



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels



Master's degree in Applications
and Technologies for
Unmanned Aircraft Systems

MASTER THESIS

TITLE: Manoeuvring Drone (Tello Talent) Using Eye Gaze and or Fingers Gestures

MASTER DEGREE: Master's degree in Applications and Technologies for Unmanned Aircraft Systems (Drones) (MED)

AUTHOR: Bouazza Berarache

PROFESSIONAL ADVISOR:
ACADEMIC ADVISOR: Miguel Valero García

DATE: October, 23rd 2023

Abstract

The project aims to combine hands and eyes to control a Tello Talent drone based on computer vision, machine learning and an eye tracking device for gaze detection and interaction.

The main purpose of this project is gaming, experimental and educational for next coming generation, in addition it is very useful for the peoples who cannot use their hands, they can manoeuvre the drone by their eyes movement, and hopefully this will bring them some fun.

The idea of this project is inspired by the progress and development in the innovative technologies such as machine learning, computer vision and object detection that offer a large field of applications which can be used in diverse domains, there are many researcher are improving, instructing and innovating the new intelligent manner for controlling the drones by combining computer vision, machine learning, artificial intelligent, etc.

This project can help anyone even the people who they don't have any prior knowledge of programming or Computer Vision or theory of eye tracking system, they learn the basic knowledge of drone concept, object detection, programing, and integrating different hardware and software involved, then playing.

As a final objective, they can able to build simple application that can control the drones by using movements of hands, eyes or both, during the practice they should take in consideration the operating condition and safety required by the manufacturers of drones and eye tracking device.

The concept of Tello Talent drone is based on a series of features, functions and scripts which are already been developed, embedded in autopilot memories and are accessible by users via an SDK protocol.

The SDK is used as an easy guide to developing simple and complex applications; it allows the user to develop several flying mission programs.

There are different experiments were studied for checking which scenario is better in detecting the hands movement and exploring the keys points in real-time with low computing power computer. As a result, I find that the Google artificial intelligent research group offers an open source platform dedicated for developing this application; the platform is called MediaPipe based on customizable machine learning solution for live streaming video.

In this project the MediaPipe and the eye tracking module are the fundamental tools for developing and realizing the application.

CONTENTS

INTRODUCTION	1
A- Motivation and goals	1
B- Structures	2
CHAPTER 1. TOOLS HARDWARE AND SOFTWARE	3
1.1 Hardware involved in this project.	3
1.1.1 Drone.....	3
1.1.2 Eye tracking device.....	4
1.1.3 Computing machine.....	4
1.2 Software tools	5
1.2.1 Programing code for hand recognition gesture.....	5
1.2.2 Cross platform for hand movement detection.....	5
1.2.3 DjitelloPy and SDK.....	6
1.2.4 Tobii Dynavox PCEye 5 software driver.....	6
CHAPTER 2. OBJECTIVES AND WORK PLAN	8
2.1 Project goals	8
2.2 Time plan for 33 weeks	8
2.3 Methodologies applied during the preparation	10
2.4 Relation with the supervisor	10
CHAPTER 3. EYE TRACKING SYSTEM	11
3.1 Introduction to eye tracking system	11
3.2 Description of eye tracking system	12
3.3 Types of eye-tracking devices	12
3.3.1 Remote eye tracking devices	12
3.3.2 Mobile or head-mounted devices.....	12
3.3.3 Head-stabilized.....	13
3.3.4 Integrated or Embedded eye tracking device	13
3.4 Operating method for eye tracking system	14
3.4.1 How the eye tracking system work.....	14
3.4.2 Basic hardware used in tracking system.....	15
3.4.3 Model of calibrating the tracking system.....	15
CHAPTER 4. HAND DETACTION AND GESTURE RECOGNITION	17
4.1 Hand gesture recognition definition	17
4.2 Mediapipe Overview	17
4.3 Mediapipe Hands Description and system components	18
4.3.1 Palm Detection Model	19
4.3.2 Hand Landmark Model.....	19
4.4 Hand gesture recognizer	21
4.5 Method used for developing a user guide of hand gestures	22
CHAPTER 5. TELLO TALENT DRONE	24
5.1 Introduction and description of Tello Talent drone	24
5.2 Components and technical specifications of the Tello talent.	24
5.2.1 Description of components of the Tello Talent drone	24
5.2.2 Technical specifications of the Tello Talent drone	25
5.2.3 Expansion kit.....	26
5.3 Detail in flying and operating	28
5.4 Vision Positioning System	29
5.5 Basic tools for operating the Tello talent drone	29
5.6 SDK	29

5.7	Mobile Application on Phone and Tablet.....	30
5.8	Introduction to Tello Python programing	30
5.8.1	Simple script for the basic drone maneuvering.....	31
5.8.2	Simple script for reading the battery level.....	32
5.8.3	Simple script for recording video.....	32
CHAPTER 6. EYE TRACKING TOOLS DYNVOX PCEYE		34
6.1	Overview about the Dynavox	34
6.2	Description	34
6.3	Technical specifications of the Dynavox PCEye	35
6.3.1	Dimension and characteristic of Dynavox PCEye	35
6.3.2	Gaze specifications.....	35
6.4	The operating system required.....	36
6.5	User positioning required.....	36
6.6	Calibration requirement.....	37
CHAPTER 7: INTEGRATION OF TELLO TALENT AND DYNVOX EYE 5..		39
7.1	Introduction	39
7.2	The hardware and software setting used in this solution.....	39
7.3	Setting and calibrating Dynavox PCEy5	40
7.4	Developing the programming code.....	41
7.5	Initial set-up and recommendation.....	43
7.6	Running, Operating and controlling Tello talent by eyes	43
7.7	Survey and results for this experiment.....	45
7.8	Voluntaries description	45
CHAPTER 8. INTEGRATION OF HAND GESTURE TO CONTROL TELLO TALENT		46
8.1	System architecture	46
8.2	Prerequisite for this part of the project.....	47
8.3	Software installation steps.....	47
8.4	Proposed system architecture for this application	48
8.5	Preparing the code for hand detection and fingers gesture recognition.....	48
8.6	implementing the main application for drone Control with Hand Gestures	55
8.7	Run and Play.....	58
8.8.	Results and analysis.....	60
CHAPTER 9. COMBINING EYE TRACKING SYSTEM AND HAND GESTURE TO CONTROL TELLO TALENT		61
9.1	System architecture	61
9.2	Implementation	62
9.3	Results and analysis.....	65
CONCLUSION		66
ACRONYMS		67
REFERENCES.....		68
ANNEXES.....		69

INTRODUCTION

A- Motivation and goals

Scientists today are searching for a new way to make the human interactions with digital content feel more natural and fluid, by exploring the movement of eyes and hands that enable users to move more easily through the digital world.

Hand and eye tracking technics are an attractive topic for many developer and manufacturer and are increasingly being used for educational and instrumental purpose by several universities and companies.

The main motivation for exploring hand and eye tracking can be summarized in the following points:

- Hand tracking can be integrated into a control function that allows the user can send commands by moving their hands in a specific space instead of touching and maneuvering.
- Hand tracking offers the ability to react faster by detecting the position, orientation, and velocity of the hand.
- The speed of collecting data by eyes and hands movements can be an effective solution for replacing the habitual system control based on maneuvering by fingers. An object can be looked or pointed faster than it can be reached by hand or other tools so this scenario increases the velocity of the user input.
- The eyes are naturally pointed to look at the objects of interest, no require any specific training for the eyes orientation.
- The data collected are able to be streamed and create real-time interactions with virtual real for different filed of use.

The use of gaze data can be an effective solution to replace the control by hands in some situations where the use of hands is prohibited and also for the people who are not able to use their hands.

In addition to the eye and hand tracking system, the project also focuses on the Drone which is a topic science for developers and used for educational and experimental purpose.

In this project, we will only discuss the positive and useful aspects of drone applications, in contrast to non-humanitarian operations such as military operations including intelligence gathering, target identification, precision strikes, force protection, surveillance and reconnaissance, combat operations, anti-drone operations.

B- Structures

This document is organized in ten chapters to help for better understanding the different steps done during the project preparation.

Chapter 2: discuss the Hardware and Software tools involved in the project, Tello Talent drone and Tobii Dynavox PCEy5 eye tracking device are available in the market, and both can be customized and programmed according to the application requirements.

Chapter 3: explains the rollout of the project and giving detailed work plan with tasks, goals and deadlines.

Chapter 4: is focused on explaining the method of the eye tracking system and providing further details on calibration procedure.

Chapter 5: describing the proposed system for hand detection, gesture and recognition, the processing method used by MediaPipe for hand detection.

Chapter 6: is focused on the in-depth explanation of Tello Talent Drone as it is the heart of the project.

Chapter 7: is discussing in details the Tobii Dynavox Eye 5 device.

Chapter 8, 9 and 10: are focused on the developing the main application, integration and simulation of drone control by eye tracking, hand tracking or a combination of both, the details of these chapters are based on the knowledge gained from the previous chapters.

A final conclusion and future work are described in chapter conclusion.

Finally, in appendix there are the additional contributions of this work, such as references, publications, research papers and software.

CHAPTER 1.TOOLS HARDWARE AND SOFTWARE

This chapter focuses on the hardware and software used for developing an application that control drones only by eyes and or hands movement.

It states the essential tools and platform used to build the basic platform and bringing new features such as Pycharm, mediaPipe, Python, Tkinter and etc.

1.1 Hardware involved in this project.

The hardware used for in this project are drones, eye tracking device and computing machine (Laptop) as described in this table.

Table 1.1 Hardware used in the project

Hardware	Device type	Manufacturer	Specifications / Model
Drone	Tello talent	DJI	RoboMaster TT tello talent (GL)
Eye tracking	Dynavox	Tobii AB	PCEYE5 Gn, Driver Gaze-Point
Processing unit	Laptop	Hewlett Packard	HP Laptop with Intel i7 2.80 GHz CPU, 8GB RAM, 512 GB SSD, Windows 10. 64 bit HD full screen

1.1.1 Drone

There are various drones available in the market and many of them are dedicated for educational and experimental purposes, these drones have been studied during the preparation of this project and understanding how the people and organizations used them.

Several drones have similar characteristics and specifications which can be suitable for this project but in the end the **Tello talent drone** have been selected for this task as it is being used for experimental and educational purposes by several companies, organizations and universities and also recommended by many developers in the forum TelloPilot.

This **Tello Talent drone** has an easy, simple, and reliable platform that can be integrated with all operating system; it can be used by beginners to start learning and also by expert for developing complexes applications.

The drone used in this project is provided by the Universitat Politecnica de Catalunya campus "Castelldefels School of Telecommunications and Aerospace Engineering (EETAC)"

More details about the **Tello Talent drone** are available in chapter 6.

1.1.2 Eye tracking device

At the beginning of this project, we began exploring the possibility of using a regular camera to capture gaze data based on eye movements, similar to a hand tracking system. This scenario has not been tackled due to the interoperability gap between different dependencies, which requires downgrading and upgrading some features like Dlib which is mostly used for face recognition purposes, using camera for detecting gaze is still under development and need improving more the accuracy and precision.

There are a few companies in the market that manufacture eye tracking devices. During the background and study phase, the devices produced by GazePoint and Tobii have been examined; both are intended for experimental and educational purposes, are suitable for this project.

Tobii offers various eye tracking devices such as Pro Fusion, Pro Spectrum, Pro Spark, Pro Nano and Dynavox, each device has a specific features and tutorial that make them ideal for researchers and easy for users who want to measure gaze data. These devices are still considered new technology and their price is not compatible with the budget.

In the end, the **Dynavox PCEye5** device (see figure 2.2) is used because it is readily available in the ETTAC.



Figure 1.2 Dynavox PCEye 5

For more details about the **Dynavox PCEye5** device, see Chapter 7.

1.1.3 Computing machine

The basic computing hardware required is:

- Operating System 64-bit Windows 10
- Processor 2.0 GH or faster
- Memory 4.0 GB RAM
- Availability of 450 MB free in Hard Disk

In this project the HP Laptop with Intel i7 2.80 GHz CPU, and operating system Windows 10- 64 bit are used.

1.2 Software tools

The Tello Talent drone supports multiple computer programming software. After exploring different combinations of operating system, platform and coding languages to select the simple and available configuration on the open sources, In the end the following software configuration was chosen for this project based on their characteristics easy to use, open source and compatible with Tello Talent software.

- Windows 10 as an operating system,
- Pycharm as a platform
- Python as coding language

1.2.1 Programing code for hand recognition gesture

In this project the Python programing platform have been used for different reasons as:

- It is simple and easy for learning and developing the complex data structure
- It is used for educational and general programing purpose
- It is an interpreted programming language
- It is a multipurpose coding language and supports object oriented programming approach and procedural coding styles for creating and developing different applications
- It is an ideal coding language for fast developing applications
- It is an open-source and utilized in several sectors and disciplines
- And others ...

In addition to the above advantages, Python provides more specifications which help for developing and implementing modules in our application such us:

- It provides a large libraries for various fields such as machine learning **MediPipe** and image processing **OpenCV**
- It is equipped with several Graphical User Interface libraries, such as **Tkinter** GUI library, that is used for developing different application
- It is a cross-platform language; the same code can run on different platform such as Windows, Linux, UNIX, Macintosh, etc.

In the project, The **Tkinter** Graphical User Interface library is used for creating graphical user interface (GUI) such as button, frames and windows on the control screen which are used for eye movement interaction with control commands.

1.2.2 Cross platform for hand movement detection

There are various 2D and 3D open source models or frameworks for eye detection such as CPN (Cascaded Pyramid Network), Deep CNN, Random

Forest, ANN. Each model has some advantages and disadvantages as well as streamed data output options that can be used in different applications.

For this project, while researching on the Internet, forums and research papers, many developers recommend exploring the lightweight 2D and 3D real-time hand gesture estimation framework developed by Google and used for educational purposes.

This framework is an open source and is updated regularly. It belongs to the field of computer vision for augmented and virtual reality technology.

This Google framework is called **MediaPipe**. It supports cross-platform compatibility (i.e. Android, iOS, Web, Edge devices). **MediaPipe** methodology provides better detection results compared to the other traditional method.

The output result obtained by using MediPipe for hand detection model and the recognition method is match the specification of this project in Computer Vision Image Processing , then the **MediPipe** cross platform have been used.

Moreover, **MediaPipe** also provides several frameworks and machine learning models like Face Detection, Object Detection, Auto Flip, Automatic video cropping pipeline, Pose Detection and more. It is not used only for detecting it also able to track the behavior of gestures.

MediaPipe provides an interface API (Application Programming Interface) compatible to Python coding platform, it contains lightweight palm detector model and hand landmark detector model that are tailored to work in real-time, high performance and less processing.

In Chapter 5, there are more details about **Mediapipe**

1.2.3 DjitelloPy and SDK

Since the Tello Talent drone was chosen for this project, the software of this drone includes a Python interface called DjiTelloPy, which contains a function library using the Tello SDK protocol to perform the following tasks:

- Implementing Tello commands in python program
- Recording and retrieving a video stream and pictures
- Receive and parse state packets
- Control a swarm of drones

DJiTelloPy interface support only Python language released 3.6 or higher.

The version of the SDK implemented in the Tello Talent drone is SDK3 and contains several packages and functions as well a list of predefined commands that make it easier to explore the data and methods to control the drone.

1.2.4 Tobii Dynavox PCEye 5 software driver

The Tobii Dynavox PCEye 5 is using several drivers such as Magic Eye FX, Sono Flex, Windows control, Gaze-Point and etc. Each driver is used for specific application.

In this project, the driver called Gaze-Point has been used because it allows the eye control of the mouse cursor to make single click, comparing to the others driver, are offering multi-choice for making single click then need to scroll more option for selecting the desired one then to increase the probability to reduce the accuracy.

The Gaze-Point driver allows the user to perform the parameters setting and calibration.

CHAPTER 2.OBJECTIVES AND WORK PLAN

This chapter is stating the time plan and methodology followed to achieve the project objective, it discusses the steps and chronology used to create a concept for drone control and monitoring through an application without using the usual human-computer interface such as keyboard or mouse.

2.1 Project goals

The initial objective of the project is to develop and integrate a basic platform that allows anyone, even people with no prior knowledge of hand-eye tracking systems, to perform basic flying missions using only hand interaction or eyes movement or a combination of both.

Users can operate and maneuver the drones for taking off, Landing, rotating etc. even creating special mission.

As we progress further in the project, we are thinking to create more applications for circus, games, etc. However, the main objective of this project is to control the drone through eyes movement and or hand emotions.

Other reason behind creating this application is that it can be used by the people who cannot use their hand or bedridden patients who need long-term care in hospitals for restarting to move their body.

The main objective of the project is to design and create new functionalities that control the drones without basing on console or Keyboard and mouse.

The objectives of the applications:

- Help patient who cannot use hand for operating drones
- Make a player game where you can pilot the drone through the eyes and hand

The work consists of designing a hardware architecture which includes drone, eye tracking device, controlling screen system, camera and computer.

In addition, software architecture is designed that allows communication between different modules and processes the data received from the eye and hand movement to the commands which are sent to drone.

2.2 Time plan for 33 weeks

At the beginning of the project, the work was focused on exploring eye tracking system for an eventual use for controlling and operating drones.

Week eleven starting the project

After three weeks, getting the first eye tracking device manufactured by Tobii AB model Tobii Eye Tracker 5, this device cannot help for the project it is dedicated for creating a fast interactive atmosphere for gaming by eyes,

Week sixteen, getting the Tobii Dynavox PCEye5 device and the Tello Talent drone which are used as main component of the project.

Week 24, starting the first simulation by eye tracking device
 Week 25, adding hand tracking for operating the drone to the object
 Week 26, starting implementation and integration
 Week 35, the first version of the application was developed
 Week 37, starting writing the project

Some timeslot was left, due to some interoperability and eye tracking module calibration issues and also we have to improve some functionality and to add new alternative scenario.

The month of June and August was left free

The project planned to be delivered in October

In the next diagram (Figure 3.1) there is a time plane which summarizing the tasks, targets and also the estimated deadline.

This figure provides a board table of the works done for realizing the timeline and associated task, offering an idea about how this project was managed and accomplished.

Table N° 2.1 work plan

Task N°	Research phase	Objectives	Deadline
1	Background research	-Studying the theory of Eye and hand tracking system -Researching the technology available for the eye tracking and Drones - Studying some case of applications realized for controlling drone in hosting platform like GitHub and other available information on open sources - Improving my knowledge in Tkinter and Python	15 April
2	Technical Details of the hardware and software involved for designing the project	-Operating and maneuvering Tello talent (DJ xxxx) and Dynavox eye tracking Eye5 (Tobii AB) - Mediapipe - SDK - Visual basic - Pycharm and etc.	15 May
3	Implementing and coding	-Creating the first basic platform for controlling drone by eye -Adding in the same application the possibility to use hand movement	15 June
4	Data analysis and result	-Improving and adding new functionalities Passing by several version -Adopting the last version -Final demonstration.	30 June
5	Writing	Text , diagram, photos, snaps, etc.	30 September
6	Revision and reviewing		21 ctober

2.3 Methodologies applied during the preparation

As illustrated on the time plan paragraph, practically the project is divided into six steps as mentioned in figure 3.1

The first two months was dedicated to understand deeply the eye tracking system, the drone engineering, etc.

Since the goal was to know firstly the possibility to design a software and hardware architecture for drone control and monitor through an application that developed by Python, Java, C++ or others involving others cross platform like Tkinter , Pycharm , etc..; and based on the information collected, I decide to go the next step.

Step 2, the necessaries hardware were selected like Dynavox Eye5 eye tracking device, Tello Talent drone, and Standard laptop.

In this step also, the choice to use the Mediapipe for processing the video generated by the hand movement was decided.

Step 3, focused for preparing the first design for the basic platform, using Python, Tkinter and Tobii SW for Dynavox and running a demo without including the talent drone.

Adding the first design the functionality that using hand movement to be combined with eye for control and monitor the drones.

Step 4, Starting to run the first platform in real demo, some problems appear during to flying and collecting videos, updating the code for Improving all the functions and adding new functionalities like monitoring the status of the drone; In this step many scenarios was tested before adopting the last version of the code.

Step 5, starting to writing the thesis based on what documented.

Step 6, double check and verification, adapting new figures to pages size etc.

2.4 Relation with the supervisor

The relationship with the supervisor of this project, Miguel Valero, has been the fundamental for the progress of the project; the work was done in full cooperation.

The collaboration was characterized by constructive interactions by e- email and also by regular meeting.

CHAPTER 3.EYE TRACKING SYSTEM

This chapter describes the eye tracking system, including different categories of devices available on the market; we will take an in-depth look at the key component of the eye tracking system and the calibration procedures required.

3.1 Introduction to eye tracking system

Eye tracking is a process of detecting and measuring the movement and position of the eyes. It can be viewed as a human machine interface.

The retina has naturally an area of dense nerves and high visual acuity that called the fovea. The lens of the eye focuses light on the fovea, and a person moves their eyes to direct the lens and fovea where they want to look.

Historically, Eye tracking methodology has more than one hundred years; it was originally used for basic research into vision and neurophysiology.

The figure 3.1 shows the geometric position of lens and retina relatively to the eye direction.

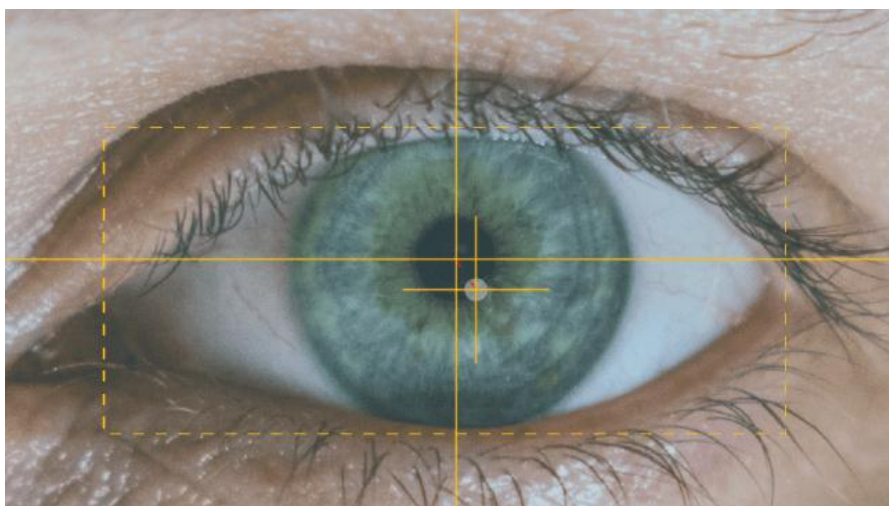


Figure 3.1 Retina and lens

The field of research and manufacturing has experienced much progress in the last twenty years, increasing the versatility and flexibility of the computer vision technology, and therefore opened the field to new applications beyond eye tracking technology; alternatively, eye movements can provide an alternate method for interaction with different environment and used as a computer interface, or introduced in virtual reality technology.

Recent advances in eye tracking technology have expanded the applications field of using eye movement to include many different areas, as a tool for research and as a source of real-time data for interaction.

3.2 Description of eye tracking system

The eye tracking is based on the data collection that focuses on a user's gaze. It gives the possibility to determine how the gaze is moving based on eye patterns which are associated with the point at which the eye is looking. The processing consists tracking data of pupil dilation and gaze direction. The information collected is:

- Movement (how the user gaze moves around a specific space which can help for determining the orientation)
- Duration (how long the user looks at the same object)
- Fixations (points where the user gaze paused or lingered, which can help to track the overall gaze pattern and focus.

3.3 Types of eye-tracking devices

There are four categories of eye tracking systems devices available on the market; each type was developed for a specific application but the methodology used for data collection is the same.

3.3.1 Remote eye tracking devices

The remote series (see figure 3.2) of eye tracking devices does not require direct contact with users; it must be placed closed to the control screen; this series is the typical tools used today for educational and experimental purpose. It is equipped with a camera and a light or laser source, Generally, it is configured via a driver that is properly developed by the manufacturer.



Figure 3.2 Model of Remote eye tracking device

3.3.2 Mobile or head-mounted devices

This category of eye tracking usually has the form glasses (see figure 3.3). This system requires placing a mirror or camera in the visual path of the eyes and an additional camera that records the field of view and scene, It is very comfortable for the user compared to the mobile eye tracking series, This type is used mostly for video gaming; the user can play as longer as they want.



Figure 3.3 Model of Mobile eye tracking device

3.3.3 Head-stabilized,

This type of head stabilized tools(see figure 3.4) is designed to immobilize the head of the user and required a high level of stability; this series is mostly applicable in neurophysiology and vision experimentations where the accuracy and precision are highly required than the user comfort.



Figure 3.4 Model of head stabilized eye tracking device

3.3.4 Integrated or Embedded eye tracking device

This type of integrated device (see figure 4.5) includes Virtual Reality and Augmented Reality features and can be used as an intuitive control method to navigate via eyes movements when the user is moving and could not use mouse or keyboard.



Figure 3.5 Model of Integrated eye tracking device

Each category for eye tracking devices has its pros, cons, and data output which can be used in various applications. For this project the remote category device have been used.

3.4 Operating method for eye tracking system

Typically the remote eye tracking device is a sensor that can detect eyes and track their movement in real-time. In practice, an eye tracking system collects information about eye movement such as pupil position and the gaze direction for both eyes, and then converts this information to a useful data stream.

An eye tracking system consists of one or more cameras, some light sources and computing capabilities that include a control screen.

3.4.1 How the eye tracking system work

In theory, an eye tracking system works as the following steps:

1. The user has to look to the target point on the control screen for a few seconds or the time required by the processing mechanism for taking the information.
2. An infrared light source is directed toward the center of the eyes (pupil) causing detectable reflections in both the pupil and the cornea.
3. These reflections represent the vector between the cornea and the pupil and are picked up by the camera.
4. Computing algorithms translate the camera feed into a scientific useful data streaming with the help of advanced image processing like machine learning and also the algorithm input is referenced to the calibrating points, user position, camera and control screen.

The data streamed at the output of the system can be used as an input modality in a wide range of application such us video gaming, virtual reality, medicine, health carry for the people who can use their hand or lost the ability to move the mouse with hands, etc.

Additionally, gaze tracking is an excellence and fast pointer than using a mouse; it can be involved as extra complimentary manual inputs for human machine interaction.

3.4.2 Basic hardware used in tracking system

Next figure 3.4 gives an overview about the eye tracking system in a simple way.

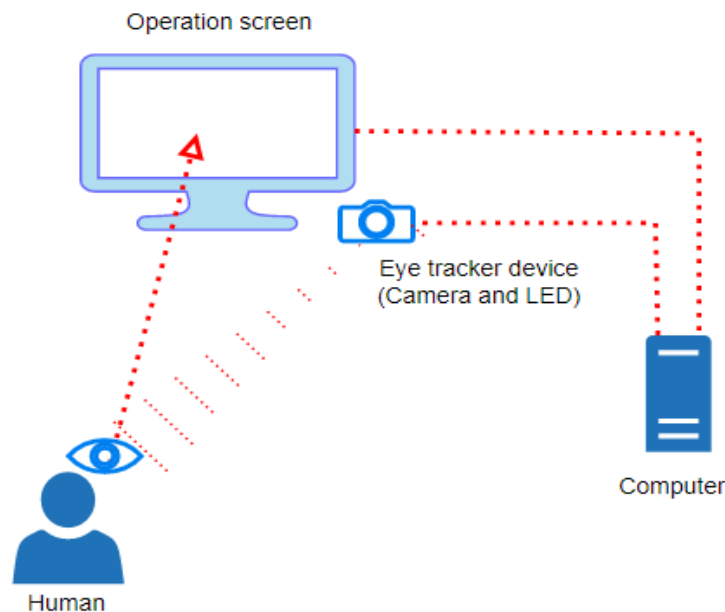


Figure 3.4 Bloc diagram of Eye tracking system

Practically, there are three devices involved in the eye tracking system.

- A special eye tacking camera includes lights sources mounted below or over the control screen and this camera is pointed to user's eyes.
- A control screen in which the user operates the system by looking at cells that are displayed on the screen for activating or pressing the cell, the user looks at the cell for a specific time, Typically, the gaze duration required to visually activate a key is about half second, can be adjusted by the user. Arrays of menu keys that are visible on the screen allow the user to navigate independently without using hands and mouse.
- A sophisticated Computer with images processing software analyzes continually the images piqued by the camera and determines which cell, the user is looking at on the control screen.

3.4.3 Model of calibrating the tracking system

The eye tracking systems do require a calibration, which is a method of algorithmically associating the physical position of the eye with the point in the

screen that the user is looking at. The calibration procedure as shown in figure 3.5 is used to compensate the variations in eye size, gaze position, control screen size and the distance between user and control screen.

The calibration procedure is required to set up the system and the user position. The user looks at a small calibration circles that have a fixed position and known on the control screen, generally the number of circles is 5, 9 or 13 and distributed on the border and center of the screen.

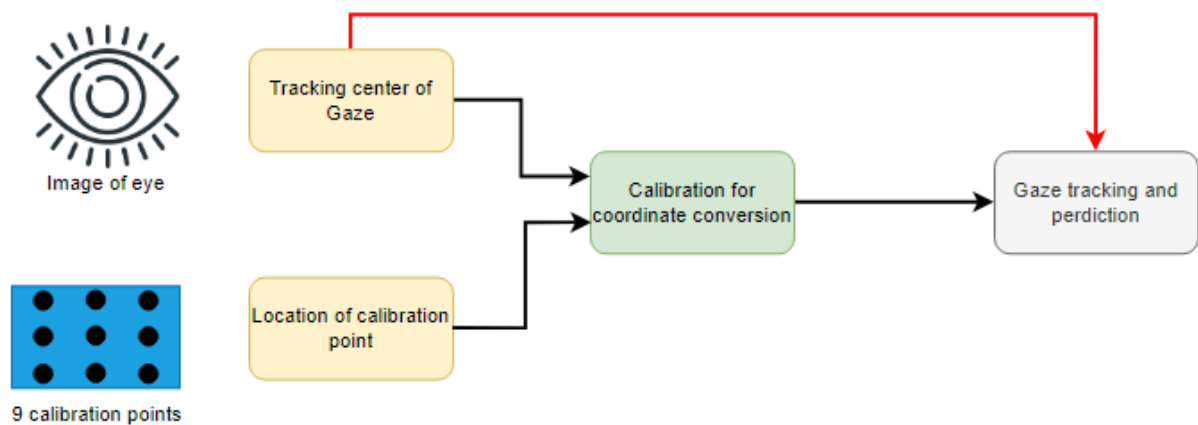


Figure 3.5 General processing flow for eye tracking

Practically, the calibration software tools is delivered with the eye tracking device and consist of pointing the eyes to the calibration circles one by one for very short time accordingly to the pop up message generated around the screen.

The calibrations require some degree of cooperation and ability from the user, There is a measuring the accuracy of the calculated gaze to validate the success of the calibration, The validation results expressed in degrees of visual angle for each measuring points.

The accuracy tolerance is depending on the application, an error of degrees between 0.25 and 0.5 degrees is considered acceptable if more than this range, the calibration considered failed and requires another attempt and improves,

The calibration procedures must be done for each user, if the same user moves away from the screen and returns in this case no need to recalibrate.

To achieve a good precision and accuracy in eye tracking system, the user needs to take the lack of alignment between the optical and visual axes into consideration.

CHAPTER 4. HAND DETECTION AND GESTURE RECOGNITION

This chapter discusses about hand detection and gesture recognition, and it is focused on the MediaPipe framework's method and programming tools for the hand gesture to create commands for different controlling uses. This method is used for educational backgrounds.

4.1 Hand gesture recognition definition

Hand gesture recognition is a part of research areas in the field of human-computer interface. It is used to develop a system that can read and interpret hand movements as commands to control devices without touching any buttons or screens.

A gesture recognition system works with a camera or sensor pointed at a specific three-dimensional space, capturing frame-by-frame images of hand positions and motions in this space. Those images are analyzed in real time by computer vision and machine learning technologies, which allows translating the hand movements and finger motions into commands, based on a predetermined library of signs.

The commands generated after processing become an input that is similar to pressing a button, mouse or touching a screen.

Gesture recognition can be used in a variety of applications such as video games, virtual and augmented reality, healthcare, automotive, controlling a presentation, commanding devices.

Next paragraphs discuss the MediaPipe and the gesture recognition.

4.2 Mediapipe Overview

MediaPipe [3] is an open source cross-platform framework developed by Google to build a pipeline for processing perceptual data from image, video and audio.

The solutions used in MediaPipe include multiple frameworks such as hand landmark tracking, posture estimation, face recognition and etc. In this project we focus only on the MediaPipe Hand Landmark, which allows the user to detect the landmarks of the hands in real time.

The MediaPipe operates on the image frame received from a simple picture, video or live streaming as an input. The machine learning performs the processing then the output is the localization of the hand landmarks in real time.

In simple terms, the supported data for input and data output capabilities are as follows:

The input needed for detecting the landmarks can be:

- Simple image
- Recorded video frame
- Live streaming video

The output for the detected hand are:

- Handedness (right or left)
- Landmarks in image coordinates
- Landmarks in world coordinates

4.3 Mediapipe Hands Description and system components

MediaPipe Hands is a tool that has a high-fidelity hand and finger tracking solution with less processing, and it can work in a normal Laptop with standard configuration.

It employs machine learning to infer twenty one 3D landmarks of a hand from a single frame picture received from a camera or extracted from a continuous stream.

MediaPipe Hands use a machine learning pipeline solution which consists of multiple models working together such as palm detection model and hand landmark model (see figure 4.1)

- The palm detection model operates on the full image received and returns an oriented hand bounding box.
- The hand landmark model operates on the cropped image region defined by the palm detector for locating the palm and defines the region of interest (ROI) then returns high-fidelity 3D hand key points, subsequently predicts all twenty one hand Knut from this ROI.

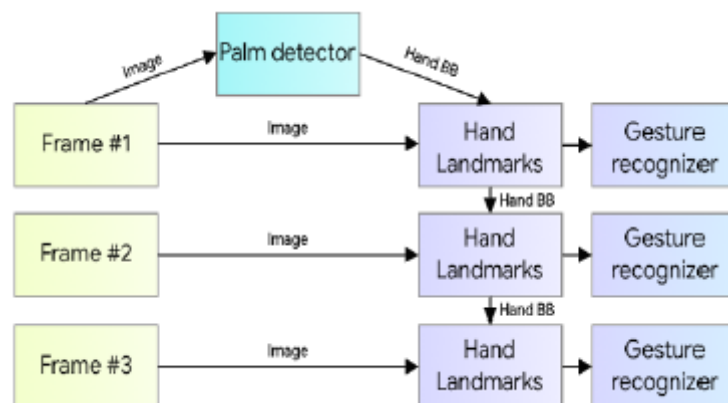


Figure 4.1. Palm and Landmark perception in Pipeline technology

4.3.1 Palm Detection Model

The palm detection model is designed to detect continually the palm locations on the image frame captured by the camera or extracted from video streaming and have to compute across a variety of hand sizes (see figure 4.2) and be able to detect occlude and self-occluded by performing a large scale span up to twenty times of the image frame.

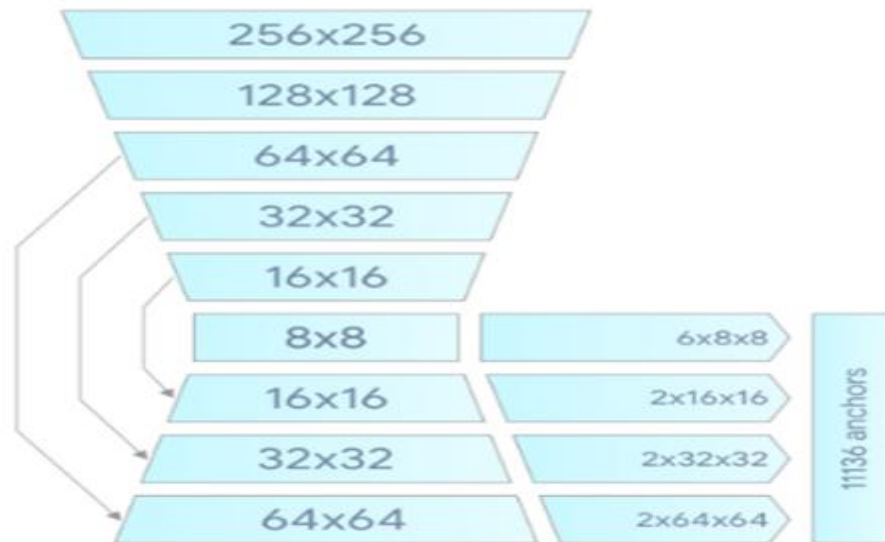


Figure 4.2. Palm detector model architecture

The detection process works by estimating the bounding boxes of rigid objects which supposed the palm then use only the first square bounding boxes that can be modeled and ignoring other aspect ratios and therefore allows reducing the number of anchors that resulting from the high scale variance by a factor varying between three and five depending to the complexity of the image and allows the system to dedicate most of its capacity towards coordinate prediction accuracy.

For improving the detection precision an encoder/decoder extractor are used for bigger scene context awareness including very small objects.

This detecting method used is able to offer an average precision of 95.7% in palm detection.

4.3.2 Hand Landmark Model

The hand landmark model uses the bounding box produced by the palm detector as an input and performs precise landmark localization of the key points or finger knuckles inside detected hand regions via regression.

The hand landmark model learns a consistent internal hand pose representation basing on prediction method in Mediapipe tools; the model has three outputs (see figure 4.3).

- Hand presence
- 21Keys points 3D landmarks
- Handedness (Right or Left)

A hand flag indicating the probability of hand presence in the input image;
Each output is trained by their corresponding datasets in Mediapipe

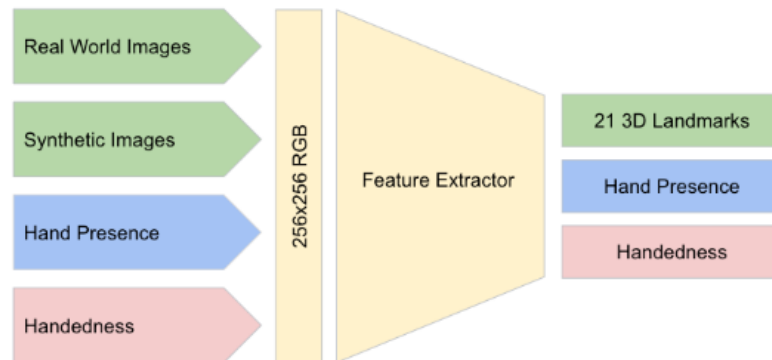


Figure 4.3 Hand landmark model architecture

At the end, this two processing steps Palm detector model and Hand Landmark detector model allow achieving precise twenty one 3D, key points coordinate as shown in figure 4.4 and figure 4.5.

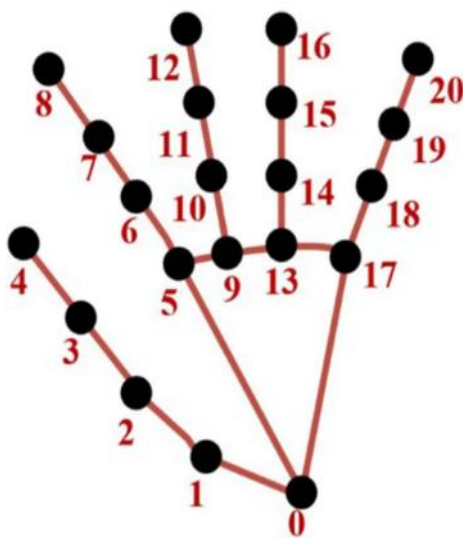


Figure 4.4. Landmark, numbering.

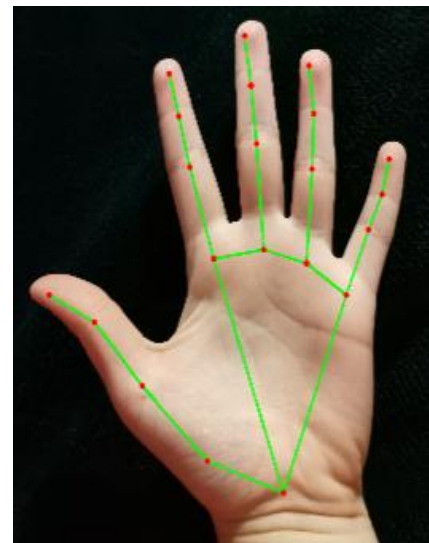


Figure 4.5 Hand landmarks on image.

A convention numbering and naming is defined for each finger knuckle and can be used for different applications, in the below table 4.1.the key points are numbered from 00 to 20.

Table 4.1. Landmark, conventional numbering and naming

N°	Description	N°	Description
00	Wrist	11	Middle finger distal interphalangeal joint
01	Thumb carpometacarpal joint	12	Middle fingertip
02	Thumb metacarpophalangeal joint	13	Ring finger metacarpophalangeal joint
03	Thumb interphalangeal joint	14	Ring finger proximal interphalangeal joint
04	Thumb tip	15	Ring finger distal interphalangeal joint
05	Index finger metacarpophalangeal joint	16	Ring fingertip
06	Index finger proximal interphalangeal joint	17	Little finger metacarpophalangeal joint
07	Index finger distal interphalangeal joint	18	Little finger proximal interphalangeal joint
08	Index fingertip	19	Little finger distal interphalangeal joint
09	Middle finger metacarpophalangeal joint	20	Little fingertip
10	Middle finger proximal interphalangeal joint		

Each key point detected has three coordinates x, y, and z in the frame, respectively x (width), y (height) and z denotes the landmark depth, more the hand is closed to the camera, the value of Z becomes smaller, The value of X and Y is normalized to [0.0, 1.0] Additionally, each landmark has world relative coordinate.

4.4. Hand gesture recognizer

Gesture recognition provides real-time data to a computer, There are different approaches that can be used to acquire information of hand gestures recognition starting from simple use till a complex application.

This project focuses on simple interaction interface using an algorithm that compute gestures based on a predetermined library of signs, this library is built by a group of fingers states such as bent finger, straight finger, finger direction angle, etc., each element of this group can be labeled, and generates a list of commands that based initially on a set of gestures, generally each specific gesture is corresponding to specific task based on its own purpose.

In Figure 4.6 show an example of five hand gesture codes, those allow to build a simple interface for remote control application.

This method works on manipulating the fingers in objective to establish the right form as indicated in the gesture code list and the programming code generate the control command accordingly.

It is noticeable that each gesture has a specific configuration of fingers states such as straight, bent and crossed that can be used for the recognition through a certain type logic and condition.

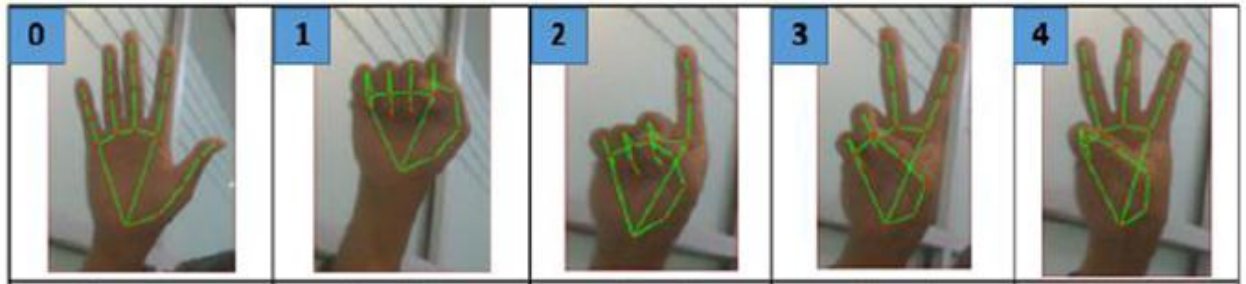


Figure 4.6 Example of hand gesture code

4.5 Method used for developing a user guide of hand gestures

This paragraph discusses the steps that the system takes by identifying hand gestures and converting the gesture into a specific command that allows the user to develop an user guide application through MediaPipe framework and a programming language.

This guide that performs hand gesture recognition is based on the captured hand pose and the twenty one key points as detailed in the previous paragraph.

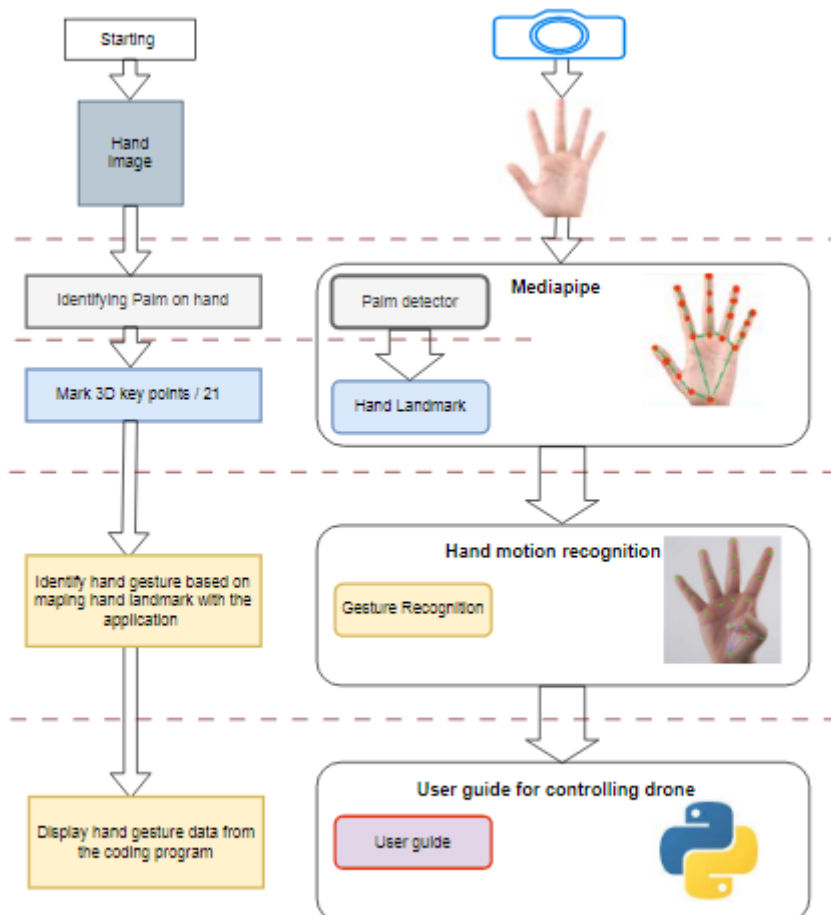


Figure 4.5 Workflow of hand gesture recognition system.

This figure 4.5 [15] show the implementation used for creating a user guide based on the hand motion and the workflow is summarized in the following tree steps:

- The image captured by the camera in real time is the input for the process, Mediapipe (Palm and landmark detector) will read the image and processing at the output the twenty one 3D keys points as explained above
- Those twenty one 3D key points are used as input in the hand gesture recognition module and will be computed and initialized as a tool for reading in real time
- The user guide is developed by an algorithm or programing code such as Python, C++, and etc. which is identifying the poses of the hand basing on the 21 key points coordinates X and Y (Z not used)

As the twenty-one key points and their corresponding coordinates are normalized, the X and Y values that are kept and tracked in real time are used as an input.

The algorithm or coding program is used to determine the finger of the hand condition relatively to the landmark coordinate, It compares the coordinate of the twenty one key points based on the position of the X (horizontal) and Y (vertical) and reads the pre-programed condition for each case, if the condition has been fulfilled then the result is true.

CHAPTER 5. TELLO TALENT DRONE

This chapter discussed the Tello talent drone including drone elements, technical specification and operating mode and in the last paragraph 6.8 there some basic example for programming the drone mission by Python.

5.1 Introduction and description of Tello Talent drone

The Tello, Tello EDU and Tello Talent drones [8] [9] are a perfect programmable drones for educational valuable learning materials and also for hobby use, It sit in the education division of DJI product, They are manufactured by Shenzhen Ryze Technology and are equipped with Intel processors and DJI flight control technology.

The drones have the same size and are distinguishable by the top color respectively white, black and red.

Historically, Tello is the first drone developed, then Tello EDU and Tello Talent is the latest version with few more features and capabilities which allow the users to configure in a simple way than the previous versions.

The Tello Talent is delivered with an extension kit that includes an Open source controller, a Dot-matrix led screen, a Distance sensing module and Extension board.



Figure 5.1 Tello, Tello EDU and Tello Talent drones

5.2 Components and technical specifications of the Tello talent.

The main components and the principal characteristic of the Talent drone are summarized in the following items.

5.2.1 Description of components of the Tello Talent drone

The Tello talent drone is a small quad-copter which includes an expansion kit and can hover in place and is suitable for flying indoors,

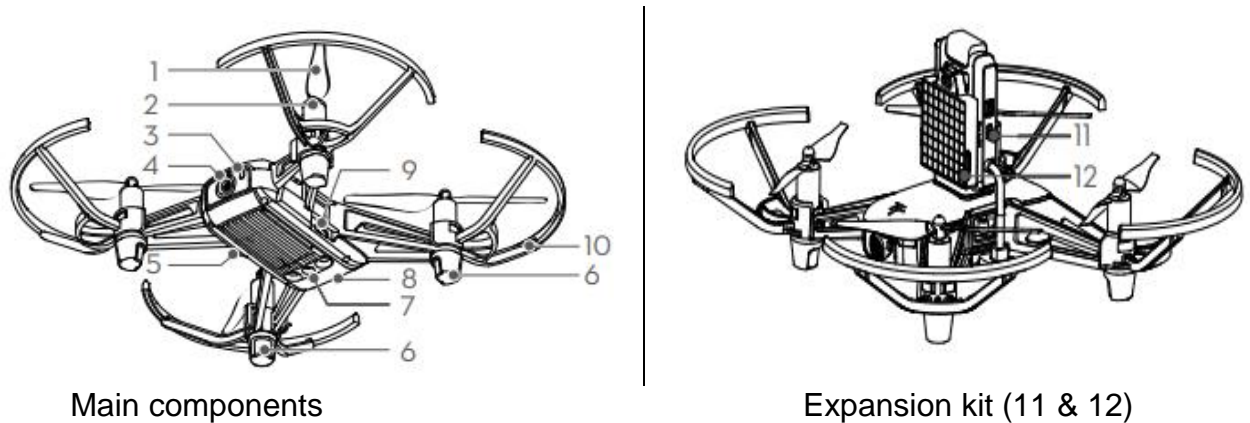


Figure 5.2 Diagram of Talent drone

The list of component is:

- | | |
|------------------------------|---|
| 1. Propellers | 8. Battery |
| 2. Motors | 9. Micro USB port |
| 3. Drone Status Indicator | 10. Propeller protectors |
| 4. Camera | 11. Open-Source Controller |
| 5. Power Button | 12. Dot-Matrix Display &
Distance Sensing Module |
| 6. Antennas | |
| 7. Vision Positioning System | |

5.2.2 Technical specifications of the Tello Talent drone

5.2.2.1 Physical characteristic

There dimensions and characteristics make it very manageable and easy to transport, see the specific in table 5.1.

Table 5.1 Physical characteristic

Descriptions	Values or type
Weight(Propeller Guards Included)	87 g
Dimensions	98x92.5x41 mm
Propeller	3 inches
Port	USB battery charging port
Operating temperature range	from 0° to 40° C
Operating frequency range	from 2.4 to 2.4835 GHz
Transmitter (EIRP)	<20 dBm (FCC) <19 dBm (CE) <19 dBm (SRRC)

5.2.2.2 Flight battery specification

The battery can be charged directly with an USB cable, or with a charging hub. The battery is totally protected by the following features and the specifications are in table 5.2.

1. Overcurrent/Overvoltage Protection: The battery stops charging if an excessive current/voltage is detected.
2. Over discharge Protection: Discharging stops automatically to prevent excessive discharge.
3. Short Circuit Protection: The power supply is cut automatically if a short circuit is detected.

Table 5.2 Battery Technical specification

Descriptions	Values or type
Battery	LiPo
Energy	4.18 Wh
Net Weight	25±2 g
Charging Temperature Range	41° to 113° F (5° to 45° C)
Capacity	1100 mAh
Voltage	3.8 V
Max Charging Power	10 W

5.2.2.3 Camera specification

The camera is located at the front of the drone and is capable to record HD videos and images, the specifications are in table 5.3

Table 5.3 Camera Technical specification

Descriptions	Values or type
Field of view	82.6°
Max Image Size	2592×1936 / 5MP
Video Recording Modes HD:	1280×720 30p
Video Format	MP4
Image format	JPG

5.2.3 Expansion kit

The expansion kit includes an open source controller, a dot matrix display, distance sensor module and an expansion board.

5.2.3.1 Open source controller

The open-source controller contains a Wi-Fi module dual frequency, a Bluetooth module and an Arduino open-source platform.

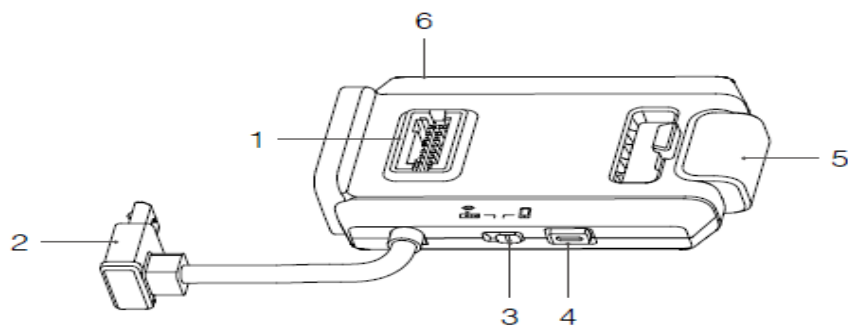


Figure 5.3 Open Source Controllers

1. Input and Output Expansion port used to connect the dot-matrix display and distance sensing module.
2. Micro USB cable used to connect to a power source 5V/2A
3. Switch used select between direct Wi-Fi Connection mode or via router mode through SDK.
4. Customizable Button used to activate the Bluetooth link.
5. Programmable RGB light used to show the status of Bluetooth connection.
6. Micro USB port used to connect to a computer for offline programming.

Table 5.4 Open source controller specification

Descriptions	Values or type
Weight	12.5 g (including the open source controller and ranging dot-matrix screen)
Dimensions	49.5×32×15.2 mm
Operating mode	Direct Connection mode and Router mode
Wi-Fi frequency band	2.4 GHz and 5.8 GHz
Bluetooth	2.4 GHz
MCU	ESP32-D2WD, dual-core at 160 MHz, 400 MIPS
Open source	Supports SDK development, Arduino, graphical programming, and MicroPython programming.
Scalability	14-pin extended interface (for I2C, UART, SPI, GPIO, PWM, and power supply)
Programmable LED indicator	Full-color LED

5.2.3.2 Dot-Matrix Display & Distance Sensing Module

This block integrates two modules an 8x8 dot-matrix display and a distance sensing, and allows the users to generate different graphics and pattern colors through programming. The below table summarize the specifications.

Table 5.5 Technical specification Dot-Matrix and Distance sensing module

Descriptions	Values or type
Dimensions	35.3×31.5×8.6 mm
Programmable dot-matrix LED indicator	8×8 red-and-blue indicator
Dot matrix driving function	IIC interface, automatic dot matrix scanning, 256-level adjustable overall brightness, and 256-level adjustable single-LED brightness
Ranging module	Infrared distance sensor (ToF)
Maximum distance measured	1.2 m (indoor with white targets)

5.2.3.3 Adapter extension board

The extension board consists of 14-pin extension port that that allows to ability Integrates external sensors.

. Table 5.6 Characteristic of the extension board

DIY adaptation	14-pin extended interface to the 2×7-pin 2.54-mm in-line pad, 2 power indicator positions, and 2 debugging indicator positions
----------------	--

5.3 Detail in flying and operating

Practically this drone is designed for indoor use, it can be used also outdoors application but the surrounding environment factors such as wind speed, weather, etc. affect the operating performance of the vision positioning system then limit the outdoor flying.

Table 5.7 flying mode parameters

Descriptions	Values or type
Max Speed	17.8 mph (28.8 kph)
Minimum speed	6.7 mph (10.8 kph)
Max Flight Time	13 minutes (0 wind at a consistent 9 mph (15 kph))
Maximum Sensing Distance of TOF	1.2 m (indoors with white wall)
Maximum distance of the flight	100 meters
Maximum flight height	30 meters

5.4 Vision Positioning System

The Tello drone is equipped with a Vision Positioning System which helps the aircraft maintain its current position and fly indoors or outdoors in windless conditions.

In addition The Vision Positioning System allows the drone to hover in place more precisely.

The main components of the Vision Positioning System are a camera and a 3D infrared module located on the underside of the aircraft.

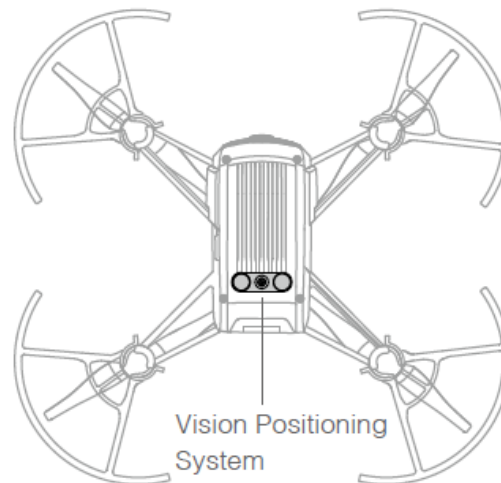


Figure 5.4 Vision Positioning System

5.5 Basic tools for operating the Tello talent drone

The Tello Talent drone supports several computers programming software, There are different operating system, platform and coding language for free or on the market to be used for operating and programing the drone.

Operating system	Windows, Mac, Linux, etc.
Platform	PyCharm, Anaconda, Visual code etc.
Coding languages	Python, Java, C++, Scratch, etc.

In this project, Windows, Pycharm and Python have been used due to easy availability.

5.6 SDK

The Tello talent drone incorporate last version of SDK 3.0 [6] which is more widespread than the previous version. This feature facilitates controlling the drone with predefined commands.

When the communication is established between drone and PC, Mac or Mobile device via Wi-Fi or Bluetooth, The SDK mode start to secure the logic connection between the Tello application and the drone by sending command in plaintext and waiting the device to reply (Ok, Error, Values, etc.)

5.7 Mobile Application on Phone and Tablet

There is an application developed by Shenzhen RYZE for Tello drone series, which allows controlling the drone manually over Bluetooth or Wi-Fi link.

This app is used to control the camera and other aircraft functions. It offers the possibility to viewing and managing photos and videos.

This application displays [10] all the necessary information related to drone and the flying status such as Battery Level, Wi-Fi Status, Bluetooth Status, Flight Speed, and Flight Altitude.

The applications allows configuring some parameters like Takeoff/Landing, Intelligent Flight Modes, Flight speed, VR , Bluetooth , and Wi-Fi settings. In addition, the Tello app can be used for activation and firmware update.



Figure 5.5 Tello application display screen

The app has various predefined intelligent flight modes such as Bounce mode, 8D Flips, Throw & Go, Up & Away, and EZ Shots which allow the used to perform various flying tasks and different basic missions,

5.8 Introduction to Tello Python programming

In this project, the Tello Python script which has been used is provided by DJITelloPy and it is available on <https://github.com/cocpy/Tello-Python> or other platform.

In the following paragraphs, there are some examples used for creating scripts to control drone Tello Talent.

5.8.1 Simple script for the basic drone maneuvering

As shown in the python code script, the first steps are importing all the classes from DJITelloPy package and Tello class and establish the Wi-Fi connection between the drone and the computer by the command **tello.connect**, The second step is initializing the objects and parameters, and then sending the commands to the drone. In this example of simple flying, the commands are takeoff, then move forward, rotate counterclockwise, and landing the code is in figure 5.6.

```
"Importing the class"
from djitellopy import Tello

"Establish the connection drone – Computer"
tello.connect

"Create Tello object"
tello.command = Tello()

"Create the commands"
tello.command.takeoff()
tello.command.move_forward(100)
tello.command.rotate_counterclockwise(90)
tello.command.rotate_clockwise(90)
tello.command.tello.land()

print("Connect to Tello Drone")
tello.connect()

battery_level = tello.get_battery()
print(f"Battery Life Percentage: {battery_level}")

print("Takeoff!")
tello.takeoff()

print("Sleep for 5 seconds")
time.sleep(5)

print("landing")
tello.land()
print("touchdown.... goodbye")
```

Figure 5.6 Script for maneuvering drone Tello Talent

The "tello.command", is only the name of the object of Command function used in this script, the user can use any other names for initializing the object of the function. Then associate the name with the commands by their function such as takeoff, rotate, move and land.

5.8.2 Simple script for reading the battery level

In this script, the reading battery percentage level is added; bellow the battery level script in figure 6.7.

```

"Importing the class"
from djitellopy import Tello

"Establish the connection drone – Computer"
tello.connect

"Create Tello object"
tello.command = Tello()

"Create the battery level object"
battery_level = tello.get_battery()

" read the battery percentage level"
print(f"Battery Life Percentage: {battery_level}")

```

Figure 5.7 Script for reading battery level

5.8.3 Simple script for recording video

In this script, we initialize the Drone cam to capture a picture and record a video then save them in the drive located on the autopilot.

In this case we have to import the clock, CV2, Thread and Stream class and initialize the file where the steaming will be recorded. See the figure 5.8.

```

import time, cv2
from threading import Thread
from djitellopy import Tello

tello = Tello()
tello.connect()

keepRecording = True
tello.streamon()
frame_read = tello.get_frame_read()

def videoRecorder():
    # create a VideoWrite object, recoring to ./video.avi
    height, width, _ = frame_read.frame.shape
    video = cv2.VideoWriter('video.avi',
        cv2.VideoWriter_fourcc(*'XVID'), 30, (width, height))

    while keepRecording:
        video.write(frame_read.frame)
        time.sleep(1 / 30)

```

```
video.release()

# we need to run the recorder in a seperate thread,
# otherwise blocking options
# would prevent frames from getting added to the video
recorder = Thread(target=videoRecorder)
recorder.start()

tello.takeoff()
tello.move_up(100)
tello.rotate_counter_clockwise(360)
tello.land()

keepRecording = False
recorder.join()
```

Figure 5.8 Script for capturing photos and video

CHAPTER 6. EYE TRACKING TOOLS DYNAVOX PCEYE

This chapter is focused on Tobii Dynavox eye tracking device, and contains a description of this product, the operating mode, technical specification and calibration procedure.

6.1 Overview about the Dynavox

The Dynavox eye tracking device (figure 6.1) is developed and manufactured by Tobii AB [1] Company, It is designed to replace the standard keyboard and mouse by the eye tracking, allowing the user to navigate and control the computer using only eyes movements.

The Dynavox solutions empower the user to individually create the most efficient way of computer interaction then allowing controlling all equipment connected to this computer.

In this project the Tobii Dynavox PCEye 5 have been used due to the availability but the desired device is Tobii Pro series which incorporate SDK Python technology and allow there integrations in the Tello applications with more features than the Dynavox.



Figure 6.1 Dynavox PCEye 5 tracking device

The Dynavox is delivered with all necessary accessories for physical installing and configuring such as magnetic mounting, USB-C to USB-A adapter and Tobii Dynavox Gaze point software.

6.2 Description

The Dynavox eye tracking device allows the user to explore all the functions of the computer with using only the eyes.

It is a clip-on eye tracker that lets the user interact in real time with the windows, button, etc; which are displayed on the screen, simply look at the screen and select commands by dwelling, zooming and/or clicking a switch. Alternatively, it allow fully the control of the mouse cursor directly with eye movements.

The Dynavox gives a fast, accurate and hands free way of accessing to the computer menu and bring greater independence for the person who cannot use their hands.

It is not complicated for operating or installing, it is delivered with some accessories that allow an easy installation closed to the screen,

For the operating, it is provided with a pack of software plug and play, and requires calibration before starting to navigate.

It can be used by several people, it is offer the possibility to create a profile individual for each person and the profile store calibration parameters and preferred setting for each user.

6.3 Technical specifications of the Dynavox PCEye

The technical specification for DYnavox PCE eye are detailed in next tables

6.3.1 Dimension and characteristic of Dynavox PCEye

There dimensions and characteristics make it very manageable and easy to transport. The standard specification and dimension are in this table 6.1

Table 6.1 Dimension and characteristic of Dynavox PCEye

Description	Values or types
Depth	8.2 mm
Height	15 mm
Width	285 mm
Weight	93 gms
Power Consumption	2.2 W typical average
Maximum Screen Size	Recommended up to 27"
Interface	USB Type C connector
Distance from user to the Eye Tracker	50 cm - 95 cm
Processing Unit	Tobii EyeChip™ with fully embedded processing

6.3.2 Gaze specifications

The following table 6.2 containing the gaze specification

Table 6.2 Gaze characteristic for Dynavox PCEye

Description	Values or types
Gaze Data Rate	33 Hz
Gaze Sample Rate	132 Hz
Data Streams	Gaze point, user position guide, Presence Primary camera image stream 33 Hz
Detected Gaze Interaction >30 Hz	98% for 95% of population

Data Flow and Data Rate	
Gaze Latency	25 ms (worst case image to signal latency)
Gaze Recovery	50 ms
Low Resolution Stream	280×280

6.4 The operating system required

The Dynavox PCEye software is compatible to all gaze viewer and others like TD Control, TD Snap, Communicator 5, Magic EyeFX, Snap Scene, Compass, etc.

In addition is compatible with Microsoft Windows 10 Eye Control and the applications which using Windows 10 eye tracking APIs. In table 7.3 resume the minimum required software and hardware configuration needed for operating the Dynavox PCEye.

Table 6.3 Operating system required by Dynavox PCEye

Description	Values or types
Computer and Processor	2.0 gigahertz (GHz) or faster, 6th generation Intel Core (i5/i7–6xxx) and later, or equivalent AMD 64 bit processor
Memory	8 GB
Hard Disk	450 megabyte (MB) available
USB	USB-C (USB-A via adapter)
Operating System	Windows 10 (64-bit) Windows 11 or newer
Eye tracker	HID compatible

In this project, Windows 10 64-bit have been used.

6.5 User positioning required

Eye tracking, or gaze interaction, is a technology used to see where a person is looking at on a computer screen. It can also be used to control a computer with your eyes instead of using a traditional keyboard and mouse, enabling individuals with physical and cognitive conditions to live richer and more independent lives.

The position of the user, screen and Dynavox device is fundamental for that the system works properly see figure 6.2.

The fixation of The Dynavox PCEye is designed to work optimally when it is parallel to the user's eyes at a distance from about 50 – 95 cm.

The optimal distance that user's position should be from the Dynavox PCEye differs and depending on the size of the screen. The user should be positioned at the optimal distance to allow for the best possible Computer Control.

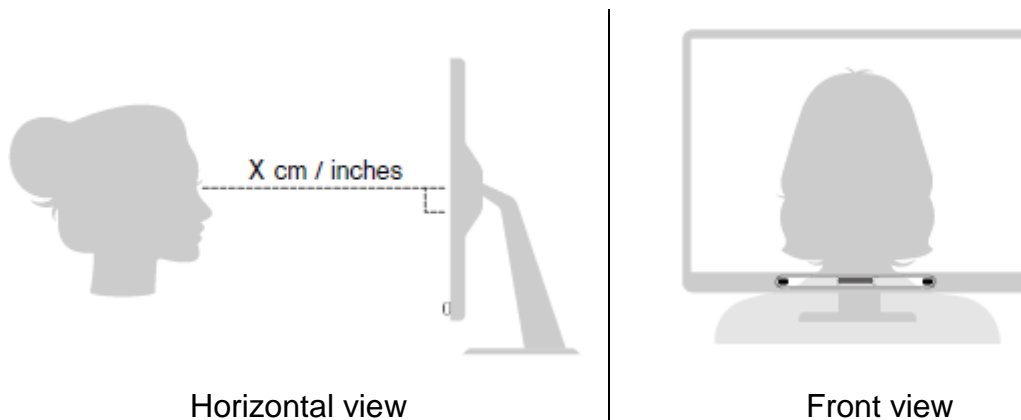


Figure 6.2 User, Screen and Dynavox positioning

The Dynavox is installed on the middle of lower part of the screen.

The user's eyes should be at the same level of the control screen center and the axe between eyes and center must be perpendicular to the screen surface.

6.6 Calibration requirement

For using The Dynavox eye tracking device with maximum accuracy possible, it requires to precede a calibration profile for each,

The calibration parameters are selectable and modifiable by the user as bellow:

- Calibration type , if it is accurate , simple or customized
- Number of Calibration points are from selectable from 1 , 2,5 or 9
- Stimulus gaze speed when moving from point to another
- Stimulus size
- Track eye used can be both, left or right

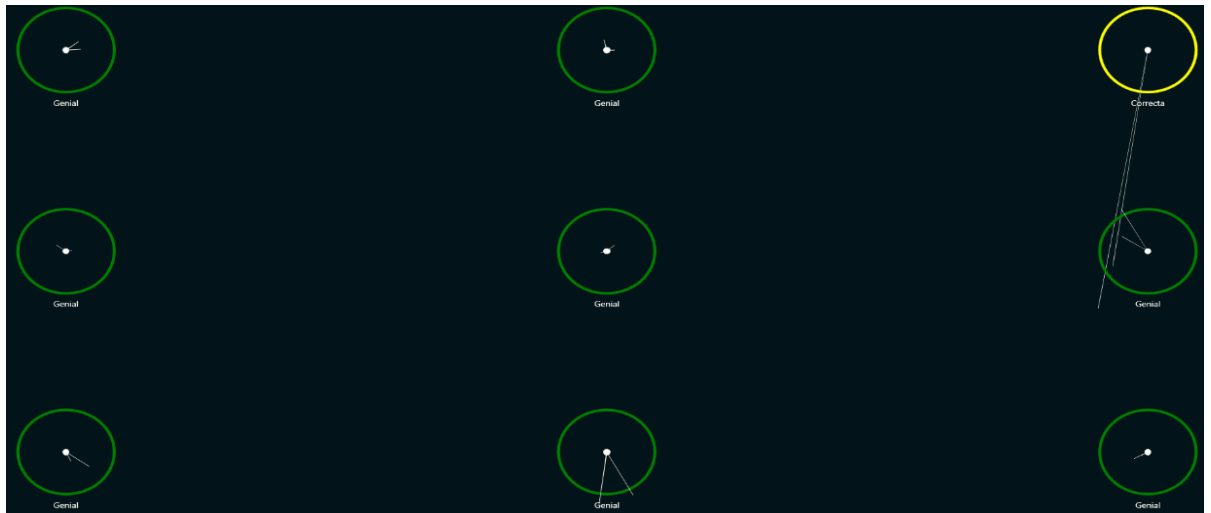


Figure 6.3 Calibrating points for Dynavox positioning

In addition, after launching the calibration, a windows pop up , the user have to look at each points demanded by the system, at the end the results of calibrating are displayed indicating the calibration status for each point such as Great, Good or No data, if it is needed to perform again the calibration as shown in this figure 6.3.

CHAPTER 7. INTEGRATION OF TELLO TALENT AND DYNAVOX EYE 5

In this section, we are going to implement and operate the knowledge that we have gained in the previous chapter's number 4 and 7, for developing a basic platform for controlling drones via the eyes,

This platform allows any person to be able to perform simple missions, even who don't have any prior knowledge of operating and maneuvering the Tello talent drone and also this chapter includes a survey for evaluating this work.

7.1 Introduction

For an interaction system, as discussed in the previous chapter, eye gaze tracking works the same way as human computer interaction (HCI), such as pointing on an object with the mouse and selecting by clicking.

In practice, eye gaze is used to move a circle overlay on a computer screen. The computer screen is divided into several restricted areas of interaction (AOI), and once the circle is on the target or on the desired button, the user must continue look at the target for a few seconds and then the Tobii pre-configured icons appears and confirm selection.

In the figure 7.1 there is an example of repartition of areas of interest (AOI) for performing the basic drone maneuver, in this case the screen is divided into six areas.

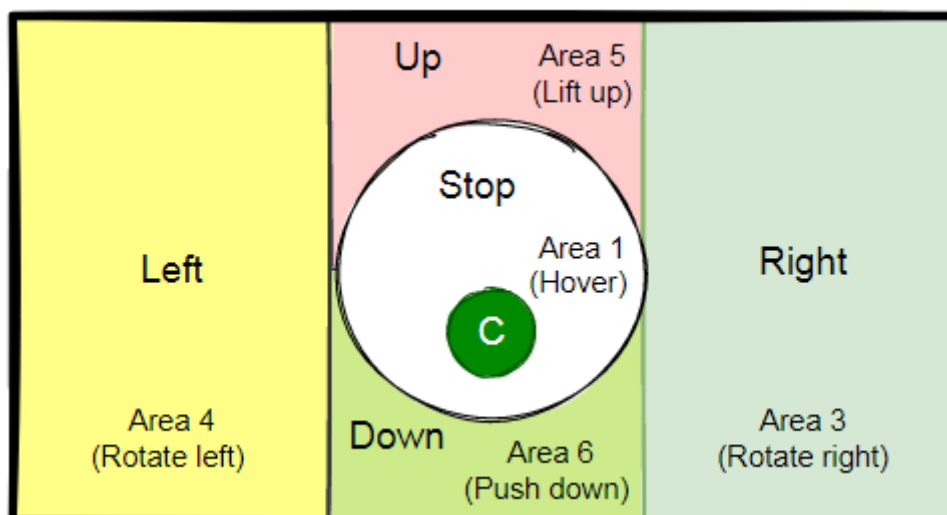


Figure 7.1 Example of controlling screen repartition

7.2 The hardware and software setting used in this solution

The hardware configuration involved in this solution is:

- Dynavox PCEy 5 device

- Tello Talent drone
- Laptop HP, Intel i7 2.80 GHz CPU running with Windows 10

The Software and packages [4] [5] used are:

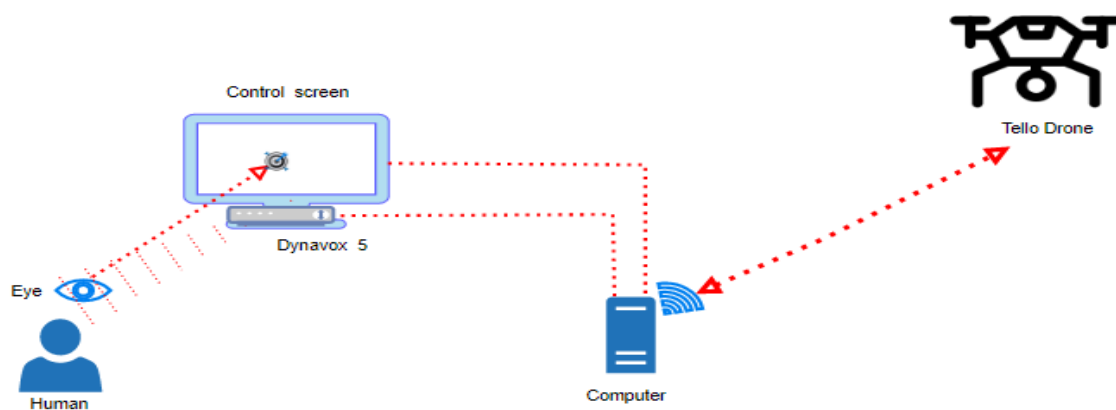
1. Pycharm version 2023.2.2
2. Python version 3.8.8
2. OpenCV version 4.5
5. Tkinter version 3.9.17

In addition, to complete the all software installation, the GazePoint driver from Tobii Dynavox must be supported by Windows and installed correctly.

The diagram in figure 7.2 shows the logic connection between different hardware.

The drone is connected to Computer via Wi-Fi or Bluetooth both are supported. The Dynavox PCEye 5 is connected to Computer by USB cable supplied by Tobii,

The light source generated by Dynavox PCEye 5 is directed to the eyes and the



gaze is tracked by the camera and the control screen,

Figure 7.2 Hardware diagram

7.3 Setting and calibrating Dynavox PCEy5

Dynavox PCEy5 is delivered with a pack of drivers for different uses; in this project the pack GazePoint is used for detecting the device by Windows and performing the basic configuration as discussed in chapter 7.

7.3.1 Setting basic parameters for Dynavox PCEy5

There are some steps to follow for installing and operating the Dynavox PCEy5 properly,

- Dynavox should be fixed correctly and positioned under the computer screen.

- Dynavox should be detected by the windows
- Installing the Gaze point software driver delivered by Tobii for Dynavox PCEy5, can be downloaded from Tobii web site
- Creating a user profile on the Dynavox application for each user
- Setting the parameters for the desired configuration for each user according to their ability
- Perform the calibration procedure

7.3.2 Calibration the position of the user, Laptop and Dynavox.

The Dynavox needs to be calibrated [1] for each user and the result of calibration should be successful as described in chapter 7- for achieving the greatest possible accuracy. Each user has a setup profile and the calibration data is stored in Tobii application, Nine calibration points are used in this demo.

7.4 Developing the programming code.

On the code programming side, to develop the interaction program that convert the eyes movement into a command to control the Tello talent drone, Python an Tkinter programming language are used in the cross-platform PyCharm environment.

The program (see figure 7.3) consists of importing a pre-set image (see Figure 8.4) containing a certain number of buttons and each button is labeled with an action and this action corresponds to the basic order for maneuvering the drone. The user looks at the desired button to activate the corresponding action.

More the button size is larger more we get the better accuracy, for some particular user who cannot concentrate their gaze to a smaller object, it is possible to use a limited number of button with large size and increase the accuracy.

For preparing this code, the picture ImageDemo is used as a screen background,

It has two coordinates Y and X representing respectively height (Max is 600) and width (Max is 800), the coordinate (0.0) is on the top and left of the screen.

This code is organized in three parts:

1. Import necessary packages.
 - Tkinter for creating button zone and adjusting the background picture
 - Djitellopy , this package is developed by Tello and used to define the drone command functions
2. Initializing the areas of interaction (AOI) for command
 - Delimiting the perimeter for each command
 - Labeling the command
 - Associating the command to the definition action
3. Creating the Tello definition action

- This parts used for creating the command and action for maneuvering drone

```
import tkinter as tk
from PIL import Image, ImageTk
from djitellopy import Tello

def scroll_start(event):
    print ('scroll ', event.x, event.y)
def scroll_move(event):
    if event.x >= 250 and event.x <= 450 and event.y >= 50 and event.y <= 150:
        print ('takeoff')
        command=action0()
    elif event.x >= 450 and event.x <= 650 and event.y >= 250 and event.y <= 450:
        print ('right')
        command=action1()
    elif event.x >= 50 and event.x <= 250 and event.y >= 250 and event.y <= 350:
        print ('left')
        command=action2()
    elif event.x >= 250 and event.x <= 450 and event.y >= 450 and event.y <= 550:
        print ('land')
        command= action3()
    else:
        print ('nada')

def action0():
    global tello
    tello = Tello()
    tello.connect()
    tello.takeoff()

def action1():
    global tello
    tello.move_right(50)

def action2():
    global tello
    tello.move_left(50)

def action3():
    global tello
    tello.land()

root = tk.Tk()
root.geometry('800x600')
root.title('Canvas Demo')

image = Image.open("ImageDemo.png")
image = image.resize((600, 800), Image.ANTIALIAS)

bg = ImageTk.PhotoImage(image)

canvas = tk.Canvas(root, width=800, height=600, bg='white')
canvas.create_image(0, 0, image=bg, anchor="nw")

canvas.pack(anchor=tk.CENTER, expand=True)
canvas.bind("<ButtonPress-1>", scroll_start)
```

```
canvas.bind("<Motion>", scroll_move)
root.mainloop()
```

Figure 7.3 Represents the code developed

The outputs of the code are:

- Pop up the "ImageDemo.png"
- Detect eye gaze and read the action button
- Send command to drone

7.5 Initial set-up and recommendation

After performing the steps mentioned in paragraph 8.3 and 8.4, the system is ready to integrate the Tello Talent drone into the scheme.

There are three steps to consider before running the script, The first step is to connect the drone Wi-Fi with the computer it a play and plug functionality for the talent drone, the second step is that the user should Keep the head position in the same level during the period of this exercise and thirdly, it is necessary to perform a new system calibration as explained above for taking account the new positions of the control screen and Dynavox PCEy5 relatively to the head of the user.

7.6 Running, Operating and controlling Tello talent by eyes

Switch the drone button to ON, wait few seconds until the Wi Fi connection between drone and Laptop establishes.

Now the drone is ready for flaying.

In this code we are using 4 commands that are corresponding to the labeled button by a text of the desired instruction to do.

Command 0: has double action to connect the laptop to drone, the second is to takeoff

Command 1: is used to order the drone to move right

Command 2: is used to command the drone to move left

Command 3: is used for landing

After launching the code, the picture "ImageDemo" pop up on the screen containing the buttons(see figure 8.4) that are corresponding to the programed coordinate on structure picture and the user looks at buttons and sends order to drone.

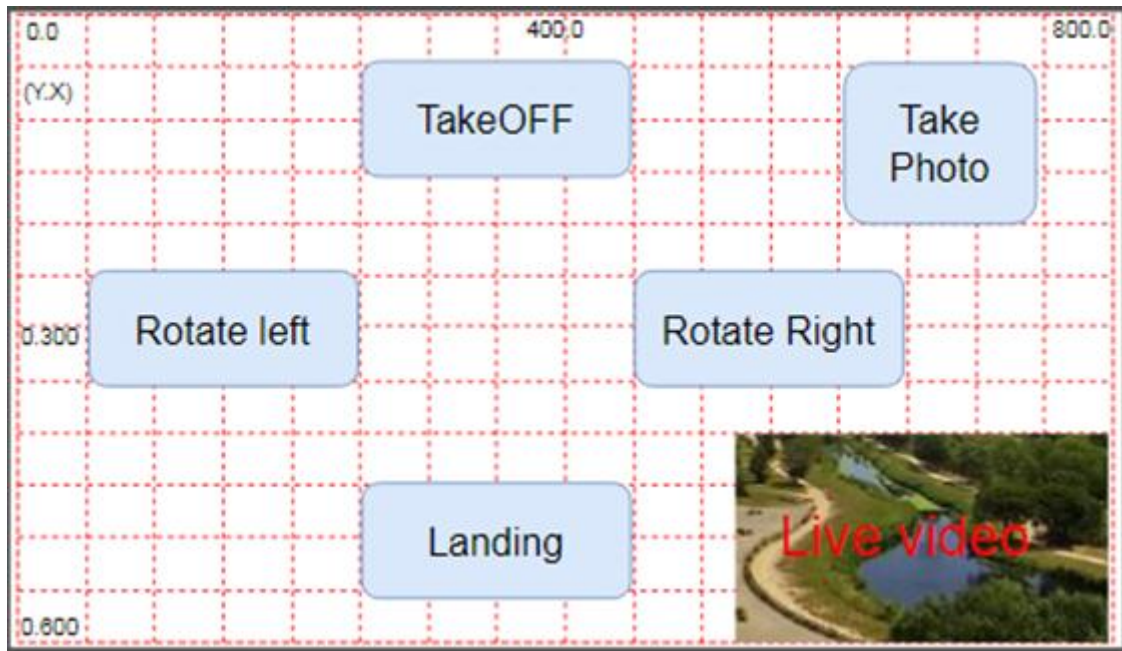


Fig 7.4 represents the action button on the ImageDemo in the controlling screen

The user can direct their gaze to any button, wait a few second, the button is activated and the command is sent to the drone accordingly.

In this programm, the drone is controlled as follows:

- While the user is looking at Takeoff button, the drone hovers.
- While looking at rotate left button, the drone moves to left at a pre-set constant translational velocity.
- While looking at rotate right button, the drone rotates right at a pre-set constant rotational velocity.
- While looking at landing button, the drone ascends at a preset constant translational velocity.

7.6.1 Upgrading the code for taking picture

For taking photos, we need to initialize the Drone camera, capture a picture and save it in the drive.

We need to import the OpenCV or cv2, create a Stream class and initialize a steaming address. Then adding new button "Taking photo" with the same manner as the previous buttons, and creating the action for recording photos, After takeoff, the user read the frame or picture from the live feed of the drone camera and save the frame as "frameX.jpg" into the drive by looking to "Taking photo" button.

7.7 Survey and results for this experiment.

Since this platform can be used by anyone, a survey was conducted to evaluate this work, and what was the user's reaction after using their eyes to control the drones.

7.8 Voluntaries description

12 people, 4 men and 8 women participated in this experiment; the majority of them are student with an average age of 22 years as it was done in a public bibliotheca. Nine of these participants performed the eye-tracking activity with their naked eyes, and three wore corrective glasses.

The test criteria is preforming a takeoff; a turn right, a turn left and a landing.

Only six persons who participated in this experiment successfully met the four test criteria. Four participates perform successfully three test criteria, and two are accomplishing two tests criteria.

The user gap for failed criteria is resumed on three points:

- Users are not familiarized with gaze movement tracking for taking into consideration the alignment between the optical and visual axes
- For selecting target require period and constant fixation of gaze on the target
- Users' eyes cannot keep the optimal distance from the eye tracking hardware

There is always a possibility that an eye movement may be done totally different from person to another, a person may move their head for pointing the gaze instead of only looking at the target and keeping the performance added by calibration parameters in the predefined norm.

For improving the accuracy of this application, it is recommended that the users must be familiarized to use the system in a proper way such as calibration, keeping head position fixe and etc.

7.8.2 Experiment data

The summary of this survey and experiments is presented in this table 7.1.

Table 7.1 Survey result

Descriptions	Experiment data
Gender	
Male	4
Female	8
Conditions when moving the drone	
With naked eyes	9
With a corrective glass	3
Person impression of the experiment	
Enjoy/fun	8
Not enjoy	2
Neutral	2

CHAPTER 8. INTEGRATION OF HAND GESTURE TO CONTROL TELLO TALENT

In this chapter, we are going to implement and operate all the knowledge that we have gained in the previous chapters, to build a hand gesture recognizer code using OpenCV and Python, the MediaPipe framework are used for the hand detection and gesture recognition respectively. After that, we move further to integrate hand recognition to control the drone and to perform various missions by using hand movements.

This project concentrates on how a system could detect, recognize and interpret the hand gesture recognition through computer vision.

8.1 System architecture

The general structure for any system to detect the hands and recognize the gestures can be schematized as shown in this figure. There are a user, camera and computer.

The user manipulate their fingers in order to generate the desired gestures; the camera takes the video frames of this gesture and the computer perform the processing and return the status of the gesture and convert it to a command.

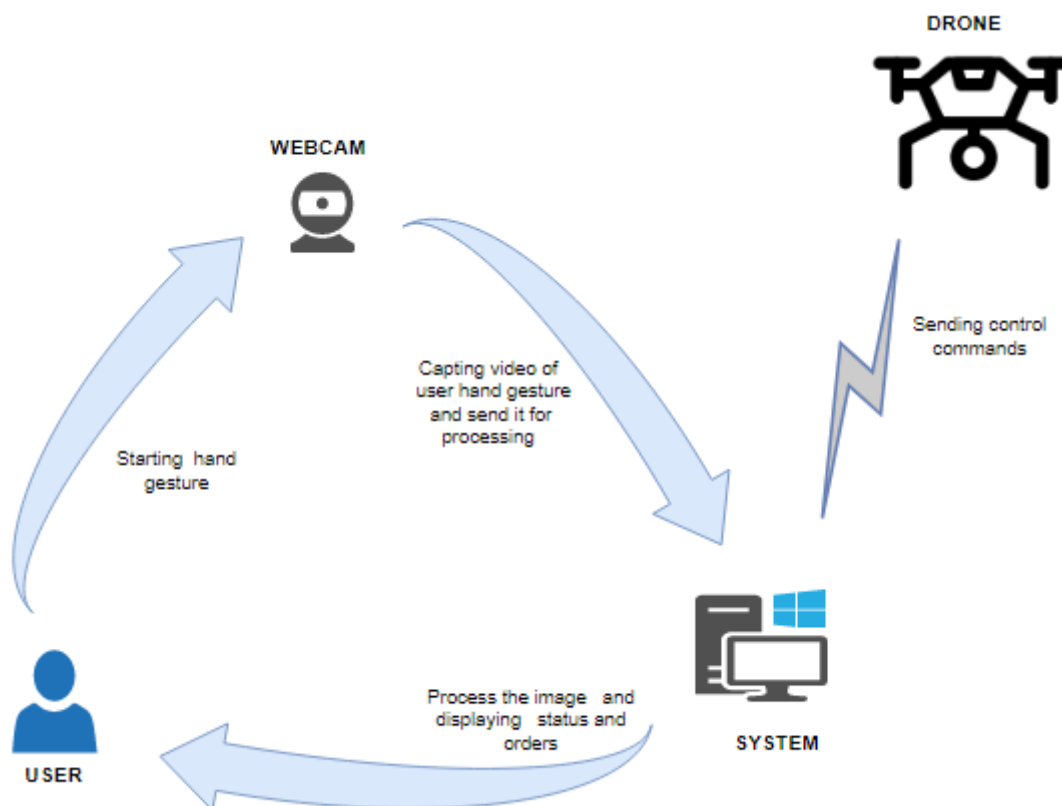


Figure 8.1. General structure of the hand gestures recognition system

8.2 Prerequisite for this part of the project

The hardware, software and packages needed for developing and running the application are summarized in next paragraphs.

8.2.1 Software and packages

The pycharm software and packages must be installed correctly without error, The package such as Python, OpenCV, MediaPipe and Numpy are preconfigured in Pycharm platform and are part of their dependencies.

- Pycharm version 2023.2.2 - Windows (exe)
- Python version 3.8.8
- OpenCV version 4.5
- MediaPipe version 0.8.5
- Numpy version 1.19.3
- Tkinter version 3.9.17

8.2.2 Hardware required

The minimum hardware configuration required is:

- 64-bit version of Windows 10
- 2 GB free RAM minimum, 8 GB of total system RAM recommended
- 3.5 GB hard disk space
- 1024x768 minimum screen resolution

In this project the Laptop HP have been used with this configuration:

Intel i7 2.80 GHz CPU, 16 GB system RAM, 350 GB hard disk space, 1024x 768 screen resolution and running with Windows 10, 64-bit.

8.3 Software installation steps

First step install Pycharm cross platform, which is a dedicated Python Integrated Development Environment (IDE) providing a various tools for developers, tightly integrated to create a convenient environment for productive Python and data science development.

Pycharm is available in two editions:

- Professional edition which is dedicated for companies and organization use and it is commercial; it provides an outstanding set of tools and features.
- Community edition which is an open-source project and has fewer features, it is the famous edition for educational and instrumental purpose.

The Pycharm software is available and downloadable from www.jetbrains.com
Second step install the packages Open CV, MediaPipe and Numpy by running the following instructions:

Run “pip install opencv-python” to install OpenCV
 Run “pip install mediapipe” to install MediaPipe
 Run “pip install numpy” to install Numpy

After installing packages, need to confirm if the pack version is compliant to the system operating requirements as discussed in paragraph 9.2.1.

8.4 Proposed system architecture for this application

The proposed software architecture used for hand gesture recognition consists of four phases, the figure 8.2.shows the block diagram for the solution.

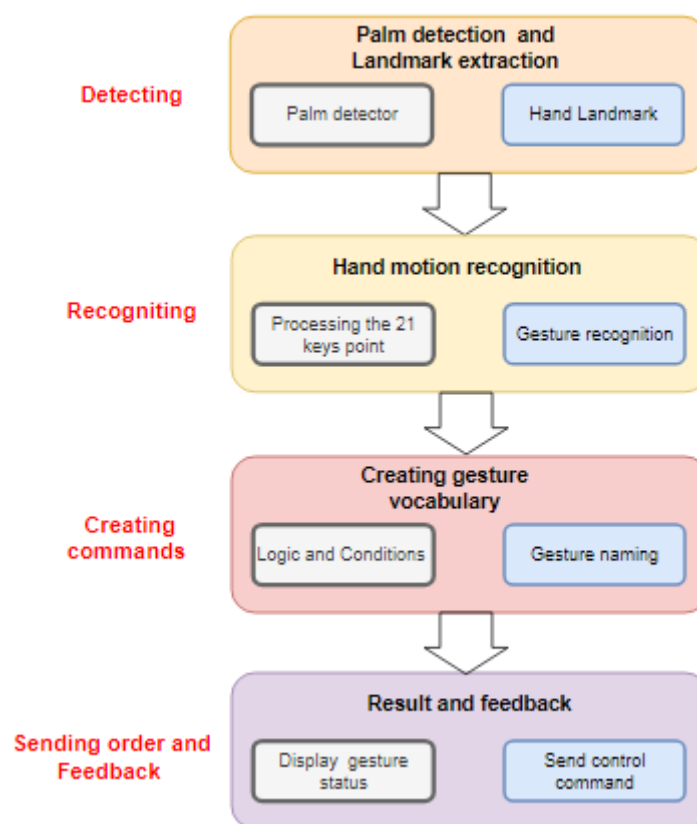


Figure 8.2 block diagram of the proposed system

8.5 Preparing the code for hand detection and fingers gesture recognition

Steps 1 – Importing the necessary packages.

First, we need to import all the relevant libraries necessary for building this code (see figure 8.3), we need to import the MediaPipe and some other python libraries Python, OpenCV, etc. as described in paragraph 9.2.

```
import cv2
import numpy as np
import mediapipe as mp
import djtellopy as tello
```

Figure 8.3 codes for importing libraries

Step 2 – Creating and initializing finger detector class in Mediapipe

In this step, we create an object-oriented FingerDetector class, and then we create some Constructors inside of this class (see figure 8.4); Constructors are used to initializing the state of the object and to assign values and parameters for this class.

- The self.mp_hands variable is assigned to the mp.solution.hands module that performing the hand detection functionality from MediaPipe. This line is for creating the object and stores it in self.mp_hands cell.
- The self.mpHands.Hands is used to assign the value for the model complexity, minimum detection confidence and minimum tracking
- The self.mp_drawing variable is assigned the mp.solutions.drawing_utils module which used to draw the detected 21 key points and connections for each hand.
- The self.hands variable creates an instance of the Hands class from npHands. This is where hand detection and tracking will be performed based on the specified parameters.

```
class FingerDetector:
    def __init__(self):
        self.mp_drawing = mp.solutions.drawing_utils
        self.mp_drawing_styles = mp.solutions.drawing_styles
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(
            model_complexity=0,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5)
```

Figure 8.4 Class

- The model complexity of the landmark is 0 or 1, Landmark accuracy and inference latency generally go up with the model complexity by default the value is set to 1.
- For the minimum detection confidence value for the hand detection model is between 0.0 and 1.0, by default the value is set to 0.5 which is considered successful in palm detection model.
- For the minimum tracking confidence value ([0.0, 1.0]), this value is related to the bounding box threshold between hands in the current frame and the previous frame from the landmark-tracking module, if the

tracking fails, Hand Landmark triggers hand detection otherwise, it skips the hand detection, the default value is set 0.5.

Step 3 – Reading frames from a webcam and creating landmarks table

In this part the coding consist capturing images, creating and processing as defined in the first part of the code (see figure 9.5),

- The VideoCapture is used to create an object and pass an argument (0) that is the camera identification. In this case, we have only one webcam connected so the argument value is by default set to zero,
- The image.flags.writeable function is used to read each frame received from the webcam.
- The cv2.cvtColor() function is used to convert the frame from BGR to RGB format because the MediaPipe works with RGB images but OpenCV reads images in BGR format.
- leftHandLandmarks and rightHandLandmarks are used for containing a lists of the landmark identifications for the fingertips (as described in chapter 5), each landmark has two values X and Y being normalized to [0.0,1.0] and corresponding to the coordinate in image. These landmarks lists will be used later for additional hand analysis or gesture recognition.

Step 4: Implementing hand detection and extracting hand landmark positions

In this step, the palm detector and hanlandmark detector are applied in real time on the video frames received from the camera for localizing the 21 keys point for each hand and extract the positions of individual hand landmarks, and store them in the leftHandLandmarks and rightHandLandmarks lists which are created above in step 3. At the end of this step, it returns the hand image detected with drawing connection and the list of landmarks coordinates, see second part of the code (see figure 9.5),

- The “results” variable is assigned to self.hands.process(image) function that is used to detect the hand on the image, the input for this function is the image or frames of the video.
- If results.multi_hand_landmarks , this loop is used to identifying the Handedness Left and right in case of two hands are detected on the image, then it returns two hands.
- handLabel is used to contain the Handedness it can be “right” or “left”
- The rightHandLandmarks.append ([landmarks.x, landmarks.y]) is used to detect landmark positions and store them in the list.
- Self.mp_drawing.draw_landmarks is used to draw all the landmarks in the frame knuckles with connection lines.
- The return “leftHandLandmarks, rightHandLandmarks, image” is used to display the image with the drawing of the keypoints and the list of X, Y that corresponding to handmarks position.

```

image.flags.writeable = False
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
results = self.hands.process(image)
image.flags.writeable = True
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
leftHandLandmarks = []
rightHandLandmarks = []
if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        handIndex =
results.multi_hand_landmarks.index(hand_landmarks)
        handLabel =
results.multi_handedness[handIndex].classification[0].label
        if handLabel == "Left":
            for landmarks in hand_landmarks.landmark:
                leftHandLandmarks.append([landmarks.x,
landmarks.y])
        if handLabel == "Right":
            for landmarks in hand_landmarks.landmark:
                rightHandLandmarks.append([landmarks.x,
landmarks.y])
        self.mp_drawing.draw_landmarks(
            image,
            hand_landmarks,
            self.mp_hands.HAND_CONNECTIONS,

            self.mp_drawing_styles.get_default_hand_landmarks_style(),

            self.mp_drawing_styles.get_default_hand_connections_style())
    return leftHandLandmarks, rightHandLandmarks, image

```

Figure 8.5. Code for reading frame, detecting hand and extracting landmark position

The figure 8.6 and table 8.1 are representing the output of this coding step, is to display the leftHandLandmarks and the rightHandLandmarks in the image with the connection drawing and the list of X, Y, Z that corresponding to landmarks position in the image frame coordinate.

The coordinate X is the landmark position in the horizontal axis, coordinate Y is the landmark position in the vertical axis, and Z is the landmark depth from the camera.

This result will be used as an input of next step.

The 21 keys points are localized on the hand picture and the values of landmarks are extracted

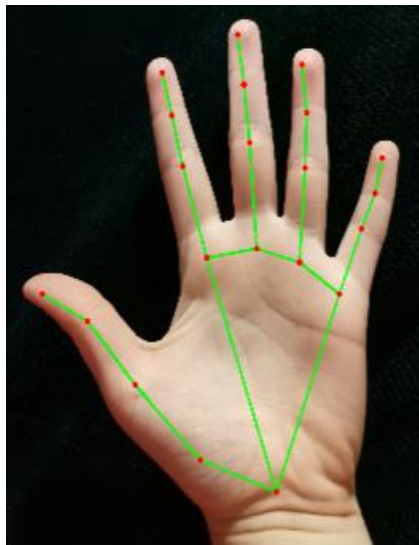


Figure 8.6, landmarks are localized on the hand picture

Table 8.1, The values (X,Y,Z) of landmarks on picture coordinates;

Landmark key point	x	y	z
0	0.19527593	0.6772005	-7.258559e-05
1	0.263733	0.63610333	-0.039326552
2	0.3196355	0.5412712	-0.058143675
3	0.3613177	0.4677803	-0.075389124
4	0.39756835	0.43665695	-0.093960665
5	0.26121178	0.3753401	-0.030742211
6	0.28375435	0.26732442	-0.061761864
7	0.29418302	0.19642864	-0.08401911
8	0.30149087	0.13136405	-0.1029892
9	0.21288626	0.3534055	-0.032817334
10	0.21505088	0.22275102	-0.058613252
11	0.2152167	0.1385001	-0.08102116
12	0.21389098	0.06872013	-0.09661316
13	0.16952133	0.36720178	-0.04239379
14	0.15782069	0.2474725	-0.07075888
15	0.15325233	0.16784605	-0.09313752
16	0.15079859	0.102125764	-0.108897485
17	0.12903559	0.41147107	-0.05464202
18	0.09621665	0.3332698	-0.08286363
19	0.07500376	0.28210545	-0.10173174
20	0.056006864	0.2304765	-0.11720086

Step 5 – Recognizing the hand gestures

In this step we will see how to create certain numbers and sign languages of hand gesture that are suitable to be used for one hand in a time.

A list of criteria can be defined by comparing the position of each landmark according the drawing of landmark on image coordinate as shown in figure 3 and 4 and using as input the list of X, Y of hand landmark positions extracted in step 4.

Each hand landmark is identified by a number and a values (X, Y) that corresponding to the image frame coordinate and addition information specifying if it is the right or left hand , example “leftHandLandmarks[8][1]” is

the coordinate X and Y of land mark number 8, [1] is for specifying the hand right.

The variable “fingerCount” is used to contain the value corresponding to the sum of criteria values; if the condition is fulfilled with criteria the value of “fingerCount” is incremented by one otherwise zero.

The initial value of “fingerCount” is set to zero and it is common for right or left hand,

In this code we are using only the 8 landmark for right and 8 of left hand,

`leftHandLandmarks[8][1] < or > leftHandLandmarks[6][1]`

`leftHandLandmarks[12][1] < or > leftHandLandmarks[10][1]`

`leftHandLandmarks[16][1] < or > leftHandLandmarks[14][1]`

`leftHandLandmarks[20][1] < or > leftHandLandmarks[18][1]`

and the same for right hand, In this combination The values of “fingerCount” can be between 0 till 8, so we can create 9 signs of hand gesture.

This hand gesture recognition will cover nine poses which are very simple and can be manipulated by any person without prior training only maneuvering the fingers index, middle, ring and pinky for both hands, the finger thumb and twist are not involved, then the code allows creating nine sign languages of hand gesture.

The main objective of this step is to associate a state with each finger and then create the gesture through some expression predefined based on finger states as straight, bent and crossed, each hand gesture directly set a command out for controlling drones, the table 8.2 is summarizing the gestures that a user can choose through a status of the index, middle, ring and pinky fingers..

Table 8.2. Hand gesture recognition 1



Gesture name	Gestures description	Left and Right landmarks conditions	Finger Count	Hands gesture	
				Right Hand	Left Hand
A	Left hand, The index, middle, ring and pinky fingers, all are bent. Right hand is not on the scene	<code>leftHandLandmarks[8][1] < [6][1]</code> <code>leftHandLandmarks[12][1] < [10][1]</code> <code>leftHandLandmarks[16][1] < [14][1]</code> <code>leftHandLandmarks[20][1] < [18][1]</code>	0 0 0 0	Right hand is out of picture	
FingerCount			0	Pose number 1	
B	Left hand, The index is straight and the middle, ring and pinky are bent. Right hand is not on the scene	<code>leftHandLandmarks[8][1] > [6][1]</code> <code>leftHandLandmarks[12][1] < [10][1]</code> <code>leftHandLandmarks[16][1] < [14][1]</code> <code>leftHandLandmarks[20][1] < [18][1]</code>	1 0 0 0	Right hand is out of picture	
FingerCount			1	Pose number 2	

Table 8.2. Hand gesture recognition 2


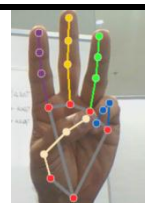

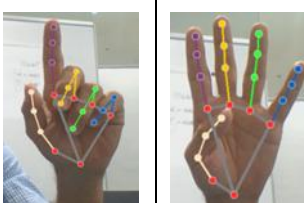
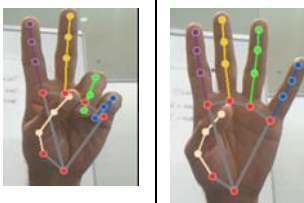
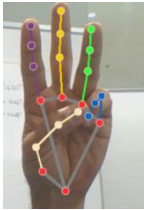



Gesture name	Gestures description	Left and Right landmarks conditions	Finger Count	Hands gesture	
				Right Hand	Left Hand
C	Left hand, The index and middle are straight and the ring and pinky are bent. Right hand is not on the scene	leftHandLandmarks[8][1] > [6][1] leftHandLandmarks[12][1]>[10][1] leftHandLandmarks[16][1]<[14][1] leftHandLandmarks[20][1]<[18][1]	1	Right hand is out of picture	
			1 0 0		
FingerCount			2	Pose number 3	
D	Left hand, The index, middle and ring are straight and the pinky is bent. Right hand is not on the scene	leftHandLandmarks[8][1] > [6][1] leftHandLandmarks[12][1]>[10][1] leftHandLandmarks[16][1]>[14][1] leftHandLandmarks[20][1]>[18][1]	1	Right hand is out of picture	
			1 1 0		
FingerCount			3	Pose number 4	
E	Left hand, The index, middle, ring and pinky are straight. Right hand is not on the scene	leftHandLandmarks[8][1] > [6][1] leftHandLandmarks[12][1]>[10][1] leftHandLandmarks[16][1]>[14][1] leftHandLandmarks[20][1]>[18][1]	1	Right hand is out of picture	
			1 1 1		
FingerCount			4	Pose number 5	
F	Left hand, The index, middle, ring and pinky are straight. Right hand, The index is straight and the middle, ring and pinky are bent	leftHandLandmarks[8][1] > [6][1] leftHandLandmarks[12][1]>[10][1] leftHandLandmarks[16][1]>[14][1] leftHandLandmarks[20][1]>[18][1] rightHandLandmarks[8][0] > [6][0] rightHandLandmarks[12][0]<[10][0] rightHandLandmarks[16][0]<[14][0] rightHandLandmarks[20][0]<[18][0]	1		
			1 1 1 1 0 0 0		
FingerCount			5	Pose number 6	
G	Left hand, The index, middle, ring and pinky are straight. Right hand, The index and middle are straight and the ring and pinky are bent.	leftHandLandmarks[8][1] > [6][1] leftHandLandmarks[12][1] > [10][1] leftHandLandmarks[16][1] > [14][1] leftHandLandmarks[20][1] > [18][1] rightHandLandmarks[8][0] > [6][0] rightHandLandmarks[12][0]> [10][0] rightHandLandmarks[16][0]< [14][0] rightHandLandmarks[20][0]< [18][0]	1		
			1 1 1 1 1 0 0		
FingerCount			6	Pose number 7	

Table 8.2. Hand gesture recognition 3

Gesture name	Gestures description	Left and Right landmarks conditions	Finger Count	Hands gesture	
				Right Hand	Left Hand
H	Left hand, The index, middle, ring and pinky are straight. Right hand, The index, middle and ring are straight and the pinky is bent.	$\text{leftHandLandmarks}[8][1] > [6][1]$ $\text{leftHandLandmarks}[12][1] > [10][1]$ $\text{leftHandLandmarks}[16][1] > [14][1]$ $\text{leftHandLandmarks}[20][1] > [18][1]$ $\text{rightHandLandmarks}[8][0] > [6][0]$ $\text{rightHandLandmarks}[12][0] > [10][0]$ $\text{rightHandLandmarks}[16][0] > [14][0]$ $\text{rightHandLandmarks}[20][0] < [18][0]$	1 1 1 1 1 1 1 0		
FingerCount			7	Pose number 8	
I	Left hand, The index, middle, ring and pinky are straight. Right hand, The index, middle, ring and pinky are straight	$\text{leftHandLandmarks}[8][1] > [6][1]$ $\text{leftHandLandmarks}[12][1] > [10][1]$ $\text{leftHandLandmarks}[16][1] > [14][1]$ $\text{leftHandLandmarks}[20][1] > [18][1]$ $\text{rightHandLandmarks}[8][0] > [6][0]$ $\text{rightHandLandmarks}[12][0] > [10][0]$ $\text{rightHandLandmarks}[16][0] > [14][0]$ $\text{rightHandLandmarks}[20][0] > [18][0]$	1 1 1 1 1 1 1 1		
FingerCount			8	Pose number 9	

The user interface created has been designed to be not complicated and user friendly, it is for everybody even who does not have a prior knowledge in this domain; the functionalities that have been included are intended to make the controlling operation process as simple as possible for the user.

8.6 implementing the main application for drone Control with Hand Gestures

In this section, We merge the classes created in the previous steps with the Tello talent classes for developing main program based on the nine gestures for drone controls,

This main programming code consists of using different state related to djitellopy functions and the program created for fingers gesture recognition see figure 8.7,

At the end the solution allows users to control the Tello drone only by maneuvering their fingers.

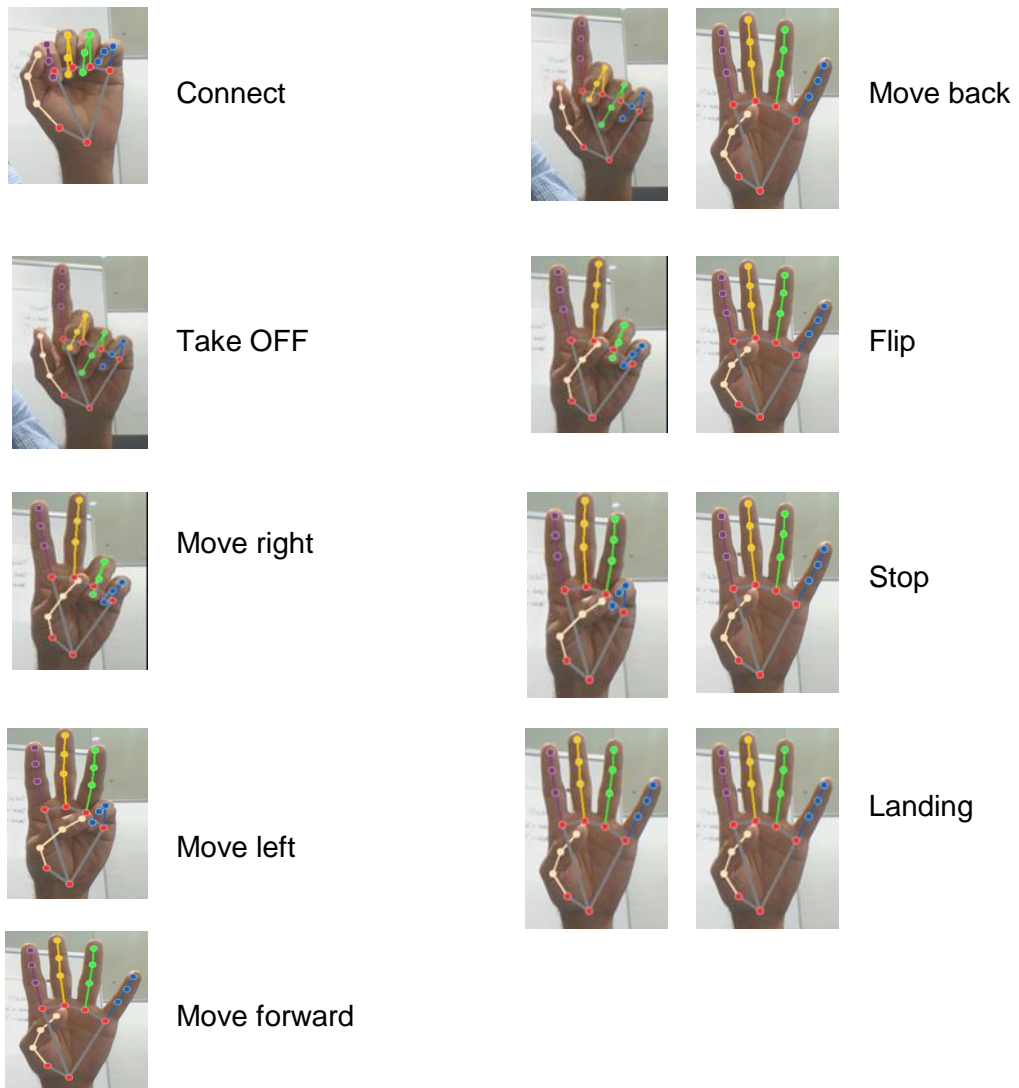


Figure 8.7 Gesture vocabulary used in this application

In this step, we associate to each FingerCounter value an action to do for controlling drone, see figure 8.8.

```
def setDirection( code):
    if code ==0
        return 'Connect'
def action0():
    global tello
    tello = Tello()
    tello.connect()
    global tello
    if code == 1:
        return 'Takeoff'
def action1():
    global tello
    tello = Tello()
    tello.takeoff()
    elif code == 2:
        return 'Move right'
def action2():
    global tello
    tello.move_right(50)
    elif code == 3:
        return 'Move left'
def action3():
    global tello
    tello.move_left(50)
    elif code == 4:
        return 'Move forward'
def action4():
    global tello
    tello.move_forward(50)
    elif code == 5:
        return 'Move back'
def action5():
    global tello
    tello.move_back(50)
    elif code == 6:
        return 'flip'
def action6():
    global tello
    tello.flip50)
    elif code == 7:
        return 'Stop'
def action7():
    global tello
    tello.stop()
    elif code == 8:
    tello.land()
        return 'Landing'
def action8():
    global tello
    tello.landing(50)
    else:
        return "
```

Figure 8.8. Associating the FingCont to a predefined djitello.py action

Step 7- Implementing the solution

When the user changes the pattern, before starting to establish the new pattern, the system waits some time and ignores the 8 video frames received, see figure 9.8.

```
def practising():

    prevCode = -1
    cont = 0
    running = True
    cap = cv2.VideoCapture(0)
    print ('camara preparada')
    while running:
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue
        code, img = detector.detect(image)
        img = cv2.resize(img, (800, 600))
        img = cv2.flip(img, 1)
        # if user changed the pattern we will ignore the next 8 video frames
        if (code != prevCode):
            cont = 4
            prevCode = code
        else:
            cont = cont - 1
            if cont < 0:
                # the first 8 video frames of the new pattern (to be ignored) are done
                # we can start showing new results
                direction = setDirection(code)
                cv2.putText(img, direction, (50, 450),
                    cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 0, 255), 10)
                if code == 7:
                    running = False

            cv2.imshow('video', img)
            cv2.waitKey(1)
            cv2.destroyAllWindows('video')
            cv2.waitKey(1)

    practising()
```

Figure 8.9 Operating code

8.7 Run and Play

After setting up the setup, you can run the program and enjoy the main application of the hand gestures for controlling drone as showed in figure 8.9. All the codes related to this project are available on GitHub (see Annex), the user can access via the link that stated in the annex.

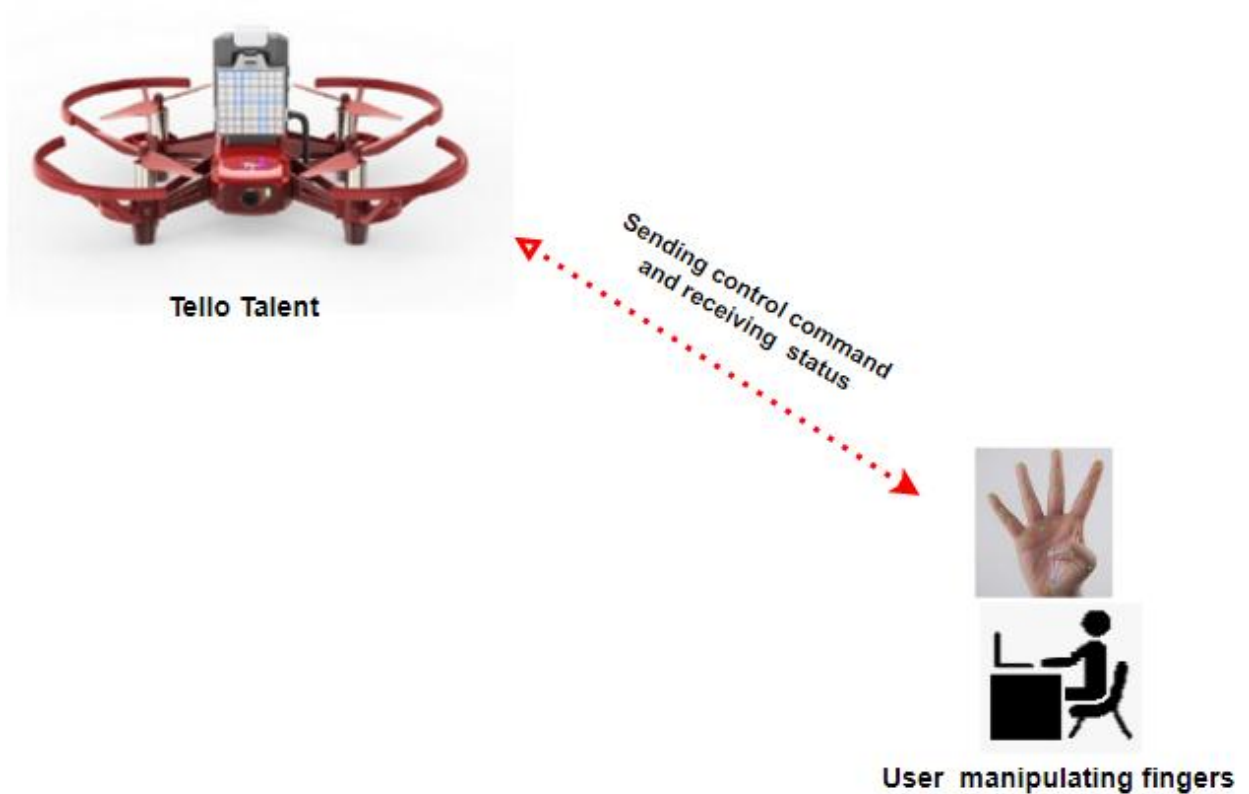


Figure 8.10 Work and test setup

8.7.1 Material and tools required

The material and tools needed for performing the simulation of the drone control by hand gesture are:

- 1 X Tello Talent
- 1 X Laptop as described in 9.1.2 paragraph
- Software tools and packs as described 9.1.1 paragraph

8.7.2 Some steps to be followed.

- Switch Drone to ON
- The Wi-Fi connection between drone and Laptop is established automatically
- Prepare the hand gesture recognition table (Table 8.1)
- Run the program and play according to the hand gesture defined in the table

8.8. Results and analysis

The nine gestures recognized in this paper are used to generate the actions of drone control. The simulation and result are performed by ten volunteers and done in different background light intensity, see table 8.3. it has given the best results when the background is clear and the light is medium, the rate of recognition reached 100%. When the background brighter than skin color, the recognition is about 85%,

This smaller degradation is caused by lighting environment and condition and also possible that the users don't fully respect the gesture form required for each code.

In the general, using MediaPipe for implementing machine learning on finger detection and gesture recognition through a python coding program and creating a user guide application using hand gestures as a command for display information and controlling drones achieves good performance.

Table 8.3 Hand Gesture Recognition Rate.

Gesture name as defined in previous table	Corresponding drone action	Number of volunteer	Number of recognized gestures	Recognition rate (%)	Total recognition rate (%)
Gesture A	connect	10	8	80%	85%
Gesture B	Takeoff	10	10	100%	
Gesture C	Move right	10	10	100%	
Gesture D	Move left	10	9	90%	
Gesture E	Move forward	10	10	100%	
Gesture F	Move back	10	9	90%	
Gesture G	Flip	10	10	100%	
Gesture H	Stop	10	9	90%	
Gesture I	landing	10	10	100%	

CHAPTER 9. COMBINING EYE TRACKING SYSTEM AND HAND GESTURE TO CONTROL TELLO TALENT

This chapter describes a method to combine the use of eye gaze and hand tracking to provide a way to control a drone,

The main objectives that will be achieving for this project, is to establish a complete system for interpreting hand gesture recognition through computer vision and a gaze tracking thought an eye tracking device using Python and others packs and libraries.

9.1 System architecture

In this interaction system, the eye gaze is used to move a circle overlay in the picture window called area of interaction (AOI) that containing a list of button corresponding to commands, on a computer screen as described in chapter 8, the hand gesture is used to create the signs of commands and the return is displayed on the hand picture windows.

The control commands is mapped in both windows, area of interaction for gaze tracking and hands picture for gesture tracking, At starting the control is done by hand tracking, If there is no hand detected at the input or gesture no performed successfully, the control commands are handovered to gaze tracking, when the hand finger is detected again, the hand tracking take over de control, When the gaze is out-of- AOI, the hand finger takes over till the hand is not detected.

The solution is based on the priority for hand tracking.

The communications between different devices [11] are showed in figure 9.1

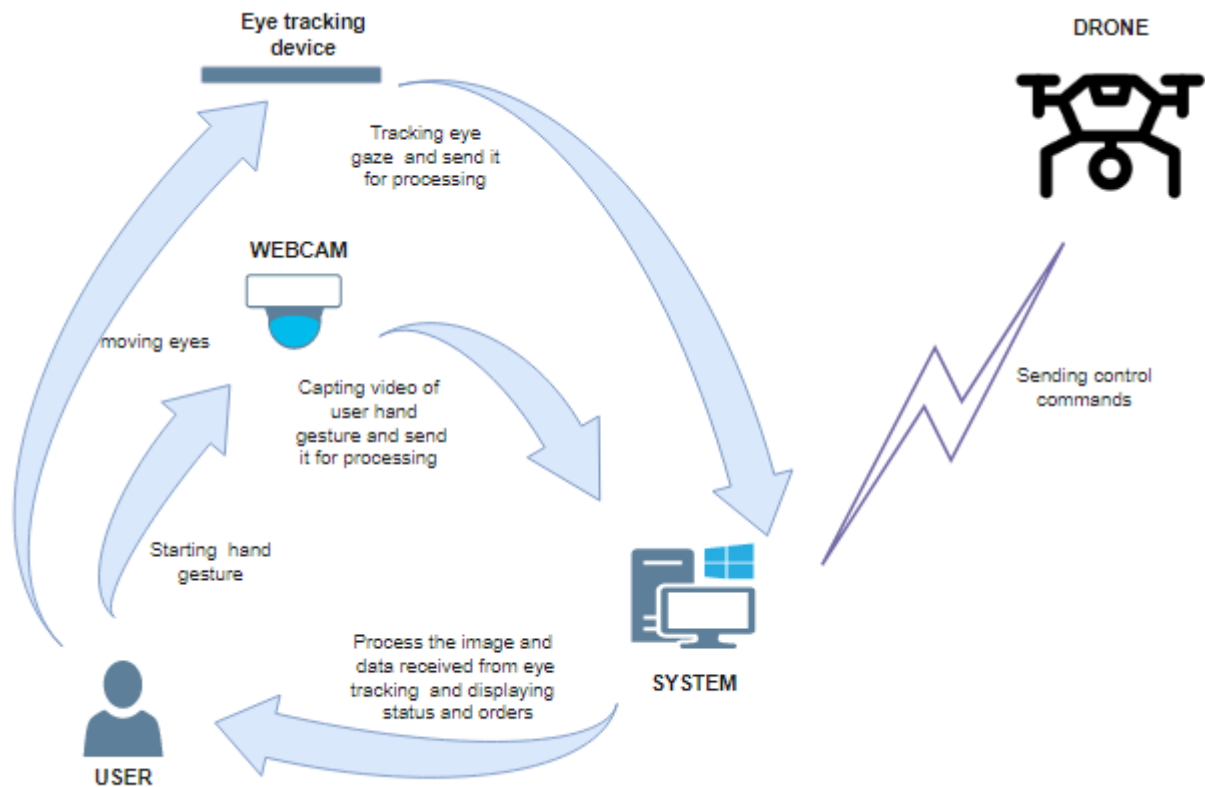


Figure 9.1 Proposed system architecture

9.2 Implementation

The same basic hardware and software which are used in chapter 8 and chapter 9 are needed for the global solution as.

- Dynavox PCEy 5 device
- Tello Talent drone
- Laptop HP, Intel i7 2.80 GHz CPU running with Windows 10 with 1620x1080 pixels

9.2.1 Eye Gaze Tracking

For eye gaze tracking hardware, we use a commercial Tobii Dynavox PCEye 5 eye tracking device which has a typical optical axis, it is used because it is readily available in the university lab.

It is positioned directly under the laptop screen with an upward vertical tilt of 25° approximated.

The Dynavox needs to be calibrated for every user in order to achieve its potential accuracy. We implement a 9 points calibration; the calibration tool is delivered with the device driver.

9.2.2 Hand Tracking

For hand tracking hardware, we use laptop camera.

9.2.3 Area of interest (AOI) Design

The window of AOI is not the full size of the screen because we reserved some margin for left, right, top and bottom and also the camera window so that:

- The user can maneuver their hands and maintaining the detection.
- Hand movements have minimal impact on users head and eyes position not influence by hand movement.

Generally, a bigger AOI can affect hand movement, It is recommended to adjust the position of Camera, eye tracking device and user hands and eyes, this is can proved by a rigorous calibration.

9.2.4 Software Design

On the software side, to develop the interaction system program, using python programming language in Pycharm platform.

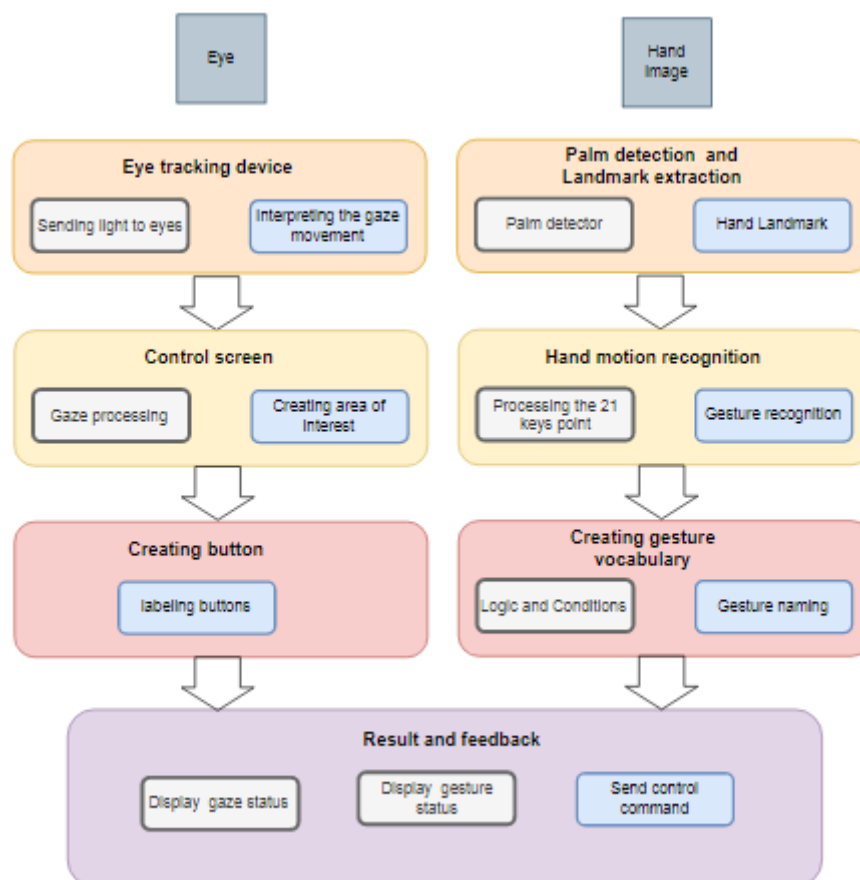


Figure 9.2 software architecture

When combining two type of tracking, we need to have a modular and systematic way to code the interaction system program, to manage the inputs

from gaze tracking and hand tracking, and implementing the rules on the determine which type of tracking is in effect as well as accommodation feature expandability and change of hardware. The overall view of the software model is shown in Figure 10.2.

The algorithm consists of privileging the commands received from the hand gesture, if the gesture is not detectable or not able to generate the FingerCont values, the application switch the commands received from the gaze, when the gesture redetected again correctly, the gesture command take over the rule. See figure 9.3.

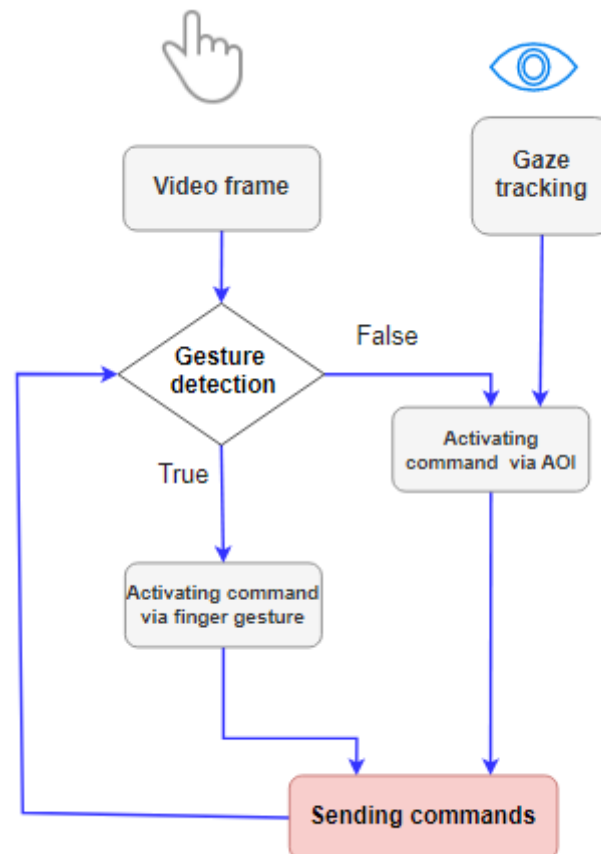


Figure 9.3 flow concept for selecting the between gaze tracking and fingers gesture

9.2.5 The interaction system and the screen display

In Screen shot of the interaction system in figure 9.4, there are two main windows, the first window is for streaming the hands gesture, the second window is the area of interest (AOI) that containing the buttons

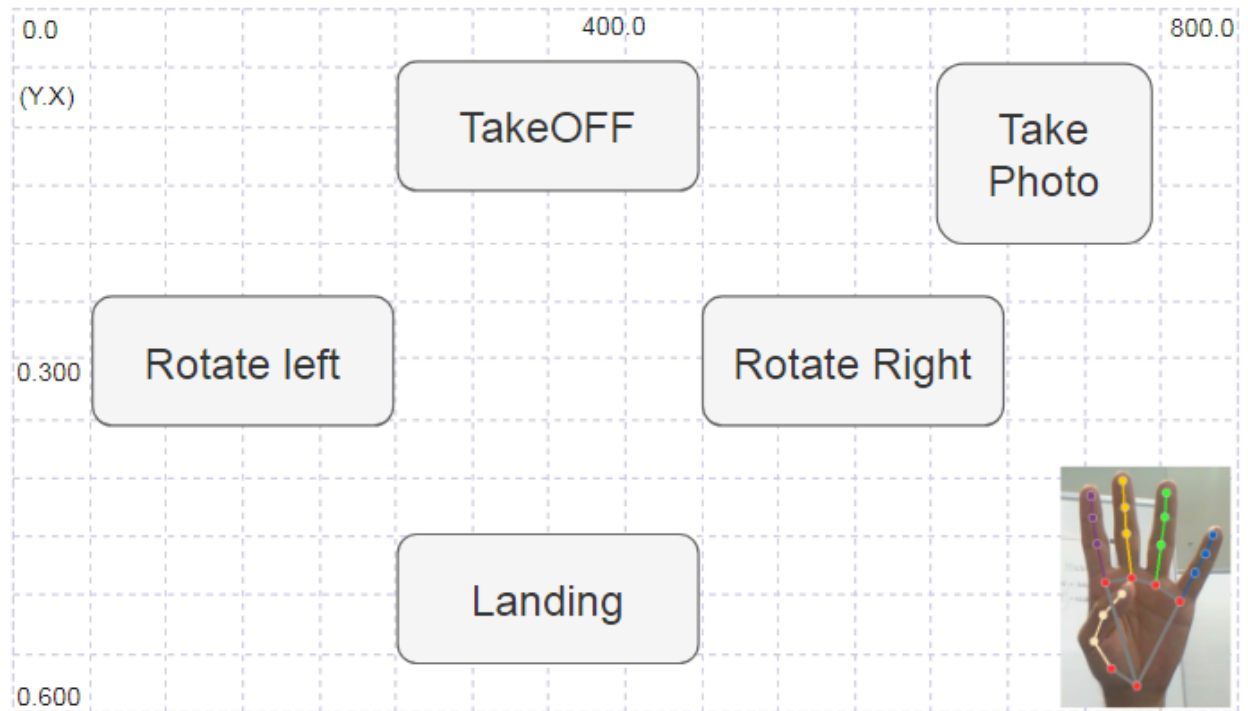


Figure 9.4 Screen display

9.3. Results and analysis

We have presented a system for human computer interacting using eye gaze and hand tracking. This method of combining two types of tracking has been tested and proven to be more accurate than just using only hand tracking method.

The system is able to utilize the advantage of gaze moving speed in eye tracking while overcoming the lack of eye tracking accuracy by switching to the hand tracking.

The hand tracking method also did not have much accuracy due the light intensity and background clearance or darkness but by combining with gaze tracking, the overall system accuracy increases. This makes the tracking system more robust than using only one of them.

CONCLUSION

Based on the result of the project and as a conclusion, it can be seen that developing a main application that merging hand gesture recognition and gaze tracking using Python and OpenCV. This application is implemented by applying tools such as MediaPipe library developed by Google for fingers gesture recognition and gaze tracking device proposed by Tobii AB for eye movement detection.

To summarize result, this system has accomplished the objective defined in this project:

- To establish a system for gaze detection and tracking through eye tracking device using Python, OpenCV, Tkinter
- To create a template or region of interest(ROI) that containing the button corresponding to commands
- To establish a system for detecting, recognizing and interpreting hand gesture recognition through computer vision using Python and OpenCV
- To create certain numbers and sign languages of hand gesture for controlling drones

At the end we arrive to establish the main control system that combine gaze tracking and gesture recognition for performing different commands that can be sent to the drone with more flexibility, the user can manipulate by hands or by eyes or both.

This combination of two type of tracking has been proven be more accurate than just using only eye or hand tracking method, This combination improves the accuracy of the global system, as the gaze alone is appropriate for choosing target by moving the pointer but for selecting target still required button clicks required period and the hand tracking is no efficient on hand skin color and background clearance or darkness.

For the future recommendation, this system could be extended to include additional gestures that will allow any users with different need to perform more functions easily specially for people who play games which require a very fast atmosphere of interaction or cannot use their hands.

In addition, the gaze detection could be based on low-cost hardware such as commercial camera or embedded Laptop camera without involving any eye tracking device, by improving the gaze detection software available but not efficient.

In the future, it is possible to implement a special Graphical User Interface for extending this application after the users know how to translate the gesture from its meaning to the sign or number and vice versa for different uses not only in drone domain such as pointing, clicking, swiping and scrolling.

ACRONYMS

API	Application Programming Interface
SDK	Software Development Kit
mph/MPH	Miles Per Hour
kph/KPH	Kilometre Per Hour
mAh	Milliamp Hours
LiPo	Lithium-Ion Polymer Battery
Wh	Watt-hour
W	Watt
HD	High Definition
MP	Mega Pixels
IR	Infrared
ML	Machine Learning
ROI	Region of Interest
PCK	Percent of Correct Keypoints
FPS	Frame Per Second
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
OS	Operating Software
GB	Gigabyte
RAM	Random-access memory
3D	Three Dimension

REFERENCES

- [1] (2023) Dynavox PCEye 5 website. [online] Available: <https://www.tobii.com>
- [2] DJITelloPy: DJI Tello drone python interface using the official Tello SDK. Feel free to contribute! (github.com)
- [3] (2023) MediaPipe. [online] Available: Landmarks - MediaPipe (<https://developers.google.com/>)
- [4] (2023) OpenCV website. [online] Available: <https://opencv.org/>
- [5] (2023) Tkinter GUI toolkit . [online] Available: <https://docs.python.org/3/library/tkinter.html>
- [6] (2023) SDK 2.0 User Guide. [online] Available: <https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf>
- [7] (2023) DroneBlocks website. [online] Available: <https://www.droneblocks.io/>
- [8] (2023) Tello FPV [online] Available: <https://play.google.com/store/apps/details?id=com.volatello.tellofpv&hl=en&gl=US&pli=1>
- [9] (2023) Tello spec. [online] Available: Tello (ryzerobotics.com)
- [10] (2023) Tello EDU spec. [online] Available: Página Oficial de Tello - Shenzhen Ryze Technology Co., Ltd (ryzerobotics.com)
- [11] (2015) Combining Eye Gaze and Hand Tracking for Pointer Control in HCI <https://ieeexplore.ieee.org/document/6504305>
- [12] (2020) An efficient hand gestures recognition system <https://iopscience.iop.org/article/10.1088/1757-899X/745/1/012045>
- [13] (2020) Hand gesture recognition on python and opencv <https://iopscience.iop.org/>
- [14] (2021) Survey of Hand Gesture Recognition Systems <https://iopscience.iop.org/article/10.1088/1742-6596/1294/4/042003>
- [15] (2021) Applying Hand Gesture Recognition for User Guide Application Using MediaPipe <https://www.atlantispress.com/proceedings/issat-21/125963795>

ANNEXES

[A] (2023) Tello-Python GitHub. [online] Available
https://github.com/b_berarache