Submission Report

Inter IIT Tech Meet 11.0

Primary Team ID#19

High Prep
Drona Aviation Pluto Swarm Challenge

February,2023

CONTENTS

Pluto Swarm Challenge

Abstract: - Drones are being developed for various applications like military, medical aid delivery, emergency first responder, surveillance, etc. They have four counter-rotating propellers placed on the same airframe. For performing tasks, the controller requires real-time feedback on the position of the drone, and for localization using vision-based navigation, we used ArUCo as mentioned in the problem statement. The detection algorithm was implemented using the OpenCV library since it provides large sets of the image-processing algorithm. Further for compensating any loss due to computation causes the tracked coordinates values were smoothened out moreover moving average statistical method was applied over the last 20 coordinates being tracked and returned, and an alternative median smoothing method was also applied to further smoothen the coordinate values. The data received from the camera and then processed by the computer for estimating the latest pose and the [x,y,z] coordinates of the drone are transmitted to the drone via Wi-Fi using Multiwii Serial Protocol(MSP) packets. Attitude dynamics are fast and key to the stability of a quadcopter. Hence, they require a computationally simple and fast controller. The position controller, on the other hand, is comparatively slower and more complex. The PID coefficient parameters have been tested and tuned for the specific scenarios and further Anti-Windup schemes have been added to prevent integration wind-up when the throttle values are saturated.

Keywords: - ***Drone,*** **Multiwii Serial Protocol*****, Image processing, PID controller, ArUCo, OpenCV, Vision-based Navigation, Pluto 1.2***

Introduction: -

Drones are being developed for use in various applications like military, medical aid delivery, emergency first responder, surveillance, etc. They have four counter-rotating propellers placed on the same airframe. They are controlled by differential adjustment of the angular velocities of the rotors. Drones can move freely in 3D space i.e., they have 6 degrees of freedom (DOF). But they have only four rotors (actuators) to control all 6 DOF. Thus, they are underactuated systems. Independent control of 2 states namely x (forward/backward), y (right/left) translation is lost. These translations are coupled with the Roll & Pitch angles.

Drona Aviation Pluto 1.2 is an open-source programmable nano drone allows users to integrate with external hardware. So, that we can program our drone to add multiple applications. The drone has an onboard microcontroller that helps implement various algorithms to stabilize and fly the drone. Its firmware is a modified version of the popular, open-source flight software, Cleanflight.

Our solution to this challenge comprises of various strategies for the control of this drone.

In this challenge we have created the python wrapper for the interaction with the Pluto official firmware API. The challenge is divided into three tasks, (a) Task-1: It comprises of the implementation of control commands for the drone movements, such as Pitch Forward, Roll Left, Take-off, landing etc., (b) Task-2: It comprises of detection and tracking of ArUCo tag using overhead camera module and thereby pose estimation of the ArUCO tag and then implementation of PID controller for drone and using it for hovering and making the figure of a rectangle of (1x2 meter) with drone., (c) Task-3: It comprises of drone swarming and making the figure of rectangle as stated above.

Methodology and Approach: -

For performing our task, the controller requires real time feedback of the position of the drone and for localization using vision-based navigation, we used ArUCo as mentioned in the problem statement.

ArUCo Tag Detection & Tracking: -

We used 4X4 ArUCo tags dictionary in our solution because they are computationally simple and were clearly visible during detection and had occlusion avoidance properties. For tracking purpose, we used the mobile camera input with the help of iVCam App. Depending on the height of the ceiling the camera could be calibrated and settings for image resolution can vary from 1080p to 4K at 30FPS. We found that better image quality resulted in better ArUCo detection but on the expense processing time so we have taken both the parameters into our consideration for best results. `calibration.py` contains the code for the calibration of the camera setup and it generates two files in the `calibration_data` folder, namely `calibration_matrix.npy` and `distorsion_coefficients.npy`.

The detection algorithm was implemented using the OpenCV library since it provides large sets of image processing algorithms. The algorithm starts by applying adaptive threshold to the image by calculating for each pixel a threshold value using the histogram of its neighbourhood. It is of particular interest for situations with multiple lighting conditions. To determine the threshold block (neighbourhood size), one block size is tested on each frame, the block size chosen is the average size from all block sizes the maximum number of markers were found, the block size is retested when there are no markers visible. After the threshold is applied to the image, we perform square detection contours detection using a border-following algorithm. Even under significant perspective distortion a square is always a convex quadrilateral. To filter noise, a criterion was added: all contours composing a geometry with an area below a defined threshold were discarded. These criteria allow to properly filter squares even under heavy distortion from the contour list. The marker data is validated as ArUCo using the calibration matrix. Markers might be detected in any orientation. The algorithm tests the data with different rotations (90º, 180º, 270º), if the marker is not recognized for any rotation, it is then discarded.

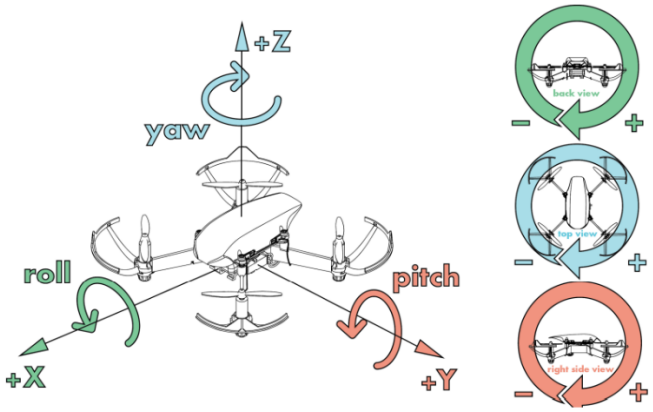
Further for compensating any loss due to computation causes the tracked coordinates values were smoothened out moreover moving average statistical method was applies over last 20 coordinates being tracked and returned and an alternative median smoothing method was also applied to further smoothen the coordinate values.

Communication: -

The data that is being received from the camera and then being processed by the computer for estimating the latest pose and the [x,y,z] coordinates of the drone are transmitted to drone via Wi-Fi using Multiwii Serial Protocol(MSP)**[4][5]** packets. The drone was connected to laptop using socket communication methods**[3]** in the python script `Cummunication.py` which generates the commands for the control of the drone movements such as Pitch Forward, Roll Left, Take-off and landing etc. in the proper form to be sent to the drone using MSP via Wi-Fi.

Controller Design: -

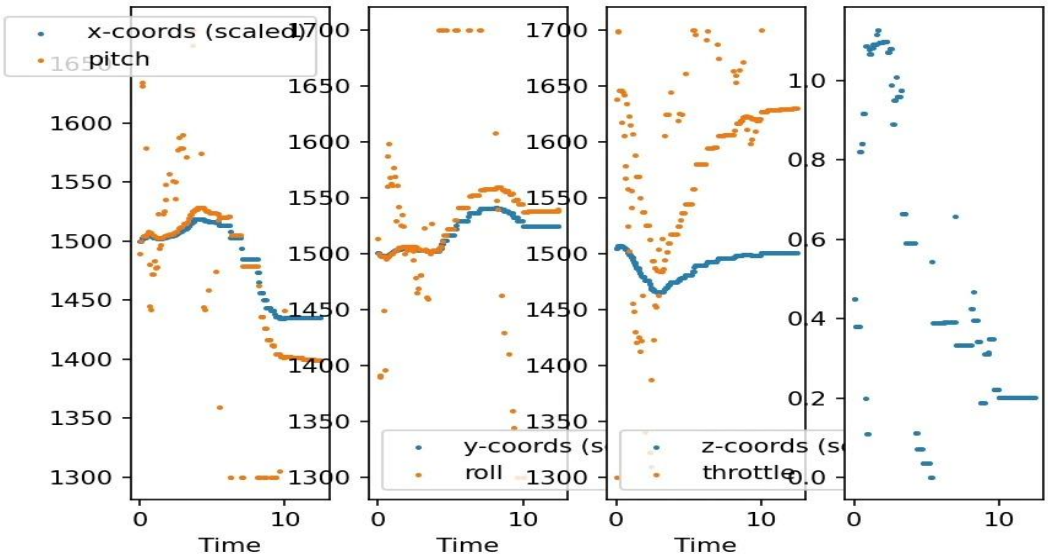
The position states (x & y) depend upon the attitude states (ϕ, θ and ψ). This implies that the attitude dynamics are faster than the position dynamics for the drone. Any control strategy thus designed can exploit this property. Attitude dynamics are fast and key to stability of a quadcopter. Hence, they require a computationally simple and fast controller. The position controller, on the other hand, is comparatively slower and more complex.



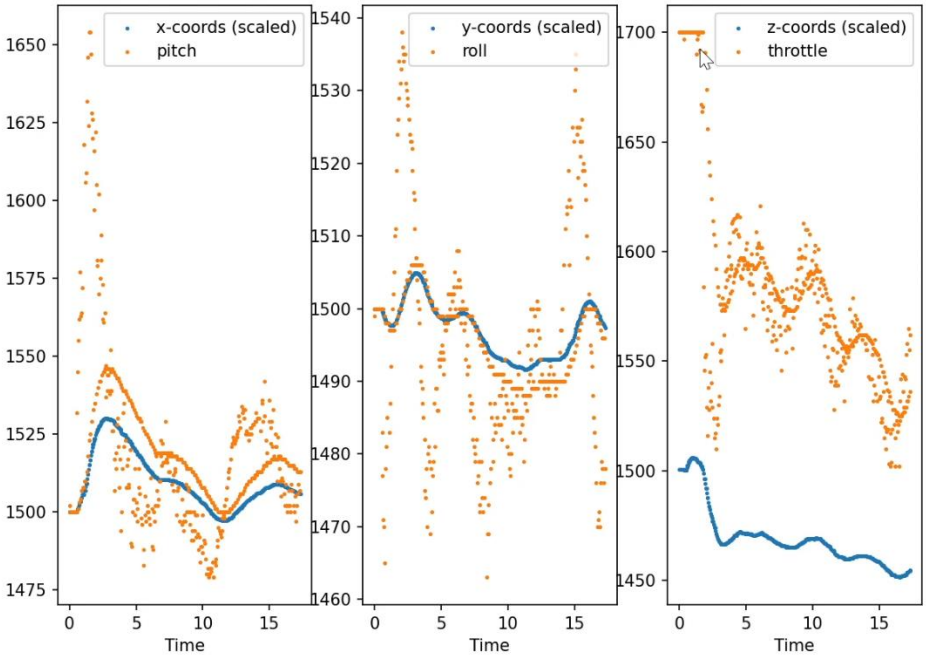
General Notation

The x position depends on the pitch angle (θ) and the y position on the roll angle (ϕ), they form a separate subsystem respectively. The Altitude (z) does not depend on any orientation angle; hence it is considered as a separate subsystem. Similarly, the heading angle (ψ) is completely independent of the others and controlled separately.

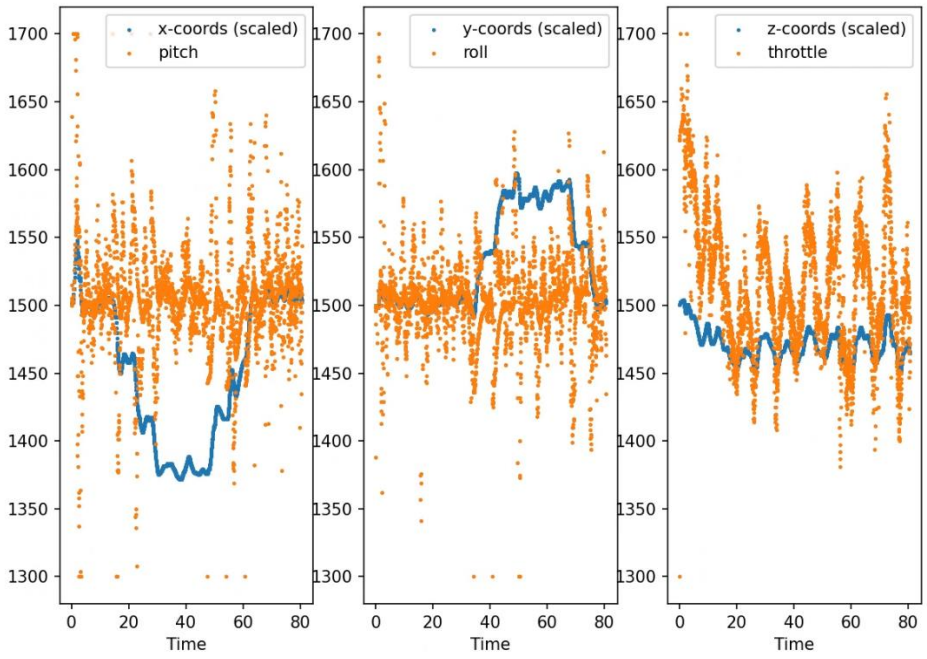
The PID coefficient parameters have been tested and tuned **[1]** for the specific scenarios and further Anti-Windup schemes **[2]** have been added to prevent integration wind-up when the throttle values are saturated. The Moving Average and Median Smoothing techniques for coordinate values have also shown significant reduction in the jitters obtained in the [x,y and especially in z] coordinate values obtained from the camera.



Controller performance during hovering without tuning and smoothing applied.



Controller performance during hovering with tuning and smoothing applied.



Controller performance during hovering in a rectangle with tuning and smoothing applied.

Swarming of Drone: -

For operating swarm of drones, we are creating a drone object from the `Communication.py` for establishing socket connection with the drones. The data containing set of commands for controlling drone is being transmitted using MSP via Wi-Fi. The tracking and communication thread is created at the initiation of the PID and then on the initiation of the `swarm.py` two threads for checkpoints accomplishment are being created which controls the drones.

Conclusion: -

The ArUCo tag on the drone is correctly identified and its position is properly estimated. The position is successfully transmitted from the estimator to the microcontroller on board. The communication script also successfully packages the data packets and transmits them to drone. Pluto 1.2 firmware identifies the MSP command received from the laptop via Wi-Fi and responds accordingly. This shows that the whole navigation & communication system works properly. We have also performed the tasks mentioned in task-2 such as hovering of drone at a fixed height and its movement in a figure of rectangle efficiently.

References: -

[1] https://in.mathworks.com/discovery/pid-control.html

[2] https://www.researchgate.net/publication/328593610_Performance_Improvement_of_Water_Temperature_Control_using_Anti-windup_Proportional_Integral_Derivative

[3] https://docs.python.org/3/library/socket.html

[4] http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol

[5] https://bit.ly/plutomsppackets