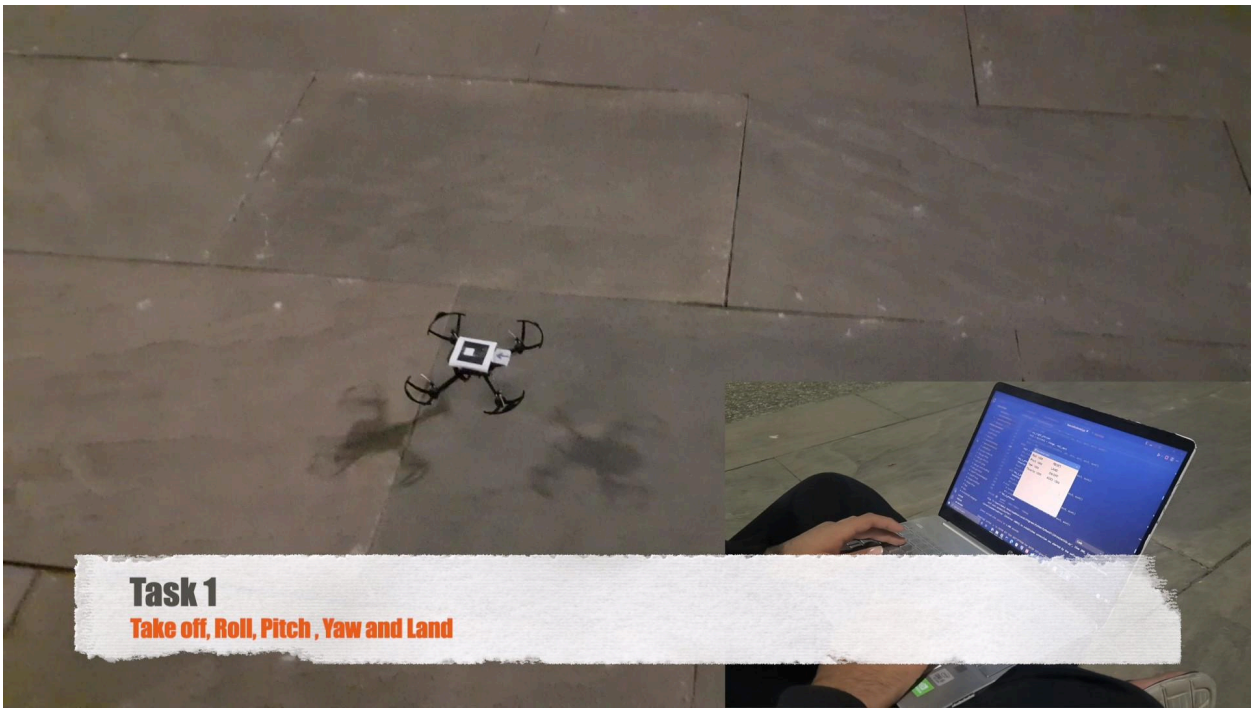
Inter IIT Tech Meet 11.0

PLUTO DRONE SWARM CHALLENGE

High Prep

By
Primary ID 49

1. Task-1



Keyboard Control of Drone using MSP

Objective:- Fly a drone using a Python wrapper that controls its movement like forward pitch, roll left, take off, land, pitch backward, roll right, clockwise yaw, counterclockwise yaw, throttle etc.

Code:- https://github.com/harsh-mandalia/TechMeet11/blob/main/task1.py

Our code is a python wrapper that implements an 8-key remote control for a PLUTO drone using Pygame and Telnet libraries. The code allows a user to control the drone's roll, pitch, throttle, and yaw by pressing keys on the keyboard. The code starts by defining the host (IP address of the drone) and port number for the Telnet connection. The Telnet library is used to establish a connection with the drone, which is then used to send commands to control the drone.

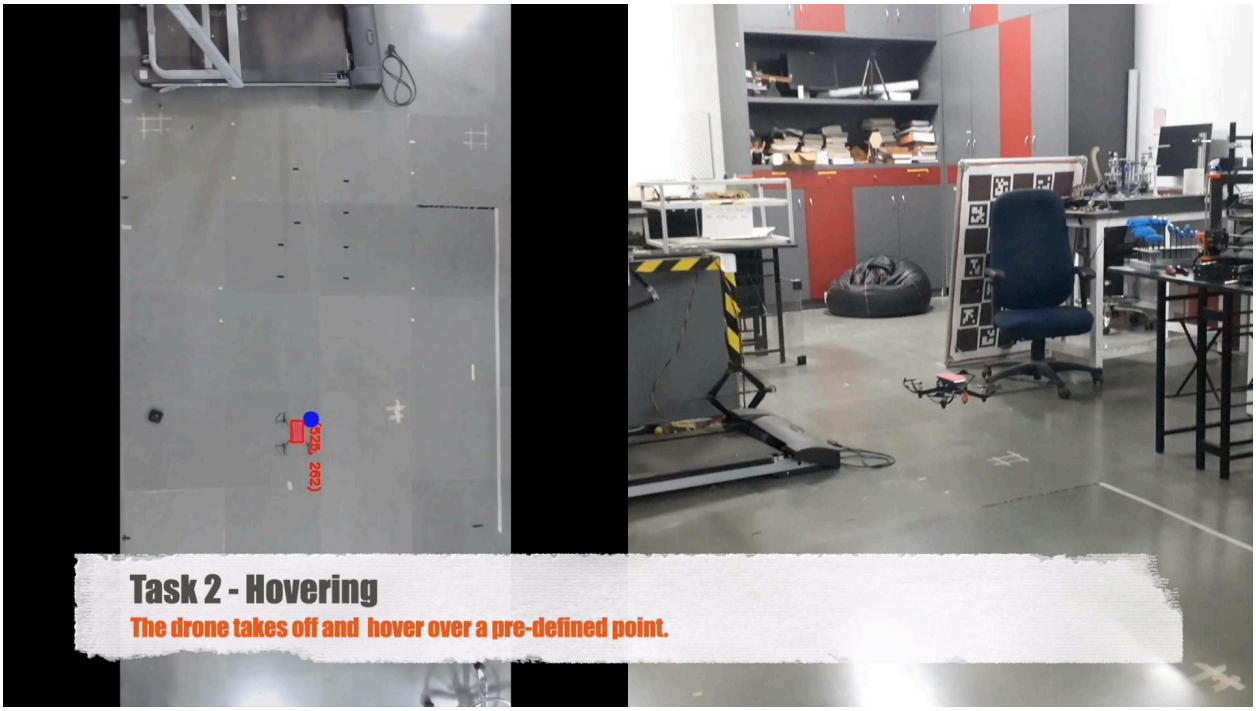
Next, the code defines three functions, add_checksum, rc and set. The add_checksum function takes a command and calculates its checksum by XORing all the bytes of the command except the first three bytes. The checksum is then appended to the command. The rc function takes eight arguments, which are the roll, pitch, throttle, yaw, and four aux values, and returns a bytes object representing the command to control the drone. It uses the *MSP_SET_RAW_RC* command to set roll, pitch, throttle, yaw, and four aux values. The function generates RC command for a drone by creating a bytearray of the RC command format, populating it with the given parameters, and adding a checksum to the command.

The set function in the above code is used to send a command to the flight controller. It takes one parameter which is the “data”. The "data" parameter is an integer that represents *MSP_SET_COMMAND*. The code then initializes Pygame, creates a window, and sets its caption. Four variables, roll, pitch, throttle, and yaw, are defined and set to their initial values. The variables have limits, and their values are updated based on the keys pressed by the user.

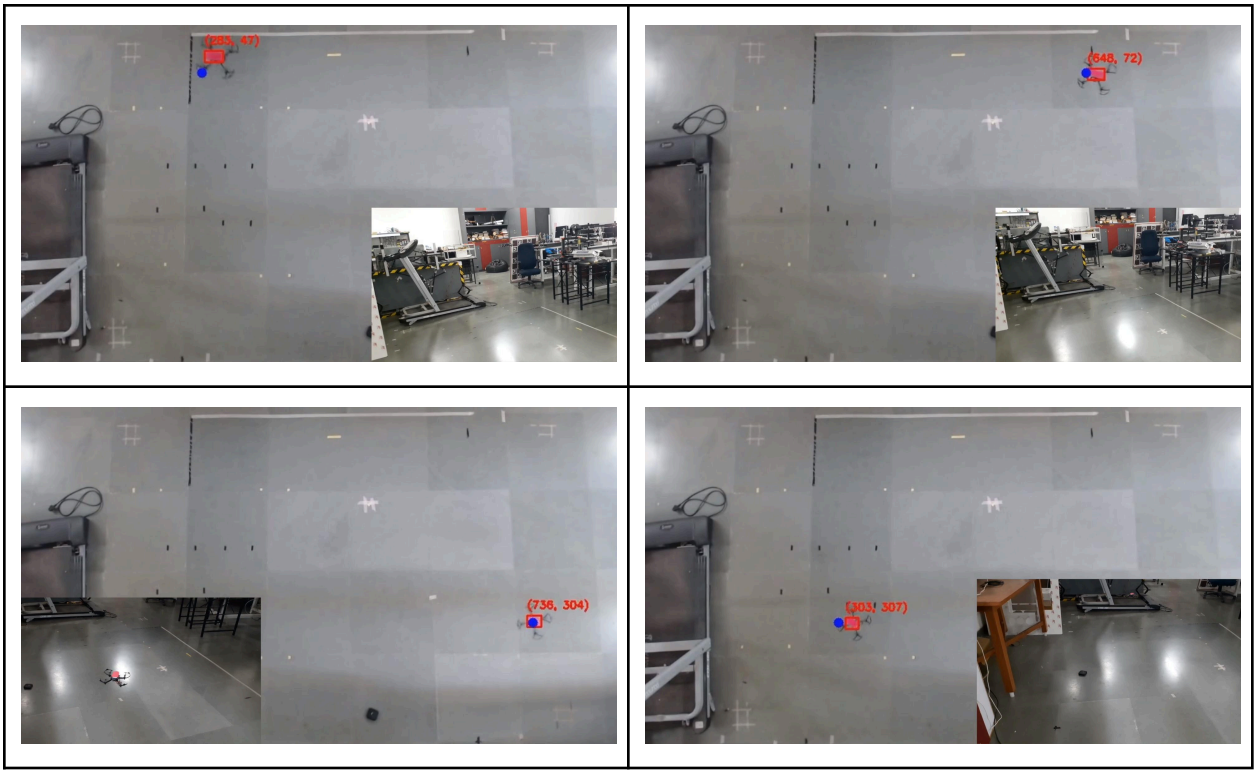
The wrapper then listens for events in a game loop. When a key is pressed, the corresponding flag is set to True, and when the key is released, the flag is set to False. The script also listens for specific key press events to control the drone. For example, if the *'r'* key is pressed, the script sends a command to reset the drone to its initial position. If the *'l'* key is pressed, the script sends a command to change the drone's mode. If the *'o'* key is pressed, the script sends a command to turn off the drone. The script also updates the values of the roll, pitch, throttle, and yaw based on the keys pressed. For example, if the up arrow key is pressed, the throttle value is increased. If the down arrow key is pressed, the throttle value is decreased. The updated values are then sent to the drone as a command to control its movement.

In conclusion, the code implements an 8-key remote control for a PLUTO drone using Pygame and Telnet libraries. The script allows the user to control the drone's roll, pitch, throttle, and yaw by pressing keys on the keyboard and sends the corresponding commands to the drone over a Telnet connection.

2. Task 2



Task 2.1: Hovering at a point using PID



Task 2.2: Rectangular motion using cubic trajectory

Objective:- Hovering a pluto drone on a particular height in one position using color detection and move the drone in rectangular motion (1X2 meter)

Code:- https://github.com/harsh-mandalia/TechMeet11/blob/main/task2.py

Our code is a Python Wrapper for controlling a drone's movements and performing image processing on the video feed received from the webcam on the ceiling. Here is a more detailed explanation of the code:

- **Importing Libraries:** The code starts by importing various libraries including OpenCV (cv2), NumPy (numpy), Telnet library (telnetlib), Matplotlib (matplotlib), and Pandas (pandas). These libraries are used for various purposes such as image processing, mathematical operations, connecting to the drone over Telnet, and working with data frames, respectively.
- **Setting up Telnet Connection:** The code then sets up a Telnet connection to the drone using the Telnet library. The IP address and port of the drone are specified in the host and port variables, respectively.
- **Defining Color Ranges:** The code then defines the range of red color in the HSV color space, which will be used to detect red color in the video feed.

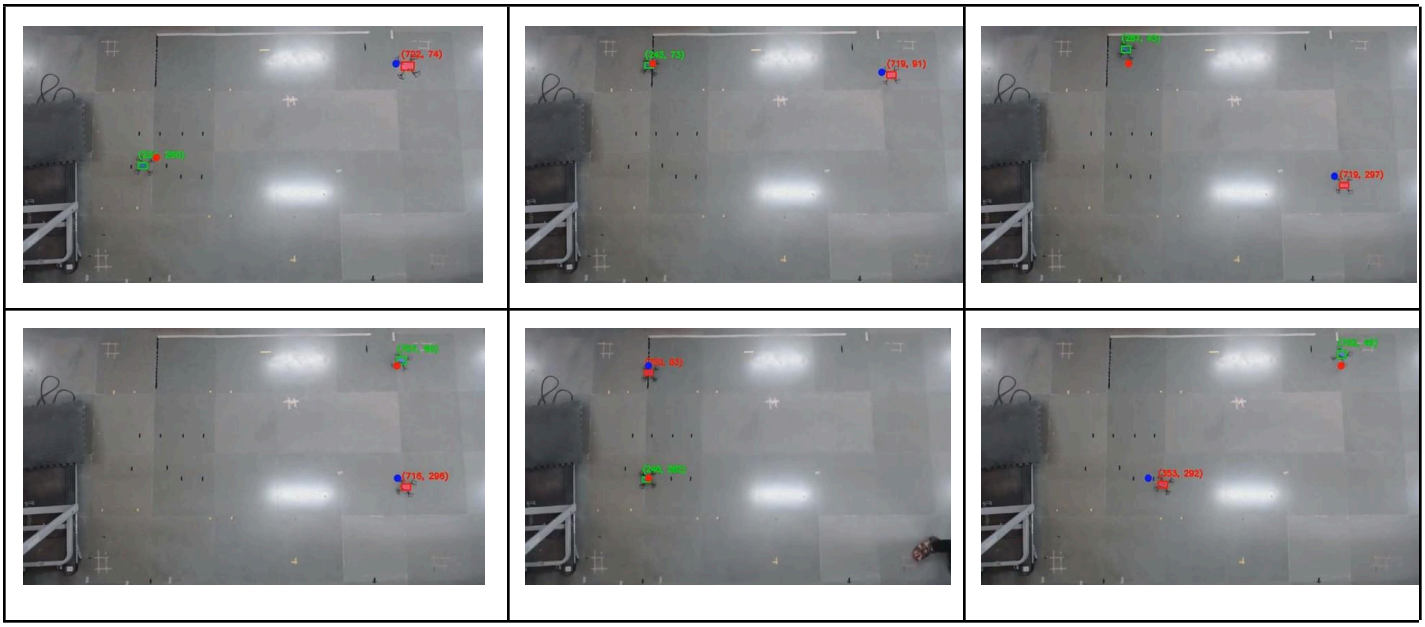
- ➔ **Setting up Video Capture:** The code sets up a video capture object using the OpenCV library and sets the frame width and height to 864 and 480, respectively. The size of the output video is also set using the VideoWriter function.
- ➔ **Initializing Variables:** The code then initializes a number of variables to store the error, position, and velocity information of the drone along the x, y, and z-axis.
- ➔ **Logging Drone Data**: The code creates a CSV file to log the drone data, which can later be used for analysis.
- ➔ ***distfunc():*** This function calculates the distance between two points in a 2D plane, given their x and y coordinates.
- ➔ ***add_checksum():*** This function calculates the checksum of a command and appends it to the command. The checksum is used to ensure that the command is not corrupted during transmission.
- ➔ ***rc():*** This function sends control commands to the drone over the Telnet connection. The function takes 8 parameters, roll, pitch, throttle, yaw, aux1, aux2, aux3, and aux4, which represent the control inputs for the drone. The function makes sure that the inputs are within the specified range, and if not, it limits them. The function then calculates the checksum of the command and sends it to the drone.
- ➔ **Main Loop:** The main loop in the code performs the tasks of capturing and processing images, computing control signals, and sending control signals to the drone. The PD controller and moving average filter are used to stabilize the drone and reduce noise in the control signals.The main loop in the code appears to be the while loop that starts with the line *"while True:"*. Within this loop, there are several tasks performed repeatedly in every iteration:
    - **Image capture and processing:** The code captures a video frame from the camera using the cv2.VideoCapture object 'cap', and then performs color thresholding to find red pixels.
    - **Detection and tracking of color markers**: The red pixels are detected using the *'cv2.inRange'* function, which filters out all pixels that are not within the specified range of specific color defined by say '*lower_red'* and '*upper_red'*. Then for tracking we use distance between different contours and initial contour.
    - **Computation of control signals**: The code implements a Proportional-Derivative (PD) controller for controlling the drone. The errors in the x, y, and z directions are computed and used as inputs to the PD controller. The outputs of the PD controller are the control signals for roll, pitch, throttle, and yaw.
    - **Cubic trajectory:-** The above code uses a cubic trajectory for generating a path. A cubic trajectory is a type of motion that follows a third-degree polynomial equation. In this case, it's used to generate the desired path for the drone to follow. The cubic trajectory is used to smooth the motion of the drone and reduce any sudden jerks or sudden changes in direction. The cubic trajectory calculates the desired path of the drone based on the starting and

ending points, and the rate of change at the start and end. This ensures a smooth and controlled motion of the drone, which is essential for safe and stable flight.

- **Moving average calculation**: The code implements a moving average filter, which computes the average of the past five values of x, y, and z errors, as well as their derivatives. These moving average values are used to reduce noise in the control signals.
- **Logging of drone data:** The code logs the drone data (x, y, z errors, control signals, etc.) in a CSV file named "data.csv".
- **Sending control signals to the drone**: The control signals are sent to the drone using the 'rc' function, which sends the control signals over the Telnet connection to the drone.

The code is a complex script that involves various aspects of drone control, image processing, and data analysis, and it requires a good understanding of these topics to make sense of it.

3. Task-3



Task 3: Two Drones Following Rectangle.

Objective:- Pluto Swarming - both the drones should fly at the same time. Drone2 will be at position0, and drone1 will be at position1 and drone1 from position1 to position2. When drone1 reaches position2, drone2 should follow drone1 and reach position1 automatically in the same rectangular(1X2) motion.

Code:- https://github.com/harsh-mandalia/TechMeet11/blob/main/task3_1.py

This code is an implementation of an image processing and control system, which is written in Python and uses the OpenCV library. The main objective of this system is to detect the color red and green in a video stream, and then perform control actions based on the detection of these colors. The video stream is acquired through a webcam, which is connected to the computer, and the resulting processed frames are written to an output video file. The color red is detected by defining a range of red colors in the HSV color space, and then filtering the frames in the video stream to only show pixels that fall within this range. Similarly, the green color is detected by defining a range of green colors in the HSV color space, and filtering the frames accordingly.

The system also includes a control loop that operates on the filtered frames to perform actions based on the presence or absence of the red and green colors. The control loop uses the Telnet protocol to communicate with two separate devices, drone and laptop, and sends commands to these devices based on the detected colors. The code also includes several global variables that are used to store the state of the system, such as the position and velocity of the two devices, as well as errors in the position and velocity. These variables are updated at each iteration of the control loop, and are used to calculate the next set of control commands that will be sent to the drone. In terms of the code structure, the main processing and control loop is contained within a while loop that continues to run until the user stops it. The loop performs the following steps at each iteration:

Acquire a new frame from the video stream. Convert the frame from the RGB color space to the HSV color space. Filter the frame to show only the pixels that fall within the defined ranges of red and green colors. Perform morphological operations on the filtered frame to improve the detection of objects. Find contours in the filtered frame, and determine the location of the centroid of each contour. Based on the location of the centroids, calculate the control actions that will be performed by the drone. Send the control actions to the devices via Telnet. Update the state of the system based on the control actions. Write the processed frame to the output video file. The code also includes several helper functions that are used to perform specific tasks, such as calculating the centroid of a contour, or sending a command via Telnet. For each frame, the x and y coordinates of the center of each object are smoothed using a moving average filter. The filter calculates the average of the last 5 frames to reduce noise and make the movements of the object smoother. Next, the code uses a proportional-derivative (PD) controller to track the center of the objects. The controller has two separate parts, one for x-axis control and another for y-axis control. The error between the desired position and the current position of the object is calculated and its derivative is also taken to incorporate velocity into the control. These error values are then filtered using another moving average filter, and the filtered errors are used to update the position of the objects.

In conclusion, this code provides a simple example of how image processing and control can be combined to perform a specific task, in this case detecting the color red and green in a

video stream and controlling two devices based on the detection. It demonstrates the use of several common computer vision techniques, such as color filtering, morphological operations, and contour detection, as well as the use of a control loop to perform actions based on the processed image data.

4. Conclusion:

All tasks have been completed successfully. The tracking accuracy can be improved by better tuning of gains.

$$x(t) = x_1 + 3(x_2 - x_1) \cdot \frac{t^2}{t_f^2} + 2(x_1 - x_2) \cdot \frac{t^3}{t_f^3}$$