HUMAN DRONE INTERACTION (HDI) USING GESTURES, FACE AN HOLISTIC APPROACHES

Alessandro Lambertini - 1938390
Denise Landini - 1938388
Gianluca Lofrumento - 1956579

July 15, 2022

"La Sapienza" University
Course: Elective in AI, HRI and RA parts
Professors: Luca Iocchi, Fabio Patrizi

For this project all the students contributed equally.

Abstract

In the last years, drones, also called UAVs (unmanned aircraft vehicles) have played an important role in several applications like industry, delivery, photography, robotics, surveillance, guidance, navigation in unknown terrain, traffic monitoring and control systems.
The goal of our work is to develop a new system that can control an UAV by using body gestures, hand gestures and multimodal approach.
To test the capabilities of our methods, we have used the DJI Ryze Tello Drone. However, our developed system is drone independent, so adaptable to several different drones by a minimal effort. Of course, the drone performance, stability and motion smoothness affect the result.
Experimental results of our system showed that:

- **it can correctly interpret gestures commands;**
- **it can recognize and follow users' feature – like face or body – by single body parts or holistic tracking approach.**

Keywords

HRI (Human-drone interaction), HDI, UAVs, RA (Reasoning agent), Body gesture, Hand gesture, Holistic

Introduction

For a long time, the researchers had studied a lot the Human Robot Interaction (HRI) to have collaboration and cooperation with the robot.
There are different areas in which HRI has been developed like human supervisory control of robots in performing routing tasks, automated vehicles in which humans have the role of passenger, social interactions between them and remote control of a space.

In the last years, we started to talk about drones, quadrotor vehicles able to fly autonomously. Initially, drones were controlled with a joystick, so they were only a sort of remote-controlled object. Very soon it was realized that this kind of technology can be very useful in day life to replace some activities done by humans or to collaborate with human in order to accomplish some objectives.

For example, one application is implemented by Avila et al. [1] in which

they use small drones for navigating visually impaired persons. In this application, user has a bracelet and by doing some **verbal commands** that the drone understands, he is able to arrive in some part of the environment because the drone computes the shortest path and starts to fly followed by the user. The distance between drone and bracelet is continuously computed by Bluetooth.

Another common application is the **package delivery** using drones [2] in which they consider a skyway node, and they implement the A* algorithm to compute the optimal path the drone has to follow to go from the source to the destination, considering all the needed constraints.

So, the research of HDI can be divided in four classes:

1) **Control modalities**: in which human can communicate with drone by using gestures (by body or hand), speech, brain-computer interfaces, touch or applying a sort of multimodal control;
2) **Proxemics**: in which they study different behaviour related to the distance between human and drone;
3) **Human-drone communication** like [3];
4) **Other use cases**: in which we have an interaction in art, social companions etc. [4]

Our study is in the direction of the control modalities, but we consider also other aspects related to the proxemics topic.

We have also to distinguish between different roles that humans have in HDI. In fact, human can be:

- **Active**: human control the drone by using a control interface;
- **Recipient**: human don't control the drone, but it interacts with it;
- **Social companions**: human might or might not be able to control the drone movement, but it is able to holds a social interaction with it;
- **Supervisors**: human has the role of supervisor in case of emergency because the drone is autonomous.

Reasoning agent is another important aspect that given a set of facts it can derive a new one in a predefined way. The important operation that we can do in reasoning are:

- Planning: applying a reasoning in order to achieve a goal;
- Problem solving: study a way to achieve the objective in case there are every type of obstacles;
- Study how to pass from a state to another state.

To do this there can be a lot of decision that can be done.

In Section I, we discuss the current works in the state-of-the by presenting a few publications done by researchers. In Section II, we present our innovative solution considering gesture and face interaction with drones. In Section III, we explain in detail how we have implemented our project. In Section IV, we present the experiment we have done and the results we obtained by using our technology.
Concluding, in Section V, we summarize the whole project.

SECTION I

HRI: Related works

In Human-Drone Interaction one important thing to take into account is the proxemics topic.
The problem is that sometimes, when the drone flies near the human, if it has some

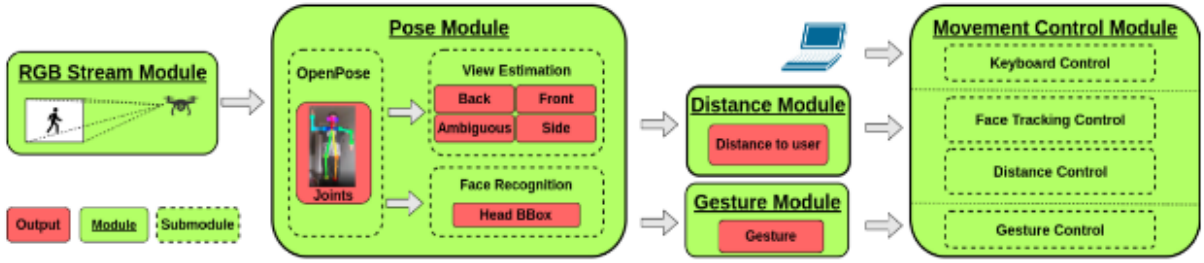

Figure 1: Framework implemented in [9].

types of malfunctions and it could be very dangerous for the human. For these reasons, it is important to study the distance between human-drone in order to have a safe performance of the activity that we have to do. Proxemics is a field that studies the amount of space that people feel it necessary to set between themselves and drones.
It can be studied by considering different factors depending on the goal we have to achieve.

For example, in [5] they consider two factors: size of drone (big and small) and the altitude; doing the experiments, they conclude that the social space is the distance that most of the participants take from the drone.

Instead in [6], they study proxemics in different cultures, and it derives that Chinese participant approach the drone closer than American ones.
Proxemics take an important role also in our project.

One application, that is also of our interest, is done by doing somebody or hand gestures to the drone and, based on which it receives, it is able to react in different ways.

For example, in [7] researchers design and develop a Natural User Interface that allows the user to command the drone by using **body gestures** instead of physical controller.
In this kind of experiment, they use Kinect, a device to extract spatial information and recognize postures that is located in front of the user, and it receives in input the humans' body gestures. The role is to recreate the 3D human skeleton and sent it to the computer. The computer processes these data and it sends the correct command to the drone.
The visual feedback is used only if the drone is not close enough to the user.
To map the body gesture received to a drone command, they use three directions (forward/backward, right/left, up/down) and one rotation around the z-axis. To recognize the movement of the humans, the data are processed to compute the angle values of the body.

There can be other applications related to **hand gestures** like in [8] that can be used to control drones.
There also different methods that can be used to recognize hand gesture like using biosensors, Electroencephalography (EEG) or Electromyography (EMG).
The most common way is to use video frames. The technique they use is:

1) UAV processing: computer receives the image frames from UAV camera to be processed;
2) Computer processing: computer processes the frames, and the algorithm

detects 21 hand key point coordinates and locates them in 3D key nodes;
3) The Artificial Neural Network (ANN) classifies the hand gesture using the positional relationship;
4) Command sends to UAV: the command is generated based on the results of classification of hand gesture.

The work from which we take the inspiration is [9] in which the framework used is divided in modules and it cannot use Kinect device.
They implement 5 modules as we can see in **Figure 1**:

1) **RGB Stream Module**: captures the images by the camera's drone and it sends them to the pose module;
2) **Pose Module**: it estimates the 2D joint locations of the people in the frame and it sends those features to estimate the view and face recognition;
   a. **View estimation**: classifies the orientation of the person into classes using the 2D joint locations as input;
   b. **Face recognition**: only one person influences the movement of the drones, and it is followed.
3) **Distance Module**: estimates the distance between user and drone;
4) **Gesture Module**: recognizes user's arm gestures;
5) **Movement Control Module**: the movement of the drone can be controlled by keyboard (like joystick), face tracking, distance control and gesture control.

RA: Related works

Nowadays, an important field in which the drones are used with some reasoning is in the surveillance systems. Surveillance tasks are characterized by two kinds of agents: observers and observed. Considering this UAV application, the observer is the drone while the observed is the target that the drone must catch or follow. In [10] they build a system that is able to compute the number of people got stuck in their homes in case of disaster situations. This allows the rescuers to help this people in a very efficient and fast way.
This can be applied also to military surveillance. As other systems, this system use Wi-Fi in order to do the communication with humans. To test this system, they use a kind of intelligent system that follows the following steps:

1) The drone receives some manual input and it answers according to that input;
2) Initialization of the camera;
3) Detection of the faces: this is done by using a centroid tracking algorithm;
4) Location and Capture of the images then stored in the detected face;
5) Count the number of people and the location in which they are;
6) The data produced are sent to ground station.

Another approach is the one in [11] which combines probabilistic reasoning with Monte Carlo simulation with long-term strategic capabilities provided by automated task planning. They focused on the SaT problem (Search and Tracking), in which the agent must be highly reactive, without spending too much time reasoning about alternative courses of action, and have limited resources, so they need to be strategic in deciding which path to follow.

Figure 2: SaT mission diagram flow.

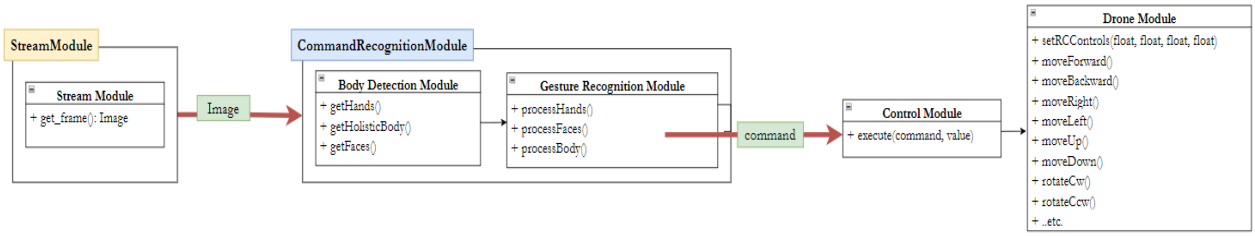

Figure 3: Pipeline of our project.

SaT mission aims to follow the target in two phases:

1) Tracking: the observer flies over the target watching its progress and tracking it;
2) Searching: the observer has lost the target and performs a series of manoeuvres trying to rediscover it. Either is found again and tracked or it is not found again after a while, so an end mission plan is executed.

The hybrid approach they propose consists of applying a Monte Carlo simulation to estimate the probable trajectories of the target and build a Probabilistic Distribution (PD) map for the target location, while using planning to reason about this map and create plans for the pursuer to maximize the probability of rediscovering the target if it becomes unavailable during the tracking.

In [12] they present an application of case-based reasoning in which they control the drone agent by using a smart watch interface. In particular, the gesture takes place by wearing the smart watch that gives to the ground station the sensor data. At the same time, the ground station can recognize the gestures and it returns a feedback. Once it has recognized the gesture, the motion control is given to the UAV. In fact, in this paper, they demonstrate the gesture-action mapping that works as follows:

1) ACTION FOLLOW: drone autonomously follows the user because this is the predefined action when there is not a specific gesture;
2) ACTION FREEZE: when there is a sharp upward hand movement of the user, the state of the drone changes because it stops to follow the user. At the same time, the drone is aware that he must stay still.

SECTION II

Solution

Our framework is displayed in Figure 3. The pipeline consists of several modules:

- **Stream Module**: allows to get the frames from the drone camera or any other camera;
- **Command Recognition Module**: it processes the frame in order to get a command to be performed by the drone; it consists of two inner modules:
  - **Body Detection Module:** it processes the frame to get the gesture of the hand, the face or the holistic body;
  - **Gesture Recognition Module:** it processes the feature extracted by the previous module (gestures, face data, etc.) into a command that is given to the control;
- **Control Module**: it processes the command and translates this in a drone motion;
- **Drone Module**: performs the action that the previous block applied.

Stream Module

The input can be taken from the video source present from the drone, or from any camera put on that. This is done by the **Stream Module**, which in our project provides the images captured by the camera of the drone. The frames can be retrieved by using any connection with the drone – Bluetooth or WiFi for example – or can be taken directly by the computer's camera. This module was developed in order to abstract any type of video source, so that it can be used also any type of external camera mounted on the drone, instead of only the built-in one.

Command Recognition Module

Since the input is an image, we can track multiple features. We decide which one by using the keyboard command, that allows to choose among:

- Hand;
- Face;
- Holistic tracking.

The behaviour of tracking is different from each other.

For both hand and face tracking the first thing that we do is to recognize the area in which we have the face or hand, and we do that by using the MediaPipe Box Tracking that allows us to draw a square around the hand or the face. For the face, it builds the bounding box considering the marker locations of the face that are ears, eyes, nose and neck. Since we have to take all the head, this bounding box is extended considering a fixed margin across the width and height. For the face, we also use the MediaPipe face detection that poses six landmarks and it supports multi-face detection in the same image.

Control Module

The control module takes in input the command given by the Command Recognition Module and it calls the function of the drone related to that command. In this way, the drone is able to perform the action. As said before, the Command Recognition Module can give to the Control Module the command gesture or the speech command depending on the experiment that we are performing.
The command that we give to the Control Module can be take off, land, stream on, stream off, move forward, move backward, move up, move down, set velocities, rotation clockwise, rotation counterclockwise, follow me and stop execution. For some commands, together with the command type, also a value is provided – for example, when moving the drone forward, the distance to run must be specified, instead when taking off or landing, a value is useless, hence not necessary.

Drone Module

The drone module stores all the information about the drone, and it contains all the functions that allow the drone to perform actions depending on the command received. It provides an abstraction for a generic drone, so that the whole application is drone independent. Hence, this allows future implementation for different drones, just by implementing the provided interface. In this project however, only the DJI Tello implementation was made.

HRI: Solution

In order to interact with the drone, the previous pipeline is used to recognize hand gestures that later would be converted into commands to give to the drone. This pipeline indeed, was thought to be flexible and reusable, so it is independent from the type of task to perform. Going into details, different concrete implementations of each module were built – in particular of the

Command Recognition Module – so that body feature data were processed in the most natural and easy way for a human, so that he could interact with the quadcopter.

Hand command recognition

For tracking hands, we also use MediaPipe high-fidelity hand and finger tracking that uses machine learning (ML) to infer 21 3D hand's landmarks from a single frame. The first thing that the MediaPipe does is to detect the palm of the hand and secondly it localizes the landmarks on the hand-knuckle detected by using regression as we can see in Figure 4.

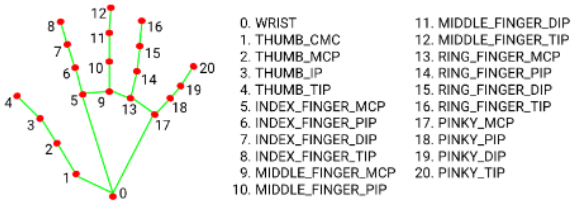

Figure 4: 21 hand landmarks.

As concern hand recognition we have use the two hands in a different way:

- **Right hand**: is used to do the *main* gestures to the drone;
- **Left hand**: is used to set the value of the velocity or other parameters based on the right-hand gesture that is being doing.

The procedure that we use to compute the distance from the left hand is based on the distance between index and thumb and then we quantize this result between 0 and 10.

Figure 5: distance between index and thumb.

The gestures that we have choosen to use for command the drone are 8:

- Up;
- Down;
- Back;
- Go forward;
- Land;
- Stop;
- Left;
- Right.

as shown in Figure 6.

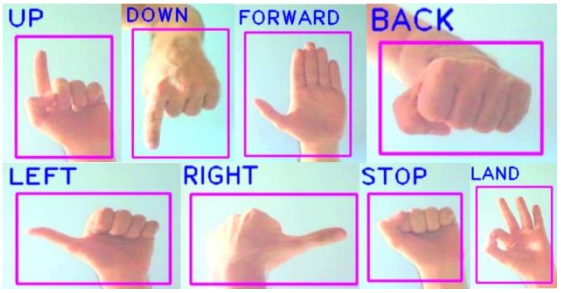

Figure 6: Hand gestures command seen by the drone's camera.

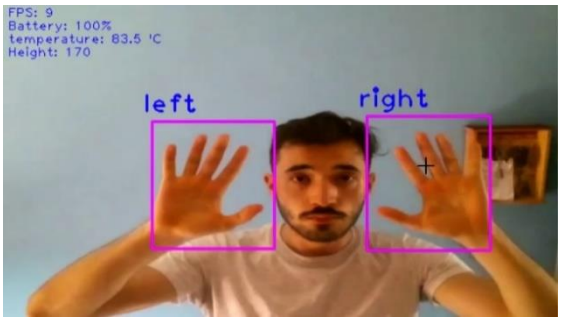

Figure 7: Example of a gesture that allows the drone to go forward.

Figure 8: Stop signal for the drone.

In Figure 7 and Figure 8, we have the drone camera view. As shown, we have the two pink boxes that detect the corresponding left and right hands with a blue labels.

Furthermore, in the upper left corner of the figures, we have some parameters:

- **FPS**: frame rate;
- **Battery**: percentage of the drone battery;
- **Temperature**: temperature of the drone in Celsius degree;
- **Height**: height the drone is with respect to the take-off terrain.

RA: Solution

In order to perform autonomously and intelligently, a set of possible states, in which the drone can be, have been thought. In particular, these routines were built for the face tracking and for the more complex surveillance system developed in this project. For the latter in particular, a concrete implementation of a pipeline has been built, so as a separated application.

Face command recognition

In Figure 9, is illustrated the view of the drone with an example. It always takes care to have the center of the face correspondent to the center of the screen. Moreover, it performs a kind of depth estimation to be at a safe distance to the user. So, we have three axes with the following functions:

1) **X-axis**: drone rotates clockwise or counterclockwise to place the face at the middle of the screen;
2) **Y-axis**: drone rises, or lowers, achieving the same goal of X, but on the height of the frame;
3) **Z-axis**: drone moves upward or backward by measuring the width of the face and tracking to bring it to a target value.

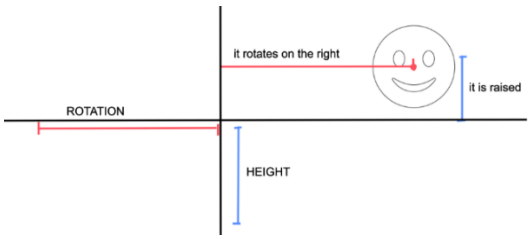

Figure 9: This is an example in which the face is on the right-side of the center, so the drone has to rotate clockwise; the face is above the line and so the drone has to raises to maintain the interaction.

We have created a *FaceCommandRecognition* procedure for this experiment.
In particular, we can schematize by using 4 states in which the drone can be:

1) **NONE**: this is the state in which the drone has not detected a face yet;
2) **DETECTED**: when the drone is in the none state and it finds a human's face, it passes in the detected state;
3) **LOCKED**: when the drone is in the detected state and detects it for more than two seconds, it passes in the locked state; we also study the case in which the drone is in the lost state and it passes to the locked one. This happens when it is in the lost state and it does the detection face for one second continuously;
4) **LOST**: when the drone is in the locked state, it passes in this state when it lost the face for half a second continuously. We have also the inverse procedure as discussed in the previous point. From the lost state, we can also pass to the none state if the drone cannot find again the face for more than two seconds.

The procedure is shown in Figure 10 by using the state diagram.

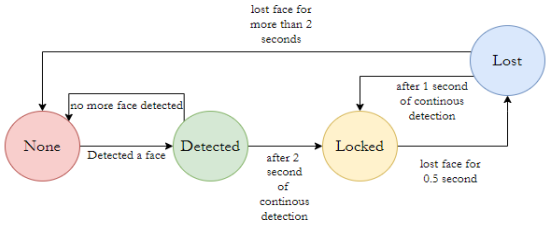

Figure 10: State diagram for face command recognition.

Holistic command recognition

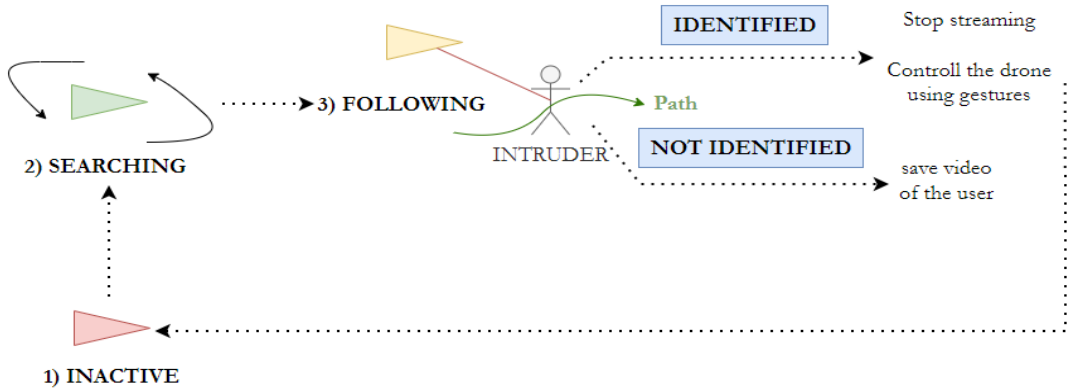

Figure 11: Drone from inactive phase passes to searching phase, it recognizes the intruder, at the same time the video starts; it follows the intruder unless the user does the stop command. So, the video is saved and then it stats to follow gestures command.

The holistic tracking is intended as the hand tracking, face landmarks plus the pose of the user's body. So, it estimates the human pose with BlazePose as we can see in Figure 12.

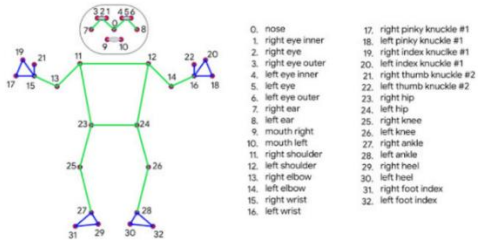

Figure 12: BlazePose landmarks.

For this part of the project, we have created an *HolisticCommandRecognition* procedure. In particular, we can schematize by using 4 states in which the drone can be:

1) **INACTIVE**: after three seconds that it is inactive, it becomes active meaning that it passes from this state to the searching state;
2) **SEARCHING**: it is in this state after passing the `takeOff` command and drone starts to detect.
   The detection is done by using the `searchIntruder` function. If it sees a face and if this case is verified, it called the `followFace` function and it also measures the time. In other words, if the drone finds a person for more than three seconds, then it starts to follow it (we have found the intruder);
3) **FOLLOWING**: once we have found a person, the drone follows the face and as consequence, it follows the intruder. This is done by using the function `followIntruder`. So, when there is the intruder, we start to record a video.
   The intruder is always followed until we give a specific command to the drone. We called this specific command "special pose" that is to have your hands in a *"X"* position over your chest. After the user makes this pose for three seconds, the drone stops making the video and it passes to identified state;
4) **IDENTIFIED**: the drone starts following the commands of this person (e.g., commands with gestures hand).

This procedure is represented in Figure 12, and it is also represented with a state diagram as illustrated in Figure 13.

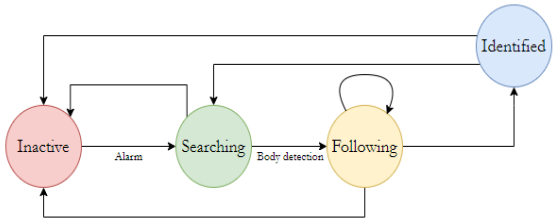

Figure 13: State diagram for holistic command recognition.

SECTION III

Implementation

To implement all the modules, we have used several design patterns studied in the software engineering because they allow to obtain a more understandable code and it ensures the reuse and the maintenance of the code in terms of conventions, refactoring and extensibility.

In the following paragraph we explain:

- Simple Factory pattern;
- Singleton pattern;
- Template pattern;
- Adapter pattern.

Simple Factory Pattern

This pattern is a factory object that decides which product class instance to create. We have three roles of the SimpleFactory model:

- **Creator role**: it creates the logic of all instances. The factory class provides all the static method needed to create the required product objects based on the parameters that we pass in;
- **Product role**: is the parent class of all the objects produced by simple factory and it describes the common interface for all the instances. Depending on the problem, it can be an abstract class (like in our project) or an interface;
- **Concrete Product role**: it creates all the objects that are instances of a class.

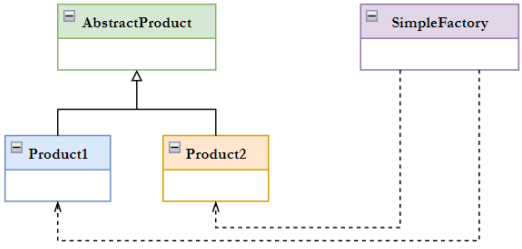

Figure 14: General scheme of the Simple Factory pattern.

In our project we have for example the class *StreamFactory* that is the *SimpleFactory* class.
AbstractVideoStream is the *AbstractProduct* that is the standard specification class.
The *WebCamStream* and *VideoDroneStream* classes inherits all the attributes and methods of the *AbstractVideoStream* superclass.
So, the class *StreamFactory* creates and returns an instance of *WebCamStream* or *VideoDroneStream* depending on the case.

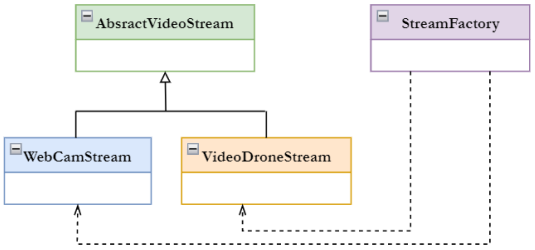

Figure 15: Example of Simple Factory used in our project.

This is only an example; in Figure 22, we have reported all the class diagram.
The reason why we have decided to use this design patter is because the factory class is able to dynamically instantiate the related classes considering the parameters passed and it removes the dependency of specific products.

Singleton Pattern

The singleton pattern is a creational design pattern. It creates unique objects for which there is only one instance and provides a global access point to it. This is a common pattern present in almost everything software product because, by creating a single copy of an object, it avoids initializing every time some components. In this project for instance, the drone must be initialized just once, so it is a unique object.

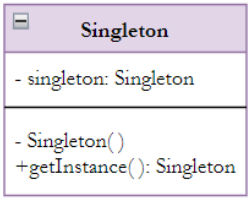

Figure 16: General scheme for singleton pattern.

Template Pattern

The template pattern is a behavioural design pattern. This pattern allows to create a base class in which we insert the necessary ordered steps to do to complete a process. It can be used to implement all the steps or only some of them.

To implement the template pattern, we must create an abstract class such that we have not to instantiate the whole base class to get access to all the steps defined in template method, but we are able to create subclasses in which we subscribe only the necessary steps. We can do that by exploiting the inheritance property.

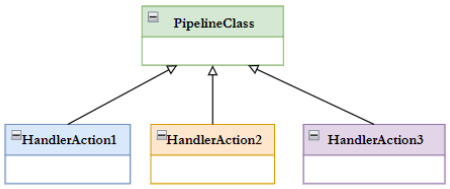

Figure 17: General scheme of the template pattern.

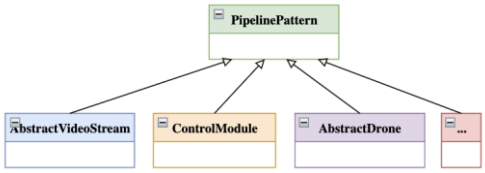

Figure 18: Example of the template pattern used in our project.

Since we use Python to implement this project, we make use of the ABC library that provides infrastructure for managing abstract base classes.

In our project we have, for example, *PipelinePattern* that correspond to the *PipelineClass*.Then we have *AbstractVideoStream, AbstractDrone, ControlModule,* etc., that has the role of a handler actions.

Adapter Pattern

The adapter pattern is a structural design pattern, and it has the role to satisfy the client by adapting an interface to another. One of the advantages of this pattern is that the client only sees the target interface. To implement this pattern, we have:

- **Client**: call the method of the adapter's interface to do the request;
- **Adapter**: translates the client request into some calls to methods of the adaptee's interface;
- **Adaptee**

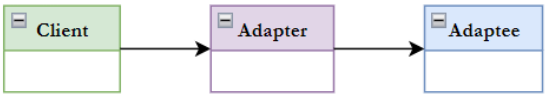

Figure 19: General scheme of the adapter pattern.

In our project, we have the client that is the Template that makes a request to the ControlModule and it translates this request by calling a function of the Drone.

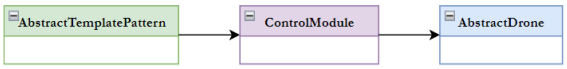

Figure 20: Example of adapter used in our project.

Finally, the total class diagram is illustrated in the following Figure 22.

Tracking

For both face command recognition and holistic command recognition, in order to control the drone, we use PID controllers (proportional, integral, derivative). These are control loop feedback techniques that allow us, given a measured process value and the desired value, to compute the deviation between them and they correct it considering the proportional, derivative, and integral terms. In Figure 21, we can see how it is structured the PID controller.

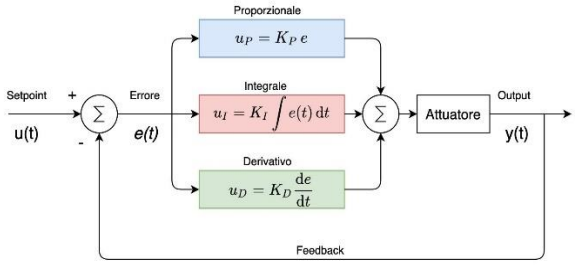

Figure 21: Block scheme of the PID controller.

So, as shown in the block scheme, we come up with the following formula:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

In which:

$K_p e(t)$: is the proportional action;

$K_i \int_0^t e(\tau)d\tau$: is the integral action;

$K_d \frac{de(t)}{dt}$: is the derivative action.

$e(t)$: is the difference between the setpoint, the measured process value and the control signal.

When we are working with a PID controller is important to tune it based on the value of the parameters that guarantee a smooth trajectory of the drone. These have been set after some experiments in which we tried multiple times the tracking in order to have an almost perfectly smooth motion.

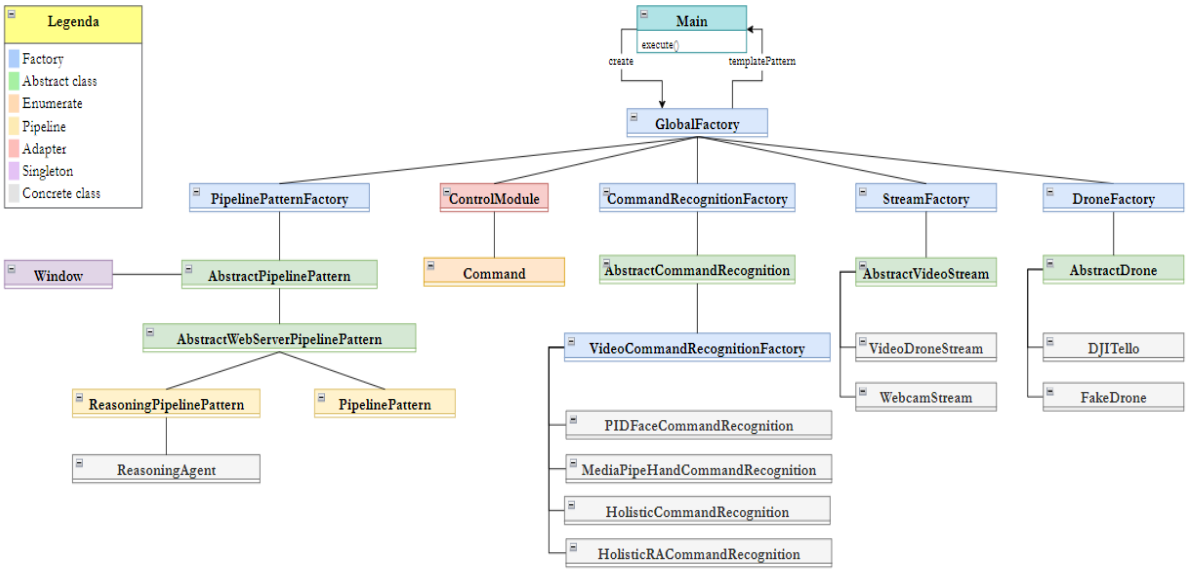

Figure 22: Class diagram of the whole project.

SECTION IV

Experiments and testing

All the experiments are done by using a the DJI Tello Drone EDU version [13]. It is a programmable drone that we have programmed by using Python language.
It has a 5MP 720p RGB frontal camera. It is also equipped with a distance sensor placed on its bottom size that allows the drone to stabilize the flight.
It supports the SDK 2.0 [14] that allows us to connect the drone using a UDP connection to the drone's Wi-Fi connection. This SDK is available for multiple programming languages; in this project Python is the chosen one, being easier and complete, also considering the Machine Learning libraries that have been used.

Figure 23: DJI Tello EDU Drone [13].

In order to test the project, we have selected some participants that do not know anything about the code. Before the testing phase, we have instructed the participants on which are the gestures or movements accepted by the drone in order to obtain a correct answer.

HRI

Face recognition experiments

The **first** experiment is based on side movements. When the user moves sideways, the drone performs a small rotation that varies according to the magnitude of the movement made. This rotation is controlled by the PID controller that we have explained before.
The drone is able to see the movement of the user thanks to its front camera.
In Figure 24, we can see the application of this experiment.

The **second** experiment is based on the upward/downward movements of the user. When the user moves upward or downward, the drone must follow his face by modifying the value on its y-axis, so that also the drone moves up or down respectively.

The **third** experiment is based on the forward/backward movements of the user. When the user moves forward or backward, the drone must go forward or backward respectively, by modifying the value on its x-axis always by following the user's face.

These experiments highlight what the drone can do. But when we start to run the program, the drone can do these 6 actions (move left, move right, move up, move down, go forward, go backward) also at the same time, based on the user movements.

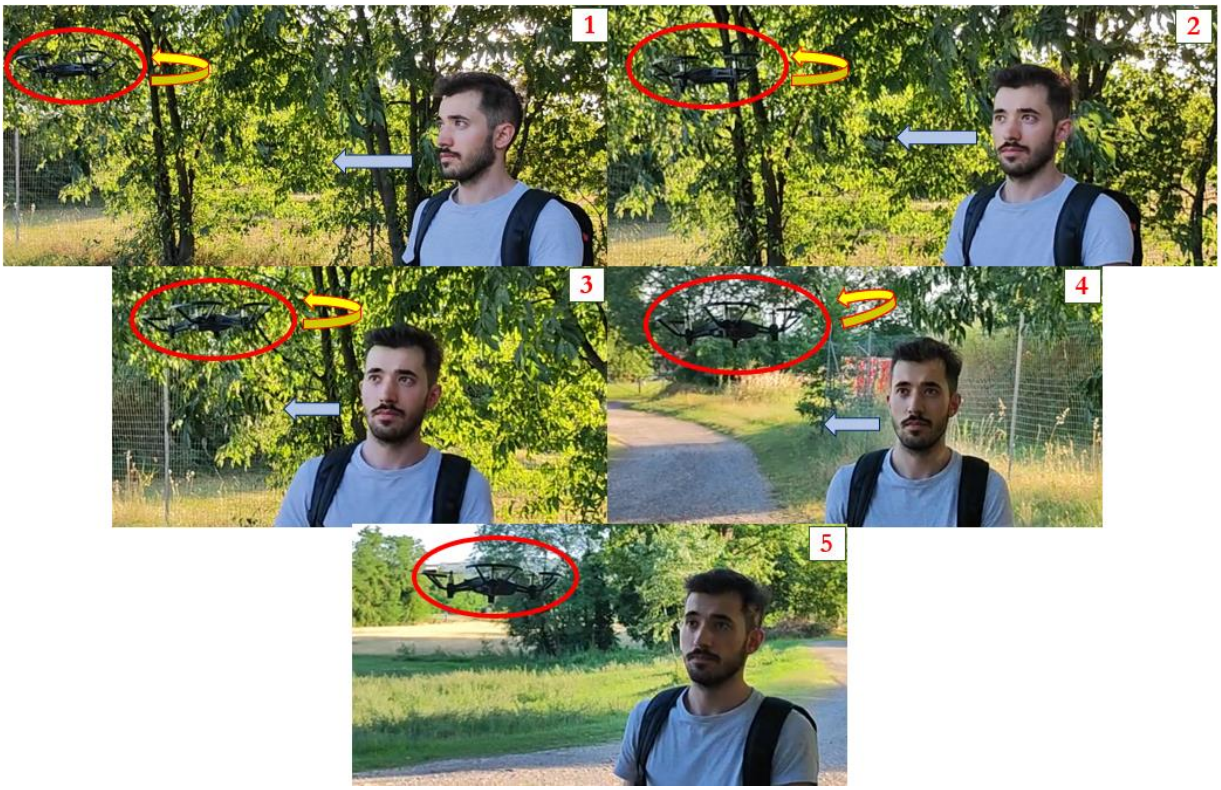

Figure 24: The first image represents the starting position and after some side movements of the user, the drone always rotates to follow the face until the user stops.

Gesture recognition experiments

For gesture recognition, the drone must wait a command done by the user's hands. In details, firstly the drone waits for two seconds to receive the same fixed command; secondly it takes the value that is in the left hand (distance value), and it takes the command from the right hand. Then, the drone waits two seconds before taking the next action. The allowed gestures are those represented in Figure 6.

In the following figures, we can see our experiments:

- Figure 25, we represent the up action;
- Figure 26, we represent the down action;
- Figure 27, we represent the forward and stop actions;
- Figure 28, we represent the left action;
- Figure 29, we represent the land action;

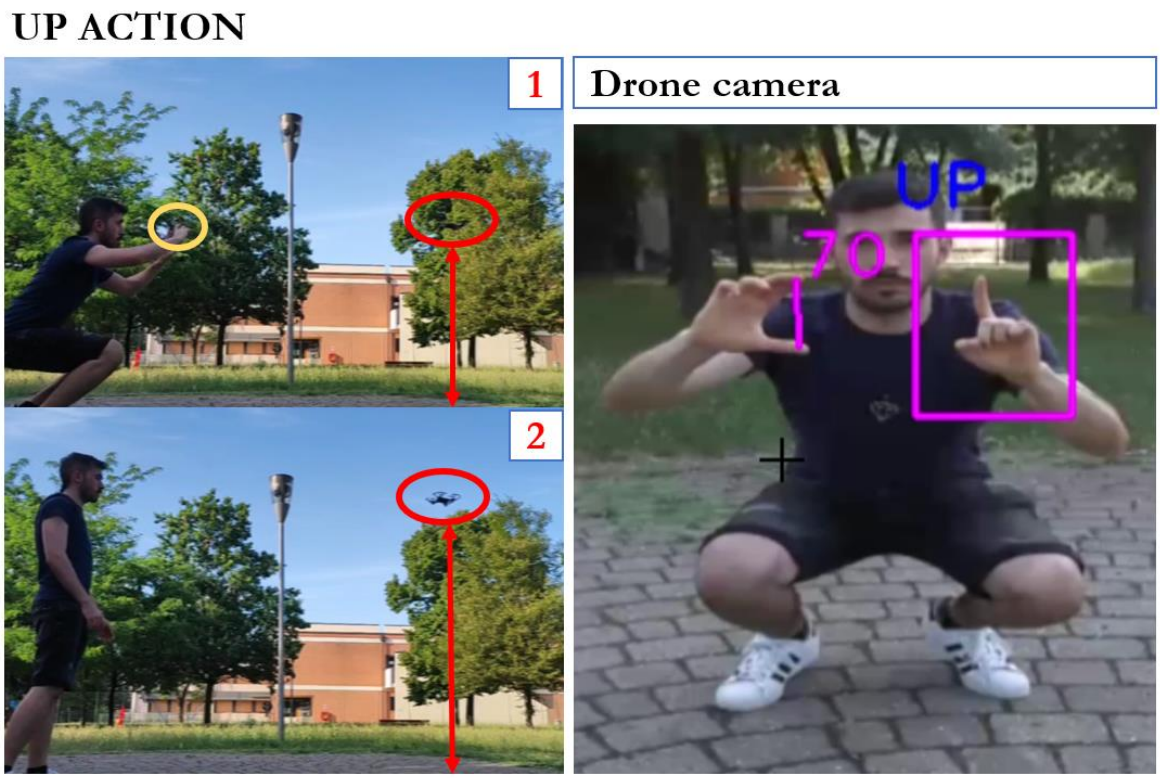

Figure 25: This represents the up action performed by the drone. In fact, in the first figure we can see that the distance between the drone and the floor is smaller than that in the second one.
In the left-side image there is the representation of which the drone sees by its monocular camera.

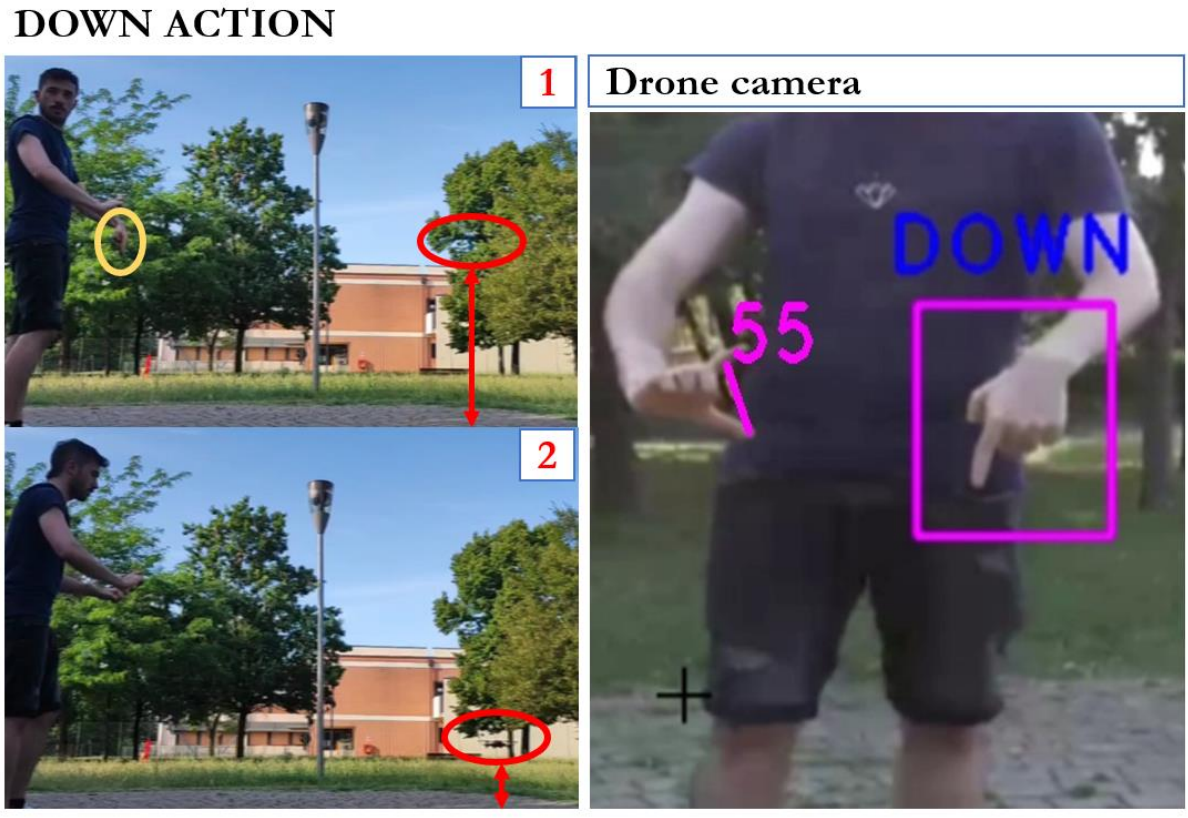

Figure 26: This represents the down action performed by the drone. In fact, in the first figure we can see that the distance between the drone and the floor is greater than that in the second one.
In the left-side image there is the representation of which the drone sees by its monocular camera.

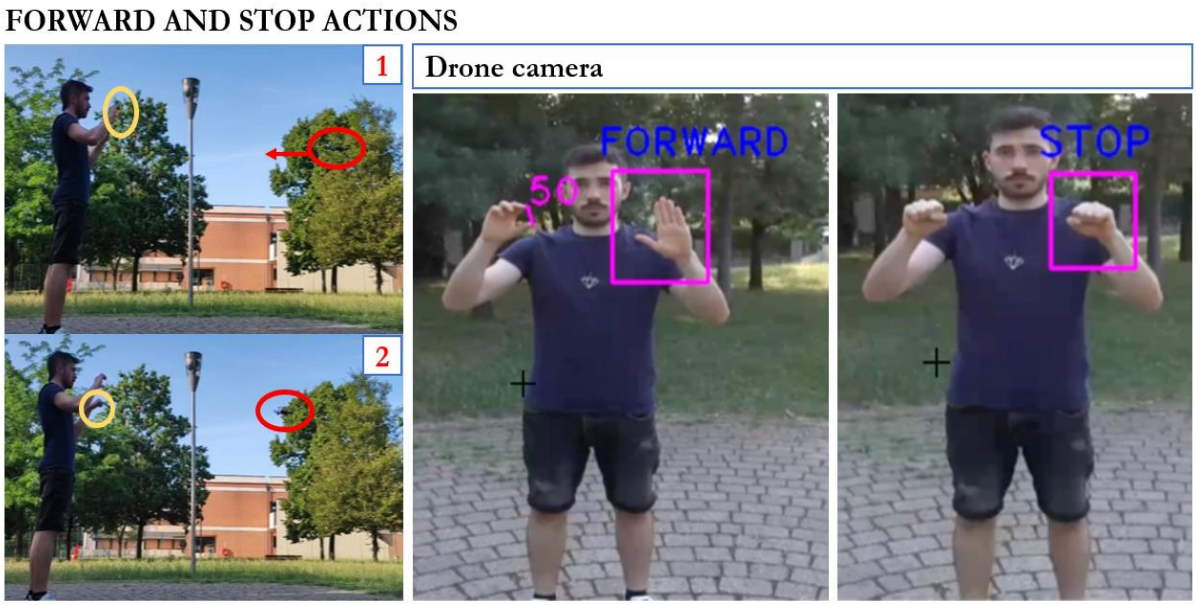

Figure 27: This represents the forward and stop actions performed by the drone. In fact, between the two images we can notice that in the second image the drone is closer to the user.
In the left-side image there is the representation of which the drone sees by its monocular camera.

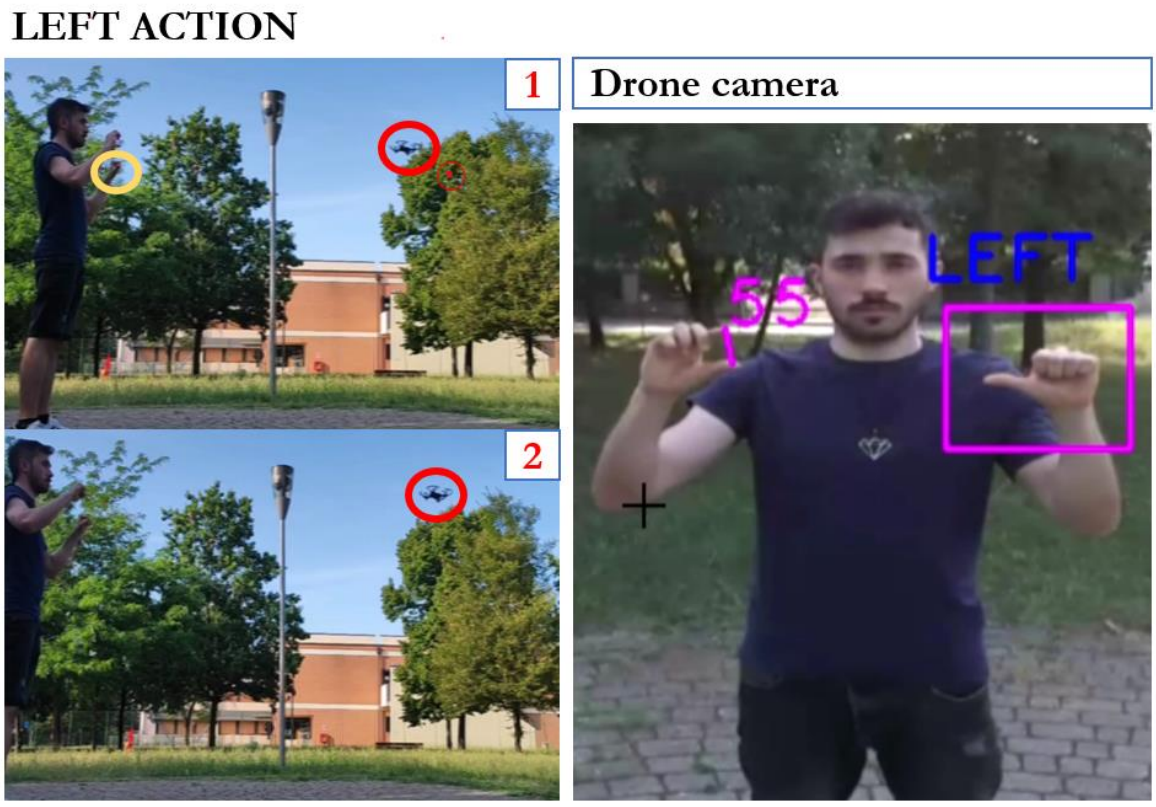

Figure 28: This represents the left action performed by the drone.
In the left-side image there is the representation of which the drone sees by its monocular camera.

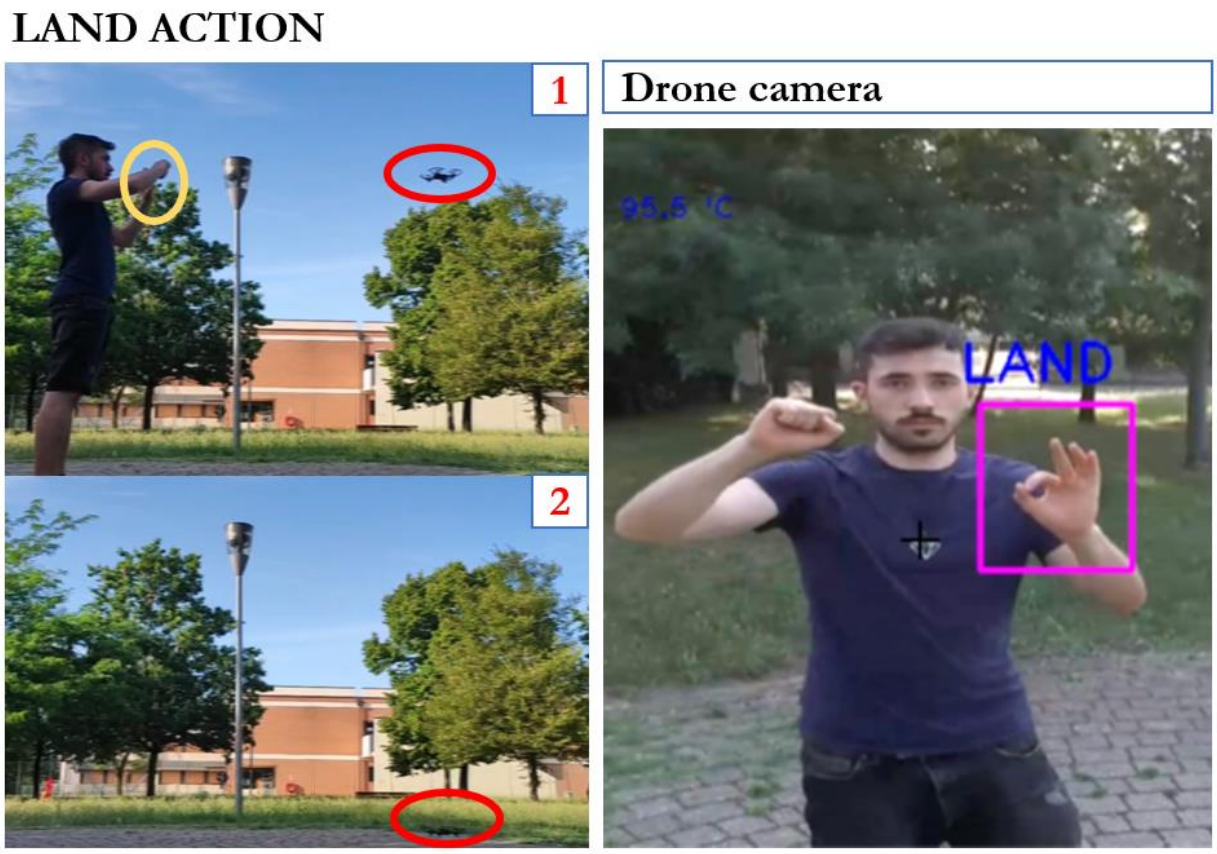

Figure 29: This represents the land action performed by the drone. In fact, given this command we can notice that in the second image the drone is perfectly on the floor.
In the left-side image there is the representation of which the drone sees by its monocular camera.

RA

Holistic experiment

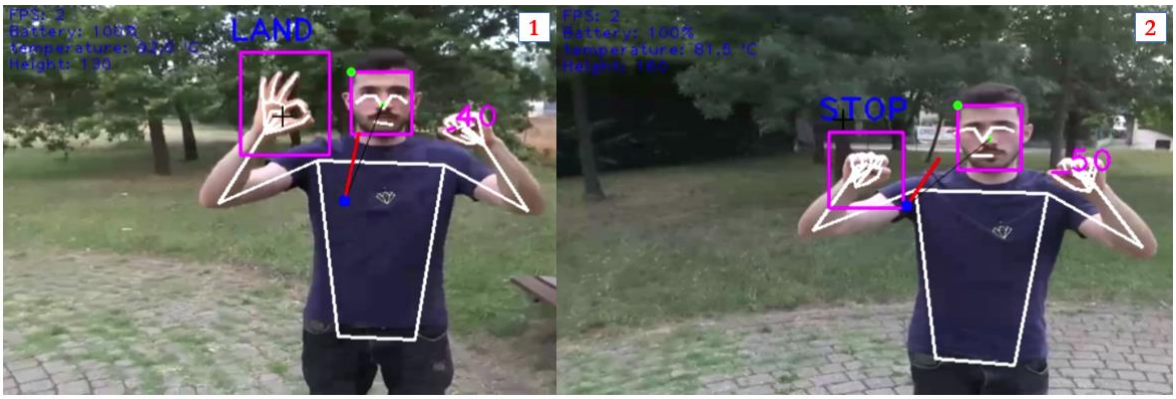

Figure 30: The first image represents the holistic approach considering the land action; the image 2 represents the holistic approach considering the stop action.

For the Holistic approach, the drone follows the body of the person, if the user shows the hands to interact with the drone, it stops moving on the x-axis and starts to follow the face of the user and waits to receive the command from the user. The drone, based on the data that it reads, can decide to do some action without the command of the user. For example, if the battery is low, the drone can decide to land and shut down.

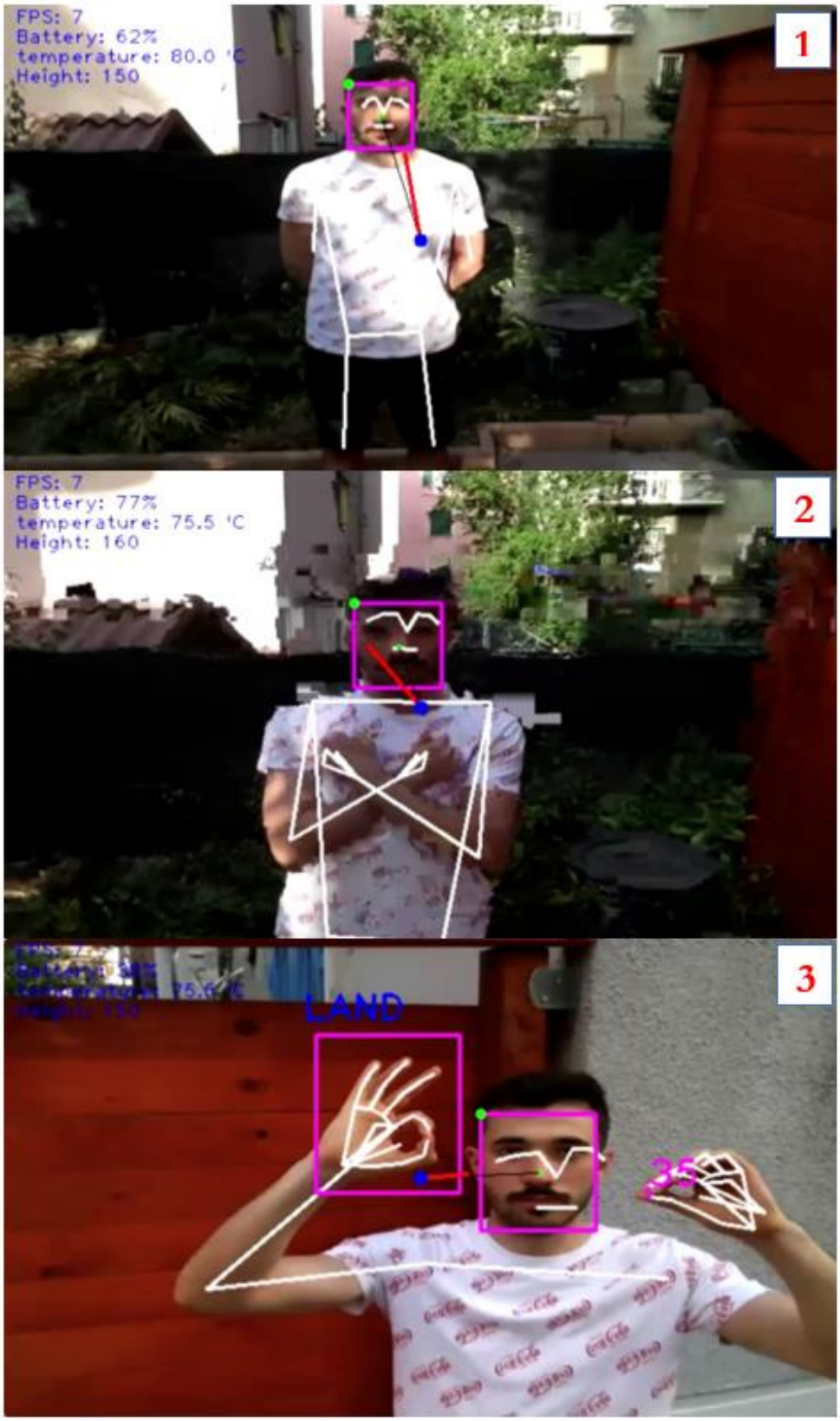

Figure 31: In the first figure, there is the initial position of the user, in which the drone follows the face of the drone; in the second image the user does the "special pose" because wants that the drone stops to follow his face; in the third figure the user does the land gesture to terminate the experiment.
All these figures are done by the camera of the drone.

The images represented in Figure 31 have also the correspondent figures recorded by a person. These can be seen in Figure 32.

Figure 32: The same experiment of the previous figure but by recording with an external camera.

SECTION V

Conclusions and future works

In this work, we were able to achieve a good result in terms of gesture recognition. The DJI Tello drone has been proved to be useful and effective to conduct these initial studies, mainly thanks to its SDK which is really powerful and to its low cost that enables small teams like our to do these experiments. Using a better drone with a better camera, however, would have surely improved the capabilities of recognition in all the light conditions, which were poorly tested in this project. As future works, there are a lot of points of improvements:

1. Using a different drone by implementing the drone interface, since this was not tested at all;
2. Improve the flow of commands to the drone because sometime, due to low network conditions, they could be skipped;
3. Trying to implement this system directly on drone with micro controllers like Raspberry Pi or recent control boards.

HRI

The implemented gestures are quite simple and natural, and we had quite good in performances during tests. Several points of improvements can be found, especially when more than one person is present in the scene. In particular, the libraries support multi people detection – only for faces and hands detection, not holistic – but it is quite difficult to match a hand – or face – with a person, resulting in a confusional control because of a non-distinguishable gesture command. Moreover, a facial recognition – not only detection – to distinguish who can give the command, would be a valid solution in case of multi people detection. It can be also thought the introduction of an Audio Capture Module would allow the drone to be driven by voice. In particular, the audio could be taken from the microphone of the computer or from a mobile device; in general, it is not convenient from the drone itself due to the propellers' noise. Eventually, the drone's behaviour and the actions computed by it can be changed according to the emotion of the user as done by [15]. So, the distance from the user can change if he is happy or angry. Furthermore, the age of the user could lead changes in the behaviour. For example, if a person is old, the drone should move slower than if a person is young.

RA

The face tracking future works as expected, although it could gain some reactiveness by using a more reactive communication system – radio wave against Wi-Fi in our case – since the commands are processed on the computer. The surveillance system works as expected too, although should be tested in a bigger environment and, for now, is not particularly robust to occlusions. A major keypoint improvement on this part would be applying a multi agent system to form a swarm and search and watch over all together. Another point of improvement would be changing the searching pattern to a more sophisticated one, in order to explore the area searching for the intruder.

References

[1] Avila, Mauro, Markus Funk, and Niels Henze. "Dronenavigator: Using drones for navigating visually impaired persons." Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility. 2015.

[2] W. Lee, B. Shazaad, B. Alkouz, A. Bouguetaya. "Package Delivery Using Autonomous Drones in Skyways.".

[3] H. Dermot Doran, M. Reif, M. OChler, C.Stohr. "Conceptual Design of Human-Drone Communication in Collaborative Environments." IEEE/IFIP International Conference on DSN-W. 2020.

[4] S.G. Zwaan and E. I. Barakova. "Boxing against drones: Drones in sports education".

[5] J. Han and I. Bae. "Social Proxemics of Huma-Drone Interaction: Flying Altitude and Size", 2018.

[6] Jane L. E, Ilene L. E, James A. Landay, and Jessica R. auchard. 2017. Drone & Wo: Cultural Influences on Human-Drone Interaction Techniques. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 6794–6799.

[7] Gio, N., Brisco, R., Vuletic, T. (2021) "Control of a Drone with Body Gestures", in Proceedings of the International Conference on Engineering Design (ICED21), Gothenburg, Sweden, 16-20 August 2021. DOI:10.1017/ pds.2021.76.

[8] Van Khoi Nguyen and R. Alba-Flores, "UAVs Control Using 3D Hand Keypoint Gestures", 2022.

[9] Zdravko Marinov, Stanka Vasileva, Qing Wang, "Human-Drone Interaction".

[10] A. C. Janbandhu, S. Sharma, I. A. Ansari and V. Bajaj, "Drone-based vision system: surveillance during calamities".

[11] S. Bernardini, M. Fox, D. Long "Combining temporal planning with probabilistic reasoning for autonomous surveillance missions". 2017.

[12] D. Srivastava, D. M. Lofaro, T. Schuler, D. Sofge and D. W. Aha, "Case-Based Interface for Multiagent Formation Control", 2020.

[13] Tello: Ryze robotics. https://www.ryzerobotics.com/tello-edu

[14] Tello SDK 2.0 User Guide. https://bit.ly/3qpnsI3, 2018.

[15] E. Malliaraki (2018) Social Interaction with Drones using Human Emotion Recognition. Companion of the ACM/IEEE International Conference on Human-Robot Interaction. https://doi.org/10.1145/3173386.3176966.