Person detection, to track a person using the onboard camera of a drone

Wednesday 20th January, 2021 - 11:16

Paul Houssel
University of Luxembourg
Email: paul.houssel.001@student.uni.lu

Jose Luis Sanchez Lopez
University of Luxembourg
Email: joseluis.sanchezlopez@uni.lu

Abstract—**This document is the final report of the Bachelor Semester Project of the student Paul Houssel which was conducted with the help of his tutor Jose Luis Sanchez Lopez. This project is part of the Bachelor in Computer Science program at the University of Luxembourg during the third semester of study. This project seeks to develop an autonomous drone able to recognise and follow a person. With the help of Artificial Intelligence we can inform the detect people and track them, this useful tool has indeed an essential role in the scientific part.**

1. Introduction

Despite the general concerns and ethical problems that involves Artificial Intelligence, the world of technology is thriving with this new but very old concept.
Inspired by the human brain, scientist have begun to develop this new technology already in the 1950's.
Nowadays AI driven systems have taken great importance in our day to day life. From responsive translations in your pocket to self driving cars, the world of AI has conquered our daily life. As a matter of fact, Artificial Intelligence has become stronger then Human minds, as proven by google's deep mind team who defeated the world champion of GO with their own AI called "Alpha GO".
The rise of Artificial Intelligent in recent years has proven to to be fruitful for a lot tech driven sectors. One of the most advancement has been made in the drone industry, hanks to AI, drones have become more autonomous, stable and safe for human use. We can now use drones for difficult and dangerous tasks without sacrificing reliability and human life's. Drone's are being use In the military, By rescue teams, for filming, or even to plant trees in remote areas.
The main objective of the Bachelor Semester Project will be to create Artificial Intelligence powered software that will track a person using a drone (aerial robot). The drone will be able to estimate the person's position and movement. Thanks to this estimate, the robot will be able to perform a person following task. Via WIFI the drone will be connected to the pc. The drone will send image footage to the pc, the pc will use the video to know where the person is, in function of the position the pc will sent out movement orders to the drone. The drone used for this project is a small indoor drone called the TELLO mini drone from DJI Electronics, the drone can be seen in figure 1.

2. Project description

2.1. Domains

2.1.1. Scientific. . In the scientific part of our project I will ask myself "How a drone is able to track a person?". The scientific part evolves around a scientific structured presentation that will answer this precise problematic.

2.1.2. Technical. The technical part of this project is split is two different parts, firstly the development of drone that is able to recognise a person to follow it autonomously. The second technical deliverable consists of creating a Graphical user interface that allows the user to fully monitor and control the drone.

2.2. Targeted Deliverables

2.2.1. Scientific deliverables. The scientific deliverable will be the research and reflexion about person recognition in AI. Once explained, we will structurally and scientifically develop on how a drone is able to follow a person once detected. Furthermore we will consider all the challenges and errors than can occurs, to finally present a solution to it.

2.2.2. Technical deliverables. In the technical part of this Bachelor semester project we will develop a drone that is able to follow a person that was autonomously recognized. The program was written in the python programming language. The computer is connected to the drone via wifi, this system allows us to run the python program on the computer and give back orders to the drone. Secondly, we will develop a Graphical User Interface which will make the system more user friendly. The user can see displayed information about the drone and control it. He can also take pictures and record a video.

3. Pre-requisites

3.1. Scientific pre-requisites

To successfully understand the study of neural networks used in Artificial Intelligence, a small background in mathematics is needed in order to understand the computation applied to matrices.

3.2. Technical pre-requisites

As the technical part mainly involves coding, it is advantageous but not essential to have some background in this area (mostly in the python programming language). Finally the technical deliverable evolves a lot around basic networking concepts. Thus, a brief understanding of the subject is necessary.

4. How can a drone track a person ?

4.1. Requirements

The primary purpose of this deliverable is the study and explanation is to answer the following scientific question : *How can a drone track a person ?*
- [4.3.1] The deliverable shall define Artificial Intelligence
- [4.3.2] The deliverable shall define trained AI models
- [4.3.3] The deliverable shall define the implementation of AI on a drone.
- [4.3.4] The deliverable shall define tracking and how it works.
- [4.3.5] The deliverable shall define the challenges of tracking a person with a drones onboard camera.

4.2. Design

The scientific report should briefly give an introduction to the Artificial Intelligence in order to understand the sanctioning of the used deep learning model. We will then apply this knowledge to explain the use of AI on a drone in order to track a person. While the main part of this scientific deliverable still is focused on how a drone can track a person.

4.3. Production

4.3.1. AI. In 1955 the concept of Artificial Intelligence (AI) was firstly introduced to the world of Computer Science by defining it as an academic discipline. A group of researches including the famous computer scientist Marvin Minsky, study the field. This study will lead to a conference the year after which is considered as the birthplace of AI. The term of AI is defined by the Cambridge dictionary as follows : "the study of how to produce machines that have some qualities that the human mind has, such as the ability to understand language, recognize pictures, solve problems, and learn". A big subset of Artificial Intelligence is the Machine Learning. Machine learning embodies "algorithms whose performance improve as they are exposed to more data over time". In this project we mainly used machine Learning. For the sake of this project, we used a trained deep learning model called `Haar Cascade`. This model is a machine learning object detection algorithm that is able to recognize a certain object if trained on it. For this project we used a trained model that is able to recognize a persons face.

4.3.2. Trained AI models. The training of a deep learning model is an extremely crucial part. The training of an AI generally works as follows :
- First, we establish a dataset containing thousands of images of different person faces. For every single image a human takes a look at it and labels the persons face. This data then tells the computer where the face is. In this way,the machine already knows the real output of the image. With this knowledge the machine will adjust the weights if his output was wrong. The operation that consists of adjusting the weights is called backpropagation.
- Then, we establish a test set, this set will allow the machine to test itself in order to obtain an accuracy level that goes from 0 to 1. This set has no expected output and thus the machine does not know the real output. The machine will therefore pass the image through the neural network to evaluate the images output. The use of a test set allows us to obtain the actual accuracy of the machine.
- Furthermore, we will diversify the training set in order to make the neural network more efficient. This is achievable by taking the images from the data set and apply some little modifications on it. They could be zoomed out or in, turned and so on.
- With this data set, the machine will parse through every image in the training set. For every image the neural network will give a prediction. If the prediction is false then the weights are adjusted such that the prediction is correct. The machine knows the actual output because the images are stored indifferent folders in function of their actual output.The last steps of the training takes an important amount of computational resources and thus time.

4.3.3. Implementation of AI on a drone. As the drone possesses an on board camera, we can use this camera to detect faces around the drone. We can see in figure 2, that the drone is able to recognize a face by using the recognition model. For a visual representation, a virtual rectangle is drawn over the face. The recognised face is treated as a tuple, with two coordinates in a 2D Cartesian space. Let (x,y) be the coordinates of the upper left corner of the rectangle, let w be the width of the rectangle, and let h be the height of the rectangle :

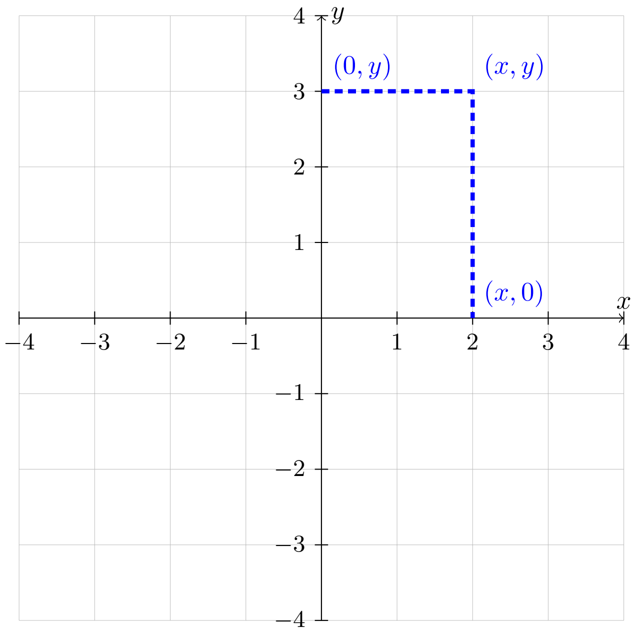
$$Face = (x, y, w, h)$$

Let's consider figure 6, let L be the upper left corner and R the lower right corner of the rectangle, L and R have the following coordinates :

$$L = (x, y)$$

$$R = (x + w, y + h)$$

Using this coordinates we can place the face in the 2D space. Since the space is not three dimensional we cannot know if the face is far away from the drone or close. In order to solve this problem we use the width of the rectangle. The further away the face is from the camera, the smallest the width is. Thus there is a correlation between the width and the distance from the face to the drone. We can use this correlation to know where the face is placed on the z-axis of a three dimensional Cartesian space.
The drone needs to follow the person, thus the person on the camera footage always needs to be in the center of the image. We therefore need a controller that can move the drone in function of the offset between the face and the center of the image.



Let's say the center of the face has the coordinates (x, y), then there is an offset on the x axis and on the y axis. We firstly handle the x-axis offset, by moving to the right if $x > (0 + deadzone)$ or to the left if $x < (0 + deadzone)$. The deadzone is a sort of accpetence interval. If for example the center of the face has a small distance with the center of the image, the the face is located in this deadzone. We then adjust the drone on the y-axis. And finally, we compare the width of the rectangle, w, to compute the z offset.
Furthermore, we don't want the drone to move smoothly with one single movement. We therefore need a proportional controller that gives out a command in function of the distance of the drone to the face.

4.3.4. Tracking. For now we saw how a drone can use AI in order to recognize a person's face with the help of the on-board camera. However the recognition does not always work as wanted. We therefore need to handle these unwanted errors. Errors that were noticed while working on the technical deliverable are for example the detection of a second face in the image, losing the face for a small amount of time (while looking to the right or while nodding the head to the sides). Finally, it can happend that the video gets blurry in some parts due to UDP packet losses. In this cases we want to have a solution that jumps right in to replace AI recognition. By using tracking technology we are able to tackles those problems. Tracking is a method that allows us to know in which parts of the image depicts the same objects in different frames. Let's say we know the position of my Face on the frame x, tracking will tell us the position of the face in the frame x+1 (without using recognition) and so on. The specific tracking model we are going to see further is called Meanshift. This tracking model works as follows :
- We firstly start by initialising the position and the size of the object that we would like to track. This happens in frame x.
- In the next frame, x+1, the tracker is going to search for a similar description in the neighbouring region of the object in frame x.
- We then found the new position of the object in the frame x+1.
In figure 8 we can see a visualisation of this Meanshift tracking algorithm as described previously. The blue dots represent the description of the tracked object, the red circle represents the position of the object in the previous frame. In each consecutive frame the algorithm will search for the blue spots in the neighbouring area of the red circle, the red circle will "shift" to the center of gravity of these points. The new position of the object was found.
One problem with meanshift is that the size of the object is not changing when it goes further away from the camera. This doesnt allow us to track the object it the third axis. This means the drone will not be able to know if a person is far or really close from the drone. For this reason, we are going to look at camshift, another tracking algorithm that works a bit differently. Finally, we can see that tracking is a great way to complete AI recognition in order to make the hole process safer. Furthermore, tracking also allows us to make less computation and this takes less ressource. A mix between those two is perfect for our drone.

4.3.5. Challenges of following a person. Following a person with a drone is very challenging and can cause many problems to solve. The most important part of this, is to guarantee the integrity of the program. A drone always needs to know where the person to follow is located.
In this first part we are going see how the drone correctly follows the right person : As we saw in the last subsection about `Tracking` (4.3.4), tracking is more efficient then recognition. However, tracking can fail, lose the face, it then needs to be updated. For this reasons a complementary mix of Tracking and recognition is the best solution. But what happens if the tracking loses the face ? In this cases the program recognizes the situation and will update the tracker again by recognizing the face. One way to detect if the tracked face is lost, is if the tracked face is of null.

$$trackedface = (0, 0, 0, 0)$$

Another situation is when the tracked face is stuck on an object (a lamp for example) and completely lost the face. In this case the face will have exactly the same coordinates every time, since it is stuck. To prevent this the program compares the coordinates of the face every second, if they are exactly the same it means that the tracker possibly got stuck. The tracker will then be updated with recognition.
If the tracker got stuck :
At frame x :

$$trackedface = (193, 100, 55, 90)$$

At frame x+27 (1 second later):

$$trackedface = (193, 100, 55, 90)$$

If the tracker didnt get stuck :
At frame x :

$$trackedface = (193, 100, 55, 90)$$

At frame x+27 (1 second later):

$$trackedface = (153, 200, 60, 90)$$

Another situation that could prevent problems is that when the tracker gets updated by the recognition, a wrong face gets recognized instead of the face that got tracked before. Or if there is more then one face in the frame then the tracker got updated with another person then before. To prevent this problem, the tracker should only be updated with the face that overlaps with the previous tracked face. If no face is overlapping, then the nearest face of the previous tracked face will be considered.
Let's consider the figure 7. In the figure above we can see three squares, let the black one be the previous tracked face. The other two squares in red and are the new recognised faces. In this situation the tracker is being updated with recognition. The recognition has the choice of two different faces to update the tracker. As only a few milliseconds passes between the black face was tracked and the new faces were recognised, the same face that was previously tacked is logically overlapping with the red square. The tracker will therefore be updated with the overlapping red square. This method allows the drone to always follow the same person and to never jump onto the face of someone else.
In order to check if two square are overlapping we take into consideration the bottom right corner and the upper left corner of each square. Let these points be respectively R and L, each point has two coordinates (x and y). Compare square 1 and 2 :
If one square is on the left or the right of the other square:
If, $L_1.x \geq R_2.x$ or $L_2.x \geq R_1.x$, then the two square are not overlapping
If not, they are overlapping.
If one square is on top or below the other square :
If, $L_1.y \leq R_2.y$ or $L_2.y \leq R_1.y$
If not, they are overlapping.

4.3.6. Autonomous Controller. In the robotics field , a robot needs to have a so-called controller in order to function. This controller allows the robot to be guided step by step to fulfil a defined task. The most primitive of those controllers is a Remote Controller (RC), that uses a human input to send out tasks to the robot. A perfect example for an RC powered robot, would be a drone controlled by a person. The person would give the drone orders with the help of a RC device. As advancement have been made, many industries start to use intelligent robot to fulfil missions where safety is an important asset. These robots use a more developed controller called Autonomous Controller (AC). This controller allows the robot to execute a mission completely autonomous without the input of a human being. However, it is important to note that the robot depends on a human to be supervised and to be told when and if a mission should take place. A great example for an AC powered robot would the technical deliverable of this project paper, a drone that is able to autonomously follow a person using an onboard camera. A drone can be called autonomous, if it has the ability to decide over itself to conclude some tasks that will lead to a precise goal. In the case of this project, the drone has the goal to follow the detected person, the drone is able on itself to do task such as turning, moving up and more, in order to complete this goal. In order to decide which task to take, the drone is using control algorithms that decide an output in function of the cameras input. To dive a bit deeper in this complex subject, we will talk about a specific autonomous controller, the PID controller. PID stands for Proportional, Integral and Derivate controller, these three controllers are all combining to form a PID controller to control a task given a parameter (an input). A PID could be used to control the steps a drone should take to efficiently follow a person. Since a drone needs to be fast, reactive and smooth in his actions, it cannot simply move in a direction. The drone needs exactly to know how much he needs to move in a certain direction to follow the person. For example if the drone follows a person that is at the border of his visibility to the right, then it needs to move a lot more then if the person would be really close to the drone. A PID controller allows the drone to fulfil this task, a more common controller like a step controller couldn't do it. In order to explain this concept, we will firstly start by taking into consideration a Proportional controller, which is a main part of PID controller. This controller also called P-controller, computes an error which is the offset (the difference) between the actual state and the state in which it should be next. So for a drone following a person, the offset would be the distance between the person and the middle of the drones camera frame on an axis in a Cartesian space. This error is then multiplied with a constant called Proportional gain, this constant will convert the distance (pixels on a frame) into centimetres. We then obtain the output, which tells the drone how many centimetres he needs to move to a certain direction. This controller is proportional because it proportionally adapts the output in function of the input error. We want to achive an error of zero such that the drone positions exactly in function of the persons postion.

$$m = (K_p \times e)$$

Where,

$$m = Output$$

$$K_p = ProportionalGain$$

$$e = error$$

Let's imagine the K_p is wrongly estimated, in the case where it is too high it can happen that the drone moved too much in a certain direction. The drone needs to adjust the output once again because the person is still not in the middle of the drones frame. In order to avoid this error there exists a Proportional-Integral Controller (PI- Controller) that is based on the P-Controller. The PI-Controller takes into consideration these small errors are added up to balance the final output in order to get an error close to 0 after a period of time. We obtain the following formula,

$$m = (K_p \times e) + \frac{1}{T}\int edt$$

Where,

$$m = Output$$

$$K_p = ProportionalGain$$

$$e = error$$

$$T = timeconstant$$

$$t = time$$

The error is represented by the difference of the setpoint and the function, to cancel this error we want to add the area between the function and the setpoint, which is

$$1/T \times \int edx.$$

Every T time the error is added. The smallest T is, the fasted the controller will be. Lastly we will consider the PID-Controller, which adds the derivative controller to the PI-Controller. The derivative will finds the ration of variable change. Let's say the drone quickly advances to the persons face, then the error rate will quickly drop, e will get smaller so fast then the drone will go a bit to much to a certain direction. To prevent this, the derivative controller will take into account how fast the error changes. If the error decreases fast, thus the derivative will Be negative. This negative value will be added to the final output, which will correct the error change. Finally, the drone will not go too much in a certain direction. Generally the derivative controller tells the drone when it is moving to fast to the wished goal. We obtain :

$$m = (K_p \times e) + \frac{1}{T}\int edt + T_d \times \frac{de}{dt}$$

With, $T_d = Timeconstant$

Finally we can see that a PID controller allows a drone to be really efficient on following a persons face, this concept is a big step on making a drone completly autonomous on his own tasks. However this doesnt solve all problems that could occur when a drone is following a person. Imagine the case in which a drone sees two people, but only follows one, how can the drone continue to follows the same person if these two people cross each other on the camera (Person A and B change position, A goes in front of person B while Person B goes behind person A). In order to make this possible a State Estimator is needed. We are going to focus on the state estimator using the Kalman Filter algorithm, which one of the most common used. A state estimator is generally used to predict the position of an object in a certain space using defined parameters, using a state estimator, the drone is able to predict the position of the face in frame $x + 1$, given the position in frame x, the speed of the face, and it"s positioning.

4.3.7. Conclusion. Finally we can see that this complex topic involves a lot of mathematical concept that allow us to solve challenges and errors that can occur when the drone tries to follow a person, since the situations can be vary different. These concepts make the drone more viable and effective for real-life using.

4.4. Assessment

We successfully presented scientific methods that result in a general structure that allows a drone to detect, and follow a person viably.

5. Drone tracking and following a person

5.1. Requirements

Here are all the listed requirements of this technical deliverable :
- [A] The deliverable shall use open CV with python in order to recognize people on a video.
- [B] Once recognised, the deliverable shall use tracking method in order to get a stable recognition of the person.
- [C] The deliverable shall have an algorithm to follow the tracked person.
- [D] The deliverable shall use recognition in case tracking fails.
- [E] The deliverable shall differentiate the person to track from other persons in the video.
- [F] The deliverable shall allow the drone to land automatically when it battery level becomes critical
- [G] The deliverable shall use the Tello SDK. The Tello SDK connects to the aircraft through a Wi-Fi UDP port.

5.2. Design

- [A] The deliverable uses a graphical python library called `opencv` used for analysing images. In order to recognize faces, a trained Deep Learning model is used, it is called

HAAR CASCADE. The model outputs coordinates on the frame that locate the recognized face. Open cv then visualize the face with fictual rectangles over the frame, as seen in figure 2
- [B] This deliverable uses two different methods to recognize a person. It first recognizes a face and the tracks it. The tracker is updated with a new face recognition every 2 seconds or when the face seems to be lost.
- [C] Using the coordinates of the face center, an offset between the center of the frame and the face is computed. To finally compute an offset that will let the program know the directions to take in order to get the face in the middle of his frame.
- [D] As explained in point [B], when the tracking loses the faces, or the faces is stuck even though the face is moving, then the recognition system jumps in to update the tracker.
- [E] The first time the tracker considers the biggest face that was recognized. When the tracker is updated with the recognition, then the face that overlaps with the previous tracked face will be tracked.
- [F] A thread check constantly the battery level, when it is below 10 then it sends a landing command to the drone.
- [G] DJI Electronics, the producer of the drone, is providing a Software Development Kit (SDK) which is a tool to develop applications for a special hardware, the drone in this case. This SDK allows us to understand the functioning of the drone communications system, in order to be able to send and retrieve information from and to the tello drone through a UDP port.

5.3. Production

5.3.1. Computer Vision. In this first part of the technical deliverable we are going to see how the drone recognizes faces using its onboard camera. This concept is generally called Computer Vision, this term is very broad and regroups all the technologies that allow a computer to analyse and understand images, and videos. Mathematically, an image is to be considered as a matrix containing the colour values for each pixel in the image. An image that has a resolution of 1200 by 800 pixels, then the corresponding matrix would have 1200 columns and 800 rows. Computer Vision will use this matrix as an input and give out an output. The artificial model used in this deliverable, Haarcascade, is one of the many subsets of Computer Vision, Deep Learning.
Deep Learning, Haarcascade Model
The drone is communicating with the computer via UDP (Network Protocol) packets. The communications are the responses to requests, signals to tell the PC the drone is still functioning, and the video stream (figure 9). All these packets categories are sent to different ports on the PC. Once the PC obtains the constant stream of UPD packets of the video stream from the drone camera, the PC decodes them to finally show the frames to the user, the user can watch the live stream of the drone's camera. Furthermore, the python program analyses each frame, as explained before, the Haarcascade model gets the frame as an input and outputs the coordinates of the faces it found. With this coordinates, open cv will draw rectangles over it to visualize them. One of the problems with recognition is reliability to always recognize the same person. From time to time the DL model jumps from the face to a small object or recognizes objects instead of the actual face in the frame. To tackle this problem we use another Computer Vision technology called, Tracking, described in the scientific part 4.3.4.

5.4. Processing the Video Stream

Once the computer is receiving the video stream of the drone, the program uses every frame as an input and outputs the coordinates of every face detected (for every frame, 27 times a second). Making computations on a RGB image takes a lot of computing resources, we therefore convert the frame into gray-scale (black and white) with the help of the python library, openCV :

```
gray = cv2.cvtColor(myFrame,cv2.COLOR_BGR2GRAY)
```

We then detect the faces using the face_cascade model loaded from an XML file.

```
faces = face_cascade.detectMultiScale(
gray, 1.1, 4)
```

The second parameter corresponds to the scale factor, specifying how much the image size is reduced at each image scale. The third parameter is the minimum of neighbors specifying how many neighbors each candidate rectangle should have to retain it.
This results a list of tuples, every tuple represents a face present in the frame. With this tuple we need to decide which face the drone needs to take into account. The first time the drone recognize a face, it will take into account the biggest face it sees. If no face is visible then the process will continue until one face is visible. From there on the face will be tracked, the tracker will be updated every two seconds. If another person comes into the view field of the drone then drone will stick to the previous tracked face. This happens thanks to a special algorithm. When the tracker is being updated with the recognition system, the detection will detect more then one face. The algorithm will compute the distance between the previous tracked face and all the detected faces. The face that was tracked before is logically overlapping with one of the new face, since theres only one frame of difference (1/27 of a second). This concept can be visualised in the figure 7. If no face is overlapping with the previous tracked face then the nearest face will be continued to track. If the face is too far, then it will not be taken into consideration. This method allows the drone to always follow the same person. Furthermore a dedicated thread is used to check if the tracked face is stuck, if the coordinates of the face are exactly the same in a small interval of time (1 second) then the tracker will be updated earlier then planned. The drone now always has access to the face he needs to follow via a global variable that stores the coordinates of the face as a python tuple.

5.4.1. Following a person using the onboard camera of the drone. In order to communicate with the drone we use a python library called `djitellopy`, this library manages network protocols and makes it easier to send and receive udp packets. The most important aspect of this library is the efficiency and low computing used. This library uses object oriented programming, we firstly create an instance of a drone object to be usuable afterwards.

```
    me = Tello()
    me.connect()
    me.for_back_velocity = 0
    me.up_down_velocity = 0
    me.yaw_velocity = 0
```

The drone has 4 different attributes that will be used the most by the program, these are the velocity of every direction. The for_back_velocity and up_down_velocity describe the number of centimeters the drone goes up or down, or forwards and backwards. The yaw_velocity defines the angle of which the drone needs to go right or left. The drone now needs to know how much he needs to move in every frame. To do that subs tract the coordinates from the center of the image to the center of the followed face. We then need to convert this difference in pixels to centimeters in order to adjust it to the drone. We then obtain the number of centimeters the drone needs to move up down, the value is negative if moving up and positive if moving down. Separately, we convert the difference in pixels to the angle the drone needs to move right or left, the value is negative if moving to the right and positive if moving to the left. Finally the drone needs to know if the face is near or far away from the drone. This method will be a bit different since the drones video is in two dimensions, it is not possible to compute the coordinate offset. We therefore compare the width of the person's face to an ideal width (defined manually). The difference will tell be positive if the drone is too far away and negative if to near. Using this difference we create a proportional distance the drone needs to move forwards or backwards. For every frame we set the attributes of the drone such that the commands are executed. Example if the person is on the top left corner of the image :

```
    me.for_back_velocity = 0
    me.up_down_velocity = 0
    me.yaw_velocity = 0
```

A separate working thread is then sending these commands to the drone, the drone reacts. This entire system makes up the proportional controller controlling the drone in order to follow the person.

5.5. Assessment

The final technical deliverable got tested in different environment resulting in different outcomes. Firstly the drone got tested in a bright room with only one person. In the second test the drone flew in the university's drone lab, which contains a lot of object on the walls and was not very bright.
In the first test, the drone successfully recognized the person and followed the person fluently without any problems. The drone successfully stayed focused on the same face during the process without staying stuck or jumping on another face.
Nevertheless, the second test was less successful. This time the drone was able to track and follow the person, however the liability was decreasing over the long term as the drone got stuck on objects.
The main solution to make the drone more efficient is too only use tracking, to track the face determined by the user. The problem with this method is it's inconvenience which we wanted to eliminate by adding an autonomous recognition system.

6. Developing a GUI

6.1. Requirements

Here are all the listed requirements of this technical deliverable :
- [A] The deliverable shall be a user friendly GUI that will allow the user to monitor and control the drone from the computer.
- [B] The deliverable shall display if the drone is connected to computer or not, if yes then the user can see information about the drone such as the battery state, the height and the flight time.
- [C] The deliverable shall allow the user to control the drone with basic commands.
- [D] The deliverable should record a video when the drone is user wants to, it is saved with a dynamic name taking into consideration the time and date.
- [E] The deliverable should take a picture when the drone is user wants to, it is saved with a dynamic name taking into consideration the time and date.
- [F] The deliverable shall allow the user to turn on or off the following mode.

6.2. Design

This Graphical user interface (GUI) allows the user to interact and to monitor with the drone and the following program without having the need to understand code, it therefore needs to be user-friendly and intuitive to navigate.

- [A] The GUI is fit to the resolution of the user's screen. On the upper-side important information about the drone is displayed, such as the battery level or the flight time.
- [B] In the upper part of the GUI, where the general information of the drone is displayed, the user can see if he correctly connected the drone to his computer, if not he is alerted.
- [C] On the lower part of the GUI application, several buttons allow the user to command the drone. The buttons

responds to the user interactions, turns red if the user deactivates something for example. The buttons are arranged such that the command system feels intuitive.
- [D] One of these buttons allows the user to record a video of the drone camera. Clicking on it, the buttons shows that the video is recording, clicking one more time, the recording will finish. The user can later on watch the video in the application.
- [E] Same as for recording a video, the user can take a picture with a similar button. He can later on see all the taken pictures in the application.
- [F] To follow a detected person, the drone needs to have the follow mode activated, by clicking a red button the follow mode can easily be turned on and off again.

6.3. Production

6.3.1. Developing environment. For this second technical deliverable we used the python programming language in order to easily use the tracking and recognition program together with the Graphical User Interface. Indeed, both technical deliverable are in one single python script.
In order to create a Graphical User Interface in python we used a common python library called `tkinter`. This module allows us to create views with several objects such as text, buttons or even progress bars.
The GUI breaks the gap between the code and the user, it is a user friendly way to execute a code and view outputs. The user has no need to understand programming languages in order to execute the programm.
The GUI we created will allow the user to monitor his drone, with all the important information, to control it, to take pictures and videos, and finally to make the drone follow the person.

6.3.2. Structure. As we can see in figure 4, the GUI is structured in 2 main sections. The main section on the bottom allows the user to command the drone, takeoff, land, and turn for example. These buttons each execute a python method when clicked on. These python methods each send commands to the drone via UDP packets. The method will await for a confirmation that the drone received the packet in order to know if the drone will execute the command or not. After a certain time limit, if no confirmation is received the method will resend the UDP packet in order to be sure that the drone will execute the command.
Generally, `Tkinter` allows us to create graphical user interfaces containing widgets. A widget is a GUI component, a visual element such as text, buttons, pictures, which display information and makes the user able to interact with the GUI.
For example, below we create a simple GUI with a text widget.

```
from Tkinter import *
# Create the root interface
root = Tk()
# Create a label including text
widget = Label(root, text="Hello, world!")
# Put the label into the window
widget.pack()
# Start the main loop
# The GUI is updated in every iterations
# in order to display information dynamically
root.mainloop()
```

In order to get dynamic information about the drone, the python program sends a request to the drone every second. The drone responds with the asked information, if the drone does not respond at all the program knows the drone is not connected and notify the user by changing the color of the GUI.
Furthermore, the user is able to record the video from the drone onboard camera. Once the user records the video, python takes every single frame coming from the drone and writes it in a file. We firstly define the writer, by defining the file name and path, the decoding method, and the dimensions of the video. The write() method will then write every frame in the file.

```
writer= cv2.VideoWriter('video.mp4',
cv2.VideoWriter_fourcc(*'DIVX'),
20,
(width,height))
# iterate through every frame
# of the video
while True :
    # frame is the variable associated
    # to the current video frame.
    writer.write(frame)
```

The user can then click a second time on the record video to finish the recording. Using the release() method,

```
 # end the iteration (while True loop)
writer.release()
```

If the user wants to take a picture, we use the same technic with one frame to save it in a png file.

6.4. Assessment

The GUI is a perfect way to bind the user and the program functionality, a user without any experience can understand with ease what he can do, even though he has no coding experience. The most important is to let the user know what happens, in case the program does not work he could understand where the problem lies.

Acknowledgment

The author would like to thank the BiCS management, the education team and his tutor Jose Luis Sanchez Lopes for the fantastic help as well as their availability.

7. Conclusion

This project allowed us to understand the importance of optimisation in order to obtain a reliable final product.

With the development of this program we combined AI and autonomous robotics that allowed us to gain helpful and great first experience in the field of robotics. Overall we are satisfied with the final deliverable and consider the GUI a great way to show the power of autonomous drones and AI in a real life application.

References

[1] [*Cambridge Dictionnary(2020)*] Definition of Artificial Intelligence. https://dictionary.cambridge.org/fr/dictionnaire/anglais/artificial-intelligence

[2] [*Mathworks (2017)*] Train a Cascade Object Detector www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

[3] [*Will Berger Blog (2017)*] DEEP LEARNING HAAR CASCADE EXPLAINED http://www.willberger.org/cascade-haar-explained

[4] [*Open CV Documentation (2020)*] Object tracking https://www.learnopencv.com/object-tracking-using-opencv-cpp-python

[5] [*PyImage Search (2018)*] Simple object tracking with OpenCV https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv

[6] [*Open CV Documentation (2020)*] Meanshift and Camshift algorithm https://docs.opencv.org/master/d7/d00/tutorial_meanshift.html

[7] [*Matlab Documentation (2018)*] Understanding PID Controller https://www.youtube.com/watch?v=wkfEZmsQqiA

[8] [*Matlab Documentation (2017)*] Understanding Kalman Filters https://www.youtube.com/watch?v=ul3u2yLPwU0

[9] [*Karl-Erik Åarzén (1999)*] A simple event-based PID controller https://www.sciencedirect.com/science/article/pii/S1474667017574820

[10] [*F.Ponci (2016)*] State Estimator - an Overview https://www.sciencedirect.com/topics/engineering/state-estimator

[11] [*Instrumentation Blog (2020)*] What is a PID controller ? https://instrumentationblog.com/what-is-a-pid-controller

8. Appendix

The apprendix shows all images used in this scientific and technical report.

Figure 1. DJI TELLO MINI DRONE

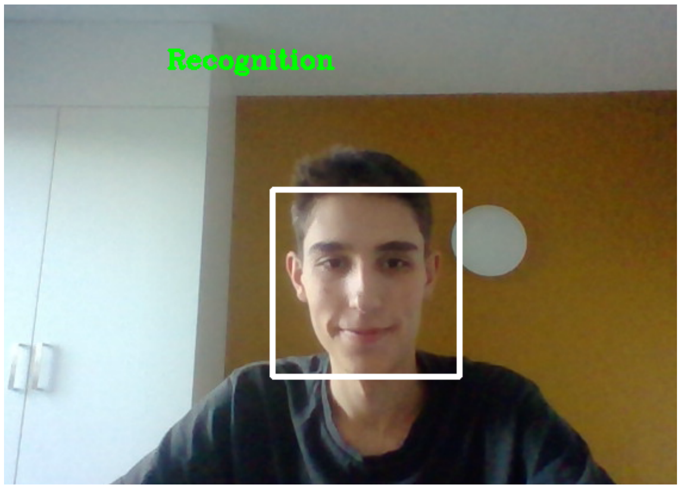


Figure 2. Drone Recognizing a face

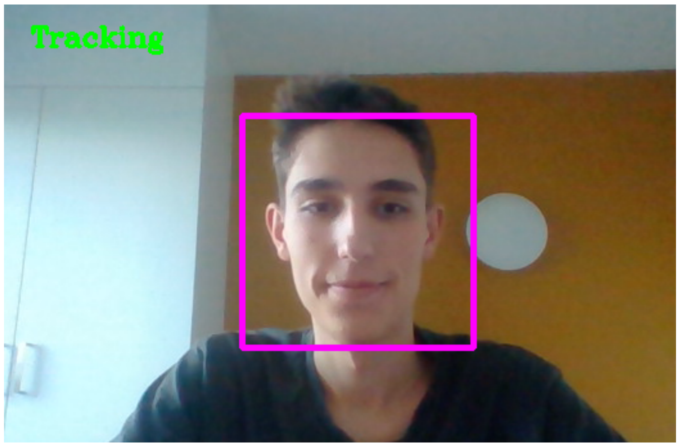
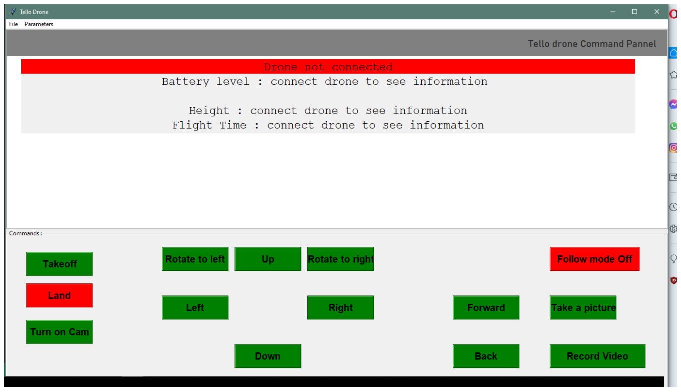


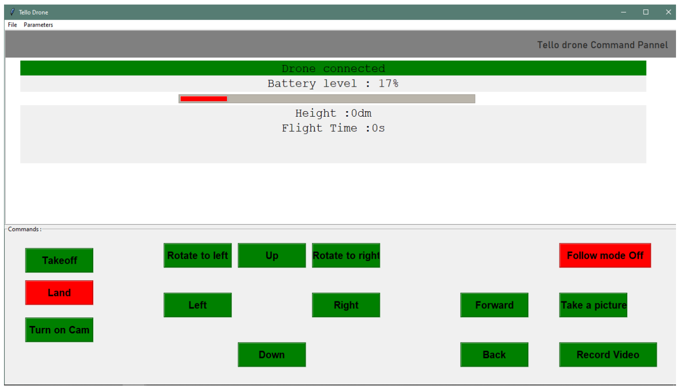
Figure 3. Drone Tracking a face



```
Drone not connected
Battery level : connect drone to see information

Height : connect drone to see information
Flight Time : connect drone to see information
```

Figure 4. Snippet of the GUI command tool when the drone is not connected



```
Drone connected
Battery level : 17%

Height :0dm
Flight Time :0s
```

Figure 5. Snippet of the GUI command tool when the drone is connected

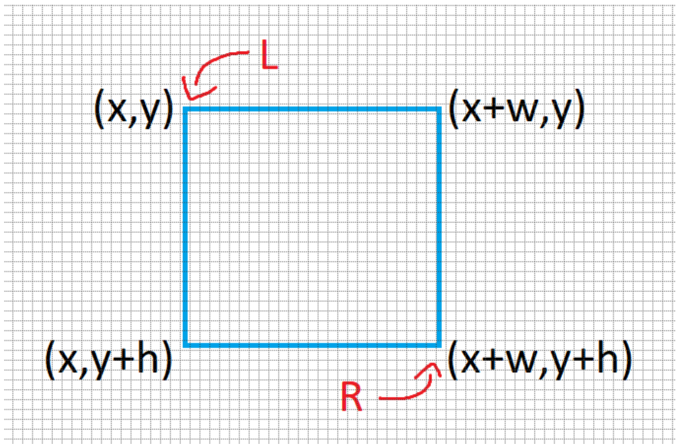


Figure 6. Representation of a face in openCV

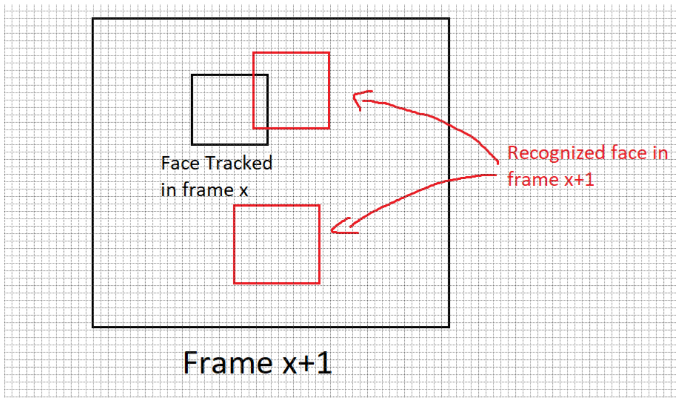


Figure 7. Update of the tracker using detection.

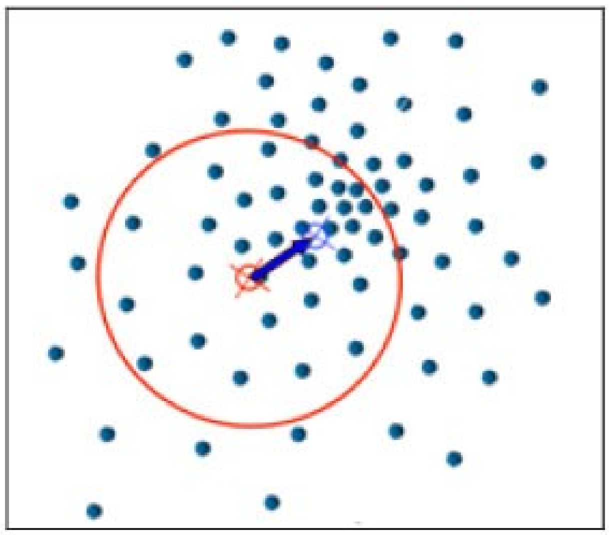


Figure 8. Meanshift tracker

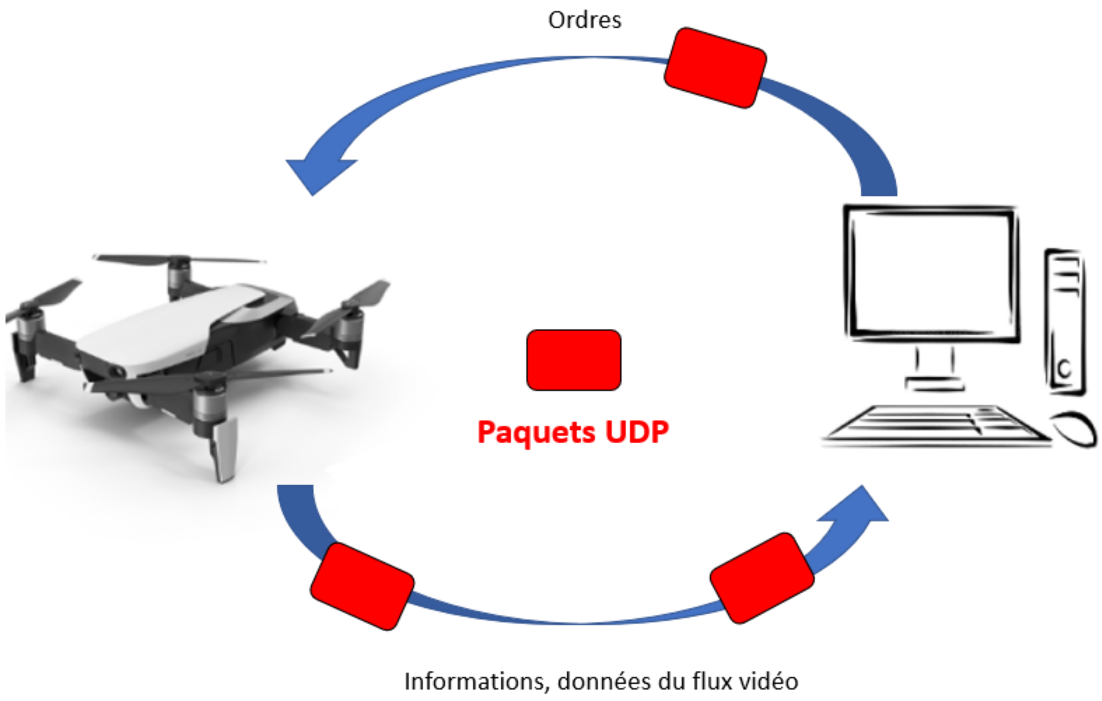


Figure 9. Communication system between the drone and the computer