

# **Drones with Face Recognition and Tracking**

**Brandon Sinclair**

2636160

Dr Andrea Bracciali

*April 2021*

**Dissertation submitted in partial fulfilment for the degree of  
Bachelor of Science with Honours *Applied Computing***

Computing Science and Mathematics  
University of Stirling

## Abstract

Drones' technology has been advancing rapidly over recent years, with the ability of adding computing vision technologies it has become a new and exciting area with many different possibilities.

Face recognition has been receiving increased attention due to the development of new deep learning models with increased accuracy rating to near human performance. This project seeks to use such higher accuracy and tracking technologies with drone-supported search of missing individuals. Organisations such as the Scottish Mountain Rescue Team could exploit such a framework and the drone's capability of agile movement in dangerous terrain for their rescue operations for missing individuals. This project aims to develop an application that controls an autonomous drone with the capabilities of detecting and identifying an individual in an area using only a single photograph.

for this to be possible it requires 3 components.

- A fast and accurate face recognition method that does not require to be trained on the individuals face in order to identify them.
- An application to communicate with the drone to capture live images, process and display the results.
- A controller to control the drone's movements based on the face's location, with necessary safety features to avoid any collisions.

This address some of the issues for the Rescue team as it creates a method to search over difficult terrain, detecting and identifying missing individuals and track that individual in the case that they are not stationary, without the need of volunteers or pilot which saves valuable time and resources.

The development process followed a typical software development approach. The development was split into 2 components. The face recognition network and drone tracking. The Project was developed in python using PyCharm with the project's overall objectives in mind throughout development.

The development started on the neural network, instead of creating a new network from scratch which would have taken more time, instead the network used was based on a pre-existing model ResNet pre-trained on the VGG2 dataset. The model used for face recognition is FaceNet, this model allowed an image to be passed into the network and then process the features from that image and return an embedding value, this value could then be added to an embedding space. this method allows you to determine the closets matching face stored in the dataset by distance, the distance determines how similar a face is to one another so rather than searching for an exact match, we search for a face most similar.

The development for the tracking capabilities used were based on a PID controller where an error is calculated to determine how far of the drone camera is from a face. The PID controller consists of 3 parts Proportional, Integral and Derivative which deal with present, past and future error, respectively. Based on the error calculated the drone's movements are adjusted to minimize the error and keep the individuals face within a pre-set safe distance.

The final build of the project successfully achieves the goals and targets set out. The application can successfully detect and identify an individual with high accuracy and the drone can successfully track and follow that individuals face as they move with restrictions. The main issue due to the face recognition being so CPU intensive resulting in low fps it is not practical to use face recognition and tracking at the same time because the tracking requires as smooth

fps as possible otherwise important frames might be skipped leading to inaccurate errors being calculated and delays in commands being made. The face tracking however works smoothly when only using face detection therefore is more practical to use this software to keep a face within the frame.

## Attestation

I understand the nature of plagiarism, and I am aware of the University's academic misconduct policy.

I certify that this dissertation reports original work by me during my University project except for the following (*examples given below: adjust according to the circumstances*):

*The PID controller code allowing for the rotation of a drone was taken from by Murtaza's Workshop [34]*

<b>Signature</b>	<i>Brandon Sinclair</i>	<b>Date</b>	16/04/2021
------------------	-------------------------	-------------	------------

## **Acknowledgements**

Firstly, I would like to thank my project supervisor Dr Andrea Bracciali, for his guidance and support through the development of the project.

I would like to also thank Michael and Robbie for their support and guidance as we were working on similar projects using the same programmable drone.

# Table of Contents

Abstract	i
Attestation	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
1 Introduction	1
1.1 Background and Context	2
1.2 Scope and Objectives	2
1.3 Achievements	3
1.4 Overview of Dissertation	3
2 State-of-The-Art	4
2.1 Drone	4
2.2 Drone Projects	5
2.2.1 TelloMe Mobile Application	5
2.2.2 Tello Vision 1D Mobile Application	6
2.2.3 Drone Face Tracking with a PID Controller	7
2.3 Face Recognition	8
2.3.1 Deep Learning	8
2.3.2 Face Recognition Methods	9
1.1.1.1. DeepFace	9
1.1.1.2. DeepID3	9
1.1.1.3. FaceNet	9
2.3.3 Face Recognition Mobile Application	9
2.3.4 Face Recognition Conclusion	10
2.4 Technologies	10
2.4.1 DJITelloPy	10
2.4.2 OpenCV	11
3 Problem Description & Analysis	12
3.1 Drones	12
3.1.1 Safety	12
3.2 Face Recognition	13
3.2.1 Illumination	13
3.2.2 Pose Variations	14
3.3 Neural Network	14

3.4	Application Architecture Analysis	15
4	Technical Overview	17
4.1	Development Environment and Language	17
4.2	Neural Network Framework	17
4.2.1	PyTorch	17
4.2.2	TensorFlow	17
4.2.3	Conclusion	17
4.3	Neural Network Components	18
4.3.1	Neural Networks	18
4.3.2	Convolution Neural Networks	18
4.3.3	Convolution Layers	19
4.3.4	Activation Layer	19
4.3.5	Pooling Layer	20
4.3.6	Fully Connected Layers	20
4.3.7	CNN Architecture Chosen	21
4.3.8	InceptionResNet Model	21
4.4	FaceNet Architecture	22
4.5	Face Detection Architecture	23
4.5.1	Haar Cascades	23
4.5.2	MTCNN	24
4.5.3	Face Detection Chosen Method	24
4.6	Face Tracking	24
5	Implementation of Core Components	25
5.1	The Model	25
5.2	Drone Connection and Live Feed	25
5.3	Implementing Face Recognition	26
5.4	Face Tracking	27
6	User Interface Design	28
6.1	Requirements	28
6.2	Navigation	29
6.3	User Interface Wireframes	30
6.3.1	Main Page	30
6.3.2	Add Missing Person Page	31
6.3.3	Search Missing Person Page	31
6.4	Implementing the UI	32
6.4.1	The Main Page	32

6.4.2	The Add Missing Person Page	33
6.4.3	Search Missing Person Page	34
7	Testing	35
7.1	User Interface	35
7.1.1	Buttons	35
7.1.2	Drone live feed displayed	35
7.2	Add new individual	35
7.3	Search for existing user	35
7.4	Face Detection	36
7.5	Face Recognition	37
7.6	Face Tracking Testing	38
8	Conclusion	39
8.1	Summary	39
8.2	Evaluation	39
8.3	Future Work	40
8.3.1	Limitations	40
8.3.2	Added Functionality	40
8.3.3	UI improvements	40
References		41
Appendix 1	Test Plan	44
Appendix 2	Face Tracking Screenshots	47



## List of Figures

Figure 1.	Facial Recognition Process .....	8
Figure 2.	[13] shows the vector recognizing facial features.....	11
Figure 3.	Illustration on how illumination affects images .....	13
Figure 4.	Illustration on how changing illumination side effects image.....	13
Figure 5.	FaceNet distance for different poses and illumination .....	14
Figure 6.	Shows Initial System Architecture .....	15
Figure 7.	Shows a convolution layer filtering an input and producing a feature map .....	19
Figure 8.	Example of Max Pooling .....	20
Figure 9.	CNN Architectures Benchmarks .....	21
Figure 10.	Triplet Loss function .....	22
Figure 11.	FaceNet Network Results .....	22
Figure 12.	FaceNet Inception Architecture .....	23
Figure 13.	Shows Haar Feature.....	23
Figure 14.	MTCNN Structure .....	24
Figure 15.	Flow Chart of Face Recognition Method .....	26
Figure 16.	Navigation Map .....	29
Figure 17.	Main Page Wireframe.....	30
Figure 18.	Add Missing Person Page Wireframe .....	31
Figure 19.	Search Missing Person Page Wireframe.....	31
Figure 20.	Shows the application main page.....	32
Figure 21.	shows add missing person application page .....	33
Figure 22.	Shows the search missing person page.....	34
Figure 23.	shows the face detection test results.....	36
Figure 24.	Shows Face Recognition Test Result .....	37

# 1 Introduction

Drones' technology has been advancing rapidly over recent years, with the ability of adding computing vision technologies, it has become a new and exciting area with many different possibilities. One such technology is face recognition which has seen advancements where its accuracy is beginning to surpass human capabilities thanks to advancements in machine learning and neural networks. Neural networks are essentially simplified models of the human brain which can be trained with large amount of data to recognise patterns quickly and accurately. With face recognition combined with mobile drones a wide range of practical applications can be developed with this project focusing on the possibilities of helping with search and rescue.

When developing an effective application limitations and constraints found from other existing drone and face recognition applications must be considered. When dealing with any physical object safety is the number one concern, with a mobile drone we must keep in mind a safe distance between an individual and the drone. Another issue is that the individuals may not always be in view of the drone's camera. Challenges with face recognition must also be considered such as illumination and pose variations that contribute to the overall accuracy.

The project aims to develop an application that commands an autonomous drone which is capable of face recognition and tracking and can be executed on any desktop platform, tackling some of the issues mentioned above to aid in search and rescue in detecting and identifying missing individuals over tuff terrain.

## **1.1 Background and Context**

In 2019 Scottish Mountain Rescue Teams received 502 independent incidents, 243 were related to mountaineering going over tough terrain like rock climbing and hill walking. This required a total number of 672 or 730 including continuations separate team call outs or an average of 2.0 call out per day for 263 days that year [1]. Scottish Mountain Rescue is completely reliant of volunteers, in 2019 26,934 hours were given up by volunteers this number can be greatly reduced with the help of drones. Unlike people drones can cross difficult terrain with ease, crossing large distances without putting others at risk.

“...many search and rescue operations in Scotland simply cannot be carried out without large numbers of volunteers able to go out on foot and who can be mobilised quickly. It often is simply not possible to search for people by helicopter in Scotland, because low clouds often prevent the helicopters from flying.” – [scottishmountainrescue.org](http://scottishmountainrescue.org)

Many volunteers need to risk their own safety for the sake of others, not only giving up their own time but also putting themselves at risk. For this reason, the project will be extremely helpful in assisting with the search and rescue efforts by reducing the required number of volunteers and giving a risk-free way of searching for missing individuals, identifying them from a photo and tracking their location so that information can be passed on.

The primary use case of this application is for rescue teams to use the drone in order to search an area for a missing individual from their photo. The rescue team would start the application, the application would then command the drone to begin the search for the missing individual and stream the drone live camera feed to a user. The drone will then search an area for a face and once a face is found run face recognition to determine if that is the missing individual. Once the drone has found the missing individual and identified them as the correct person a notification will display to the user notifying them that the individual has been found, using the camera feed the user would be able to determine the location of the individual and send the rescue team to that location. The drone will also continue to track and follow the missing person so that person location is not lost.

## **1.2 Scope and Objectives**

The aim of the project is to aid in search and rescue efforts by implementing an application that controls a drone to detect and identify missing individuals, using the drones live feed and a highly accurate facial recognition method which requires only one photograph of the individual in the dataset, track that specific individual so their location is not lost, and that information can be passed on to the user.

The project will be a smaller scale of what would be a real-world application to act as a proof of concept. The restrictions to the application would be only detecting faces in a small clear area with no obstacles with the future goal to development object avoidance and large-scale pathing to deal with real world situations like searching a large area avoiding objects such as trees, rocks and walls.

### **1.3 Achievements**

The overall aim of the project has been achieved, as the application can successfully control the drone, to detect and identify an individual using the drone live camera feed with only one photograph of that individual and can also track that specific individual as they move around a small area with restrictions, the necessary safety features are also in place, always keeping the drone at a safe distance from the individual. Although the core functionality is complete there are room for further developments and improvements to be made such as solving the CPU restriction, implementing new functionality like pathing and body detection to be more real world viable.

### **1.4 Overview of Dissertation**

This paper consists of 8 chapters; each one of them describes and analyses a specific aspect of the project which are the following.

Chapter 2- State of the Art: this chapter aims to provide the reader with the research and technical background for drone programming and face recognition technologies. Also, this chapter investigates other similar projects to give a critical analysis and evaluation of their work and see where improvements can be made.

Chapter 3- Problem's and Analysis: this chapter deals with the various problems and challenges the project may face. It covers the problems that face recognition faces and what issues working with a drone may have.

Chapter 4- Technical Overview: this chapter goes into more technical detail of the technologies used and why some architectures were chosen over others for the project.

Chapter 5- Implementation of Core Components: this chapter goes into detail of how the core components; face recognition, detection and tracking were implemented.

Chapter 6- User Interface Design: this chapter goes into detail on the design and implementation of the UI, it includes the requirements of the UI as well as the finished application.

Chapter 7- Testing: this chapter goes over the tests run to determine that the application functions correctly. The tests include UI tests, unit testing where each component is tested independently before integration testing where components are tested together.

Chapter 8- Conclusion: this chapter summaries the achievements of the project giving a critical evaluation of the application and the process used for the project, highlighting any limitations the application has and any future work that may be implemented if the project was to continue.

## 2 State-of-The-Art

Drone Programming is a relatively new area so there is not an abundant number of papers or applications available but the ones that are available are exceptional for their level of detail and clarity. In this section I will discuss and analyse these examples highlighting features that could be implemented and improved for the project.

Following the critical analysis of these projects, a review of the technologies, frameworks and algorithms available to create the application giving an analysis of each of them and providing a detailed conclusion of my choices.

### 2.1 Drone

For this project as mentioned a drone will be used that will stream a live feed to an application, the drone that was chosen for the project is the DJI Tello EDU developed by DJI [2]. This Drone was chosen as it is a programmable drone with an SDK, this SDK however is built into a visual scripting language like Scratch which uses blocks to issue commands, which makes it very restrictive in terms of programming freedom and compatibility with other libraries. Fortunately, libraries have been developed like the DjitelloPY [3] which bypass this restrictive making it possible to program using the SDK in python, the DjitelloPy will be discussed more in a later section [2.4.1]. The SDK allows an application to connect to the drone using a WI-FI connection, allow for some commands for basic movement and speed controls and receive read back commands to retrieve information such as speed, battery and temperature.

The drone connects to a device over a Wi-Fi UDP port with a range of 550 feet (170 meters) with a built-in safety feature that will land the drone automatically if it has not received a command after 15 seconds. The drone uses a sensor to detect the ground rather than a traditional GPS seen in other commercial drones, using this sensor it can determine how far from the ground it is and keep it self-hovered at a safe height automatically.

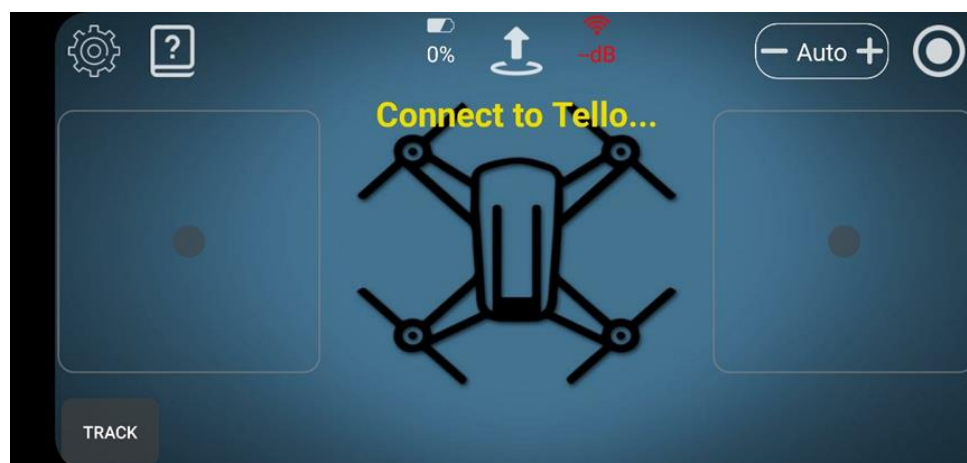
Using this drone does have its benefits and its weaknesses. The benefits being it is a cheap and widely available programmable drone that most WI-FI capable devices will be able to use. The disadvantage for using this drone is that the application will only work with a Tello drone because of the SDK only being compatible with Tello drones. However, since the SDK is only required as a controller and communication for the drone it is possible to change this to an alternative SDK for different drones, so one of the project goals will be to make the application more modular so swapping the SDK for a different drone's SDK will not be difficult, this will allow for other drones to be used with little change to the application. The other restriction is that the SDK is written in python, so I am restricted into using this programming language which is not a bad restriction as python is a very popular easy to learn language with many guides and tutorials available to learn from.

## 2.2 Drone Projects

### 2.2.1 TelloMe Mobile Application

The TelloMe app [4] works with the Tello drone, it allows a user to be tracked using a person's face or body. The app seems to work well however the body tracker did not work as it said I was too close, so more room seems to be required before using this feature which currently is not possible due to covid lockdown and available space indoors. The face tracker feature worked well, the app was able to detect my face and rotated as I went around it, the restriction seems to be that it is unable to move from its launch area only allowing rotation of 360 degrees. unfortunately, only the demo was tested as the full version cost extra. The demo comes with a few restrictions such as no video recording, no screen recording and limited testing. The app also recommends a powerful recent phone as the real time image processing takes a lot of resources.

The UI of the application was clean and easy to follow, with important icons being displayed at the top such as the battery life and connection range. The controls for the drone were a bit tricky as I could not at first tell which controller was for what (rotate or move), the only way to find out was to try it which is a potential safety hazard as you do not want the drone to fly into a wall or your face. The tracking was simple to activate as there was a button placed at the bottom left corner called "track" which let you choose between certain options like body or face and once pressed that was it activated, and the user could let the drone fly autonomously.

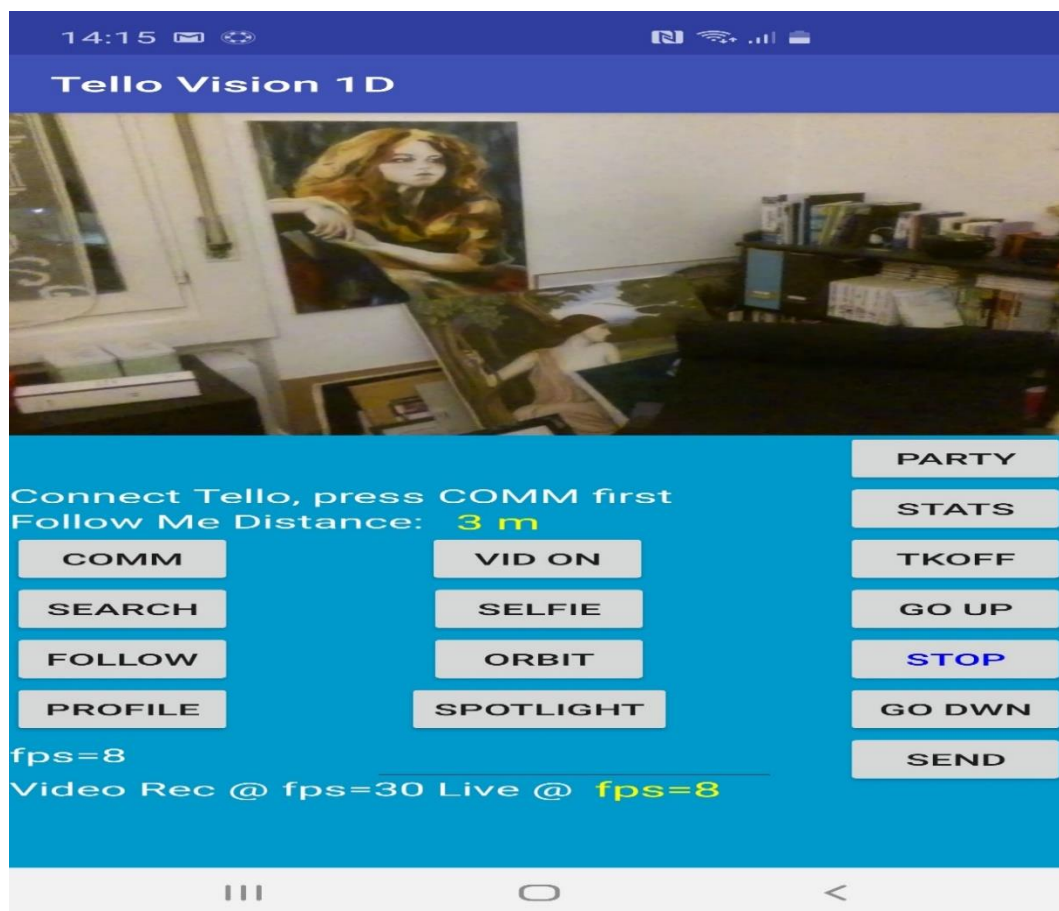


In conclusion after trying the app the strengths where it is UI was very clean and easy to use the only minor thing would be adding possible labels to each of the controls so the user can accurately tell what each control does. The facial tracking worked well after multiple tests it was able to detect my face every time with high accuracy, the only thing I could improve here would be to add the ability for the drone to move from the launch area and of course specific facial recognition so the drone would track the correct person. I was unable to test how multiple faces in the area would affect the accuracy for example would it be able to detect all the faces in the location or just the one. the few issues that it had mentioned above, for the project we will look at each of these restrictions to see what can be implemented and what improvements can be made

### 2.2.2 Tello Vision 1D Mobile Application

The Tello Vision 1D app [3] works with the Tello drone, it allows the user to control the drone basic movement. The app also has some object detection which worked well, being able to detect people as well as objects. flying of the drone was extremely limited as you could not move the drone from it is launch area other than going up or down, there was no control stick for the drone. I was unable to test the search and follow options as the drone kept crashing due to very unstable flying controls. There were also restrictions like the Tello me app [2] as this app was also a demo so we are unable to test the full functionality to see if it improves the stability of the app.

The UI was extremely cluttered with every option being placed on the lower half of the screen, this made everything easy to find but is a very inefficient use of the screen space as only the top half of the screen was the live video.



In conclusion although the app runs correctly it needs some fine tuning. The cluttered UI needs a rework, a recommendation would be to use a menu with useful headings for better usability and the video can cover a larger portion of the screen. I was impressed on how well the object detection worked as not only was it able to identify me as a person but other objects around me such as my monitor and tv. The flight was the main issue like the Tello me app it was very restrictive only allowing stationary searching by rotation. For the project improvements a goal will be to allow less restrictive flight controls.

### 2.2.3 Drone Face Tracking with a PID Controller

From what we have seen from the last two applications, the way the control the face tracking is unknown as they are mobile applications and source code is not visible. This python application developed by Murtaza's Workshop [34] shows how a PID controller is used with face detection boundary box to control the drone's rotation allowing it to track a face.

boundary box is created with face detection methods which result in a box being drawn around faces. These boxes contain data such as coordinates, width and height which can be used to determine the location of a face within a given frame.

A PID (Proportional, Integral, Derivative) is a controller that uses a control system to get an output you want and keep it at that output using feedback. The basic structure of a PID is there is a command variable which acts as the goal of the system to achieve, this variable is entered into a system and that system outputs a controlled variable which is what the system managed to achieve, this variable is usually determined from a sensor such as a temperature gauge. Using this output variable as feedback we can determine how far off the desired command variable; we are which is known as the error term. Using this error, we can use a controller to try to reduce this error to zero meaning we hit the desired output. The controller is made up of 3 parts Proportional (P) which is the present Error, Integral (I) which is the past error and Derivative (D) the future error [35].

The Proportional determines how far away from your desired goal presently and adjusts the command variable accordingly until the goal is reached. This controller seems to work well on its own for some scenarios, the issue with using only P starts when we introduce the drone into the equation. with only using P we would control the propeller speed to gradually increase speed until we hit a desired goal for example a certain altitude, but what happens when we hit our goal? The error would be zero and so the propellers would stop since we hit our goal and there is nothing left to do. This is an issue as when the propellers stop the drone starts to fall which increases the error which would then start the propellers again and this loop would continue, this is known as the steady state error. To solve this issue an integrator is introduced [35].

The Integral keeps a running total of the input over time, so it keeps a memory of past input values, it works with the proportional to reduce the error to 0. When the error hits 0 we know that the proportional stops doing anything, at this point the integral starts taking over as it has been calculating the correct rpm that is required to make the drone hover at the desired altitude which is then sent as the command variable. There is however an issue that can occur when the error is so small that it is practically zero the integral can calculate a hover rate that is slightly over the correct hover rate for the desired altitude, this makes the drone overshoot the target altitude which may not be desired. This is where the Derivate comes in [35].

The Derivative produces a measure of the rate of change of the error, that is how fast the error is increasing or decreasing. Using this measurement, the derivative can determine if the drone is reaching the goal to quickly and if so, a negative value is produced which is then added to the controller's output, so the propellers gradually lower their speed solving the overshoot problem [35].

Murtaza's application only uses PD excluding integral, this may be because his application only allows the drone to rotate so the issues that Integral solves are not required. His solution does work well and so will be implemented into the project with some adjustments. The problem of how to allow the drone to move forward and backward based on the person distance from the drone have not been solved so a solution to this problem is still needed, which will be discussed in a later section.



## 2.3 Face Recognition

Face recognition is not a new concept there are plenty of papers and applications out there that make use of face recognition daily. However, what is not so abundant is drone applications that use face recognition.

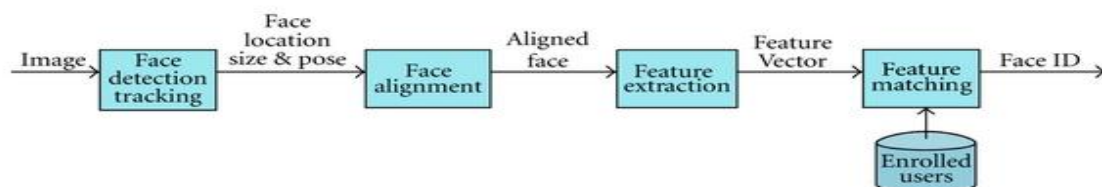
This section will discuss and analyse examples of facial recognition projects highlighting features that could be implement and improved for the project. Following the critical analysis of these projects, a review of the technologies, frameworks and algorithms available to create the application giving an analysis of each of them and providing a detailed conclusion of the choices made.

### 2.3.1 Deep Learning

Deep learning, specifically neural networks have been becoming more popular in the efforts to make face recognition more accurate and efficient [5]. There are many new network architectures being developed in the attempt to improve from the last with the error rates decreasing drastically each time [4].

It is important going forward to understand a few things. Deep learning is a very powerful method that uses neural networks, these networks use large amounts of data to recognise patterns quickly and accurately which is key for facial recognition as facial recognition is a visual pattern problem [4]. Before the network can retrieve a person identification it first needs to extract the features vectors from an image, then using the features vectors values it can check that against the dataset.

**Figure 1. Facial Recognition Process**



These methods are classified by a benchmark called Labelled Faces in the Wild (LFW). The LFW is a database of faces which contains more than 13,000 images and 5000 unique individuals, this allows different neural networks to be tested and return a performance score.

### **2.3.2 Face Recognition Methods**

There are many methods that have been developed to deal with the problem of face recognition, this section will discuss and analysis some of them.

#### **1.1.1.1. DeepFace**

Deep face is a deep learning facial recognition system created by Facebook. This method focuses on using their own alignment process. This process allows them to assume the location of the face is at set pixel location which lets them take the pixel values rather than going through conventional layers for feature extraction seen in other facial recognition methods. This method results in a highly accurate rating of 97.35% on the LFW benchmark [6].

#### **1.1.1.2. DeepID3**

DeepID3 is another facial recognition system that uses deep learning. They proposed two different networks hierarchies which incorporates a stacked convolutional neural network (CNN) layer before each of the pooling layers and including nonlinear feature extraction layers [7]. This method results in an exceptional accuracy rating of 99.53% on the LFW benchmark [8].

#### **1.1.1.3. FaceNet**

FaceNet differs from other face recognition methods as it provides a unified embedding method for face recognition, verification and clustering tasks. Using a CNN neural network to extract feature embedding values which are put into a Euclidean space. in this space the distance between images corresponds to face similarities, so rather than searching for an exact match it instead searches for a face that is the most similar. What makes FaceNet so appealing for the project is it achieves state of the art performance while only requiring 128-bytes per face meaning only 1 image of a person is required in order to be identified [9].

### **2.3.3 Face Recognition Mobile Application**

The face recognition app [10] is a mobile application that focuses on facial recognition and training methods including neural networks. The app works by first asking the user to take photos of themselves, this is done in an automated process that takes 20 photos of the user in succession. Looking through the settings the app was using OpenCV facial recognition method, a discussion about OpenCV will be made in the technologies section [2.4].

The app worked well, being able to recognize and identify a face after training with a very high accuracy rate being able to correctly identify each time after multiple tests. the UI was clean and simple to navigate and the number of options to change the method of facial recognition to test the accuracy rate was extremely useful, the default settings using OpenCV resulted in the highest accuracy rating however an issue found was if the turned away from the camera to either the left or right the accuracy drop considerably this being the limitation of facial recognition technology because you all ways have to be facing the camera directly in most cases. In conclusion this app works well, the inclusion of different methods of facial recognition it offers as well as the solution to adding the photos to the database in a quick efficient way. Improvements that can be made would be to find a solution to increase the angle of a face being recognized i.e., a face turning to the right will still be detected and correctly identified.

### **2.3.4 Face Recognition Conclusion**

To summaries there are many deep learning methods that are extremely accurate at face recognition. The ID3 and DeepFace methods although prove to be efficient and accurate they have two restriction which makes them unsuitable for the project which is that the individual needs to be trained on, in order to be identified which is a form of classification and both networks require more than one image of an individual to give an accurate prediction. This is unsuitable as it is unrealistic to believe that a missing individual would have been trained on beforehand with multiple suitable images available. This is where the face net method seems to be the logical choice as not only does the individual not need to be trained on to be identified, it also only requires one image of the individual to be accurately identified making it the ideal choice for this project.

## **2.4 Technologies**

This section will discuss what technologies have been developed that can be used to help create the application.

### **2.4.1 DJITelloPy**

DJITelloPy [11] is an open-source library available on GitHub that interfaces with the Tello EDU SDK interface which has the following features:

- Implementation of all Tello commands
- Easily retrieve a video stream
- Receive and parse state packets
- Control a swarm of drones
- Support for python >=3.6

The library makes use of the Tello EDU SDK allowing you to connect to the drone through the devices WI-FI connection without needing to know the IP address of the drone which was a limitation of the SDK. It also allows you to run commands to control the drone movements very easily only needing to specify a distance in cm or controlling the drone with your keyboard. Recording and taking videos is also made easier using this library as it is compatible with OpenCV [2.4.2] as both libraries are written in python which makes it suitable for this project [11].

### 2.4.2 OpenCV

OpenCV is the most popular open-source computing visions and machine learning software library with the focus being on image processing, video capture and analysis including face detection and object detection which can be used in real-time applications [12].

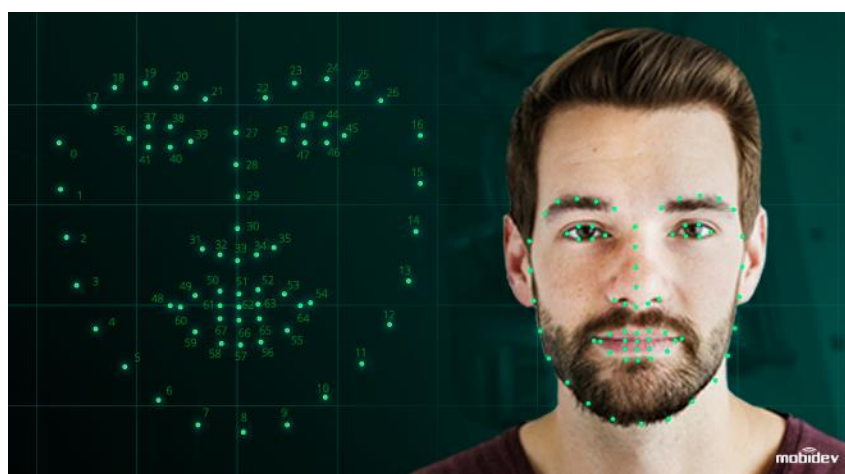
OpenCV uses various algorithms of deep learning to compare live image or digital images with the stored facet. A facet is derived from a classification of biometric software which facial recognition is a part of which compares the facial features or components on one's face mathematically and that data is stored as a facet. The facial recognition uses the deep learning algorithms to compare the images with the facet data for verifying someone identity. [12]

OpenCV was designed to be cross-platform as it was written in C so any device that can run C can run OpenCV. It also has wrappers for python and java to encourage adoption by a wider audience. It can be both run on desktop (windows, Linux, macOS) and mobile devices (android, IOS)

OpenCV has been used to develop a wide range of uses and applications for example using facial recognition for security camera to recognize someone trying to enter a building using a live video camera and alert someone if that person was unrecognized [13]. For this application to function it had 3 main steps. The first step is that it must detect a face and recognize that face instantaneously, it did this by using the TensorFlow object detection model and the Caffe face tracking which are both methods of the OpenCV library. Once a face had been captured that image is cropped and saved by the API with an appended person ID. After the face has been captured an algorithm uses the Dlib function to generate the 128-dimension vector which is used to identify the person face attributes. Lastly the algorithm cross-references this vector with all the facial entries in the database using Euclidean distance to find out whether this face matches any of the other faces in the database [13].

In conclusion OpenCV is a powerful library that can support a wide range of computer vision-based applications making it suitable for the project for its video and image processing capabilities.

**Figure 2. [13] shows the vector recognizing facial features**



### **3 Problem Description & Analysis**

As discussed, the aim of the project is to create an application that controls a drone and using that drones live feed detect faces to search for a specific individual and track that specific individual as they move. From this we can determine that they are three components: (i) communication between drone and the application, (ii) the face recognition, and (iii) drone tracking. In this section the three components will be broken down, analysed and discussed.

#### **3.1 Drones**

As we are working with hardware It is important to know the limitations of the Tello Drone going forward not only to prevent damage but for safety as well. This section will discuss some of the important limitations of the drone and how they can affect the project.

The range of the Tello Drone EDU is 100 meters and max height of 30m which is not that far. If we are testing out a wide area like this, it is important to key these figures in mind so we do not go outside the maximum range which could possibly lead to the loss of the drone [14].

The Tello Drone has a 13-minute battery life, this is important to know as you do not want your battery to run out and the drone to fall from the sky. So, when developing the application, the battery percentage should be clearly visible, so the user is kept informed [14]

The Tello Drone uses an IMU sensor, this sensor monitors how far the drone is from the ground, this keeps it from crashing into the ground, however a limitation of this sensor is that it is extremely inaccurate in low light. During tests during low lighting the drone was unable to hover and crashed repeatedly, this seems to be a hardware limitation, so it is unable to be fixed. So, during the project before any tests of the flight accurate lighting will be a requirement.

The drone does not run any code natively so in terms of the application the drone does not restrict any computational requirements like developing for other lightweight devices like mobiles might do.

##### **3.1.1 Safety**

When carrying out any project it is important to think about safety, even more so when dealing with a physical object. Even though the drone is small it can still pose a risk to others, so it is always important to follow the safety guidelines that the manufacture gives. The Tello manufacturer has many safety guidelines some of the key ones are always Maintain visual line of sight, stay away from rotating propellers and motors, do not fly under the influence of alcohol or drugs, when at low battery land the aircraft promptly, do not fly too close to people [12].

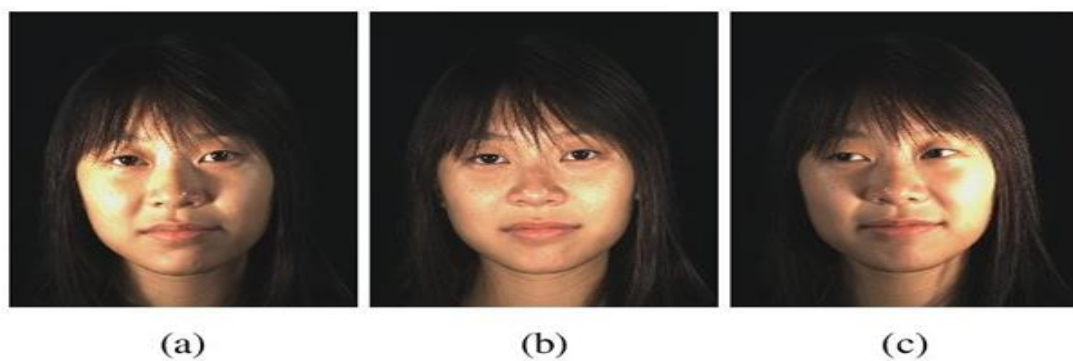
## 3.2 Face Recognition

There are many challenges that can affect face recognition accuracy and reliability, this section will discuss and analysis each of and if possible, provide solutions to these issues.

### 3.2.1 Illumination

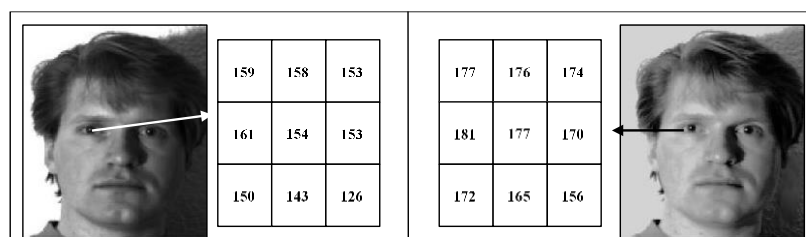
Illumination can be considered a complex problem for pattern matching. Studies have revealed that two problems of textural based illumination handling in face recognition to be very common. Firstly, textural values change during illumination normalization due to the increase of contrast that changes the original pixels of the face [16]. Secondly illumination minimizes the distance between inter-classes which increases the false acceptance rates [16].

**Figure 3. Illustration on how illumination affects images**



In [16] they attempt to tackle this issue by proposing an algorithm that aims to overcome these limitations through transforming pixels from non-illumination side to illuminated side.

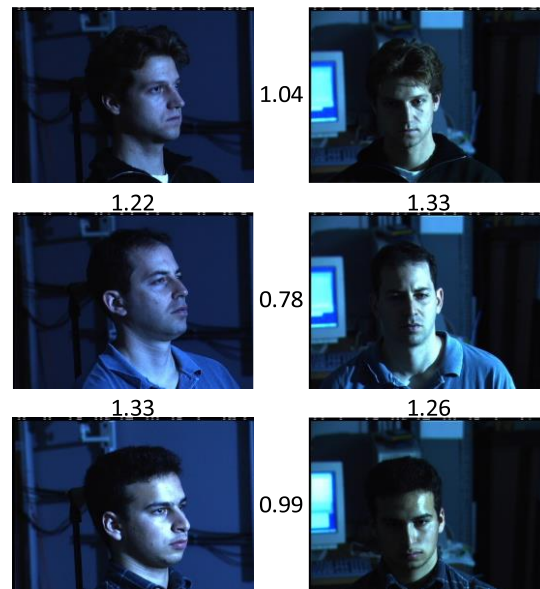
**Figure 4. Illustration on how changing illumination side effects image**



### 3.2.2 Pose Variations

Pose Variation is another long-standing problem in face recognition, where different poses can drastically change the result.

**Figure 5. FaceNet distance for different poses and illumination**



As seen in figure 5 shows the distance (lower is better) between pairs of the same person and a different person in different pose and illumination combinations [9]. FaceNet solution to this problem is to assign two images a value based on their similarities 0 meaning it is the same person and 4 corresponding to the opposite spectrum [9]. In the example shown they show a threshold value of 1.1 would classify every pair correctly regardless of pose and illumination.

### 3.3 Neural Network

Neural Networks can cause a variety of issues especially considering creating a new network from scratch which would be a very complex and time-consuming task, a pre-existing network however would remove a lot of these issues and reduce the development time considerably so may be a viable method to solve this problem.

### 3.4 Application Architecture Analysis

With the general problems analysed and the main components selected but not yet specified which will be explained in a later section, we can create an initial system architecture to gain a better picture on how the application components will function together.

**Figure 6. Shows Initial System Architecture**

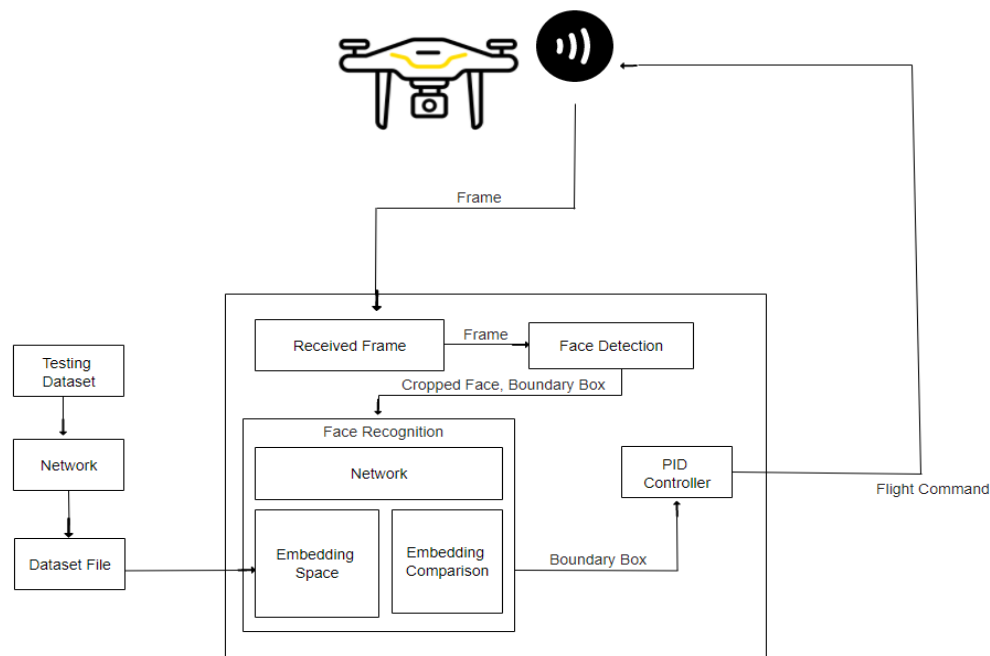


Figure 6 shows the system architecture. It consists of 3 main components: face detection, face recognition and a PID controller.

The face detection architecture has not yet been decided but we can assume its functionality based on previous examples such as Murtaza's [34] application using Haar cascades which produces a bounding box over a detected face.

The face recognition method as discussed is googles FaceNet architecture which uses embeddings and embedding space to compare embeddings over distance to match the face most similar.

The tracking method as discussed [2.2.3] is Murtaza's [34] example using a PID controller with adjustments and improvements necessary to allow for less restrictive drone movement.

Figure 6 shows that the drone will send a frame of a WI-FI connection as mentioned, this frame should then be passed through to the face detection method which will produce a cropped face and a boundary box if a face is detected within a frame. The cropped face is necessary as we do not want the face recognition to process an entire frame which would increase the computational time and decrease accuracy considerably.

The face detection component sends the cropped face and boundary box to the face recognition component where the cropped face should be embedded using a neural network which has not yet been chosen. The embedding should then be compared to the other embeddings in the embedding space and the lowest embedding verified to be the missing person which if true will send that bounding box to the PID controller.



The embedding space is generated by using an undecided dataset such as LFW, VGGface2, cassia-webface. Each dataset contains images of people's faces and some contain names to act as labels for training neural networks. As mentioned FaceNet does not need to be trained on the person we are trying to identify beforehand so this dataset will not be used to train the neural network but rather generate the embedding space so an accurate result can be obtained. The reason we need a large embedding space is that only having the person we want to identify in the space will result in only that person being returned regardless of which face is being recognised so would not be an accurate result. The embedding space should be generated into a file for easy access.

The last step is that the boundary box is sent to the PID controller, this box should contain the cord, width and height of the face that has been verified to be the missing person, within the frame giving an idea of its location from the drone. This data as discussed should generate an error which should be used to generate flight commands to adjust the drone position to reduce the error, this command is then sent to the drone via a WI-FI connection. In the case that the face recognized is not the missing person the PID controller should not be used as one of the requirements is that only the missing person is tracked. This process should be repeated with each received frame and should only be terminated once the application is closed.

## **4 Technical Overview**

The aims and objectives have been made clear along with the potential problems these components may have. However, a more technical understanding of the components is needed to give a critical analysis of the technology and understand the choices made. This section will cover the technical descriptions and choices of the technology that were used for the project.

### **4.1 Development Environment and Language**

As mentioned earlier the language the application was developed in is python. This is not problematic as it is a free and open-source programming language which is also easy to learn.

The Development Environment chosen for the development of the application is PyCharm which is a commonly used IDE for python-based applications. The reason this IDE was chosen as it is the most popular IDE that I myself am familiar with.

### **4.2 Neural Network Framework**

There are many popular frameworks for developing neural networks, they may follow similar structures, but they are separated by their ease of implementation and community support due to my self being unfamiliar with developing a neural network. This section will discuss the two popular frameworks Pytorch and TensorFlow.

#### **4.2.1 PyTorch**

PyTorch [17] is an open-source machine learning library based on the Torch library with a focus on applications used for computer vision and natural language processing developed by Facebook. It is built to provide maximum flexibility and speed adopting a dynamic graph structure, which lets you process variable-length inputs and outputs. PyTorch has grown in popularity in recent years accumulating a wider community meaning it is easier to learn as many tutorials as possible, guides and problems being discussed are available online [18].

#### **4.2.2 TensorFlow**

TensorFlow [19] is an open-source machine learning framework designed for large scale distributed training and inference, developed by Facebook. This framework differs from PyTorch as it operates on a static computational graph which must be constructed before the model can be tested or trained. Like PyTorch TensorFlow is an extremely popular framework with many guide, tutorials and problems being discussed online [18]

#### **4.2.3 Conclusion**

Comparing both frameworks came down to not just better performance but the ease of implementation, as someone who has no prior experience in machine learning frameworks, PyTorch was the clear choice for its flexibility and speed and community support.

## **4.3 Neural Network Components**

Before going into the implementation, this section will cover a brief overview of the core components in building a deep neural network.

### **4.3.1 Neural Networks**

As mentioned earlier Neural Networks (NNs) works similarly to how the human brain functions. As learns between the input data and expected output which is already provided similar as a person would learn [20].

The structure of a NN consist of three layers an input layer, a hidden layer and an output layer. The input layer is where the initial dataset weight is passed into the neural network and the output layer is where the prediction is made. The hidden layers apply given transformations to the input values until it can predict accurately (reaches the expected output).

### **4.3.2 Convolution Neural Networks**

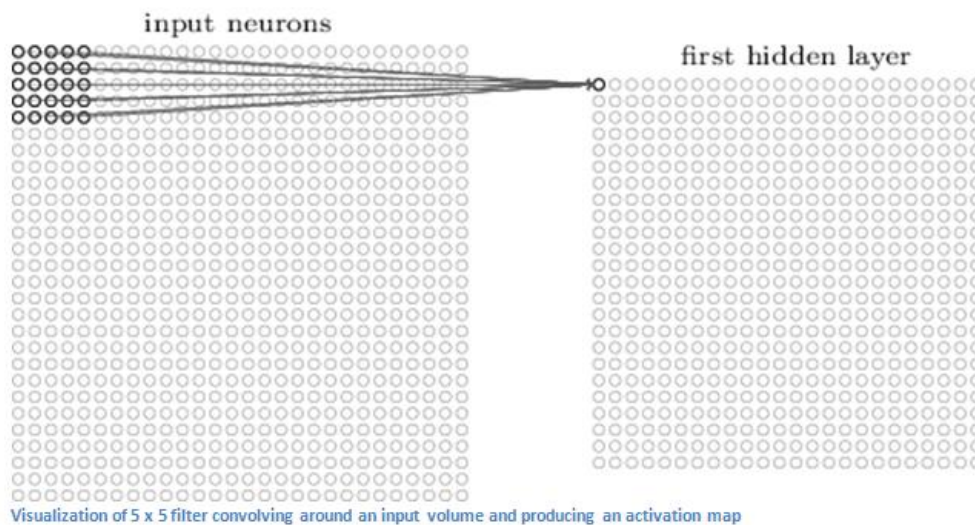
As mentioned in [2.3.4] the face recognition architecture the project will use is FaceNet. FaceNet requires a CNN to extract the embedding values from images, so it is important to understand a bit about these networks, which this section will cover.

CNN is a deep neural network that excels with processing visual information which take a grid like structure such as RGB. CNN are comprised of one or more convolutional layers and are mainly used for image processing, classification and segmentation [21,22]. Since images from the computer viewpoint are nothing but an array of pixel values the CNN task is to take in these values and output a probability of that image being a certain class (cat, dog, person etc). it does this by passing through the image though a series of layers to get the output, each of these layers will be outlined below [23].

### 4.3.3 Convolution Layers

The first layer of CNN is always a Convolutional Layer (CL). A CL consist of a series of filters sometimes referend to as a neuron or a kernel. A way to image the filter would be a flashlight going across the image covering an area of the image, this area is called the receptive filed which. these filters contain an array of numbers known as weights or parameters, they slide or convolving around the entire inputted image multiplying the filters values with the original pixel values and then summing these mathematical values resulting in a single number. These calculations are repeated for each location until a feature map is produced [23]. The feature maps are then passed through an activation layer.

**Figure 7. Shows a convolution layer filtering an input and producing a feature map**



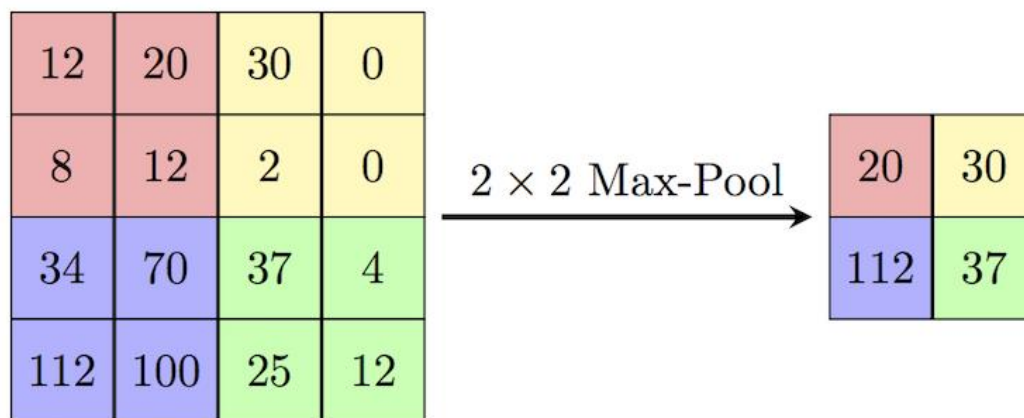
### 4.3.4 Activation Layer

The Activation Layer (AL) in CNN is used to apply non-linear activation function on the input data. The AL is important as we aim to have no linearity to the network otherwise it would have simple linear mapping between the input and output. The AL is commonly grouped with the CL to avoid any inherent linearity in the network. A common activation method used is ReLu which performs a mathematical activation function that can be repeated as many times as necessary [24].

#### 4.3.5 Pooling Layer

The Pooling Layer (PL) is responsible for several tasks which are related to down sampling of the feature maps after the AL has been applied [25]. There are many ways the pooling operating does this for example reducing the size of the image, reducing the number of parameters, max and average pooling. Max and average pooling work like the way CL convolve around an image section where the image is split into different sections and the filter slides over them, this method does the same selecting a pooling method (maximum or average) then calculating values based on the selected pooling method for each section which results in a smaller outputted new matrix [25]. PL is useful layer as it reduces memory cost resulting in an increase of efficiency.

Figure 8. Example of Max Pooling



#### 4.3.6 Fully Connected Layers

The Full Connected Layer (FCL) is the last layer of the CNN architecture where the model starts to predict or classify the output. This layer takes in an input volume from the previous layer and outputs an N dimensional vector where N is the number of classes that the application can choose from for example, if classifying an animal is a dog or cat N would be 2 [25]. Each number in the N vector represents the probability of a certain class.

### 4.3.7 CNN Architecture Chosen

At this point we now know the structure of a CNN Network, so it is now time to choose which CNN architecture that will be use. There are many different CNN architectures going back to the start where VGGNet and AlexNet were the most popular [26]. The issues of these models are the immense number of parameters used for example VGGNet was using over 138 Million parameters while also having a very small depth [27]. Although these networks worked well it just was not practical as the large parameters would result with a vanishing gradient problem. This is where newer Inception models such as ResNet were developed to reduce the number of parameters but also increasing the depth of layers resulting in higher accuracy and better efficiency [26].

**Figure 9. CNN Architectures Benchmarks**

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
ResNet50	98 MB	0.749	0.921	25,636,712	-
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
ResNeXt50	96 MB	0.777	0.938	25,097,128	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

As mentioned earlier we are not restricted by any hardware constraints such as memory size so the choice on which to use comes down to accuracy rating and efficiency. Figure 8 shows that the highest accuracy model is the InceptionResNet model with an impressive 55 million parameters and a depth of 572[26]. Since this is the most accurate rating model it was chosen for the project.

### 4.3.8 InceptionResNet Model

Before talking about the inception model first a discussion on ResNet is needed. ResNet is an artificial neural network (ANN) with the ability to skip connections or to jump over some layers. Typically, ResNet models use double or triple layer skips that contain ReLU and batch normalization in between. The reason that skipping over layers is useful is that it avoids the problem of vanishing gradients [29] by reusing activations from previous layers until the adjacent layers learns its weights [30].

The InceptionResNet model was designed by google and it was designed to solve the vanishing gradient issue mentioned earlier. Since stacking many convolution layers is computationally expensive a solution was made to have filters with multiple sizes which operate on the same level [28]. This allows the network to choose which filter size to use to learn the required information were relevant, for example comparing two persons faces 1 face takes up a larger portion of the image so the network may choose to use a larger filter size while the second person face takes up little of the image so a smaller filter size can be used. This leads to a network not only more memory efficient but also faster than other CNN models.

#### 4.4 FaceNet Architecture

As mentioned earlier the chosen face recognition method architecture chosen for this project is FaceNet developed by google. This section will go over face net architecture in more detail than previous sections. FaceNet [9] introduces a system for face verification and recognition that learns by mapping face images to a compact Euclidean space where distance corresponds to a measure of face similarity.

FaceNet trains a deep convolutional network which is optimized for embedding itself rather than having a classification layer seen on other approaches which is seen as a major bottleneck. The network is trained to output to be a compact 128-D embedding using triplet-based loss function which consists of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the two matching faces from the non-matching face by a distance margin [9].

**Figure 10. Triplet Loss function**



Figure 9 shows the triple loss function which minimizes the distance between an anchor and a positive which have the same identity. It also maximizes the distance between the anchor and a negative of a different identity [9].

FaceNet tested various amounts of CNN architectures including the inception model in order to find the most accurate one to use. Testing each network ourselves would take an exceedingly long time so we can trust that these results are accurate and can be used to determine which architecture to use.

**Figure 11. FaceNet Network Results**

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	87.9% ± 1.9
NN2 (Inception 224×224)	89.4% ± 1.6
NN3 (Inception 160×160)	88.3% ± 1.7
NN4 (Inception 96×96)	82.0% ± 2.3
NNS1 (mini Inception 165×165)	82.4% ± 2.4
NNS2 (tiny Inception 140×116)	51.9% ± 2.9

Figure 10 shows the results of FaceNet testing of networks, it shows the NN2 (Inception Model) was the most accuracy method. NN3 shows the same network but with a reduced input size of 160x160, similar with NN4 with an input size of 96x96. Lowering the input size drastically lowers the CPU requirements but as figure 10 shows lowers the accuracy [9].

Figure 12. FaceNet Inception Architecture

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	$L_2$ , 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	$L_2$ , 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	$L_2$ , 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	$L_2$ , 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	$L_2$ , 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	$L_2$ , 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

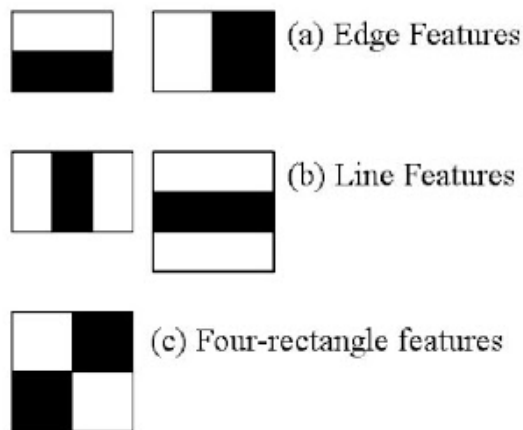
## 4.5 Face Detection Architecture

The face recognition architecture chosen discussed and analysed, a discussion on what face detection was needed. In simple terms the purpose of face detection is to determine if a face is in an image or not, it is an important step in face recognition as it stop unwanted processing of images that contain no faces. This section will go over face detection methods and the chosen method used for the project.

### 4.5.1 Haar Cascades

Haar cascades is a face detection method approach in which a cascade function is trained from a lot of positive and negative images to detect faces [37]. Haar features are used to extract features from the image, they make it easier to detect edges or lines in an image. Haar cascades offer a fast and light way for face detection but are prone to false positives [37].

Figure 13. Shows Haar Feature

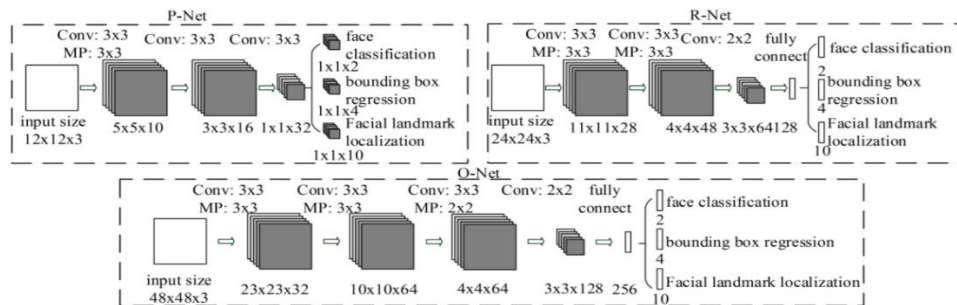




### 4.5.2 MTCNN

MTCNN is a framework for a solution for both face detection and alignment using multi-task cascade convolutional networks [32]. It is one of the most popular and most accurate face detection tools available that can deal with pose, illumination problems. It consists of 3 neural networks connected in a cascade (P-NET, R-NET, and O-Net). it can outperform many face detection benchmarks while retaining real time performance [33].

**Figure 14. MTCNN Structure**



### 4.5.3 Face Detection Chosen Method

During development both face detection methods were tested as they were not difficult nor time consuming to implement. The Haar cascade worked well but was only able to detect the face when directly facing the camera. MTCNN however worked exceptionally well with many different poses and lighting conditions for this reason it was the chosen method for face detection as working with a mobile drone we cannot assume the user will not always be directly facing the camera.

## 4.6 Face Tracking

With the face detection and recognition analysis the face tracking has yet to be discussed. This section will go over the face tracking problem and solution chosen. As discussed in the state of the art [2.3.2] the chosen method for this project was using a PID controller using Murtaza's implementation with some adjustments to work with the MTCNN face detection method. The method was chosen as it proves to be a well working implementation and rather than develop a new solution to a problem that has already been solved, it more efficient to implement this solution and adjust and improve on it where possible. Murtaza method is restricted as his implementation only allows the drone to rotate from a stationary position, so a solution to allow the drone to move backwards and forwards is also needed.

This paper introduces a way to solve this problem by using the already established face detection boundary box to determine how much area a face is taking within in a frame. Like the PID using error to adjust the drones speed, the face area size is used to control the drone's direction for example if a face is taking a large area size the drone will move backwards and vice versa. This solution will need a pre-determined safe area like PID desired outcome, an area where the drone is not to close to a face where it could cause harm and not too far that the face cannot be recognized.

As discussed, the PID requires a desired outcome which acts as the goal to head towards, and an error to show how far we are from the goal. The desired outcome is the drone camera facing the centre point of the boundary box to keep the face clearly in frame. The error is calculated based on how far the centre of the frame is from the centre of the boundary box, with the drone's flight adjusted to minimize the error thus moving toward keeping the centre point in the centre of the frame.

## 5 Implementation of Core Components

The design has now been discussed with major components been chosen, it is now time to move on to implementation, this section will discuss how the application main components were implemented.

### 5.1 The Model

As discussed, the project will use FaceNet architecture with a ResNet CNN for face recognition. To train this network from scratch with a large dataset would take a long time, according to FaceNet [9] their training method took 1,000 to 2,000 hours. So, to combat this issue a pre-trained method was introduced as discussed earlier. The use of this pre-trained network should not affect performance compared to a newly trained network.

At this point when implementing the network, a useful repository for the inception ResNet model was found. Facenet-pytorch [31] repository offered the use of the Inception ResNet architecture with the option of pre-trained methods using the VGGface2 or cassia-webface datasets. Another useful feature from the repository is they also include a MTCNN implementation which they claim to be the fastest implementation available. VGGFace2 was chosen as the dataset as it shown to have a higher LFW accuracy [31]. After the model was implemented, the next step was to begin testing by creating the embedding space. The testing dataset chosen was LFW as this is the benchmark for face recognition models, my face will also be added to determine if the model works correctly, and my face can be recognised. The next step was to create an algorithm to take the testing dataset and iterate it through the ResNet Inception model (RNI) producing an embedding value for each of the images and attaching the labels to each of the images. These values are then added to a file, from this point on the LFW dataset images are no longer required as the embedding values stored in the file is all that is needed. The file can then be loaded into any python class using Pillow [32] or similar methods.

### 5.2 Drone Connection and Live Feed

To connect to the drone as discussed, the DJITelloPy [2.4.1] library is used. The drone is connected through the WI-FI, no password required. Once the drone is connected frames can be returned, a requirement is that the method of retrieving frames be in a while loop as we are using live video. The drones live feed is resized and displayed in a window using OpenCV [2.4.2].

### 5.3 Implementing Face Recognition

Using the file generated earlier, two lists are created an embedding list which contains a list of all the embedding values and a name list that contains all the names labels. The index between these two lists will be the same meaning the first embedding value and the first name label will match correctly.

To begin recognising the face from the drone feed, the frame must be embedded, before this can happen the frame needs to go through the MTCNN face detection process to return the face area so that we do not need to process a larger image area than necessary or process a frame with no faces at all.

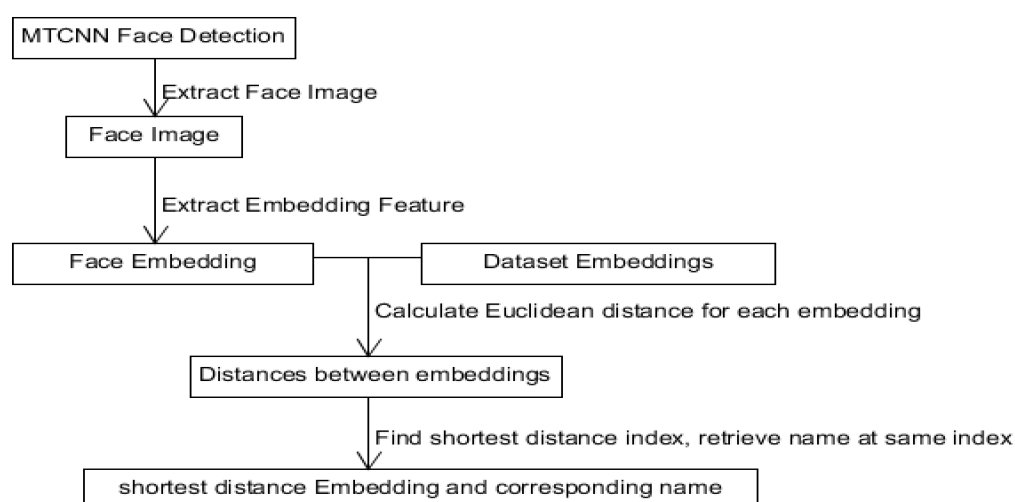
once the face is returned from the MTCNN, the next step is to turn that face into an embedding value using our CNN. With the embedding value successfully generated the next step is to start searching for the closets distance embedding value stored in the file. To do this we first need to calculate the distance from our face embedding value calculated by our CNN to every embedding value stored in our dataset which results into a list of distances.

From this list we can run a simple method to return the minimum distance value which will be the embedding value that is closest match to the targets. We can also create a confidence score using the distance since distance corresponds to face similarity we can say if it is not a short enough distance then the output is not a good enough result.

The final step is to return the embedding value and the name label with the same index to a method to add it to our bounding box using OpenCV to show the result frame to the user.

An important note when running this method, it runs the same calculation for each frame excluding frames with no faces, this means it is very CPU intensive and initial would result in low FPS for the resulting video feed. To solve this issue the frame size was reduced much in the same way that FaceNet solved their CPU requirements by lowering the size of the input. The result was a smoother FPS but still not perfect and a smaller frame box being shown to the user, the accuracy however was not affected as it was still able to recognize the correct individual.

**Figure 15. Flow Chart of Face Recognition Method**



## 5.4 Face Tracking

For the face tracking method as mentioned the project uses Murtaza's implementation of a PID controller and the solution devised to deal with direction allowing for less restrictive drone movements. Murtaza implementation uses the proportional and derivative which mentioned in the state of the art [2.2.3] deals with present and future error changes. To calculate the error the bounding box data is used to determine the centre points, width and height and area. Some changes were necessary from Murtaza implementation as he used cascades as his face detection method were, we are using MTCNN which returns different values i.e., int instead of float. Using the box data, the error is calculated and used to control the yaw of the drone keeping track of previous errors, so the drone does not overshoot missing the target.

To deal with direction the method suggested was to use the face area returned by the bounding box. With this area we can determine how far the face is from the drone and adjust accordingly. The first step was to determine how much area a face should take to find out the safe distance between the drone and the face would be. To find this out the drone camera was connected without flying the drone and then pointed at a face with the area being calculated for each frame, the drone was then moved closer to the face until it would be considered dangerous close, and that area was noted as the point where the drone should not cross. To find out how far the drone could go until it could not recognise the face, the drone was moved outward and when the drone could not recognise the face that area was noted down as the drone should not go past that area. Using these two noted down areas we can determine our desired area of between 15000-20000 and write a function to determine if the drone is in the desired area or not and adjust the flight accordingly.

The result was a working flight controller that accurately followed a face around an area not getting too close or too far from a face. To deal with the issue when a face is not within a frame the drone is simply to not move from its location until the face is back in frame. This would be replaced in the future with a pathing algorithm which will be discussed in the evaluation section.

The last step was to get the face tracking to follow a specific person using face recognition. To accomplish this face recognition is run for each frame and if the person recognised is a match to the missing person the box for that person face is passed through to the face tracking function so the drone will only follow the box matching the specified missing person. To deal with multiple faces being within the frame, only one box is drawn at a time with the goal being when the target face is recognised that box is used moving the drone towards that specific person. Due to the size of the camera and the frame it is only possible to fit two faces within a given frame and still maintain high accuracy.

The issue with this approach is that due to the CPU restrictions with face recognition mentioned earlier, the frames were not high enough for a smooth flight for the drone due to frames being skipped, delay and stuttering which result in inaccurate errors being calculated, and commands being delayed.

one solution was devised to solve this issue, which was to lower the input size of the face recognition even further which proved to be a failure as the face detection and recognition accuracy dropped too low for the tracking to use. Other possible solutions could be to only run face recognition every few frames instead of each frame or to only run face recognition up to the point the target individual is recognised however due to time restraints of the project these methods cannot be implemented or tested as it would require complete restructure of the application code, more on this topic will be discussed in the evaluation section.

## 6 User Interface Design

With the core components developed the next step was to develop the user interface with some helpful functionality to add to the useability of the application. Before diving straight into the development of the application side, design documentation is needed to give a clearer picture of what the application will look like and function beforehand. This section will cover the key requirements of the application as well as the design of the application.

### 6.1 Requirements

When looking over the objectives of the project there are several key requirements the graphical user interface (GUI) must cover.

1. The application must allow the user to connect to the drone.
2. the application must display a live feed of the drone camera with the face recognition/detection results clearly visible which in this case means the bounding box will show when a face is detected and the name of the individual that was determined from the face recognition component also clearly shown.
3. The application should allow the user to add new missing individuals photographs and names so that they can be added to the dataset.
4. The application should allow the user to check for missing individuals that are already in the dataset, so that they can determine if they need to be added or not.
5. The application should also show the battery percentage of the drone so they user is kept informed.
6. Clear instructions on how to operate the application should also be provided to the user.
7. The application should have appropriate error handling were necessary

After analysing these requirements, it was decided that the application would consist of a main page acting a central hub that would display the drone feed, battery information, a way to connect to the drone and navigation between two other pages; add new missing person page and a search missing person's page.

The add new missing person page should allow the user to add a new person to the dataset, this means the user will need to be able to enter a person's name and browse for an image of that individual so that image can be embedded and added to the dataset.

The search missing person page should allow the user to check to see if an individual is already in the dataset, this functionality should also be added to the search procedure as an error handing measure to make sure the drone is not trying to find someone it will not be able to accurately recognize. I felt it necessary to add this functionality as a separate function in the case that the user wants to check for an individual but does not want the drone to take flight beginning the search procedure.

## 6.2 Navigation

As with any application it is important to keep navigation as simple as possible, not to confuse the user.

**Figure 16. Navigation Map**

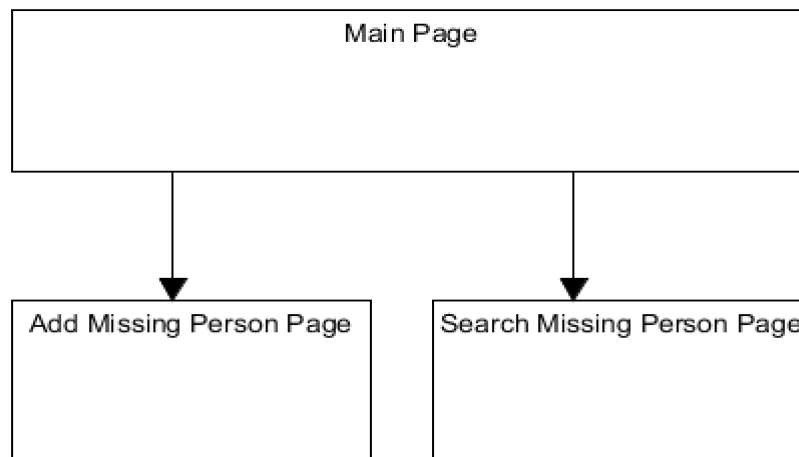


Figure 14 shows the initial navigation map for the application. As discussed, they are two pages that the user can navigate too the add missing person page and the search missing person page with the main page acting as the home of the application. The home page will allow navigation between pages, display the camera feed and some drone functionality commands such as land. Each page can be accessed from the main page, but an important note is the main page should never close as it acts as the main display so should always be visible. For this reason, the two other pages will open as separate windows that can be interacted with independently from the main page. The add missing person page should allow the user to add a new individual to the dataset in a non-complicated and convenient way. The search missing person page should allow the user to search for existing individuals in the dataset before taking flight.

## 6.3 User Interface Wireframes

This section will cover the initial design for each page, so that we can envision how the application will look and how different elements will interact with each other.

### 6.3.1 Main Page

Figure 17. Main Page Wireframe

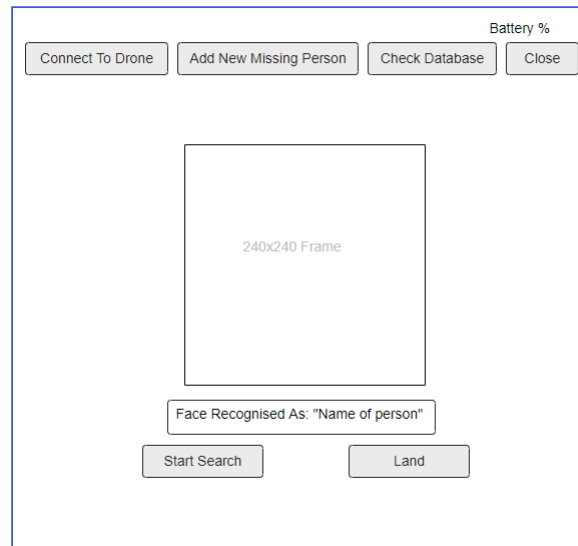


Figure 15 shows the main page design. The design is a simple interface with no distractions with clearly labelled buttons, a battery % being displayed in the corner and the frame placed in the centre for maximum visibility.

The buttons offer the following functionality:

**Connect to Drone:** will connect to the drone.

**Add New Missing Person:** will open the add missing person page as mentioned above

**Check database:** will open the search missing person page as mentioned above

**Close:** will land the drone if connected, then close the application

**Start search:** this will commence the search procedure, first opening a new window where the user will be able to enter the person's name they are currently looking for and begin the search for that specific person.

**Land:** this will land the drone if connected.

The frame window will display a constant drone camera feed with the results of the face recognition/detection clearly visible within the frame.

The face recognised message box will also display the name of the person identified for a clearer result view. This box should update each time the face recognition detects an individual.

### 6.3.2 Add Missing Person Page

**Figure 18. Add Missing Person Page Wireframe**

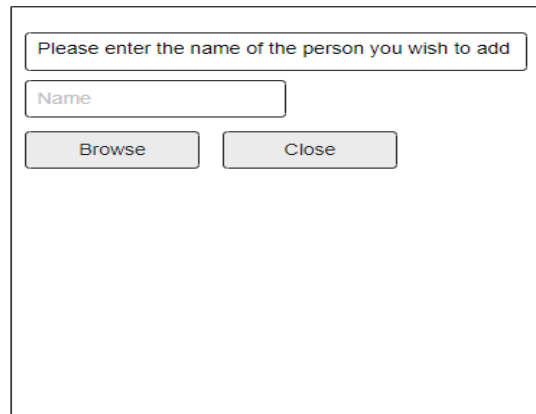
Figure 16 shows the add missing person page design. This follows the simplistic design of the main page allowing only for necessary functionality. As mentioned, this window should be opened as a separate window and live independently from the main page.

This window will allow the user to enter a name and browse for an image containing that individuals face.

The format on the name is irrelevant and does not need to match the other name formats found in the dataset as it acts only as a label. So simple error handling can be used for example entering numbers instead of letters.

The user can also add images for individuals that are already in the dataset by entering the same name and browsing for an image of that individual, since the embeddings are not classified, they are not concerned what label they are assigned.

For an image to be accepted MTCNN should be used to determine if the image contains a face first. Advanced error handling would be to use MTCNN face detection to determine how many faces are in the image, as only one face should be present.

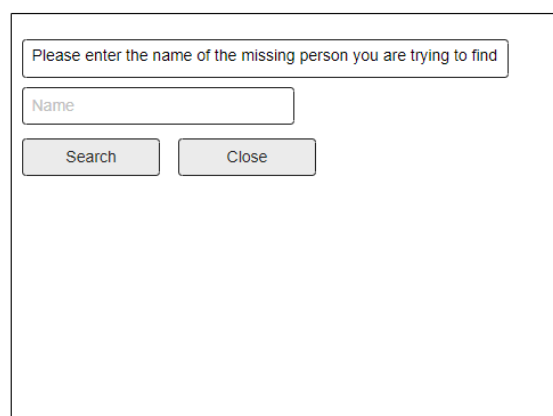


### 6.3.3 Search Missing Person Page

**Figure 19. Search Missing Person Page Wireframe**

Figure 17 shows the search missing page design. This page follows the same design as the add missing person page. It allows the user to enter a name and search the dataset for a matching name.

If the entered name is match to a name in the dataset, the appropriate message will display showing the user that there was a match found.



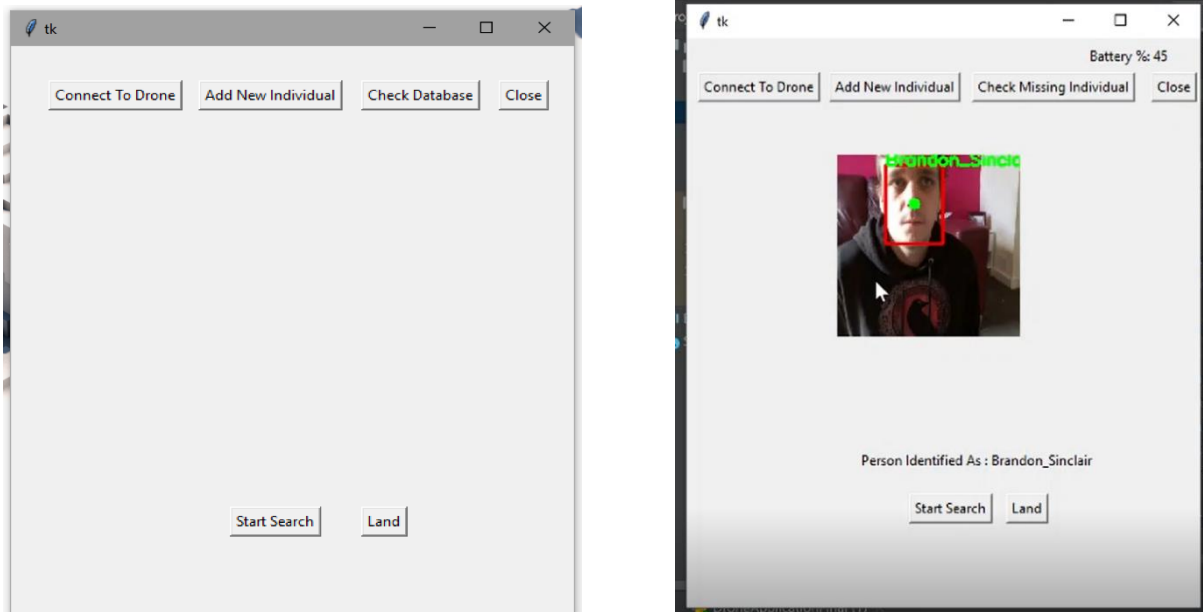


## 6.4 Implementing the UI

With the design on the application created, the next step was to create the UI for the application. To do this python has a standard GUI library Tkinter [36] was used, this was an unfamiliar library, so it took some time to learn but once learned it was a simple process to develop the UI. This section will cover what the final app functionality and appearance.

### 6.4.1 The Main Page

Figure 20. Shows the application main page



Figures 18 show the application main page final state, when the application first start this page will be the first thing the user sees. The drone feed will start to stream once the user has started the search process. Once the process starts the drone will launch from its location and given a few seconds to orientate itself before the stream will display.

The add new individual and check database buttons will open their respected pages as separate windows that can be left open or closed at the user's desecration.

The land button will land the drone once a check to determine if the drone is connected is made otherwise an exception will occur. If the drone is not connected and the user selects this button an alert box will appear to let the user know that the drone is not connected.

The face recognised message located at the bottom will update each frame displaying a clearly visible name to the user.

The battery is located on the top right corner where users would typically find the percentage, this is updated in real time as well.

The size of the frame is still a concern as we were forced to lower the frame size to deal with CPU restrictions. A solution to this was to increase the frame size after all the processing had been completed while keeping the same aspect ratio but the result was a far less clear image that would be unusable in the real world, more on this topic will be discussed in the evaluation stage.

## 6.4.2 The Add Missing Person Page

Figure 21. shows add missing person application page

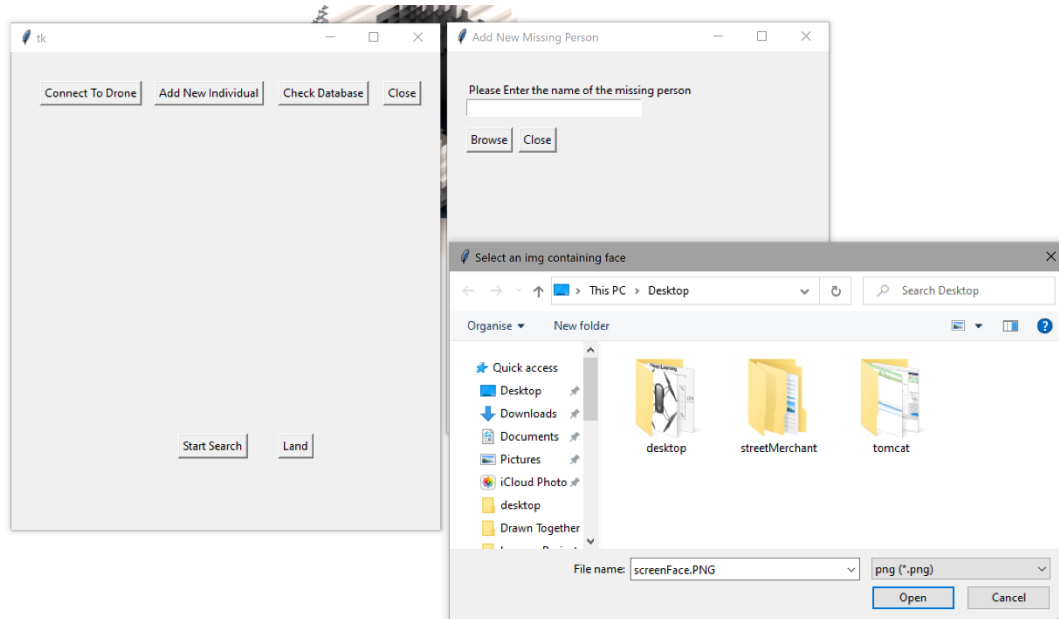


figure 19 shows the add missing person application page browsing for an image. The user opens this page by selecting the add new individual button from the main page, this opens the add missing page. Once on this page the user can type in a name and browse for an image.

Error handling has been implemented to check that the image contains at least 1 face, if the image does not contain an image an alert popup is shown to the user to inform them that the photo was not accepted and to select a new image.

Once the image is selected and approved, the image embedding, and name label is created and inserted into the dataset and saved into the dataset file for immediate and future use. Once this process is completed the appropriate message is displayed informing that the person was added to dataset successfully.

### 6.4.3 Search Missing Person Page

**Figure 22. Shows the search missing person page**

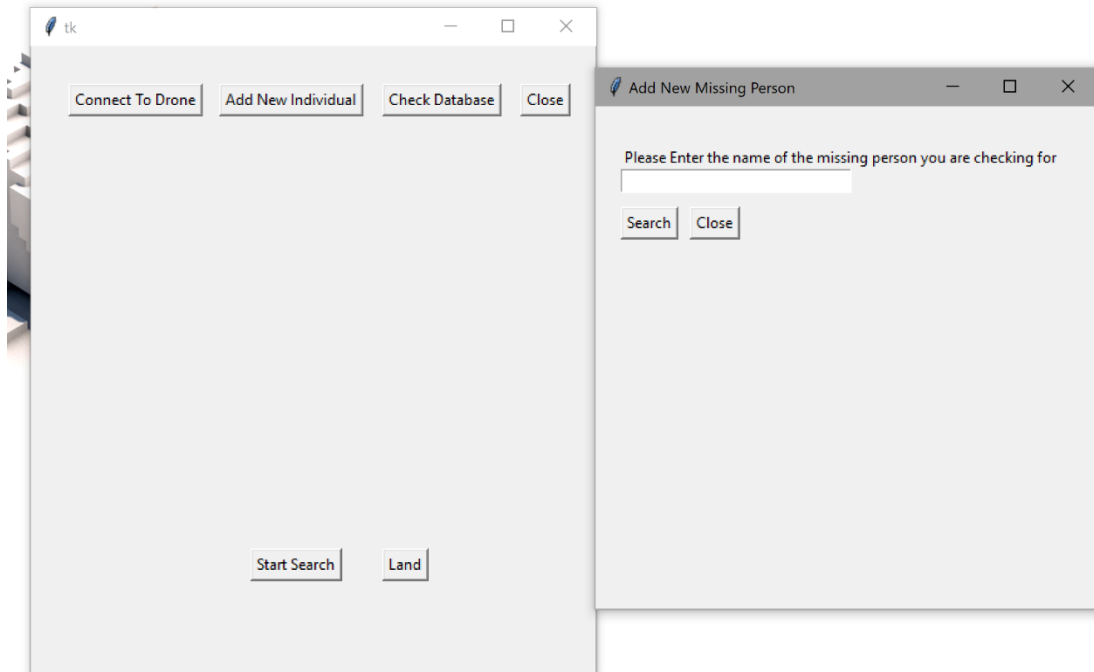


Figure 20 shows the search missing person page final state. This page allows the user to enter a name which will search the dataset for a matching name. if a match is found the appropriate message informing the user that an individual matching name was found and similarly if no matching name is found.

## **7 Testing**

Testing of the application has been performed throughout its development and each feature added to the interface has been tested extensively. To begin with initial unit testing was performed on each of the main components, UI, face detection, face recognition and tracking before integrating each component to work together. This section will go over each of the test's cases with a description on each test.

### **7.1 User Interface**

This section will go over the user interface tests, testing the components, error handling and the drone commands such as connect and land.

#### **7.1.1 Buttons**

A test plan was created (see appendix 1) to test each button to make sure they function correctly. The test plan consists of the plan, expected result and the actual result. No test resulted in a failure meaning that each button is functioning correctly.

#### **7.1.2 Drone live feed displayed**

To test the drone feed the drone was connected and the start search was initialised as mentioned the drone feed will not be until the search has started. The result is that drone feed is displayed in the centre of the application.

### **7.2 Add new individual**

For this test, a new user was added using one image of their face by first entering the new user's name in the name box label and then selecting the browse button which opens a browse interface and selecting an image containing the face which is then added to the dataset. This user was not previously added to the dataset so would not be recognized before this point. Once added the drone was connected and pointed to the new user face and that user was successfully recognised which proves that the test passed correctly.

A second test was run to determine if the photo submitted was valid meaning it did indeed contain at least 1 face. The test was conducted by first entering a name and then browsing for an image with no faces, this image is then submitted and passed through to the face detection function to determine if the image contains any faces which in this case it did not so a popup alert informing the user that the image was not valid was displayed proving that the test was run successfully.

### **7.3 Search for existing user**

For this test, a name is entered into the name box which is known to be in the dataset and the search button is pressed which results in a popup being displayed saying that a match was found. Similarly, to test if a user is not in the dataset the same test is run but with a name that is not known in the database with the result being a popup message saying that no match was found proving that this test was successful.

## 7.4 Face Detection

For this test, the drone is connected and pointed to a face to test the MTCNN face detection. The result was the boundary box was drawn over the face for each frame. Various poses were also tested to determine the degree of angle of where the face detection fails to detect the face. This was tested by turning the face to full left and then full right and marking where the face detection falls off.

**Figure 23.** shows the face detection test results

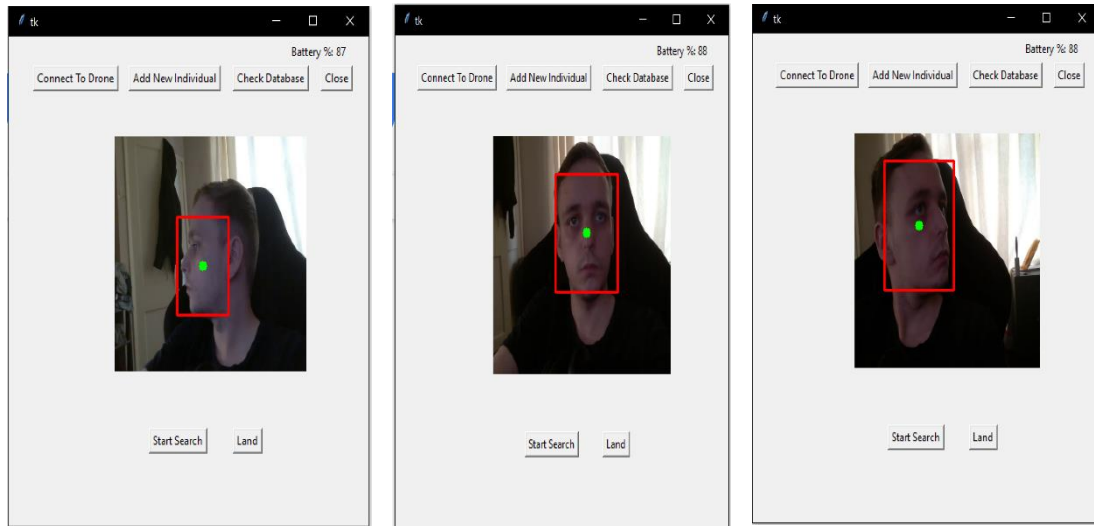


figure 22 shows that the tests were successful as the face detection was successfully able to detect a face from 3 different poses. From the figures the face detection did not fall of even at full turn when only half the face was visible.

## 7.5 Face Recognition

To test the face recognition the drone was connected and pointed towards the face like the face detection test. The test will also test different poses to see at which point the face recognition fails to recognise a face.

The parameter for this test is that the distance between the frame embedding and the embedding stored in the embedding space must be less than 1 as anything higher than this can be considered an inaccurate result.

**Figure 24. Shows Face Recognition Test Result**

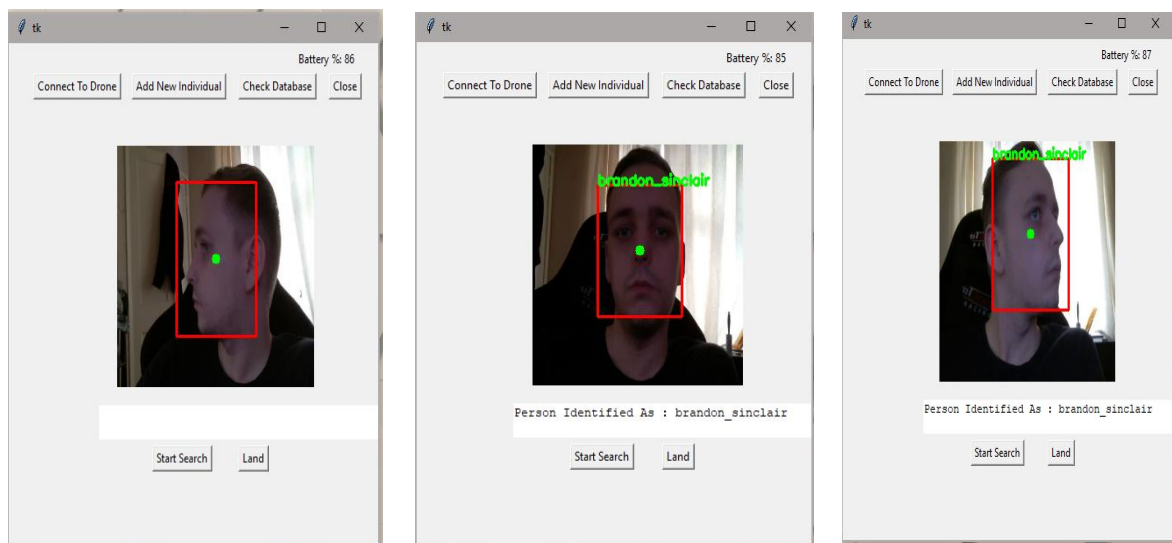


Figure 23 shows the result of running the face recognition test. As we can see the face recognition was able to successfully recognise the face in two of the scenarios with the name being clearly visible in the frame as well as the text below the frame. The image on the left however was a failure, this is because most of the face is not visible and thus an embedding with a distance value below 1 was not found, which is the expected result in this case because it proves that inaccurate results will be ignored. The image on the right was a pass even though half of the face is not visible, but the correct person is still being recognised. From this we can determine that the face recognition can still return accurate results even with half the face not visible.

## 7.6 Face Tracking Testing

This test will test the tracking and safety features, the test will start by launching from an area and start by detecting any faces it can find. For this test face recognition will be disabled for the fps issues mentioned earlier so the tracking will track a single face it detects. Once a face is detected that face should then be tracked and followed while maintaining a safe distance. The parameters for the safe area size are set to be between 15000-20000 as previously mentioned this is deemed to be a good distance between the users face and the drone.

The test will begin by placing the drone in an area and only one person will be inside that area as we are not yet testing how multiple faces will impact the test. The person should first stand in front of the drone so the face can be detected and will then move around the area while maintaining direct eye contact with the drone's camera, so the detection is not lost. The drone should maintain a safe distance as it follows the person to test the safety features. In the case of the drone getting too close the person is permitted to move back if they feel the drone will hit them as we do not want any accidents. Once the tracking capabilities are tested the person should then move out of the drones view to test the secondary safety feature which is that the drone should not move from its location when no face is detected.

[Appendix 2, figure 1] shows the initial start of the test, below we can see the console output which contains the tracking output (for visual feedback) and commands (red) being executed. We can see the error, direction positive meaning moves forward negative being move backward, speed is the yaw (rotation) and the centre points which is the green dot shown in the centre of the box.

[appendix 2, figure 2] shows the face is close to the drone with an area of above 20000 which means that it is above the desired area and thus the drone should move back which it successfully does as we can see the direction is -10 meaning it is telling the drone to move back by 10. We can also see the rotation working as well as the face is slightly to the right which increases the error (positive) meaning the speed is adjusted to rotate right to reduce the error.

[appendix 2, figure 3] shows the face being too far from the drone with an area less than 15000 meaning the drone should move forward which is successfully does as we see the direction is positive 10. The error is also very close to 0 as the face is nearly in the centre of the frame so little adjustment is needed which we can see in the low speed being calculated.

[appendix 2, figure 4] shows the drone hovering in the same position when no face is detected which is a safety feature implemented until pathing functionality can be added.

[appendix 2, figure 5] shows the test with two people in the frame, as mentioned the MTCNN should only draw 1 box at a time, with the face recognition run on each until the missing individual is recognised then only follow that person. although the face recognition will not be run, we can still test the box been drawn only on one person at a time. The test went as expected due to no specified person to follow the drone will try to follow the person with the box been drawn. The box was drawn on each person seemingly at random regardless on distance or amount of face covered, more research is needed to understand why this is the case. We can assume that once the face recognition restriction is solved the drone would only follow the missing individual specified.

In conclusion the face tracking successfully passed all tests when face recognition is not turned on. it can track and follow an individual face while maintaining a safe distance.

## **8 Conclusion**

### **8.1 Summary**

The final build of the project partially achieves the goals and targets set out. The application can successfully detect and identify an individual with high accuracy and the drone can successfully track and follow that individuals face as they move with restrictions. the main issue due to the face recognition being so CPU intensive resulting in low fps it is not practical to use face recognition and tracking at the same time because the tracking requires as smooth fps as possible otherwise important frames might be skipped leading to inaccurate errors being calculated and delays in commands being made. The face tracking however works smoothly when only using face detection therefore is more practical to use these functions together to keep a face within the frame until a solution to the CPU bottleneck is found.

### **8.2 Evaluation**

I believe that this project met its goals from [1.2] because the application can control an autonomous drone capable of tracking an individual with the restriction of the CPU, a very accurate face recognition solution that only requires one image of the individual in order to be recognised. Overall, the project works well enough as a proof of concept, I would have preferred to be able to fix the CPU limitation but unfortunately there was not sufficient time to dedicated for that.

The FaceNet architecture proved to be a huge success for the project. FaceNet accuracy and less restrictive neural network worked exceptionally well successfully achieving the goal of only requiring one image for accurate recognition, so for the project I would not of changed this chosen architecture.

The tracking method chosen also proved to be a huge success for the project. Murtaza PD implementation allowing for the rotation as well as my improvements and chosen solution for allowing for forward and backward drone movement with safety features proved to work exceptionally well so I would not of changed this chosen method for tracking.



## **8.3 Future Work**

### **8.3.1 Limitations**

As mentioned, there is a restriction on the success of the application which is that a specific individual cannot be tracked because face recognition is too CPU intensive it cannot provide a smooth enough frame rate which is a requirement of the tracking component. This was an issue that should have been addressed in the problems and analysis so a solution could have been planned and tested in the given time for the project. The reason this was not in section 3 was due to lack of research in the problems with real time face recognition, an oversight on my part. With no research being done to this issue the problem was not discovered till later in the implementation once the face recognition and tracking were implemented and tested. With not much time a few methods were tested to fix this issue, but no solution was found. If I were to do the project again, I would research the problems of real time face recognition and come up with a solution that would result in a smooth enough frame rate for the tracking to work correctly.

The current version of the application does not allow the drone to change its height. This height is set to be a pre-set value which takes it to the height of the average person which is suitable for the application's proof of concept however will need to be implemented to the tracking capabilities before the application can be deployed into the real world. To do this the same PID controller can be used in the same way the rotation is determined with adjustments to the calculation. However due to time restraints this method has not been implemented or tested.

### **8.3.2 Added Functionality**

As mentioned, this project was a proof of concept of a real-world search and rescue operation. There are many functionality components needed before this application could be deployed into the real world for example pathing, body detection/tracking, object detection and avoidance, GPS location.

Pathing would allow the drone to move to one target to the next which is a necessary functionality as currently the drone will remain stationary when no new faces/targets are detected which is not very useful in a search and rescue scenario. This pathing algorithm would also require object detection avoidance to deal with physical objects that would block its path.

Body detection would also be a significant improvement for tracking as it is a more larger and easier detected object than a face. The reason this was not chosen as the tracking method was due to another student who was doing a similar project developing this method of detection.

Having GPS location of the drone would be useful in a search and rescue scenario as currently we are relying on the video feed to determine the drone's location. A GPS would require either a device capable of pinging a GPS location attached to the drone or a new drone being used with a GPS built in.

### **8.3.3 UI improvements**

Due to time restraints the UI was not designed with much detail aiming for a more practical appearance. If given more time I would improve the UI with a much more visually appealing interface with icons and colour. This is not a functional improvement but rather an improvement to benefit the user experience.

## References

- [1] Scottish Mountain Rescue. 2020. Any Day, Any Hour, Any Weather - Scottish Mountain Rescue. [online] Available at: [Accessed 13 March 2021].
- [2] Ryzerobotics.com. 2020. Tello. [online] Available at: [Accessed 13 March 2021].
- [3] Play.google.com. 2020. [online] Available at: <[https://play.google.com/store/apps/details?id=com.vision.pgminin.tellovision1d&hl=en\\_US](https://play.google.com/store/apps/details?id=com.vision.pgminin.tellovision1d&hl=en_US)> [Accessed 14 March 2021].
- [4] P.J. Phillips, J.R. Beveridge, B.A. Draper, G. Givens, A.J. O'Toole, D.S. Bolme, J. Dunlop, Y.M. Lui, H. Sahibzada, S. Weimer, "An introduction to the good, the bad & the ugly face recognition challenge problem", *image and vision computing*, March 2012
- [5] Le, T., 2011. Applying Artificial Neural Networks for Face Recognition. *Advances in Artificial Neural Systems*, 2011, pp.1-16.
- [6] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, "Deepface: Closing the gap to human-level performance in face verification", *Conference on Computer Vision and Pattern Recognition*, 2014. 2.
- [7] Y. Sun, D. Liang, X. Wang, X. Tang, "DeepID3: Face recognition with very deep neural networks", *ArXiv* 2015
- [8] Paperswithcode.com. 2021. *Papers with Code - Labeled Faces in the Wild Benchmark (Face Verification)*. [online] Available at: <<https://paperswithcode.com/sota/face-verification-on-labeled-faces-in-the-wild>> [Accessed 16 March 2021].
- [9] F. Schroff, D. Kalenichenko, J. Philbin, "FaceNet: A unified embedding for face recognition and clustering", *arXiv*, June 2015
- [10] Play.google.com. 2020. [online] Available at: <[https://play.google.com/store/apps/details?id=ch.zhaw.facerecognition&hl=en\\_US](https://play.google.com/store/apps/details?id=ch.zhaw.facerecognition&hl=en_US)> [Accessed 18 March 2021].
- [11] GitHub. 2020. *Damiafuentes/DjitelloPy*. [online] Available at: <<https://github.com/damiafuentes/DJITelloPy>> [Accessed 18 March 2021].
- [12] L. Khurana, A. Chauhan and P. Singh, "Comparative Analysis of OpenCV Recognisers for Face Recognition," *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 485-490, doi: 10.1109/Confluence47617.2020.9058014.
- [13] Medium. 2020. *How to Build A Face Detection and Recognition System*. [online] Available at: <<https://towardsdatascience.com/how-to-build-a-face-detection-and-recognition-system-f5c2cdfbeb8c>> [Accessed 18 March 2021].
- [14] 2020. [ebook] Available at: <[https://dl-cdn.ryzerobotics.com/downloads/Tello/20180211/Tello+Disclaimer+and+Safety+Guidelines+\(EN\)+v1.0.pdf](https://dl-cdn.ryzerobotics.com/downloads/Tello/20180211/Tello+Disclaimer+and+Safety+Guidelines+(EN)+v1.0.pdf)> [Accessed 18 March 2020].
- [15] GOV.UK. 2020. *Data Protection*. [online] Available at: <<https://www.gov.uk/data-protection>> [Accessed 25 October 2020].
- [16] Hussain Shah, J., Sharif, M., Raza, M., Murtaza, M. and Saeed-Ur-Rehman, 2015. Robust Face Recognition Technique under Varying Illumination. *Journal of Applied Research and Technology*, 13(1), pp.97-105.
- [17] PyTorch () PyTorch, Available at: [20] <https://pytorch.org/> (Accessed: 22 March 2021)
- [18] Skymind, "Comparison of ai frameworks." [Online] Available: <https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch>. (Accessed: 22 March 2021)

- [19] Tensor Flow () Tensor Flow, Available at: <https://www.tensorflow.org/> (Accessed: 22 March 2021).
- [20] [26] Aurelien Geron, Hands-On Machine Learning with Scikit\_learn & TensorFlow, 1st ed. "Introduction to Artificial Neural Networks", 2017, p. 111.
- [21] Medium. 2021. *An introduction to Convolutional Neural Networks*. [online] Available at: <https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7> [Accessed 23 March 2021].
- [22] Medium. 2021. *Understanding of Convolutional Neural Network (CNN) — Deep Learning*. [online] Available at: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148> [Accessed 23 March 2021].
- [23] A. Deshpande () A Beginner's Guide to Understanding Convolutional Neural Networks, Available at: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-UnderstandingConvolutional-Neural-Networks/> (Accessed 23 March 2021).
- [24] Brownlee, J., 2021. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/> [Accessed 23 March 2021].
- [25] Brownlee, J., 2021. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/> [Accessed 23 March 2021].
- [26] Medium. 2021. *Illustrated: 10 CNN Architectures*. [online] Available at: <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d> [Accessed 23 March 2021].
- [27] Medium. 2021. *LeNet 5, AlexNet, VGG -16 from Deeplearning.ai*. [online] Available at: <https://medium.com/@shahariarrabby/lenet-5-alexnet-vgg-16-from-deeplearning-ai-2a4fa5f26344#:~:text=One%20varies%20cool%20thing%20about,simplified%20these%20neural%20network%20architectures.&text=It%20is%20a%20really%20deep,of%20about%20138%20million%20parameters.> [Accessed 23 March 2021].
- [28] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.
- [29] En.wikipedia.org. 2021. *Vanishing gradient problem - Wikipedia*. [online] Available at: [https://en.wikipedia.org/wiki/Vanishing\\_gradient\\_problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) [Accessed 23 March 2021].
- [30] En.wikipedia.org. 2021. *Residual neural network*. [online] Available at: [https://en.wikipedia.org/wiki/Residual\\_neural\\_network](https://en.wikipedia.org/wiki/Residual_neural_network) [Accessed 23 March 2021].
- [31] PyPI. 2021. *Pillow*. [online] Available at: <https://pypi.org/project/Pillow/> [Accessed 23 March 2021].
- [32] Ieeeexplore.ieee.org. 2021. *Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/7553523> [Accessed 27 March 2021].
- [33] Medium. 2021. *How Does A Face Detection Program Work? (Using Neural Networks)*. [online] Available at: <https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff> [Accessed 27 March 2021].
- [34] MURTAZA'S WORKSHOP. 2021. *Drone Programming Course - MURTAZA'S WORKSHOP*. [online] Available at: <https://www.murtazahassan.com/courses/drone-programming/> [Accessed 28 March 2021].
- [35] 2021. *Understanding PID Control, Part 1: What Is PID Control*. [image] Available at: [https://www.youtube.com/watch?v=wkfEZmsQqiA&ab\\_channel=MATLAB](https://www.youtube.com/watch?v=wkfEZmsQqiA&ab_channel=MATLAB) [Accessed 28 March 2021].

[36] Docs.python.org. 2021. *tkinter — Python interface to Tcl/Tk — Python 3.9.2 documentation*. [online] Available at: <<https://docs.python.org/3/library/tkinter.html>> [Accessed 2 April 2021].

[37] Medium. 2021. *Haar Cascades, Explained*. [online] Available at: <<https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d#:~:text=Haar%20cascades%20are%20machine%20learning,combination%20of%20%E2%80%9Cweak%E2%80%9D%20learners.>>> [Accessed 14 April 2021].

## Appendix 1 Test Plan

SOFTWARE COMPONENT NAME: Drone App Buttons			DATE: 30/03/2021
Test No	Purpose/Type of Test	Expected Results	Pass/Partial/Fail
1	Connect to drone button will connect to the drone. (requires drone to be connected by WI-FI)	Drone connects, popup alerts user to successful connection.	Pass
2	Add new individual button opens add new individual page	Add new individual page opens as separate window	Pass
3	Check Database button opens the check database page	Check database page open as separate window	Pass
4	Close button, lands drone if connected and then closes the application	Lands drone if connected, closes application	Pass
5	Start Search button opens a new window to start the search procedure	New window opens	Pass
6	Land button will land the drone when pressed	Drone will land	Pass

7	Browse (Add new Page) will open a browse interface allowing the user to select an image	Browse Interface will show allowing a png or jpg image to be selected	Pass
8	Search (Check page) will run the search operation returning a popup showing the result	Popup showing result displayed to user	Pass



## Appendix 2 Face Tracking Screenshots

Figure 1

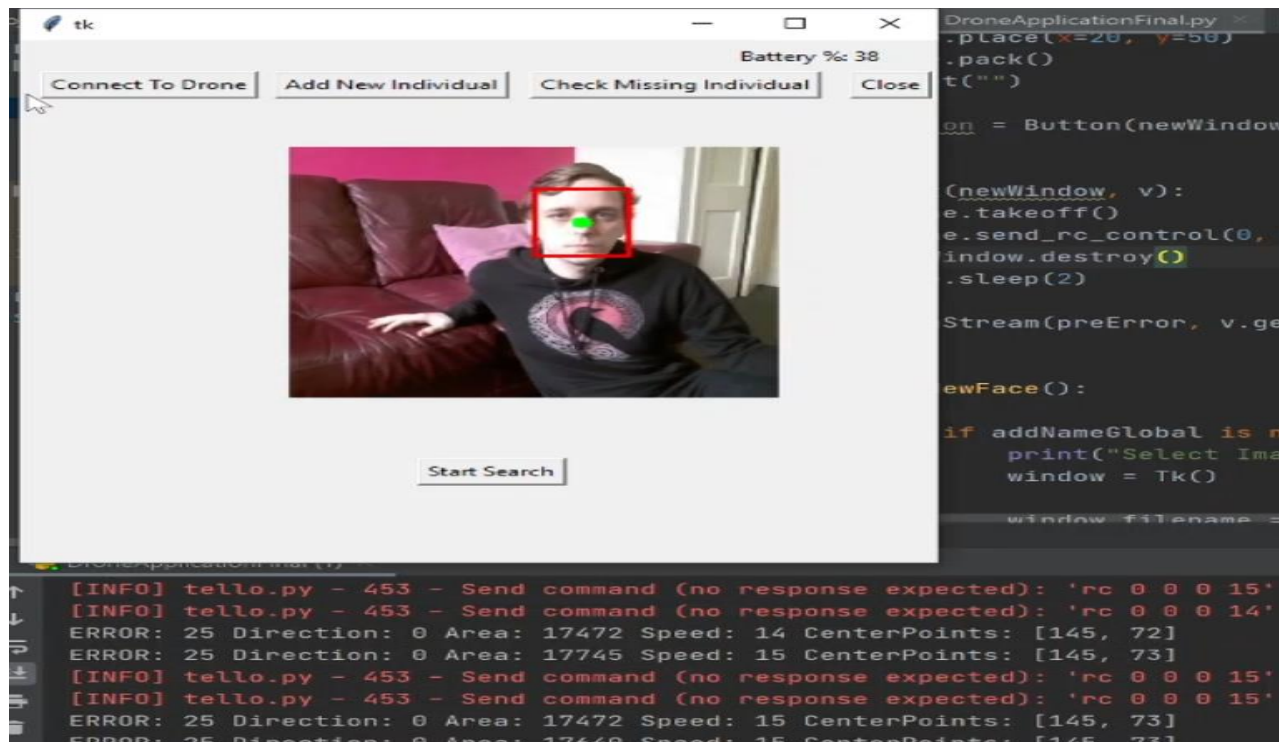




Figure 2

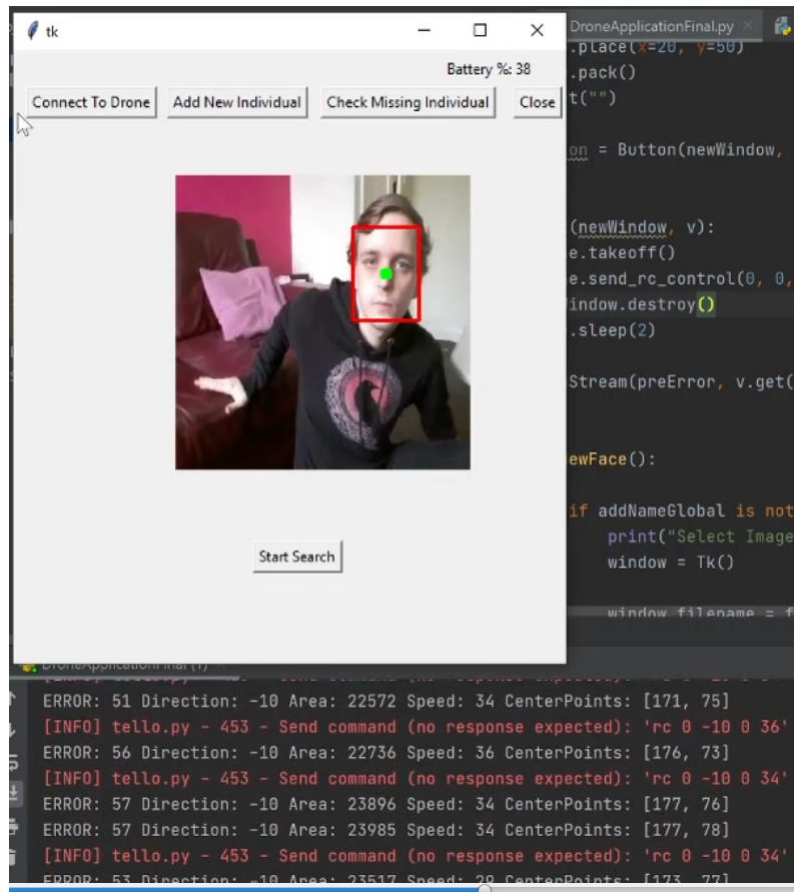


Figure 3

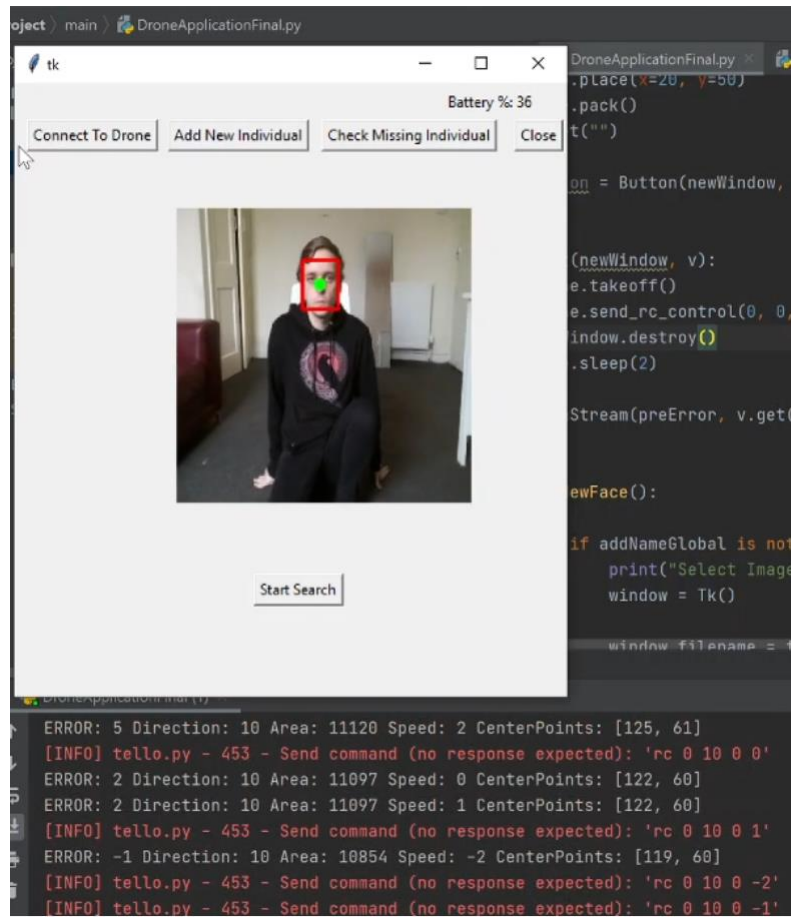


Figure 4

