response) filter is the amount of phase lag introduced by IIR filter. Since the system is pretty agile, it is very sensitive to any amount of phase lag.

General form of a M point FIR filter is given by:-

$$u[n] = \sum_{k=1}^{M} b_k * u[n-k] \tag{2}$$

Hence for an MA filter, b_k = 1/M $\forall$ k

Lastly by fine tuning all the gains, hovering was achieved.

3.5 Results

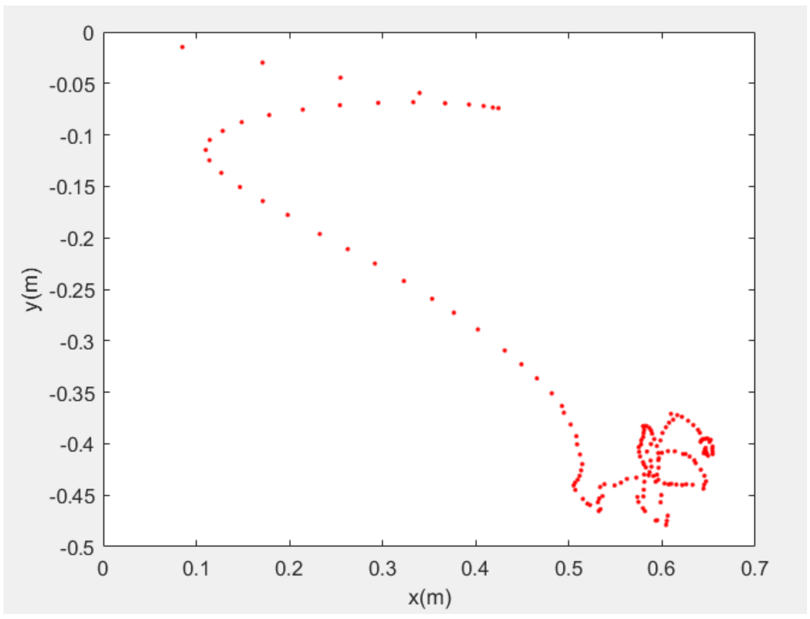


Figure 4: Obtained quadcopter positions over time during hover

4 TASK 2 - part 2

Purpose:- To make drone traverse a rectangular path using ArUCo based pose detection.

After having the PID control in place, it is possible to guide the drone by giving desired positions as a function of time. For a simple path like a straight line, just a single point as the desired point is sufficient. But the acceleration profile mostly looks like a straight line segment with negative slope, crossing zero at the mid-point.

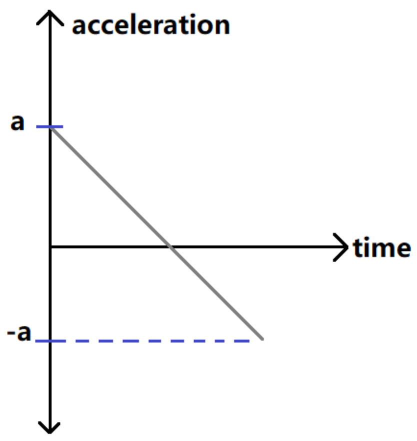
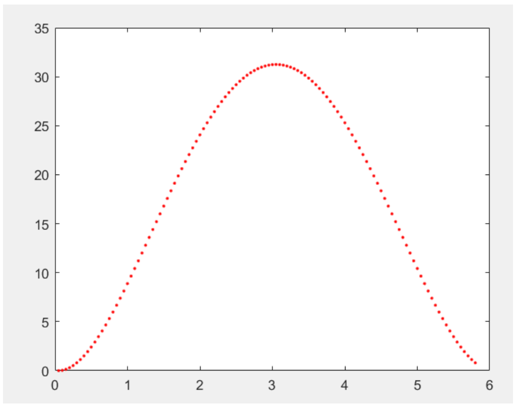


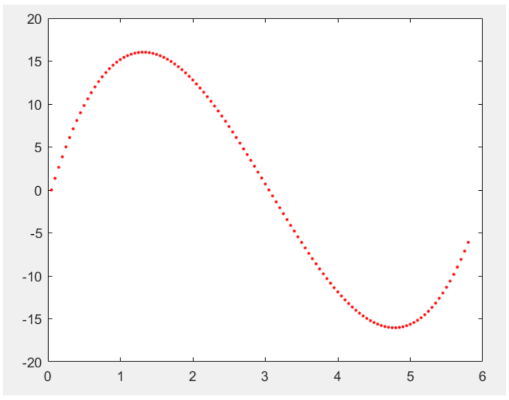
Figure 5: Acceleration profile

4.1 Quintic polynomial approximation for waypoints

Here we can notice that there are jerks at the start and end points. But with the simple PID controller, we don't have control over the acceleration, i.e. second order boundary conditions. Hence instead of giving the end point directly as the desired point, the path can be divided into fine steps, whose step size will be governed by a 5th order (quintic) polynomial. Since the polynomial is differentiable more than two times, it is possible to set a desired initial and final acceleration, which is mostly kept zero. This way jerk can be removed. Lets look at the desired velocity and acceleration profiles that comes from a quintic polynomial.



(a) Velocity profile



(b) Acceleration profile

Figure 6: **Desired velocity and acceleration profiles with quintic polynomial approximation**

Here we can observe the smooth trend in acceleration which removes jerk. However we did not adopt this approach in the final submission because of the limited update rate. The desired position values update at a certain rate, which needs to be perfectly captured by the controller to get an agile system. But due to the phase delays and other limitations it was forcing to reduce the overall travelling speed of the drone (i.e. the time for completing one straight line had to be kept long enough).

Hence directly desired setpoints are given to the controller after specific intervals to complete the required rectangular trajectory.

4.2 Results

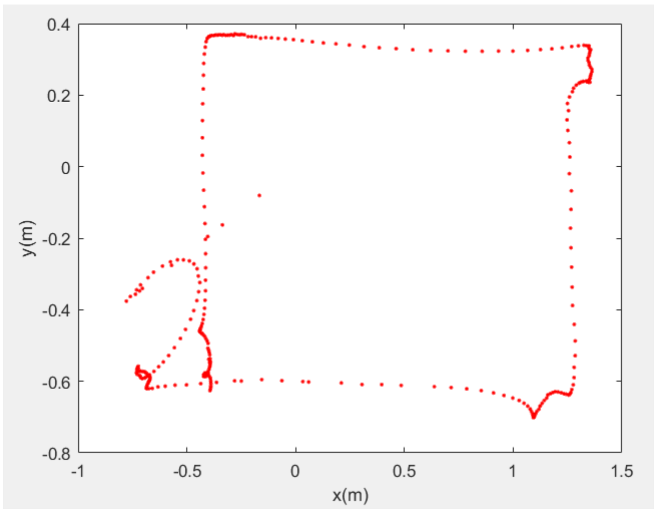


Figure 7: Obtained quadcopter positions over time during rectangular trajectory tracking

4.3 CONCLUSION

The accuracy of resultant path mostly depends on the detection of ArUco marker and position estimation. Hence the type of camera and its specifications greatly affects the overall performance. Also lighting, marker quality (printing) and drones speed creates significant effect. The performance can be drastically improved by using a high speed camera with good Field of View.

References

[1] J. Aguerrebere, E. G. Hernandez-Martinez, S. Montufar-Chavez, X. Tortolero-Baena, M. Salgado-Aguirre, G. Fernandez-Anaya, E. Ferreira-Vazquez, and J. J. Flores-Godoy. Quadcopter uav control based on input-output linearization and pid. pages 1003–1006, 2021.

[2] María Elisa Castro Fúnez, Andrés Felipe Báez Aponte, and Triana. A pid-controlled quadcopter system: The effect of parameters selection. *International Journal of Applied Engineering Research*, pages 1–6, 01 2020.

[3] Mücahid Rıdvan Kaplan, Abdullah Eraslan, Aykut Beke, and Tufan Kumbasar. Altitude and position control of parrot mambo minidrone with pid and fuzzy pid controllers. In *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 785–789, 2019.