The Hong Kong Polytechnic University
Department of Computing

COMP4913 Capstone Project
Report (Final)

CHENCW-05
Vision-based quadrotor gesture control

Student Name: Kwong Chun Him

Student ID No.: 21028468D

Programme-Stream Code: 61431-SYC

Supervisor: Prof. CHEN Changwen

Co-Examiner: Dr YANG Bo

2nd Assessor: Prof. YOU Jane

Submission Date: 11 April 2023

Abstract

The use of gesture control for quadrotor drones has become an area of interest due to its natural and intuitive nature. This study presents a gesture control system for a quadrotor drone using Mediapipe. This machine learning framework detects hand gestures by recognizing landmarks 20 hands The system then translates these gestures into commands that enable users to control the drone's movements. Additionally, the system incorporates a face-tracking function to maintain a consistent distance between the drone and the user. While the system is easy to use and requires minimal training, it has limitations such as limited range and dependence on the external equipment. This paper reviews the current state of the art in vision-based quadrotor gesture control, including the system architecture, gesture recognition methods, and control algorithms. The studies also incur users, which showed a high recognition rate for hand gestures and reliable tracking of the user's face. This system can be applied in various fields, including surveillance, search and rescue operations, and entertainment. Finally, the paper discusses the challenges and opportunities for further research.

Acknowledgement

I would like to express my sincere gratitude to Prof. CHEN Changwen for his generous support and approval of the funding for the purchase of the DJI Tello drone that was crucial for the successful completion of this project.

I am also deeply grateful to Xiaotong for his invaluable guidance and expertise throughout the entire process, his dedication and patience were essential in helping me navigate the challenges and complexities of this research.

Lastly, I would like to thank everyone who took the time to read and review this report, your feedback and suggestions were instrumental in shaping the outcome.

Table of Contents

List of figures and tables

1. Introduction

1.1 Background and Problem Statement

Gesture control is a technology that enables users to interact with devices and machines through body movements, without requiring physical contact or traditional input methods such as keyboards or touchscreens. This technology has been gaining popularity in recent years due to its potential to enhance user experience and improve accessibility for individuals with disabilities. One particular application of gesture control is in the field of quadrotors, which are flight machines that use four motors to change the angular speed of the propellers, thereby controlling thrust and torque. However, despite the versatility of quadrotors, they can be challenging to operate, requiring specific remote controls or smartphone applications, and extensive training. This limits their potential usefulness in a wide range of scenarios, including search and rescue missions and other situations where untrained individuals need to control the quadrotor.

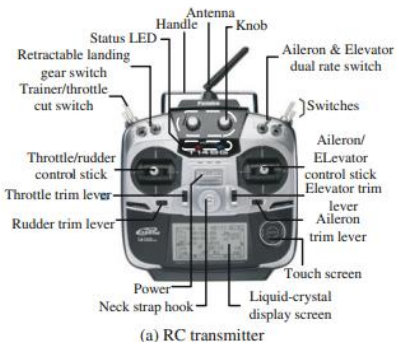


Figure 1 quadrotor remote controller [1]

The problem addressed in this research is the difficulty of operating quadrotors and the limited versatility of quadrotor technology due to these challenges. Specifically, the researchers seek to address the issue of how to make quadrotors more accessible and easier to operate for untrained individuals. One potential solution to this problem is visual-based gesture control, enabling users to control quadrotors through intuitive body movements. This technology can potentially improve quadrotor accessibility, expand its usefulness in various scenarios, and reduce the need for extensive training. However, implementing visual-based gesture control for quadrotors presents several challenges, such as ensuring the accuracy and reliability of the system, accommodating different

lighting conditions and user preferences, and ensuring the safety of both the quadrotor and the surrounding environment. The research aims to address these challenges and develop a practical and effective visual-based gesture control system for quadrotors.

1.2 Objectives and Outcome

1.2.1 Objective:

The objective of this project is to develop a user-friendly control system for quadrotors that allows individuals without quadrotor training to easily control the quadrotors using hand gestures instead of a remote control. The system will rely on the quadrotor camera to detect the user's gestures and translate them into quadrotor movements, such as turning, ascending, descending, and moving to a specific position.

1.2.2 Outcome:

The expected outcome of this project is a closed-loop control system that incorporates face and gesture detection functions to track the user's position in real-time and enable smooth and precise quadrotor operation. The system will use the face detection function to keep the camera focused on the user's face, even as the user moves or rotates around the quadrotor, and the gesture detection function to control the quadrotor's movements. All actions should have a response time of no more than 0.5 seconds. Although the system is not expected to match the precision of quadrotor pilots with extensive training, it will provide an intuitive and accessible alternative for those with no prior experience.

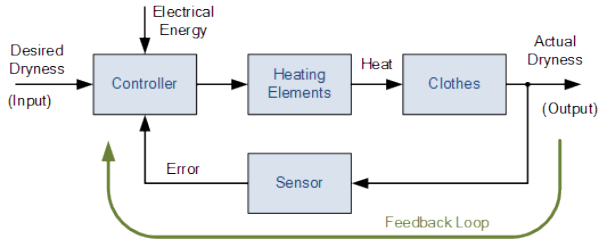


Figure 2 Flow chart of the close-Loop system [2]

2. Methodology

2.1 Function Requirements

This part discusses the functions needed to realize this visual-base quadrotor gesture control.

2.1.1 Gesture recognition

The gesture recognition part is responsible for recognizing the hand or body gestures made by the user and translating them into commands for the quadrotor. Here are some of the function requirements of gesture recognition in Vision-based quadrotor gesture control:

Real-time recognition: The gesture recognition system must be able to recognize gestures in real-time to ensure the quadrotor responds quickly to the user's commands. The system should have a low latency and be able to process the input data quickly.

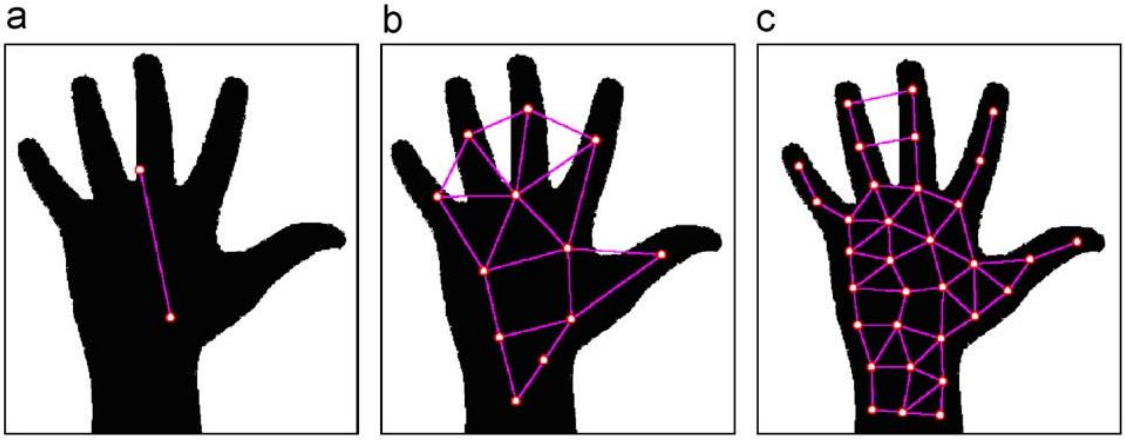
Accuracy: The gesture recognition system must have a high accuracy in recognizing gestures. The system should be able to distinguish between different gestures with high precision and recall.

Multi-gesture recognition: The gesture recognition system should be able to recognize multiple gestures and map them to specific commands for the quadrotor. The system should be able to handle complex gestures and gestures made in sequence.

Training: The gesture recognition system should be trainable, meaning it should be able to learn new gestures and commands. This involves collecting data of new gestures, labeling the data, and retraining the system to recognize the new gestures.

Integration: The gesture recognition system should be able to integrate with the quadrotor control system. This involves mapping recognized gestures to specific commands for the quadrotor, such as takeoff, land, move forward, move backward, etc.

Overall, the gesture recognition component of Vision-based quadrotor gesture control plays a critical role in allowing the user to control the quadrotor using hand or body gestures. It must be able to recognize gestures in real-time, be robust and accurate, handle multiple gestures, and integrate with the quadrotor control system.

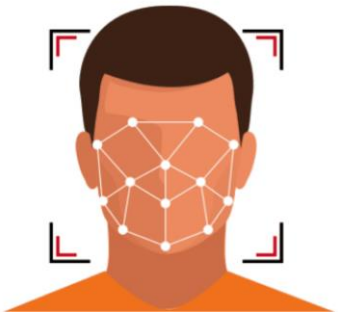


2.1.2 gesture control

The project's primary purpose is to use gesture recognition to control the quadrotor to replace the traditional GUI, or quadrotor, remote controller. The quadrotor's camera recognizes gestures and sends commands to the quadrotor. Based on the interpreted gestures, the software must generate control signals that can be used to control the movement of the quadrotor. This may involve calculating the quadrotor's desired acceleration, velocity, or position and any necessary control inputs such as pitch, roll, or yaw.

Once the hand gestures are recognized, the flight commands are sent to the quadrotor to control its motion. This is done using a proportional-integral-derivative (PID) controller, which adjusts the motor speeds of the quadrotor based on the flight commands received.

2.1.3 Face Detection



The Vision-based quadrotor gesture control system uses face detection as an important component for several reasons. One of the primary reasons is that face detection allows the system to identify the user and track their movements, which is critical for accurate gesture recognition and control.

User identification: Face detection allows the system to identify the user and distinguish them from other objects in the video stream, such as hands or backgrounds. This is important because the system needs to know who is controlling the quadrotor to ensure that the correct commands are being sent to the quadrotor. Here are some of the reasons why the Vision-based quadrotor gesture control system needs face detection:

Gesture recognition: Face detection provides an important reference point for analyzing hand or body gestures. By tracking the location of the user's face, the system can determine the position and orientation of the user's hands or body, which is important for accurately recognizing gestures.

2.1.4 Image processing

The images captured by the camera are processed to extract the hand gestures made by the user. This is done using computer vision techniques such as thresholding,

contour detection, and feature extraction. The position of the hand in the image is determined, and the hand gestures are classified based on the position and shape of the hand.

2.1.5 Face Track

Face tracking can be used as a part of a vision-based quadrotor gesture control system to improve the accuracy and reliability of gesture recognition. By tracking the user's face, the system can focus on the relevant area and ignore other distractions in the background. This can make it easier for the system to recognize the gestures made by the user accurately. When the quadrotor detects a face, the quadrotor can track the face to prevent the user from leaving the camera area and losing control of the quadrotor.

To implement face tracking, the system would need to use a camera or other imaging sensor to capture images or video of the user and then use computer vision algorithms to detect and track the user's face. The system could then use the position and orientation of the face to determine the gestures made by the user and map these gestures to specific control commands for the quadrotor.

2.2 Hardware requirement

In this part, we will focus on explaining which quadrotor platform to choose:

Figure 3 Photo of DJI Tello Drone

The DJI Tello drone is a popular choice for the Vision-based quadrotor gesture control project for several reasons:

Affordability: The DJI Tello drone is an affordable option, which makes it a great choice for research and development projects. It is significantly cheaper than other DJI drones and has a similar feature set, making it a good value for the price.

Lightweight and Portable: The DJI Tello drone is lightweight and portable, making it easy to transport and fly in a variety of environments. This is important for a project like Vision-based quadrotor gesture control, which requires the drone to be flown indoors in a confined space.

Stable Flight: The DJI Tello drone is designed to provide stable flight and hover, which is important for a project like Vision-based quadrotor gesture control, where the drone needs to be precisely controlled to follow the user's commands.

Camera Quality: The DJI Tello drone has a built-in camera that provides high-quality video streaming and recording capabilities. This is important for a project like Vision-based quadrotor gesture control, where the camera is used to capture the user's hand and body gestures and send them to the gesture recognition system.

SDK Support: The DJI Tello drone has a software development kit (SDK) that provides access to the drone's flight control system and camera, which is important for integrating the drone into the Vision-based quadrotor gesture control system. The SDK allows developers to write custom software to control the drone and access the camera feed, which is critical for this project.

Overall, the DJI Tello drone is a great choice for the Vision-based quadrotor gesture control project because of its affordability, lightweight and portable design, stable flight, high-quality camera, and SDK support. It provides a good balance between functionality and cost, which makes it a popular choice for research and development projects like this one.

2.3 system design

The drone is equipped with a front-facing camera that sends live video feed and receives commands over Wi-Fi. The video feed is transmitted to an off-board computer, which is responsible for processing the image and providing instructions to the drone. All calculations and image processing are performed by the computer, and then the instructions are sent back to the drone. In essence, the drone's camera sends the video to the computer, which analyzes it and sends back instructions to control the drone's movements.

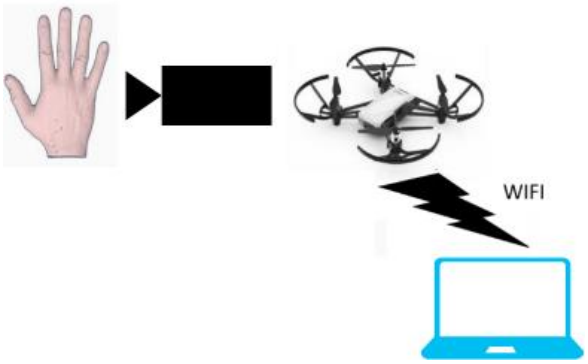


Figure 4 Conceptual Framework of gesture control

2.3.1 Flow chart diagram

Suppose the quadrotor was powered on and the computer was connected to the quadrotor through the Wi-Fi networks.

Gesture Control

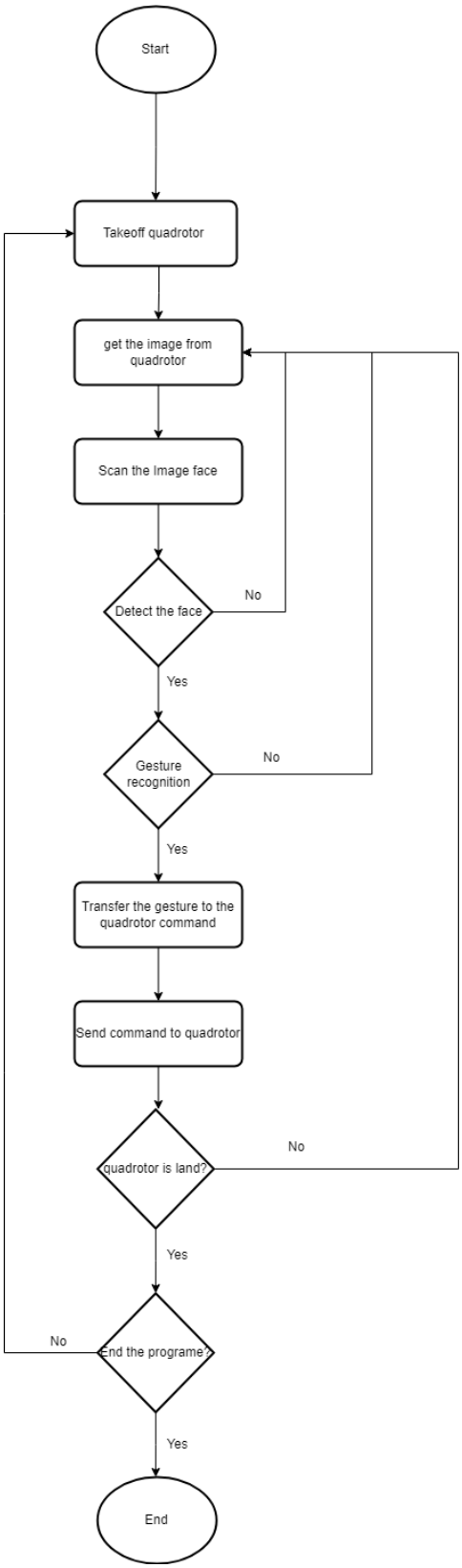


Figure 5 Flow chart of gesture control

Face Tracking

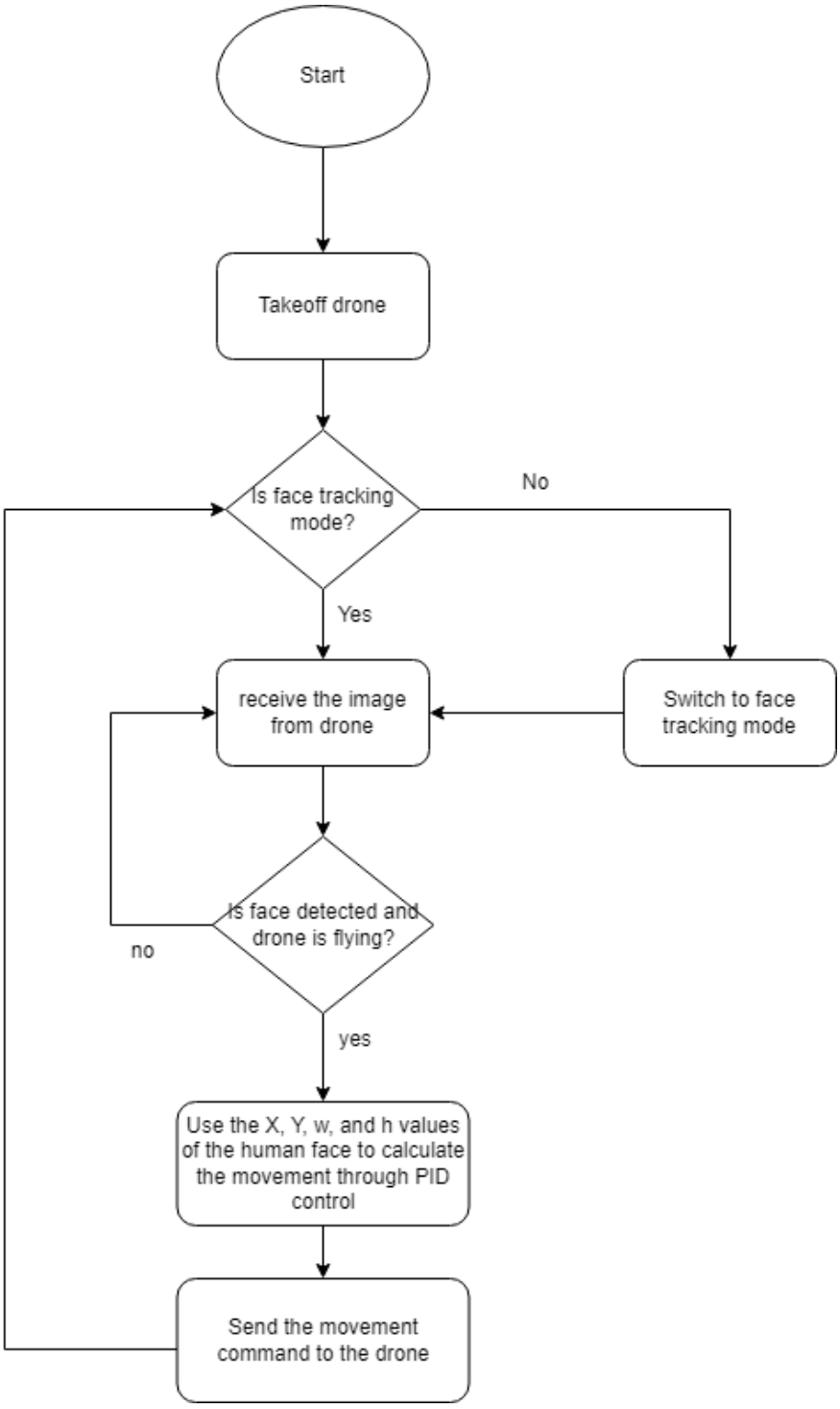


Figure 6 Flow chart of face tracking

2.3.2 sequences diagram

The concept with connect the quadrotor through WI-FI:

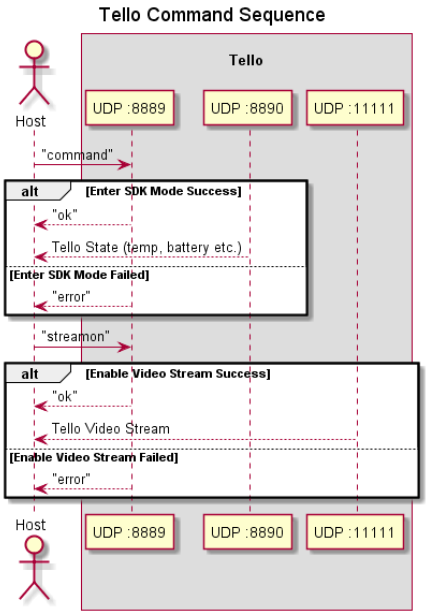


Figure 7 Sequences diagram of connect quadrotor

Suppose the quadrotor was powered on and the computer was connected to the quadrotor through the Wi-Fi networks; then the takeoff command is sent to the quadrotor.

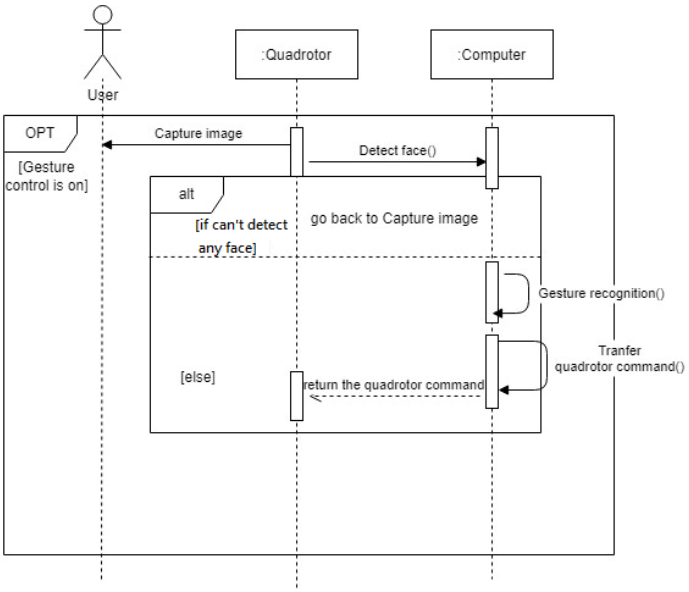


Figure 8 sequences diagram of gesture control

This diagram explains that after the quadrotor captures the image, the image is sent to a computer program for processing, identifying, and determining the type of gesture, then sending the corresponding information to the quadrotor.

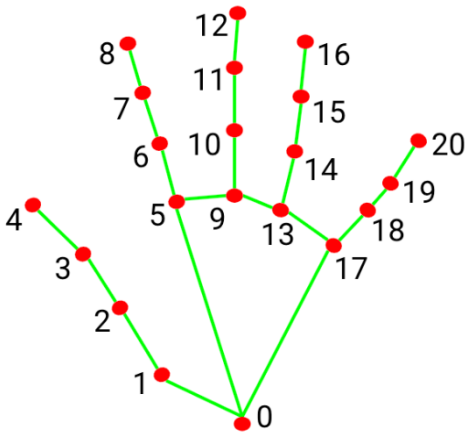
3. Implementation

3.1 Gesture recognition

This project implements gesture recognition through Mediapipe Hands. MediaPipe Hand is a framework developed by Google that is designed to enable the creation of hand-tracking and hand-gesture recognition applications. It uses machine learning to track the movements and gestures of a person's hands in real-time, using data from a camera or other sensor. MediaPipe Hands can be used in various applications, including virtual reality, augmented reality, and interactive displays. It is intended to be easy and flexible, allowing developers to quickly build and deploy hand-tracking recognition applications.

3.1.1 Technical details of gesture detector

Twenty-one hand landmarks are being returned in three dimensions using Mediapipe Hand Keypoints Recognition. We shall employ 2D coordinates for our model.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 9 hand landmarks [3]

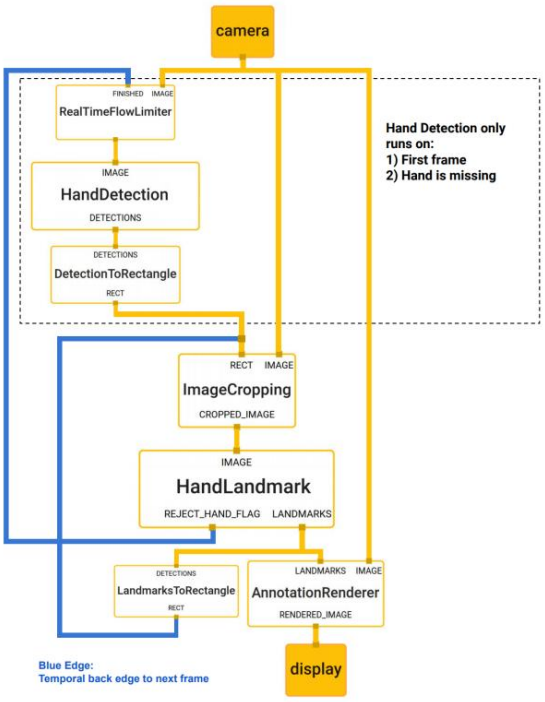


Figure 10 The hand landmark models output controls when the hand detection model is triggered. This behavior is achieved by MediaPipes powerful synchronization building blocks, resulting in high performance and optimal throughput of the ML pipeline. [4]

3.1.2 Implementation in interim report

These points are preprocessed for training the model in the following way.

(Landmark coordinates)

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[551, 465]	[485, 428]	[439, 362]	[408, 307]	……	[633, 315]	[668, 261]	[687, 225]	[702, 188]

(Convert to relative coordinates from ID:0)

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[0, 0]	[-66, -37]	[-112, -103]	[-143, -158]	……	[82, -150]	[117, -204]	[136, -240]	[151, -277]

(Flatten to a one-dimensional array)

ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-66	-37	-112	-103	-143	-158	……	82	-150	117	-204	136	-240	151	-277

(Normalized to the maximum value(absolute value))

ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-0.24	-0.13	-0.4	-0.37	-0.52	-0.57	……	0.296	-0.54	0.422	-0.74	0.491	-0.87	0.545	-1

Figure 11 Dataset sample for model train

After that, we can use the processed data to train a deep learning model on the extracted features with TensorFlow. A keypoint classifier is a simple Neural network with such a structure.

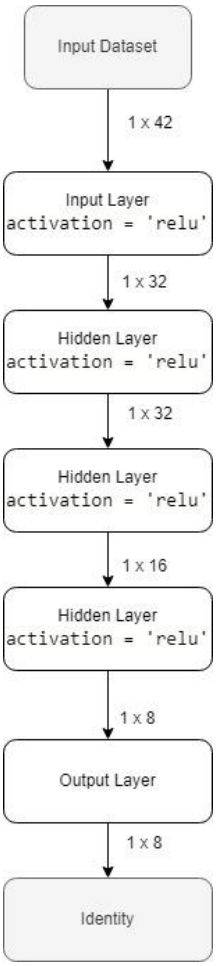


Figure 12 Neural network structure

Evaluate the model

Evaluate its performance on a held-out dataset to ensure that it's able to generalize to new examples.

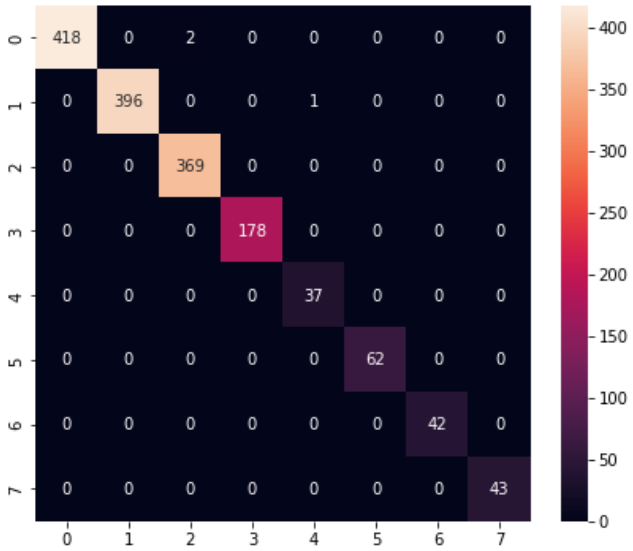


Figure 13 Confusion matrix

	precision	recall	f1-score	support
0	1.00	1.00	1.00	420
1	1.00	1.00	1.00	397
2	0.99	1.00	1.00	369
3	1.00	1.00	1.00	178
4	0.97	1.00	0.99	37
5	1.00	1.00	1.00	62
6	1.00	1.00	1.00	42
7	1.00	1.00	1.00	43
accuracy			1.00	1548
macro avg	1.00	1.00	1.00	1548
weighted avg	1.00	1.00	1.00	1548

Figure 14 Classification Report

The graphs and tables show that the model is very accurate. Using this model, we can accurately predict the gestures and send the corresponding commands to the quadrotor. In the interim report, I also used this function to successfully realize gesture recognition.

3.1.3 Final implementation

In the mid-term report, I used TensorFlow to train the model to recognize gestures, and it was accurate enough. Since my desktop computer doesn't have Wi-Fi, I used my M1 MacBook to implement this project. However, I found that TensorFlow doesn't yet support CPU on the M1, so I looked for another way to implement gesture recognition. I decided to use a simpler approach by analyzing the data returned by Mediapipe to determine the number of fingers that are raised, and then using the number of raised fingers to recognize the corresponding gesture and send the appropriate command to the drone. here are some advantages of this approach:

- Although this approach has a limit on the number of gestures that can be defined, it is sufficient for controlling the drone.
- The approach can be more easily customized to specific gestures and use cases, as the rules for recognizing gestures can be defined based on the specific requirements of the application. I can easily add or remove different gesture commands without needing to train a new model.
- The approach may be more robust to changes in lighting or other environmental conditions, as it relies on finger tracking data rather than image classification

Example:

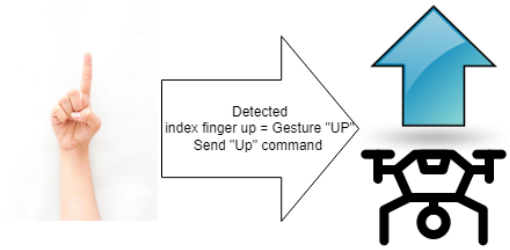


Figure 15 sample of gesture recognition

After the hand is detected, surround it in a rectangular frame and draw 20 landmarks to extract features. These features are then used to recognize gestures, and if a gesture is detected, it is displayed in the upper left corner of the screen. The following is an example:

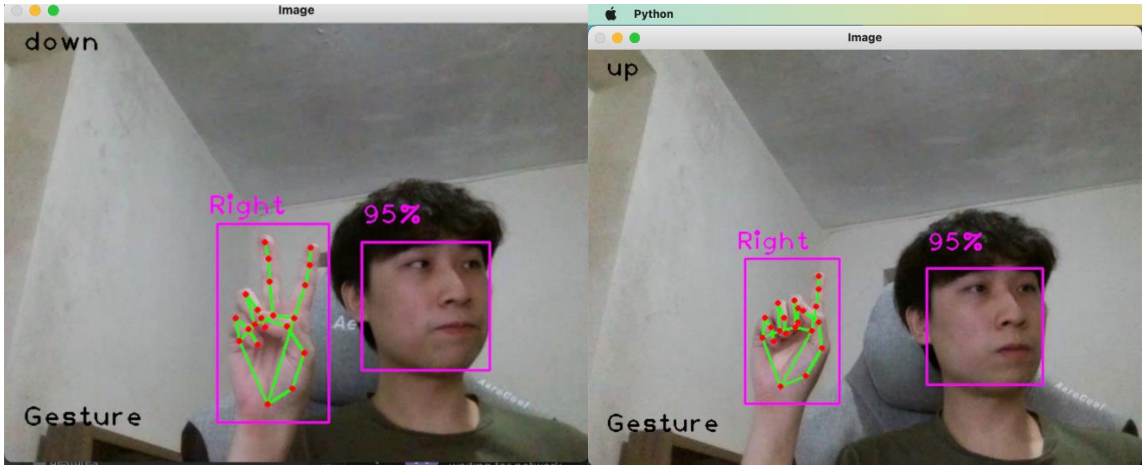


Figure 16 demo of gesture recognition

3.2 Gesture control

This part will detail of “GestureControl” function. The following is the main process:

1. The function receives three parameters: img (an image), bboxs (a list of bounding boxes), and allHands (a list of dictionaries with information about detected hands).
2. The function checks if the bboxs list is not empty.
3. If the bboxs list is not empty, the function checks if the allHands list is not empty.
4. If the allHands list is not empty, the function checks if the first detected hand is a right

hand.

5. If the first detected hand is a right hand, the function calls the fingersUp method of a detectorHands object passing the allHands list as an argument. This method returns a list of binary values indicating whether each finger is up (1) or down (0).
6. The function checks the value of the fingers list and sets the gestureText variable and adds the corresponding gesture to a gesture_buffer object.
7. The function calls the gesture_control method of a gestureController object passing the detected gesture ID as an argument.
8. If the first detected hand is a left hand, the function repeats steps 7 to 10 with a different set of gestures.
9. The function calls the gesture_buffer.get_gesture() method to get the ID of the oldest gesture in the buffer.
10. The function calls the gestureController.gesture_control() method passing the ID of the common gesture in the buffer as an argument.

The Flow chart of “GestureControl” function:

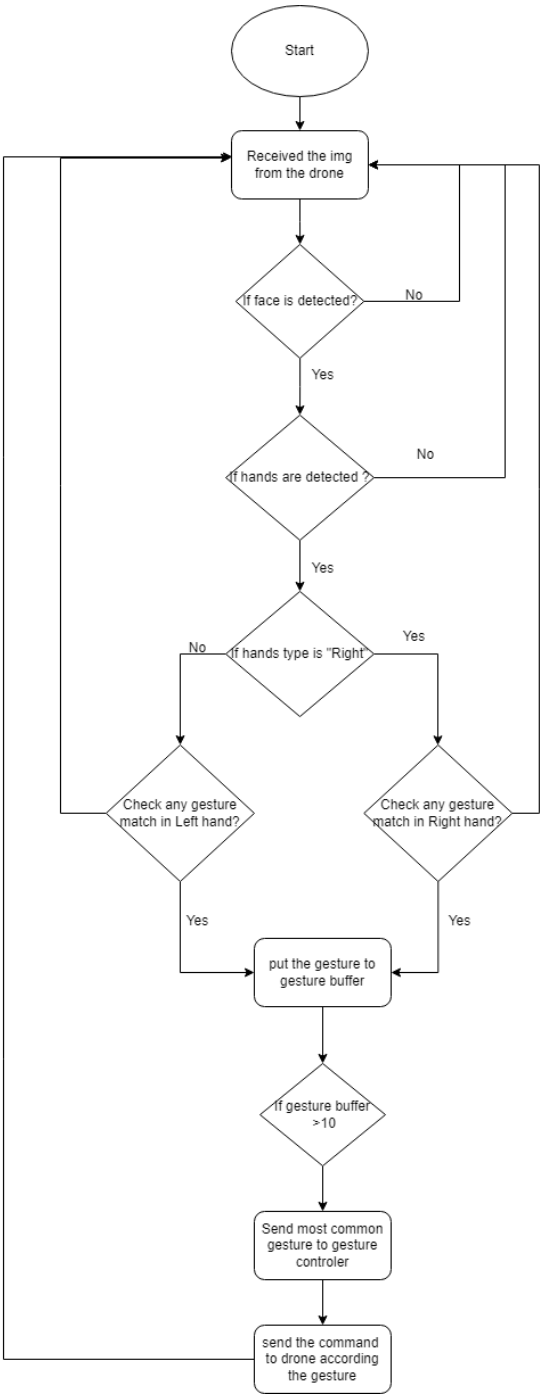


Figure 17 Flow chart of “GestureControl”

Pseudo code of Gesture Buffer:

```
FUNCTION add_gesture(gesture_id)
    APPEND gesture_id TO _buffer
END FUNCTION
```

```
FUNCTION get_gesture()
    COMPUTE counter AS Counter(_buffer).most_common()
    IF counter IS NOT EMPTY THEN
        IF counter[0][1] >= (buffer_len - 1) THEN    # buff_len=10, check is the buff full?
            CLEAR _buffer
            RETURN counter[0][0]
        ELSE
            RETURN NONE
        END IF
    ELSE
        PRINT ""    # don’t do anything
    END IF
END FUNCTION
```

Then we will explain the purpose of the gesture buffer; This is done to ensure that the gesture controller module can access the most up-to-date set of gestures when making decisions based on the current state of the hand movements. The gesture controller module reads the gesture buffer and determines which action to take based on the current set of gestures.

In other words, the gesture buffer updates the current state of the hand movements and allows the gesture controller to adapt to changes in hand movements over time. With a gesture buffer, the gesture controller might make decisions based on outdated information, leading to correct actions being taken.

Tello Commands

Control Commands

Command	Description	Possible Response
Command	Enter SDK mode.	ok / error
takeoff	Auto takeoff.	
land	Auto landing.	
streamon	Enable video stream.	
streamoff	Disable video stream.	
emergency	Stop motors immediately.	
up x	Ascend to "x" cm. x = 20-500	
down x	down "x" Descend to "x" cm. x = 20-500	
left x	Fly left for "x" cm. "x" = 20-500	
right x	Fly right for "x" cm. "x" = 20-500	
forward x	Fly forward for "x" cm. "x" = 20-500	
back x	Fly backward for "x" cm. "x" = 20-500	
cw x	Rotate "x" degrees clockwise. "x" = 1-360	
ccw x	Rotate "x" degrees counterclockwise. "x" = 1-360	
flip x	Flip in "x" direction. "l" = left "r" = right "f" = forward "b" = back	
go x y z speed	Fly to "x" "y" "z" at "speed" (cm/s). "x" = -500-500 "y" = -500-500 "z" = -500-500 "speed" = 10-100 Note: "x", "y", and "z" values can't be set between -20 – 20 simultaneously.	

Figure 18 DJI Tello Command

Another reason is that DJI TELLO provides an SDK for the movement, such as up x, down x, left x, and right x. When we send the command to the Tello drone, it cannot receive any command before returning the response "OK." However, due to the drone's design, the visibility conditions, such as low light, will affect the precision and performance [5]. If we use a command that requires a waiting return response, a "NO VALID IMU" error will occur, so I can only use the RC command to transmit the command to the drone unilaterally. Since this is a command that does not need to wait for a response if a gesture is misjudged, the wrong command will be sent out, so it is necessary to use the buffer to control it.

3.3 Face Detect

I implement face detection through mediapipe. MediaPipe Face Detection is a fast and accurate technology for detecting faces. It uses a lightweight face detection model called BlazeFace that works well on mobile devices. BlazeFace can detect faces with six landmark points and can detect multiple faces at once. This technology can be for various tasks that require detecting facial features, such as 3D facial keypoint estimation, facial

expression recognition, and face segmentation. BlazeFace uses a special method that makes it faster and more efficient than other face detection models. For more add information about BlazeFace resource are available [6].

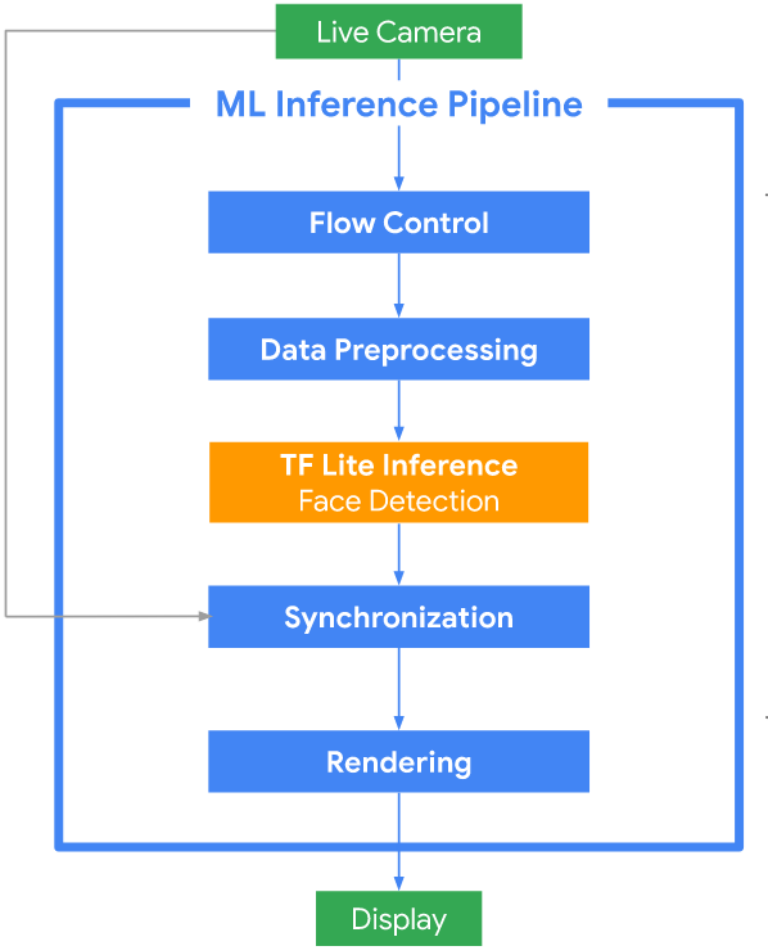


Figure 19 flow chart of mediapipe face detection [7]

If faces are detected in the image, the method loops through each detected face, extracts its bounding box information, and stores it in a dictionary "bboxInfo". The bounding box is also appended to a list "bboxs". If the draw flag is True, the bounding box is drawn on the output image using "cv2.rectangle()", and a text indicating the confidence score of the face detection is added using "cv2.putText()". The following is an demo:

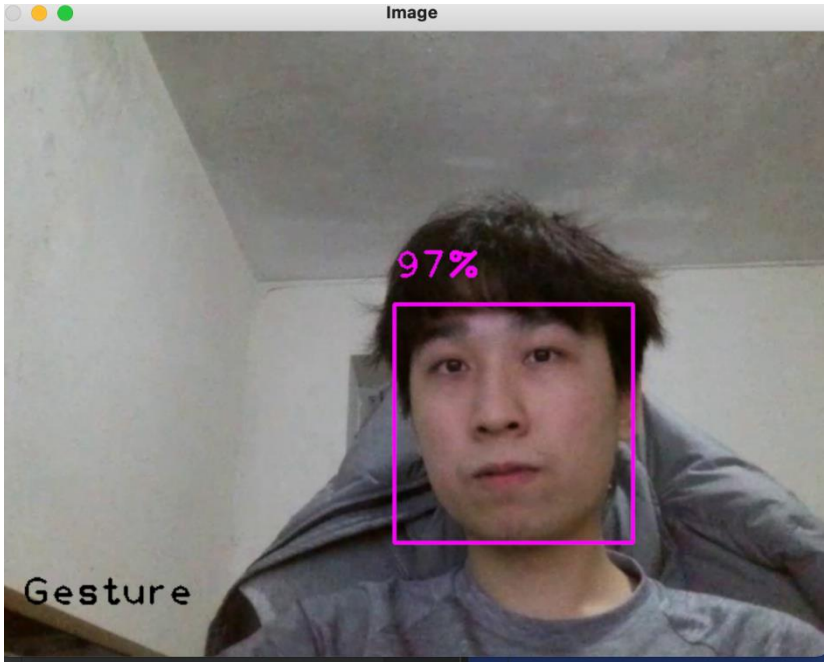


Figure 20 face detection demo

3.4 Face tracking

After detected the face, here's how to implement face tracking with drones:

1. Calculate the center point of the detected face in the video frame. calculate the center point of the bounding box by adding half of the width and height to the x and y coordinates, respectively.
2. Define a PID controller that takes the center point of the detected face as the input and adjusts the drone's movements to keep the face centered in the video frame. The PID controller should have three components: a proportional (P) component, an integral (I) component, and a derivative (D) component.
3. The P component of the PID controller should adjust the drone's left/right and up/down movements based on how far the center point of the face is from the center of the video frame. For example, if the center point of the face is to the right of the center of the video frame, the drone should turn to the right to keep the face centered.
4. The I component of the PID controller should adjust the drone's movements to

correct for any systematic errors in the P component's adjustments. For example, if the drone tends to overshoot when it tries to center the face, the I component should adjust the movements to compensate for this.

5. The D component of the PID controller should adjust the drone's movements based on how quickly the center point of the face is moving. For example, if the center point of the face is moving rapidly to the right, the D component should adjust the drone's movements to counteract this and keep the face centered.
6. Tune the PID controller parameters to optimize the drone's performance. You can do this by trial and error, adjusting the parameters until you achieve the desired behavior.
7. Finally, use the PID controller to adjust the drone's movements in real-time based on the center point of the detected face.

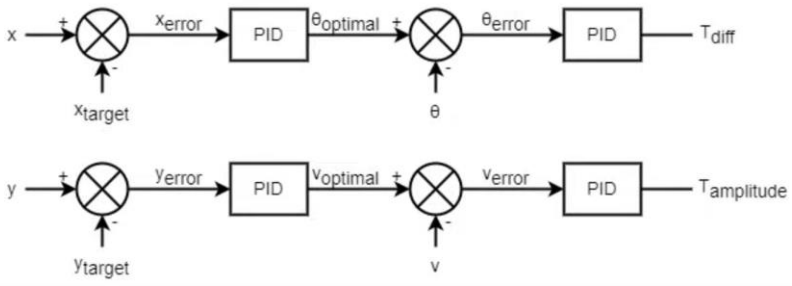


Figure 21 concept of PID control

For example, if the previous left-position was 30 in the x-coordinate and the new left-position is 40 in the x-coordinate, the code will detect that the face has moved to the right. This will result in the drone moving towards the right as well. The drone will move in other directions based on the changes in coordinates of the face's movement.

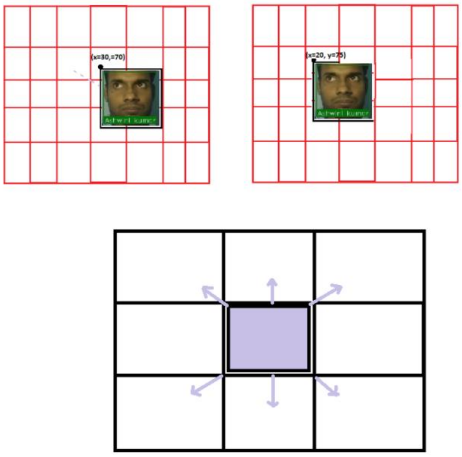


Figure 22 example of face fracking [8]

A live plot function was developed for testing and tuning the parameters of a PID (Proportional-Integral-Derivative) controller. The values of X, Y, and Z approaching zero indicate greater stability.

TargetValue of X: width // 2 (center point of X)

TargetValue of Y: Height // 2 (center point of Y)

TargetValue of Z: 12000 (the value of x * y that means the size of rectangle, if the size to big that means too close, if too small that means too far)

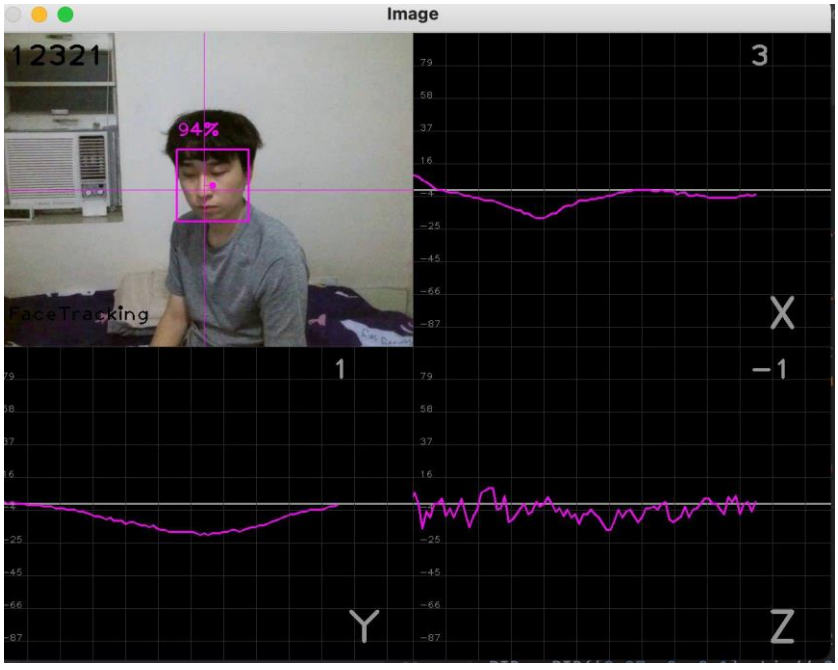


Figure 23 live plot function for testing PID parameters

As a result, this system is developed as a closed-loop system. The face-tracking algorithm continuously monitors the position of the detected face and adjusts the drone's movements in real-time to keep the face centered in the video frame. This closed-loop system ensures the drone maintains a stable position relative to the face, even in external disturbances.

In addition to face tracking, a similar closed-loop system can be implemented for drone gesture control. The system can detect and recognize specific hand gestures using computer vision techniques such as deep learning and use a PID controller to adjust the drone's movements in real time based on the detected gestures. The PID controller can adjust the drone's pitch, roll, and yaw movements to maintain a stable position relative to the hand gesture, ensuring accurate and responsive control.

Overall, the implementation of closed-loop systems for face tracking and gesture control enables precise and stable drone movements, improving the user experience and expanding the potential applications of drone technology. Future work in this area could focus on optimizing the PID controller parameters and developing more advanced computer vision algorithms for more accurate and reliable face tracking and gesture recognition.

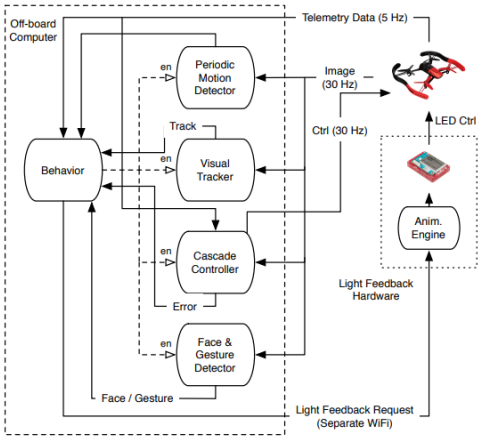


Figure 24 Sample block diagram of the System [9]

4. Improvement

4.1 Face recognition

Currently, my quadrotor can be controlled by anyone due to the lack of face recognition implementation. To address this issue, the next step would be to implement face recognition and train a model that can recognize a specific individual, allowing only authorized users to operate the quadrotor.

Facial recognition implementation typically involves utilizing machine learning algorithms to analyze and compare patterns in facial images. Here are the essential steps for developing a facial recognition system [10]:

1. Collect a dataset of facial images, which could consist of images of one person or a collection of pictures of multiple people.
2. Preprocess the images, such as cropping the photo to focus on the face, and resizing or rotating the pictures to ensure proper alignment.
3. Extract features from the images, which may involve identifying key facial points (such as the eyes, nose, and mouth) using algorithms and creating a numerical representation of the face based on these points.
4. Train a machine learning model on the dataset, potentially utilizing a supervised learning algorithm to classify faces based on the extracted features.
5. Test the model on a separate dataset to evaluate its accuracy in recognizing faces.
6. Deploy the model in a face recognition system, which could entail integrating the model into a web application or building a standalone desktop or mobile application that can recognize faces in real-time.

There are various machine learning algorithms and approaches that one can use for each of these steps. Popular choices for facial recognition include support vector machines (SVMs), neural networks, and deep learning algorithms like convolutional neural networks (CNNs).

4.2 Improvements to existing function

Improving the accuracy of gesture recognition: Depending on the complexity of your current gesture recognition model, you may be able to improve its accuracy by training it on more diverse data or fine-tuning the existing model.

Incorporating more gestures: If you have only implemented a few gestures so far, consider adding more to expand the range of controls available to the user.

Real-time control of multiple quadrotors: If you have the resources available, you could explore controlling multiple quadrotors at the same time using your existing face and gesture recognition system.

5. Limitations and Challenges

5.1 Limitations of DJI Tello drone

Limited range: Vision-based systems often have a limited range of operation, as the quadrotor must be within the camera’s field of view to be detected and tracked.

Overheat problem: The Tello drone is equipped with a compact chipset that generates a lot of heat during operation. This can cause the drone to overheat and shut down, especially if it's used continuously for an extended period without any breaks. The drone's

firmware is programmed to monitor the temperature of the chipset and automatically shut down the drone if it detects that the temperature has risen above a certain threshold.

5.2 Limitations of this project

Occlusion: The quadrotor may be occluded by objects in the environment, such as walls or furniture, making it difficult for the camera to detect and track it.

Complex environments: Vision-based systems can struggle in environments with complex backgrounds or lighting conditions, making it difficult to detect and track the quadrotor accurately.

Latency: There is often a lag between the time the gesture is made and the time the quadrotor responds due to the time required to process the image data and generate appropriate control commands.

Limited precision: Vision-based systems may provide a different level of accuracy than other control methods, such as those based on inertial measurement units (IMUs).

Dependence on external equipment: Vision-based systems require a camera and a computer to process the image data, which adds weight and complexity to the quadrotor.

5.3 Challenges

5.3.1 Difficulty of Understanding knowledge

The difficulty of Understanding knowledge with this project includes:

Understanding the principles of image processing and computer vision: To implement a

vision-based quadrotor gesture control system, the student must have a strong understanding of image processing and computer vision concepts—recognition and tracking.

Implementing a closed-loop control system: The student must develop a closed-loop control system that can adjust the quadrotor's movements based on the detected gestures. This requires knowledge of control theory, such as PID controllers, and designing and tuning a control system to achieve stable and responsive behavior.

Addressing practical challenges: The most significant difference between hardware development and software development is that hardware needs to deal with more variables, such as lighting conditions and unexpected movements of the drone. This requires significant testing to identify issues and ensure proper functionality.

5.3.2 Implement of video recording function

To implement video recording, I used the OpenCV library to capture and encode frames from the drone camera. While the video recording feature worked initially, it caused the drone's chipset to overheat, resulting in a decrease in FPS. As a result, the feature was tested multiple times, but each time, the same problem persisted.

During video recording, the FPS dropped to an unusable level, making the feature impractical. Since I am not a student of Mechanical engineering, I cannot solve this problem from the hardware side without relevant knowledge ◦ Therefore, I ultimately decided to abandon the video recording feature. The overheating problem in the drone chipset is a common issue that affects drone performance.

6. Conclusions

Based on the methodology and implementation sections, this report presents the development of a vision-based quadrotor gesture control system. The system utilizes Mediapipe for gesture recognition and face detection, enabling the user to control the drone through hand and facial gestures. The project achieved its objectives, including successfully implementing gesture recognition and control, face detection, and image processing. Additionally, face tracking was implemented using PID control, allowing for accurate tracking of the subject by the drone.

However, the limitations of the DJI Tello drone and the project itself should be noted, such as the drone's overheating problem. The report also discusses the difficulty of understanding knowledge and implementing specific features, such as the video recording function.

In conclusion, this project successfully demonstrated the feasibility of using vision-based gesture control for a quadrotor drone. The system shows potential for applications in various fields, such as surveillance, inspection, and entertainment. Further improvements and refinements can be made to enhance the system's functionality and overcome the identified limitations.

References

1. Q. Quan, *Introduction to multicopter design and Control*. Puchong, Selangor D.E: Springer Singapore, 2018.
2. Electronics Tutorials, “Closed-loop System and Closed-loop Control Systems,” Basic Electronics Tutorials, Mar. 04, 2018. https://www.electronics-tutorials.ws/systems/closed-loop-system.html
3. “Hands,” mediapipe. https://google.github.io/mediapipe/solutions/hands.html
4. F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “MediaPipe hands: On-device real-time hand tracking,” arXiv.org, 18-Jun-2020. [Online]. Available: https://arxiv.org/abs/2006.10214. [Accessed: 07-Apr-2023].
5. S. C. De Silva, M. Phlernjai, S. Rianmora, and P. Ratsamee, “Inverted Docking Station: A conceptual design for a battery-swapping platform for quadrotor uavs,” Drones, vol. 6, no. 3, p. 56, 2022.
6. V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," 2019. [Online] Available: arXiv: 1907.05047
7. Google, “Mediapipe/face_detection.MD at master · google/mediapipe,” GitHub, 04-Apr-2023. [Online]. Available: https://github.com/google/mediapipe/blob/master/docs/solutions/face_detection.md. [Accessed: 07-Apr-2023].
8. A. Sinha, “Face tracking and movement following drone,” Electronics For You, 29-Mar-2023. [Online]. Available: https://www.electronicsforu.com/electronics-projects/face-tracking-and-movement-following-drone. [Accessed: 07-Apr-2023].
9. M. Monajjemi, S. Mohaimenianpour and R. Vaughan, "UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV,"

2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4410-4417, DOI: 10.1109/IROS.2016.7759649.

10. K. H. Teoh, R. C. Ismail, S. Z. M. Naziri, R. Hussin, M. N. M. Isa, and M. S. S. M. Basir, “Face recognition and identification using deep learning approach,” Journal of Physics: Conference Series, vol. 1755, no. 1, p. 012006, 2021.

Appendix

A	Control Menu

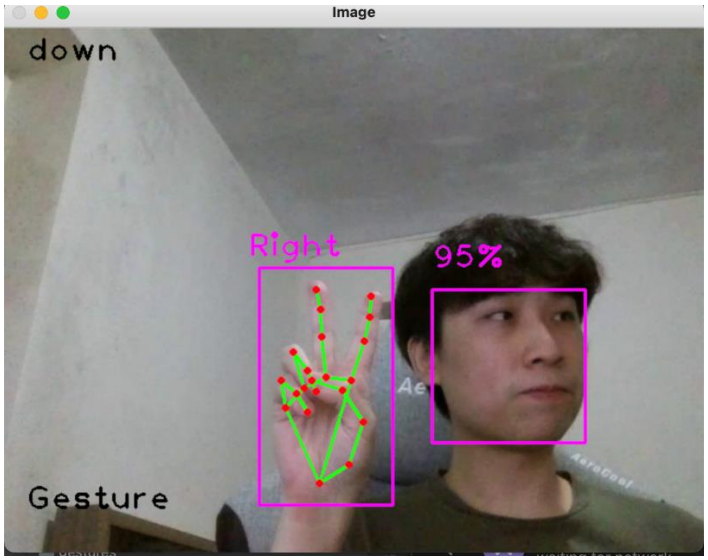
Control Menu

Keyboard Control Menu:

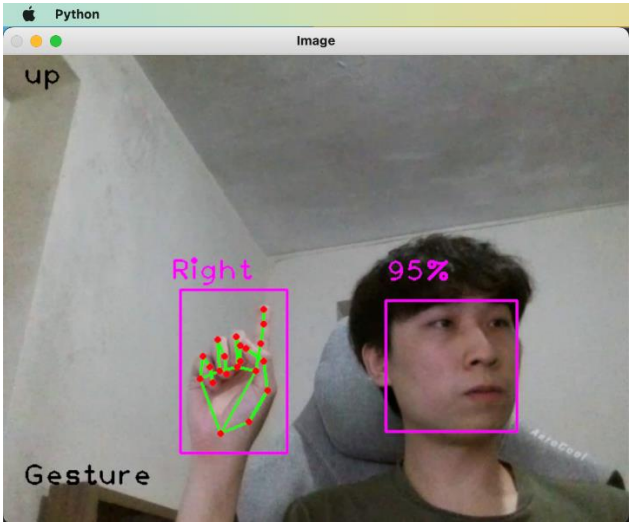
“Q”	Quit the program
“E”	Take off the drone
“G”	Switch to gesture control mode
“F”	Switch to face tracking mode

Gesture Control Menu:

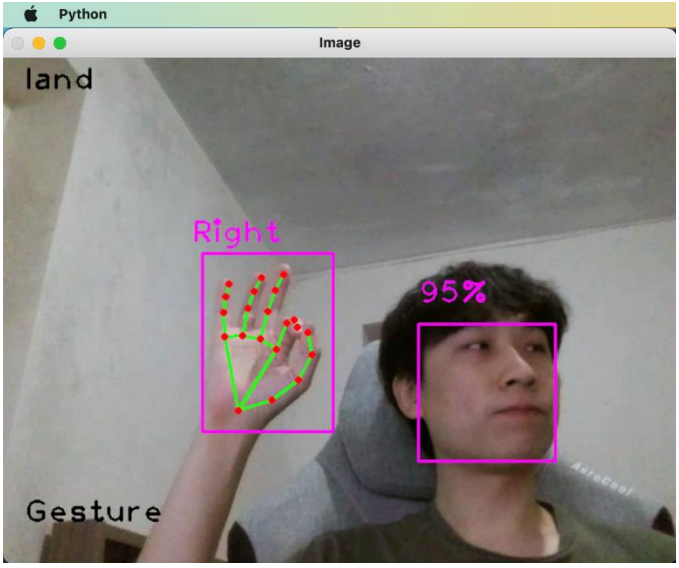
There are 13 kinds of gestures, and different instructions will be sent according to different gestures, the same set of gestures will be different depending on the left or right hand.



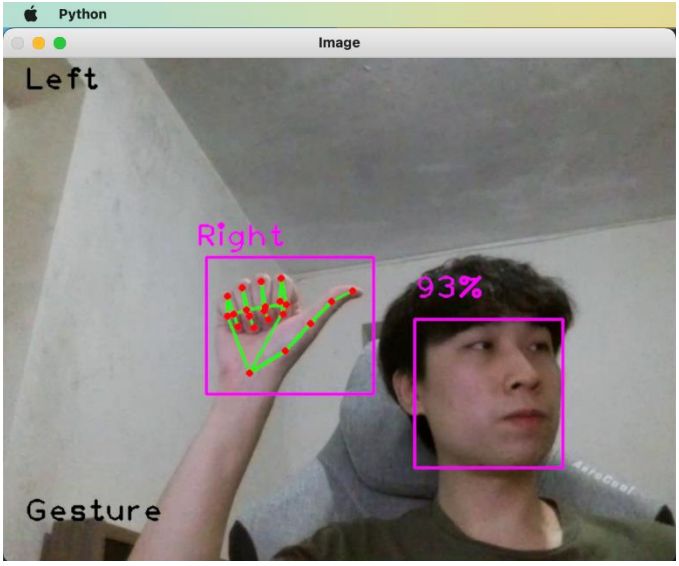
Gesture recognition demo (down)



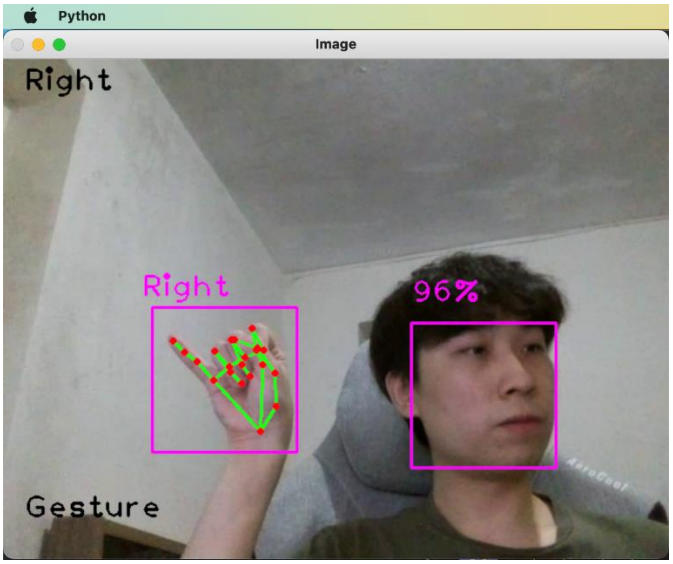
Gesture recognition demo (UP)



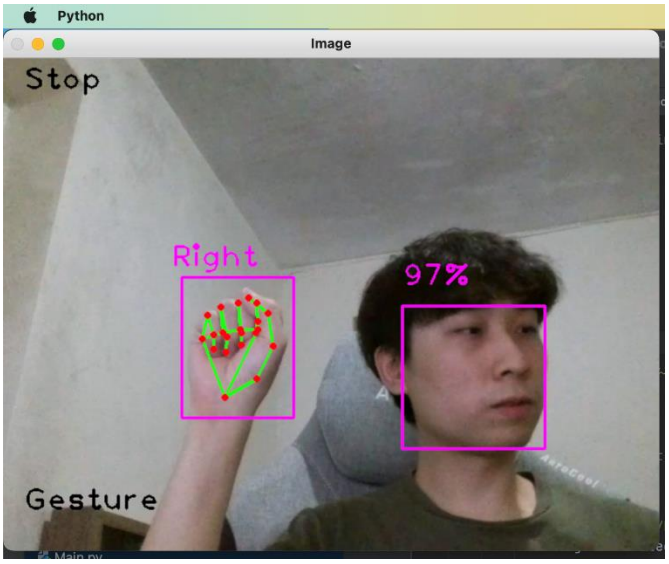
Gesture recognition demo (Land)



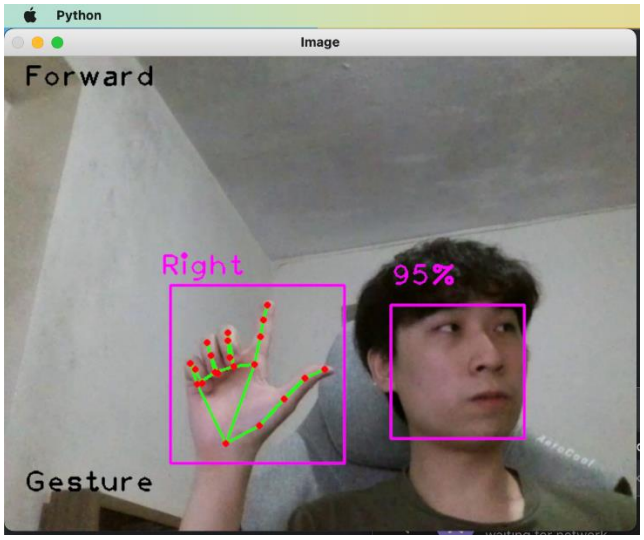
Gesture recognition demo (Left)



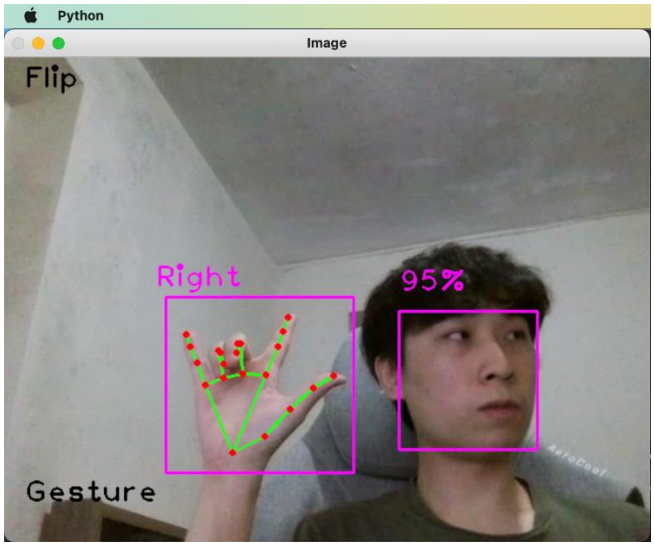
Gesture recognition demo (Right)



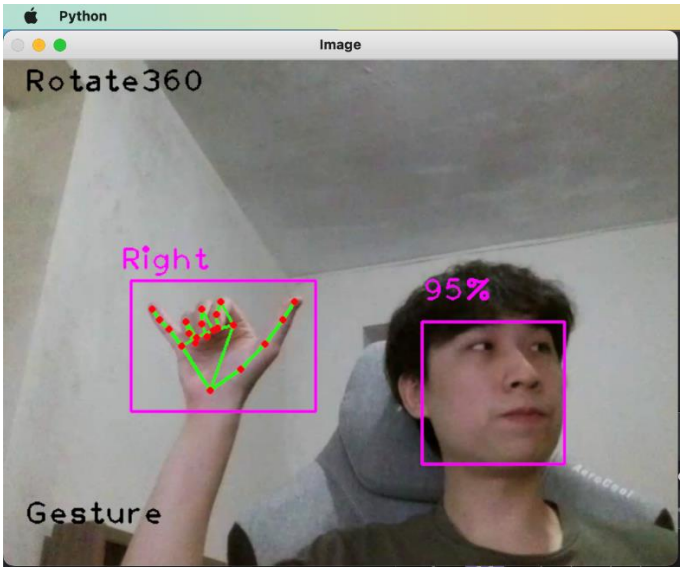
Gesture recognition demo (Stop)



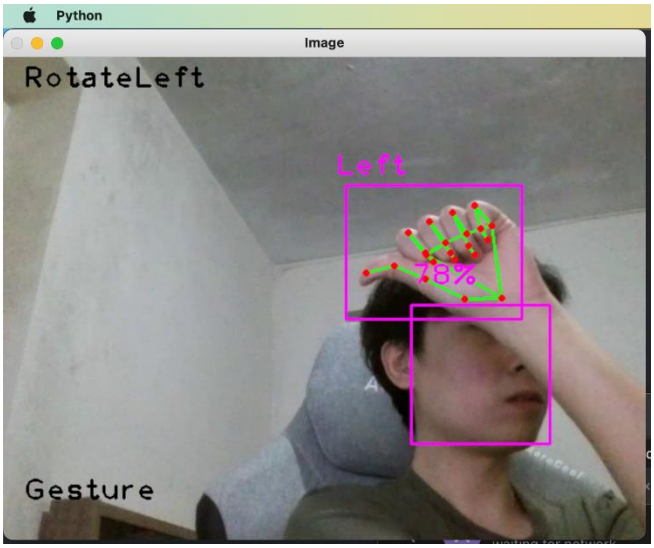
Gesture recognition demo (Forward)



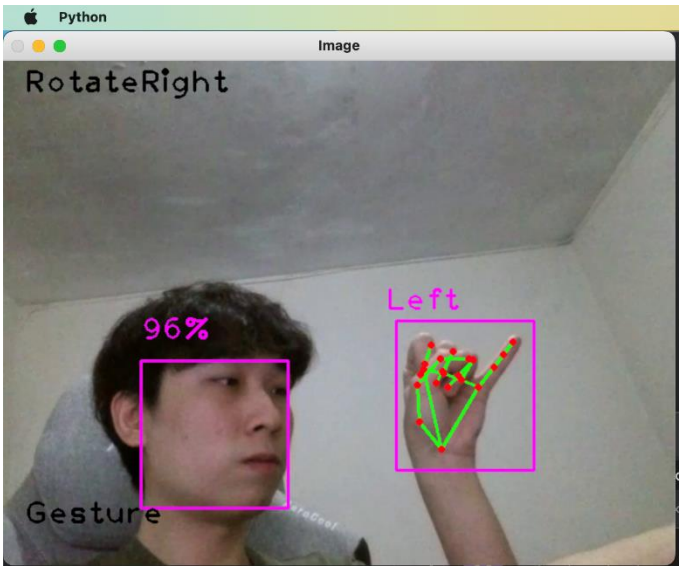
Gesture recognition demo (FlipRight)



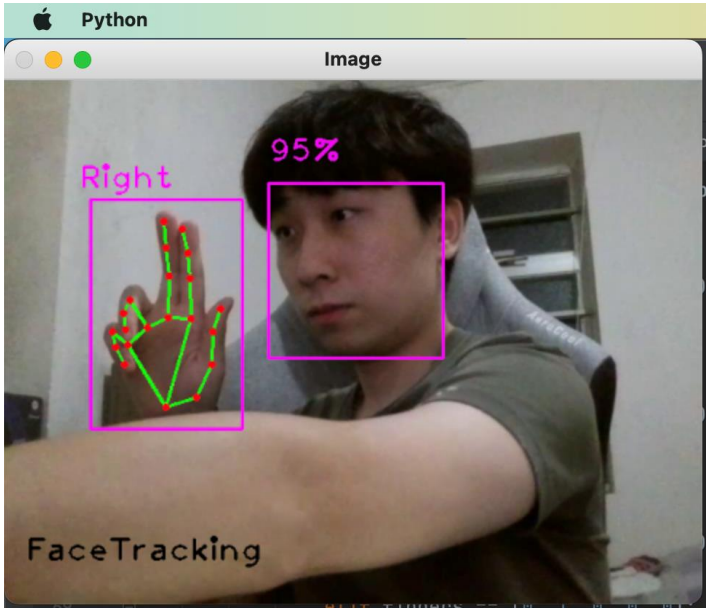
Gesture recognition demo (Rotate 360)



Gesture recognition demo (Rotate Left)



Gesture recognition demo (Rotate Right)



Switch Mode