POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Deep Learning-Based Real-Time Detection and Object Tracking on an Autonomous Rover with GPU based embedded device

Relatore
Prof. Ing. **Paolo Garza**
Politecnico di Torino

Correlatore
Prof. Ing. **Aldo Algorry**
Universidad Nacional de Córdoba

Candidato
Aldo Caliò

A.A. 2020-2021

“I computer sono incredibilmente veloci, accurati e stupidi.
Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.
L'insieme dei due costituisce una forza incalcolabile.”
Albert Einstein

Alla mia Famiglia,
Luogo di partenze e arrivi del cuore.

Abstract

In recent years, there has been a growing interest in autonomous systems with artificial intelligence, especially in the military, home automation and smart cities sectors. In this project, the aim is to develop, through an artificial neural network capable of detecting objects, a system that can make a rover-type vehicle independent. An autonomous vehicle is a system capable of perceiving its environment and moving safely with little or no human input. Based on an appropriate network according to the needs, a tracking algorithm is developed, which gives the memory to the intelligence, i.e. it can assign an identity to any detected object, around frames in a real-time video. The implementation of a tracking algorithm would allow the system to be more dynamic and to perform more complex functions than simply detecting and tracking the object frame by frame.
This paper explores the possibility to create an autonomous system that allows the user to select the desired mode of use by means of a graphic user interface. A Follow Me module is implemented, a technology that is now present in many systems of this type, such as drones or domestic robots, which allows the drone to follow a moving object around autonomously. The Follow Me module allows any person detected to voluntarily activate and, vice versa, deactivate the tracking function of the system.

The following work is part of a larger research project, entitled "INTEGRATION OF OBJECT DETECTION IN REAL-TIME IMAGES IN AUTONOMOUS VEHICLE DRIVING", supported by the National University of Cordoba, aimed at the autonomous operation of a rover-type land vehicle. The tests that will be carried out will relate to the use of a very specific board, the INVIDIA Jetson-Nano, which is mounted in the physical system supplied to the general project.

Keywords

CNN *Convolutional Neural Network.*
BBOX *Bounding Box*.

SSD *Single Shot Detector.*

ML *Machine Learning*

SOT *Single Object Tracking*.

MOT *Multiple Object Tracking*.

GPIO *General Purpose Input/Output*. Entrada/Salida de Propósito General.

CUDA *Compute Unified Device Architecture*.

API *Application Program Interface*.

CPU *Central Process Unit*.

GPU *Graphic Process Unit*.

FPS *Frame Por Second*.

AI *Artificial Intelligence*.

Index

Index of figures

Index of tables

1

Introduction and Background

1.1 Introduction

Machine learning is a method of data analysis that automates the construction of analytical models. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine Learning is forming its own name, with growing recognition showing that MA can play an important role within a wide range of crucial applications, such as data mining, natural language processing and image recognition.

Deep Learning is a subcategory of Machine Learning. It refers to the function and structure of a brain that is known as an artificial neural network. With the introduction of CNNs (Convolutional Neuronal Network) and the growth in computing power of processors, there has been an exponential increase in the use of artificial intelligences with DL techniques and especially in the category of Computer Vision. CV is one of the branches of Artificial Intelligence that has experienced the greatest growth in recent years. Computer Vision refers to the discipline that studies how to process, analyse and interpret images automatically.

1.1.1 Work methodology

Used tools:

- NVIDIA Jetson Nano
- Micro Usb 32Gb
- Camera de Raspberry
- Monitor HDMI
- Power MicroUSB 5V-2A
- Devices

The working methodology chosen to be used is iterative and incremental. This model was created in response to the weaknesses of the traditional waterfall model, and groups a series of tasks in small repetitive phases (iterations). The aim of each iteration is to evolve the product incrementally, implementing new features and new functionalities. This model consists of a series of steps that are repeated in each increment, starting with an analysis, and ending with the testing and documentation of the implemented functionality.

Some of the advantages of the iterative and incremental model compared to other software development models are:

● Users do not have to wait for delivery of the complete system before they can use it.

● Users can use the increments as prototypes and gain experience with the requirements.

● It allows the complexity of the project to be separated into smaller problems that are solved in each iteration.

● The final product that has clearly been made with incremental stages is much less likely to have a failure, and in case something does not work you can go back to the previous stage through backups of each stage.

There are two phases of the project that involve the design and implementation of the software. In these phases we proceed with the analysis of the tools we will be working with, the design of functional prototypes, and the extension of the functionality and structural improvements of these prototypes. This report describes in detail the design and implementation of the final prototypes of the pieces of software created.[1] This report describes in detail the design and implementation of the final prototypes of the pieces of software created.[1]

1.2 Background

1.2.1 Object Tracking

Object tracking is a field of Computer Vision that involves the tracking of objects as they move through various video frames. Tracking has many practical applications, including surveillance, image medical, animal tracking, traffic flow analysis, cars that are lead alone, the analysis of audience flow and the interaction between man and computer.

Technically, object tracking begins with object detection (identifying objects in an image and assigning them bounding boxes). The tracking algorithm of objects assigns an identification to each object identified in the image, and in subsequent tables tries to lead through this identification and identify the new position of the same object.

Here are some of the challenges of object tracking compared to the detection of static objects, including re-identification, appearance and disappearance, and occlusion. The two main types of Obj. Tracking commonly used are Offline Tracking, tracking objects in a recorded video where all the frames are known in advance, including future activity, and Online Tracking, tracking objects in a live video stream, for example, a surveillance camera (this is more challenging because the algorithm must work fast, and it is not possible to take future frames and combine them in the analysis).

Object tracking is simply an extension of object detection. Modern object detection algorithms can do most of the work of detecting objects and re-identifying them in later frames, and that tracking of objects can be reduced to a simple intuitive study. Other object training algorithms work in conjunction with the detection of objects

and apply deep learning techniques to bring an identified object to the following video frames.

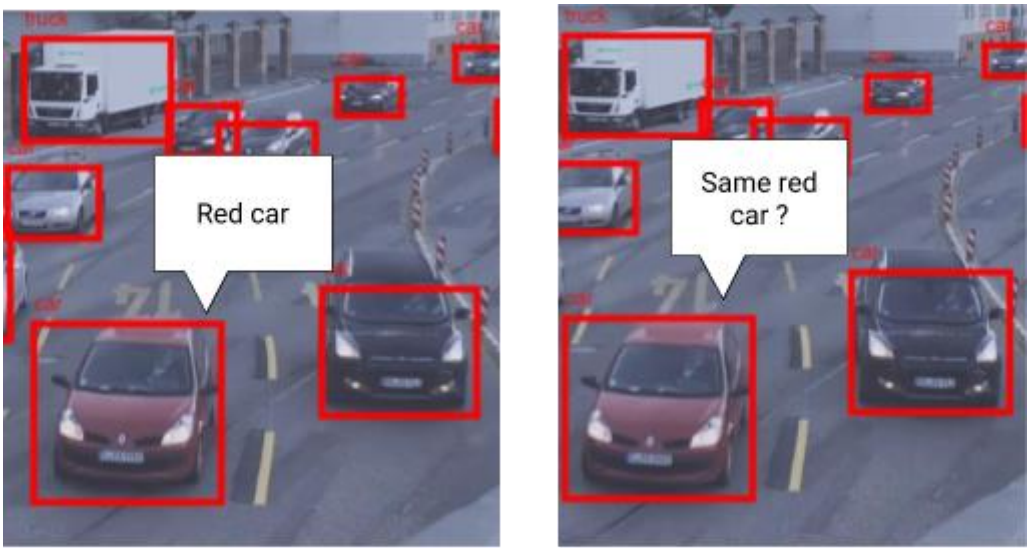

Figure 1.1: *Abstraction of the tracking issue*

At a high level of abstraction, there are mainly two levels of object tracking: Single Object Tracking (SOT) and Multiple Object Tracking (MOT). Multiple Object Tracking is a computer vision task that aims to analyze videos to identify and track objects belonging to one or more categories, such as pedestrians, cars, animals and inanimate objects, without any prior knowledge of the appearance and number of targets.

While in single object tracking (SOT) the aspect of the target is known a priori, in MOT a detection step is necessary to identify the targets that can go out or come in. The main difficulty in tracking various simultaneously derives from the various occlusions and interactions between objects that can sometimes also look similar. Tracking objects is not limited to 2D sequence data and can be applied to 3D domains. [2]

1.2.2 Challenges of Object Tracking

The main problems that can be found in real time video object tracking concern both aspects related to hardware limitations and aspects related to the movement of the tracked object and the characteristics of the surrounding environment:

1. **Re-identification** - connecting an object in a frame to the same object in the following frames.

2. **Scale change** - objects in a video can change scale dramatically, due to the camera's zoom, for example.

3. **Identity switches** - When two objects cross each other, we need discern which is which.

4. **Appearance and disappearance** - Appearance and disappearance can enter or leave the frame unpredictably and we have to connect them to the objects previously seen in the video.

5. **Monitor blur** - Blurry motion objects may appear different due to its own movement or the movement of the camera.

6. **View points** - The view-objects can look very different from different points of view, and we have to consistently identify the same object from all perspectives.

7. **Occlusion** - Occluded objects are partially or completely occluded in some paintings, as other objects appear in front of them and cover them.

8. **Illumination** - Changes in the lighting of a video can have a big effect on the appearance of objects and can make them more difficult to detect in a consistent. [3]

1.2.3 NVIDIA JETSON NANO

It was decided to use the NVIDIA® Jetson Nano™ Developer Kit for the realization of this project. This board is a small, powerful computer that can run multiple neural networks in parallel for applications such as image classification, object detection, segmentation and voice processing.

All in an easy-to-use platform that runs on just 5 watts. Precisely for this reason it was decided to use it for image processing because of its ratio between processing power and low power consumption.

Figure 1.2: *NVIDIA Jetson Nano*

1.2.3.1 Comparison between TFLOPS

FLOPS is an acronym for floating point operations per second. It is used as a general measure of the performance of a processing unit. Floating point operations are necessary when dealing with software that uses a large variety of numbers. The software stores exponentially large or exponentially large or small numbers in a predictable 64-bit encoded size. Converting these 64-bit instructions into raw numbers requires processing power. Processors can be measured in FLOPS, which refers to the speed with which they can convert bit instructions into numbers [4].

Comparison of computing power between Jetson modules:

Table 1.1: *Comparison of computing power between Jetson modules*

	Jetson Nano	**TX2 4gb**	**TX2i**	**Jetson Xavier NX**	**Jetson AGX Xavier Series**
AI Performance	472 GFLOPs	1.33 TFLOPs	1.26 TFLOPs	21 TOPs	32 TOPs

2

System design and software implementation

2.1 System Design

For the description and analysis of the system, the design of the system is divided according to the *Figure 2.1*

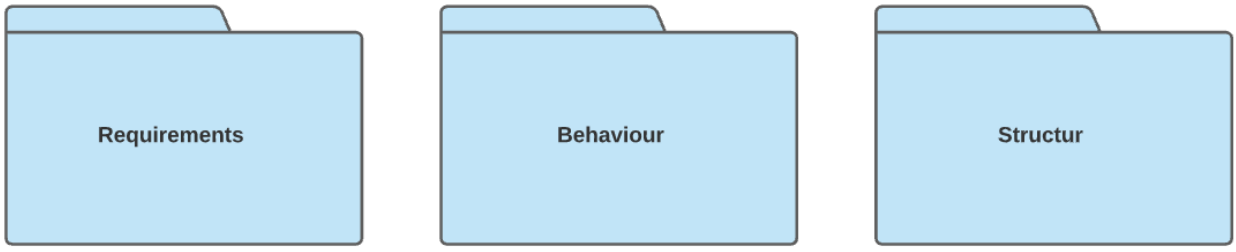

Figure 2.1: *System Design*

2.2 Functional and non-functional requirements

This chapter lists functional requirements, i.e. requirements that define the functions of a system or its subsystems, and non-functional requirements, i.e. requirements that specify the criteria that can be used to judge the functioning of the system.
In the table *Functional and non-functional requirements* all requirements are listed.

Table 2.1: *Functional and non-functional requirements*

ID Requirement	*Requirement declaration*	*Functional/ Non-Functional*	*Comment*
R01	The system shall be able to detect different kinds of objects.	Functional	Classes of COCO-Dataset
R02	The system shall parameterize the viewing space to assign a position to the object.	Functional	
R03	The system shall be able to select only one class of objects.	Functional	
R04	The system shall provide an interface for communication with the AI.	Functional	
R05	The system shall assign different IDs to each object of the selected class.	Functional	
R06	The system shall dynamically select and track the target object.	Functional	Dynamic means that the IA has a frame-by-frame memory of the target.
R07	The system shall reach a stationary or moving object.	Functional	
R08	The system shall allow an object to activate and deactivate the tracking function by a specific body movement.	Functional	Function available for objects of type person
R09	Python 3.6 will be used as programming language.	No Functional	To maintain compatibility with the API of the network used
R10	The system will have to achieve low latency between image production and algorithm output.	No Functional	Esencial para permitir el uso de la IA en tiempo real
R11	The system shall apply the safeguards for borderline cases in tracking	Functional	Case: “IDs switch”

2.3 Behaviour description

2.3.1 Use cases

Taking a High Level Software (in this case the GUI) as a user of the system, the use cases of the latter on the API are described:

Table 2.2: *CU1.1*

Name	**CU1.1** - Most Central Object
Precondition	In the most central object mode, the AI automatically sets the person in the centre of the image as the target, allowing you to follow them.
Description	In the most central object mode, the AI automatically sets the person in the centre of the image as the target, allowing you to follow them.
Normal sequence	1. The network recognises in a given frame t=x one or more persons. 2. All the distances of each recognised person are checked and only the person closest to the centre is taken. 3. This person is set as the target. 4. The target is followed
Postcondition	The network is always active, until it loses its target and the possible rover starts to follow it.

Table 2.3: *CU1.2*

Name	**CU1.2** - Farthest Mode
Precondition	The network is in operation, with the detection and tracking modules active. At least one object of the class "person" must be present (and therefore correctly identified) in the image.
Description	In Farthest Mode the AI automatically sets the person in the centre of the image as the target, allowing you to follow them.
Normal sequence	1. The network recognises in a given frame t=x one or more persons. 2. All the distances of each recognised person are checked and only the person farthest from the centre is taken. 3. This person is set as the target. 4. The target is followed
Postcondition	The network is always active, until it loses its target and the possible rover starts to follow it.

Table 2.4: *CU1.3*

Nome	**CU1.3** - Select Target
Precondition	The user chooses through the graphical interface the class of objects to be tracked. The network is in operation, with the detection and tracking modules active. At least one object of the class "person" must be present in the image (and therefore correctly identified).
Description	In Select Target mode, the AI automatically selects the object closest to the centre from all objects of the class chosen by the user and follows it.
Normal sequence	1. The network recognises in a given frame t=x one or more objects of the class chosen by the user. 2. All distances of each recognised object are checked and only the object closest to the centre is taken. 3. This object is set as the target. 4. Follow the target
Postcondition	The net is always active, until it misses the target and the possible car starts to follow it.

Table 2.5: *CU1.4*

Name	**CU1.4** – Follow Me
Precondition	The network is running, with the detection and tracking modules active. At least one object of the class "person" must be present (and therefore correctly identified) in the image.
Description	The Follow Me mode of use allows a person to be followed by the AI, through a specific movement that activates and deactivates it.
Normal sequence	1) The person requesting to be followed shall be positioned at such a distance that it is possible to detect the variation of his/her movement. 2) The person requesting to be followed must be facing the camera. 3) The person must activate, by a precise movement, the Follow function. 4) When the target wants to interrupt the following mode, repeat steps 1, 2 and 3..
Postcondition	After the target decides to end the FollowMe mode, the network remains active and ready for any further requests from the same or other targets.

Table 2.6: *CU1.5*

Name	**CU1.5** – Reach Target
Precondition	The network is running, with the detection and tracking modules active. The user has selected through the GUI interface the type of class he wants to reach.
Description	The network is running, with the detection and tracking modules active. The user has selected through the GUI interface the type of class he wants to reach.
Normal sequence	1) The network recognises in a given frame t=x one or more objects of the chosen class. 2) All distances from each recognised object to the centre are checked. 3) The object with the shortest distance from the centre of view of the camera is selected as the target. 4) The AI follows the object until it is reached.
Postcondition	After the target is close to the camera, the mode stops and the user can select another or the same class.

Table 2.7: *CU2.1*

Name	**CU2.1** - Check Target Lost
Precondition	The network is in operation, with the detection and tracking modules active, and a target has already been selected.
Description	The following module allows you to report the case when the target is lost during tracking. Lost means that the target is out of the picture for more than N seconds.
Normal sequence	1) The network recognises the loss of the target due to the absence of the target for too long. 2) The network de-selects the object with the lost target identification. 3) The system produces information about the 'lost target'.
Postcondition	The system sets the target loss parameters and returns to the mode selected by the user.

Table 2.8: *CU2.2*

Name	**CU2.2** - Check Collision
Precondition	The network is in operation, with the detection and tracking modules active, and a target has already been selected.
Description	The following module allows reporting when the camera is close to an object, either the target or all detected objects in general.
Normal sequence	1) The system checks if the lowest point of the captured objects in the image is below a certain threshold. 2) If point 1 is checked, a "stop" message is issued due to the possibility of a collision with one or more objects. 3) In case the obstacle is moving (people or animals for example) the stop message is maintained until the distance to it increases beyond the acceptable threshold.
Postcondition	The system sets the relevant parameters and returns to the mode selected by the user.

Tabla 2.9: *CU2.3*

Name	**CU2.3** - Check Target Switch
Precondition	The network is operational, with the detection and tracking modules active, and a target has already been selected.
Description	The following module allows reporting when the camera is close to an object, either the target or all detected objects in general.
Normal sequence	1) the network recognises in a given frame t=x a possible probability of exchange due to a non-respected minimum ownership distance between the target centroid and another possible object of the same class. 2) It is checked for n seconds if one of the two identifications remains pending, the pending condition is due to the fact that one or more objects due to their proximity are not recognised. 3) If this condition occurs for N seconds, the AI recognises that there has been a switch.
Postcondition	Mission abort

2.3.2 Sequence diagrams

To specify the communication and behavior of a system through its interaction with users (*Figure 2.2*).

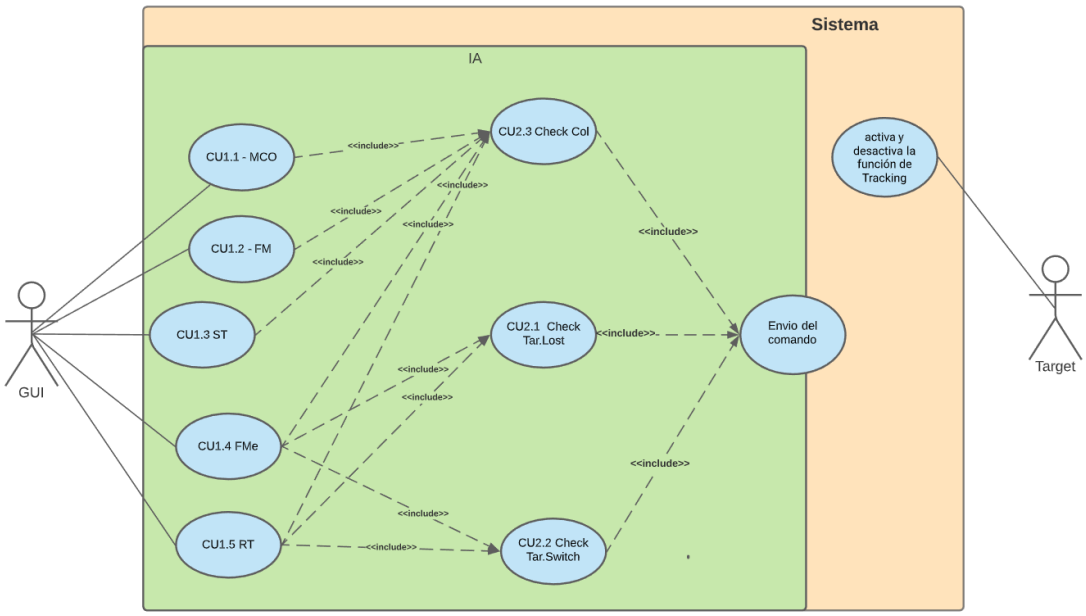

***Figure 2.2*: Use case diagrams**

2.3.3 Execution sequences

The following diagrams show the flow of operations that are executed in all functionalities by describing them at a high level. Only the execution flows of the functionalities developed in this work are presented:

- Follow Me
- Reach Target

2.3.3.1 Follow Me

To illustrate the series of steps involved, the execution of the Follow Me functionality is analysed:

1. The user, through the GUI interface, selects the Follow Me mode which calls the Follow_Me() function of the Object Detection Module.

2. The network takes the current frame as input and returns an array with all the information about the detected objects.

3. The coordinates of the bounding boxes of the current frame are passed to the Follow Me Module which updates the centroids.

4. With the updated centroids, it checks if a target is already fixed, if not, it waits for the user to activate the system.

5. When the system is activated, the functions of the Safe module are called function check_target(), check_collision(), check_switch() are called.

6. If errors are detected, error cases are handled, if not, a check is made to see if the target wants to disable the system. the target wants to disable the system.

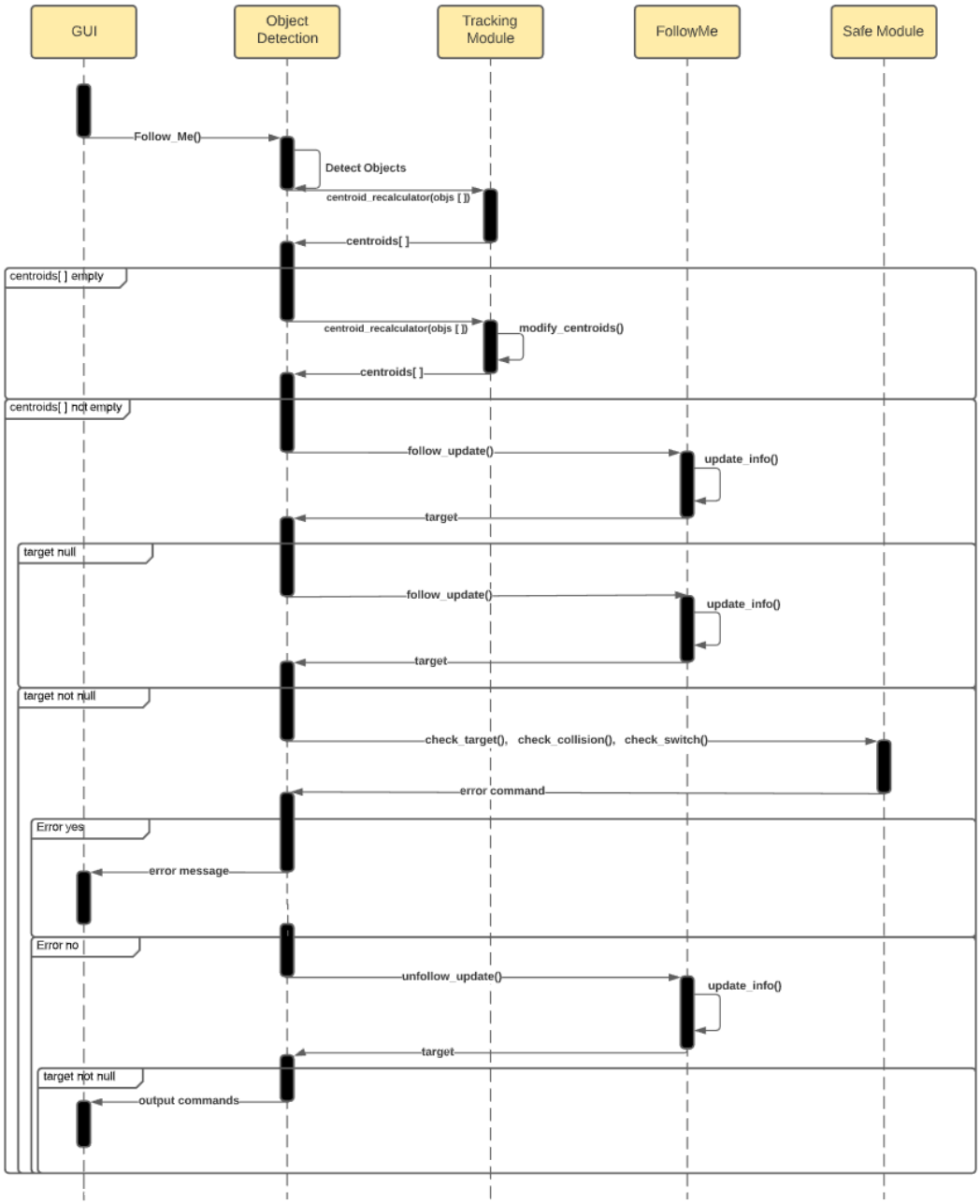

Figure 2.3: *Follow Me Mode execution diagram*

2.3.3.2 Reach Target

To illustrate the series of steps involved, the execution of the functionality is analysed:

1. The user, through the GUI interface, after entering the name of the preferred class, selects the Reach Target mode that calls the Reach Target function. class name, selects the Reach Target mode which calls the function Select_Dyn_Mode() function of the Object Detection Module.

2. The Select_Dyn_Mode() function calls the reach_target() function that manages the FollowMe. FollowMe function.

3. The network takes the current frame as input and returns an array with all the information about the detected objects. information about the detected objects.

4. The coordinates of the bounding boxes of the current frame are passed to the Tracking Module which updates the centroids.

5. With the updated centroids, the object of the user's chosen class closest to the centre of the current frame is set as the target. chosen by the user closest to the centre of the screen is set as the target.

6. Once the system is activated, the module's safe function check_target() is called to check if the target has been lost. check if the target has been lost.

7. If errors are detected, error cases are handled, otherwise the system checks if the target has been reached. check if the target has been reached. If the target has not been reached, the mission is If the target has not been reached, the mission continues and the function check_collision detects whether there is a collision.

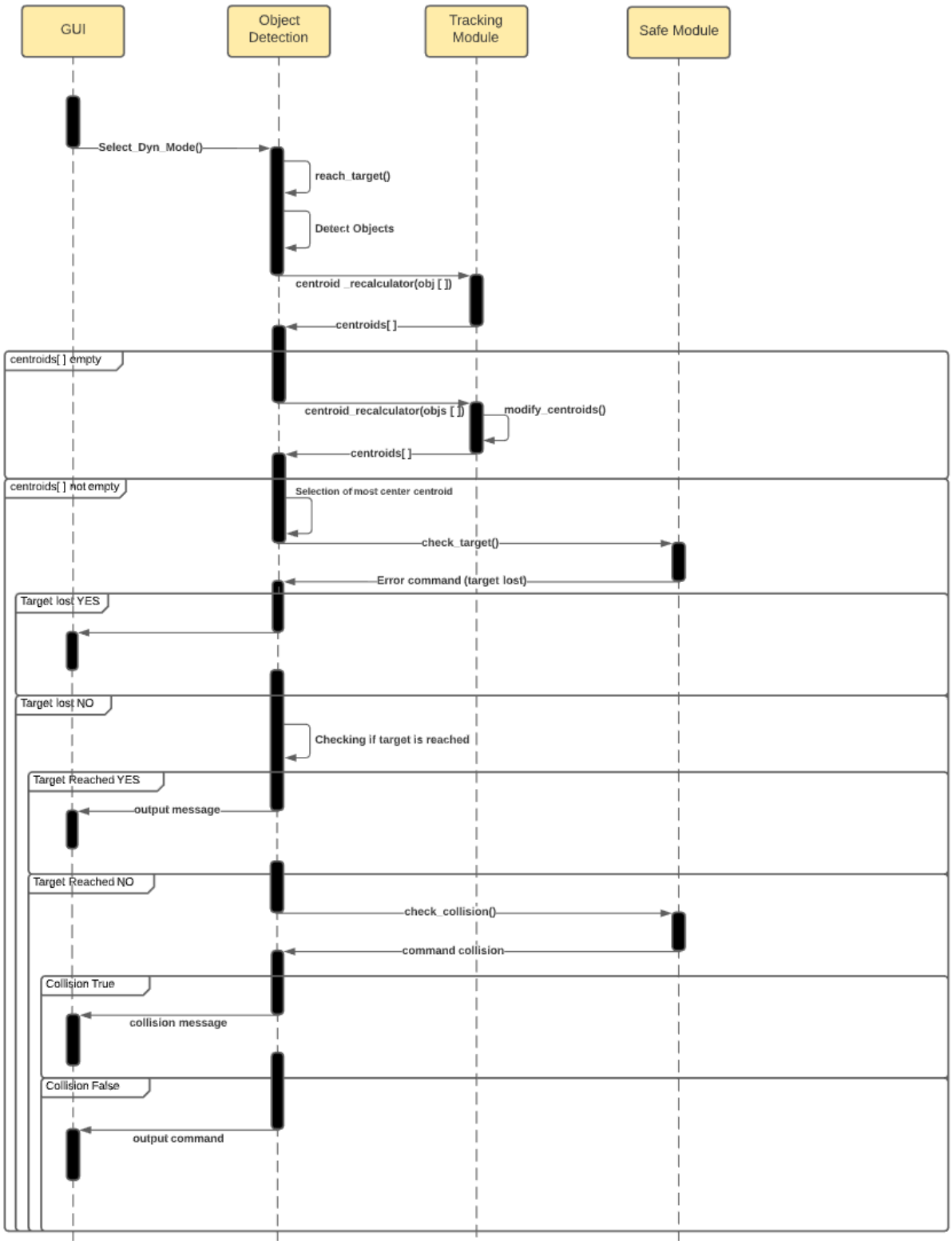

Figure 2.4: *Reach Target execution diagram*

2.4 Description of the structure

The system consists of several interrelated components which communicate for proper operation of the system:

- **Microcontroller:** This component consists of the microcontroller itself and its associated peripherals (such as the camera). Microcontroller and its associated peripherals (such as the camera) communicate with the artificial intelligence on both input and output. Artificial intelligence on both input and output.

- **Artificial Intelligence:** AI is responsible for implementing the artificial network and all the algorithms associated with object detection and tracking. It waits for commands from the microcontroller and receives in input the values passed by the camera (microcontroller component). camera (a component of the microcontroller), returning information about the decision to be made, the angle of the to be made, the angle of the object, etc.

- **GUI:** This component of the system allows the user who wishes to use its functions to choose between different ways of using artificial intelligence. Also allows to set certain options such as, for example, in certain cases, the type of class of the object to be tracked, etc.

- **System User:** The user is the actor who, through the GUI, communicates in input with the system and who receives in output the images from the camera with the changes made by the AI.

- **Target object:** The target object is the actor that refers to the target that the AI targets through its tracking algorithms. The target does not participate passively but, provided that the user chooses it and it belongs to the class "person", it can activate and deactivate the tracking of the target itself.

The system is expressed in the figures below (*Figure 2.5, Figure 2.6*, *Figure 2.7*)

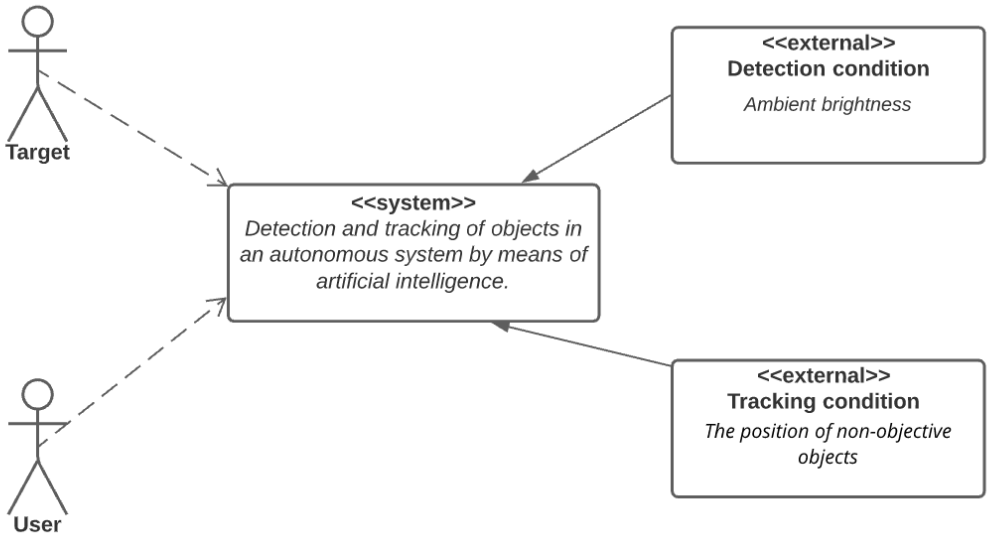

Figure 2.5: *Diagram of system context*

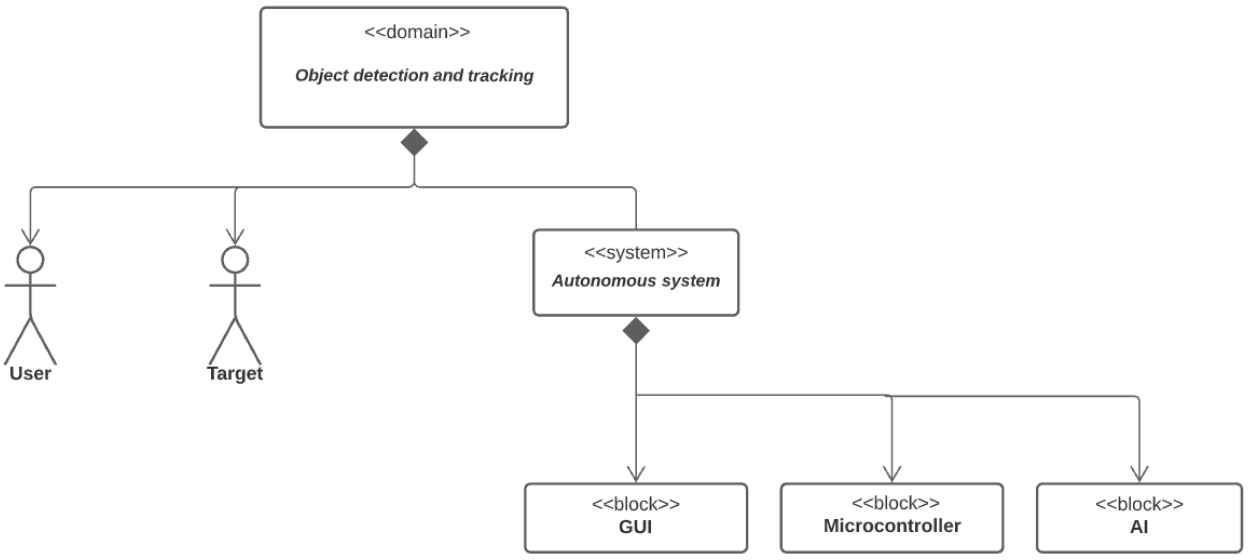

Figure 2.6: *High-level Block Definition Diagram*

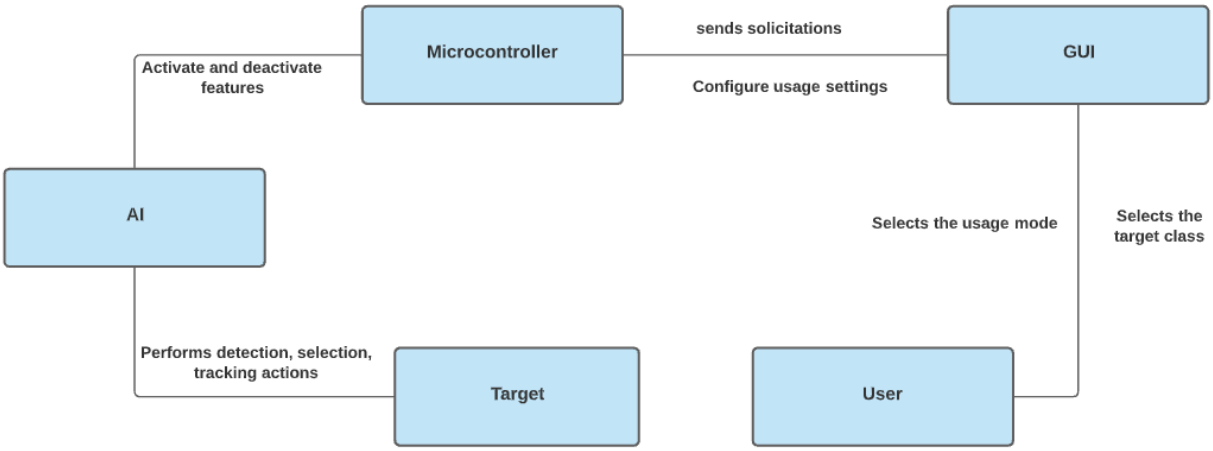

Figure 2.7: *Internal Block diagram*

2.5 API Architecture

The system consists of:

- A user interface, which through buttons allows the user to use 5. functions divided into two groups: Static Modes and Dynamic Modes.
- An Object Detection module that contains the neural network and the 5 functions of the system. of the system, also from this module are called the Tracking module, the Secure module and the Follow Me module. Secure module and the Follow Me module.
- A Tracking module that dynamically manages frame by frame all the various objects the various objects returned from the network and tracks them.
- A Follow Me module that allows a potential user to be followed by a given and vice versa.
- A Safe module to control possible collisions, swapping between the target and other objects of the same class and loss of the target.

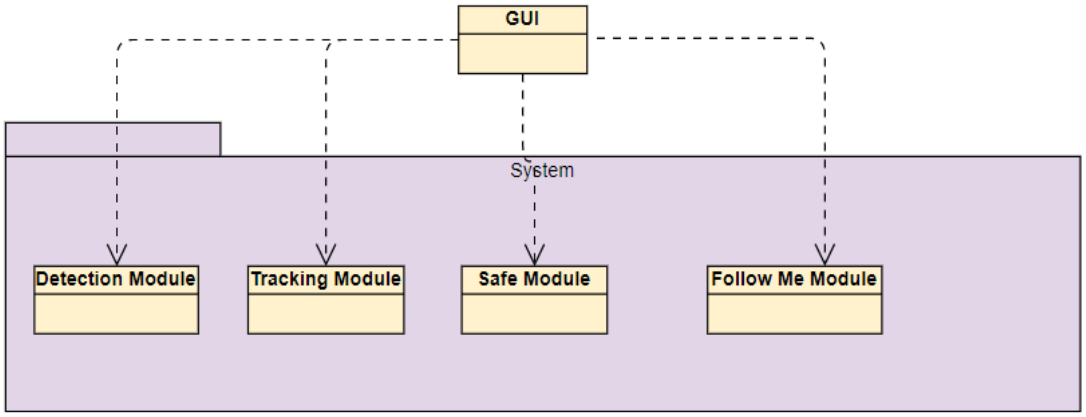

Figure 2.8: *High-level class diagram*

2.5.1 Class Diagrams

Before illustrating each individual Class Diagram, *Figure 2.9* shows the complete Class Diagram:

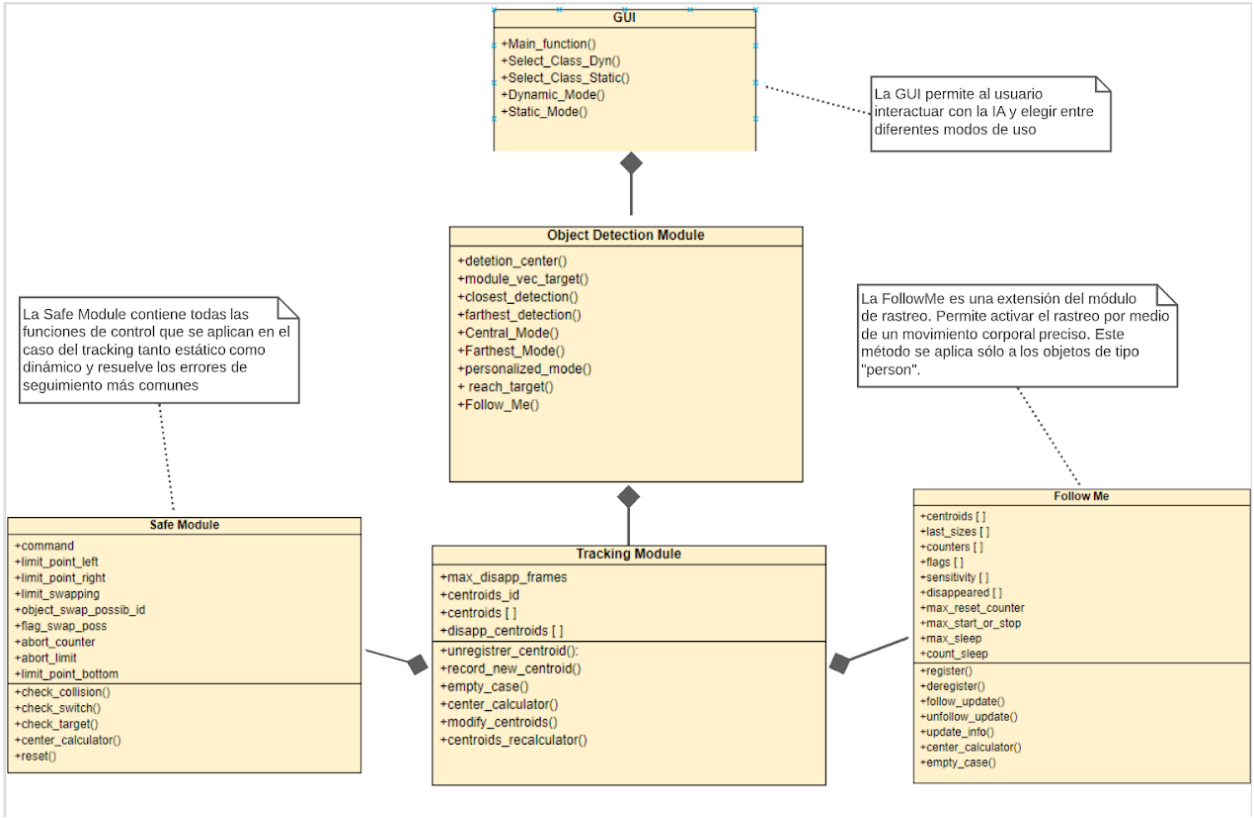

Figure 2.9: *Complete class diagram*

The various modules are explained one by one below:

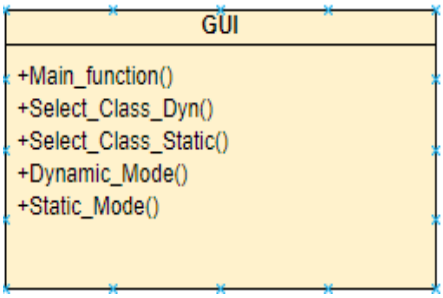

Figure 2.10: *GUI Module*

- *Main_function()*: This function creates a window and two buttons that call the two functions Dynamic_Mode() and Static_Mode().

- *Static_Mode()*: In this function a window is created with 3 buttons related to the 3 static functions, Most central object, Farthest object, Selection detection.

- *Dynamic_Mode()*: In this function a window is created with 3 buttons related to the 3 static functions, Most central object, Farthest object, Selection detection.

- *Select_Class_Dyn()* y *Select_Class_Static()*: The two functions create a window with an input for the user to window with an input for the user to choose the class to follow.

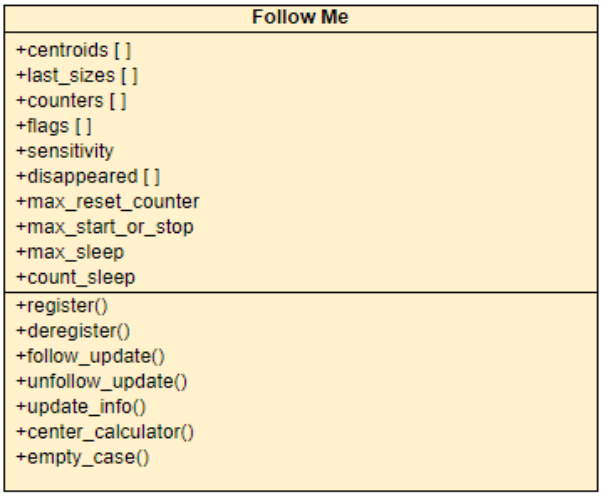

Figure 2.11: *Follow Me module*

- *centroids [ ]* : vector of all saved centroids of each frame delimiter of the selected class that is compared each time with the input centroids.

- *last_sizes [ ] :* vector in which the dimensions of the bounding box associated with each ID are stored in each cell.

- *counters [ ]* : vector in which a counter is stored for each ID that determines how many times the associated ID has performed a given movement.

- *flags [ ]* : vector that sets a flag of 0,1 for each ID.

- *sensitivity:* parameter that indicates the sensitivity with which all IDs must to make a certain movement to be followed.

- *disappeared [ ]* : vector which for each ID counts how many times a given object is not detected.

- *max_reset_counter :* parameter indicating when a counter related to the tracking activation must be reset. counter related to the activation of the tracking.

- *max_start_or_stop :* which is compared with the counter associated with each ID to understand whether a particular user has requested to be tracked or not.

- *max_sleep*: parameter that indicates for how many seconds the tracker should not detect any movement.

- *count_sleep:* parameter that is incremented to be compared to *max_sleep*.

- *register():* function that registers the new centroids.

- *deregister():* function to deregister a given centroid.

- *follow_update():* function that manages the movement of all IDs of each frame and returns a possible of each frame and returns a possible target.

- *unfollow_update() :* function which takes the current target as input and checks if the target has requested to be unfollowed.

- *update_info() :* function that checks the movement of all IDs and updates them.

- *center_calculator()*: function that takes in input all the bounding boxes and calculates the relative centroid.

- *empty_case()* : function that in the case that the algorithm in a frame t=x does not detect anythin, the associated IDs disappear.

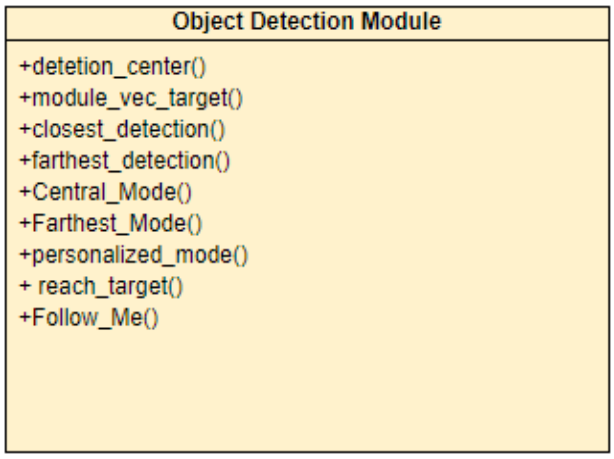

Figure 2.12: *Object Detection Module*

- *detection_center():* function that given a detection type object taken from the the network, returns the one closest to the centre.

- *closest_detection():* function that takes in input the vector of the objects detected and returns as close as possible to the centre of the camera.

- *farthest_detection():* function that takes in input the vector of detected objects and returns the farthest detected and returns as far as possible from the center of the camera.

- *Central_Mode() :* function that takes in input the vector of detected objects and returns the farthest detected and returns as far as possible from the centre of the camera.

- *Farthest_Mode()* : rutinas que manejas la funcionalidad de “Farthest Object”.

- *Personalized_Mode()* : routines that handle the functionality of "Select Detection Class Static".

- *reach_target()* : routines that handle the functionality of "Select Detection Class Dynamic".

- *Follow_Me()* : routines that handle the functionality of "Follow_Me"

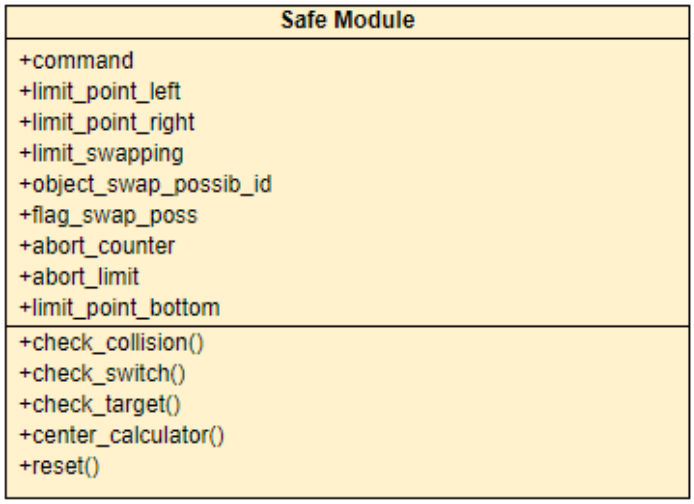

Figure 2.13: *Safe Module*

- *command* : parameter that assumes a certain value based on the Safe Module functions.

- *limit_point_left* : parameter that sets a left boundary based on the screen size in pixels.

- *limit_point_right* : parameter that sets a left limit based on the screen size in pixels. screen size in pixels.

- *limit_swapping*: parameter indicating the limit distance in pixels before the swap takes place swapping takes place.

- *object_swap_possib_id*: parameter indicating that a certain ID has a possibility of swapping.

- *flag_swap_poss*: parameter that if set to 1 indicates that there is a likely swap.

- *abort_counter* : counter to abort the mission.

- *abort_limit*: value to indicate mission abort.

- *limit_point_bottom*: lower level for check collision.

- *check_collision() :* function that takes in input all the bounding boxes of the detected objects in a frame objects detected in a frame t=x and checks for probability of collision.

- *check_switch()* : function that checks in each frame if there is a probability of swap.

- *check_target() :* function that checks in each frame if the target has been lost.

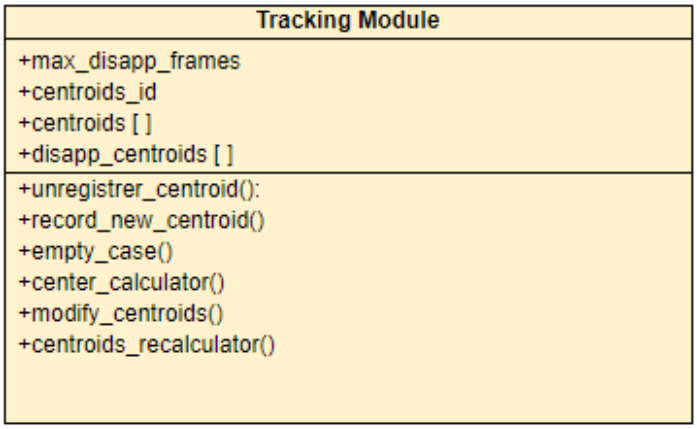

Figura 2.14: *Tracking Module*

- *max_disapp_frames*: parameter that indicates the limit in frames for which an object is removed.

- *centroids_id* : parameter used as an index. It counts the current number of centroids.

- *centroids[ ]* y *disapp_centroids [ ]* : these are two OrderedDict objects and are used to map the centroids and record those that are currently missing.

- unregister_centroid() : function that removes centroids.

- *record_new_centroid* : function that records centroids.

- *empty_case()* : function that in case the network does not detect anything sets the centroids as missing and in the case deletes them if the counter is equal to *max_disapp_frames*.

- *modify_centroids():* taking into input the current centroids, the function calculates all the distances between the current centroids and the saved ones and updates them

- *centroids_recalculator()* : given the bounding boxes in input, it calculates the current centroids.

3

Object tracking

In many situations, there are multiple objectives in the image that are of interest. Not only do you want to classify them, but also obtain their specific positions in the image. Ideally, you should use a stand-alone system with cameras that have a 360-degree view. However, this study is limited to a view of approximately 63 degrees of the system that is closer to the limitation due to the use of a single camera.

In object detection, a bounding box is often used to describe the location of the lens. The bounding box is a rectangular box that can be determined by the x and y axis coordinates in the upper left corner and the x and y axis coordinates in the lower right corner of the rectangle. The origin of the coordinates in the image above is the upper left corner of the image (TOPLEFT), and to the right and bottom are the positive directions of the x-axis and y-axis, respectively (the value of the axes is in pixels) [5].

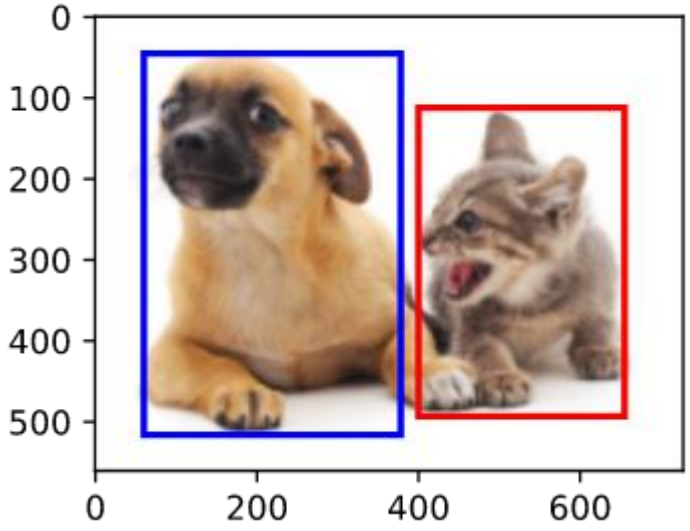

Figure 3.1: *Bounding boxes example*

3.1 Object Tracking implementation

For the practical purposes of the project, the simple detection of objects added with a "static" scan, and then recalculated frame by frame, of a pre-selected target thanks to its class or its position in the image, is lacking in many aspects.

To obtain a more robust system and to be able to track an object for a long time it is necessary to be able to guarantee a network memory, therefore an algorithm that assigns an identity to the subject. This would make it possible to follow the target, detect limit cases, such as the momentary output of the image object, and therefore correct the behavior of artificial intelligence.

Implementing a dynamic tracking algorithm would also allow for a generally more dynamic of the system, as it would make it possible, for example, to select tracking thanks to the specific movement of one of the objects, since in fact the system retains the memory of the information about the objects present and past.

3.1.1 Centroid Tracking Algorithm

The centroid algorithm is the most intuitive of the tracking algorithms, because calculates, for each object, the variation of the center from one frame to another. In comparison with algorithms based on the correlation of various characteristics of each object, the centroid algorithm is fast and capable of handling when the object being tracked "disappears" or moves outside the limits of the video frame, so it seems more suitable for application on the Jetson Nano and for project use cases where you want to track a single lens and try to keep it focused on the camera image.

The main assumption of the algorithm is that a given object will move potentially between subsequent frames, but the distance between the centroids for two subsequent frames will be less than all other distances between the objects.
The final objective is, therefore, to associate to each object an identifier corresponding to the nearest previous centroid, and in case of assigning to the new objects new identifications or deleting those that too many frames do not appear. [6]
The algorithm at time zero, in the first frame, assigns each object an identification, and then creates N centroids for N objects:

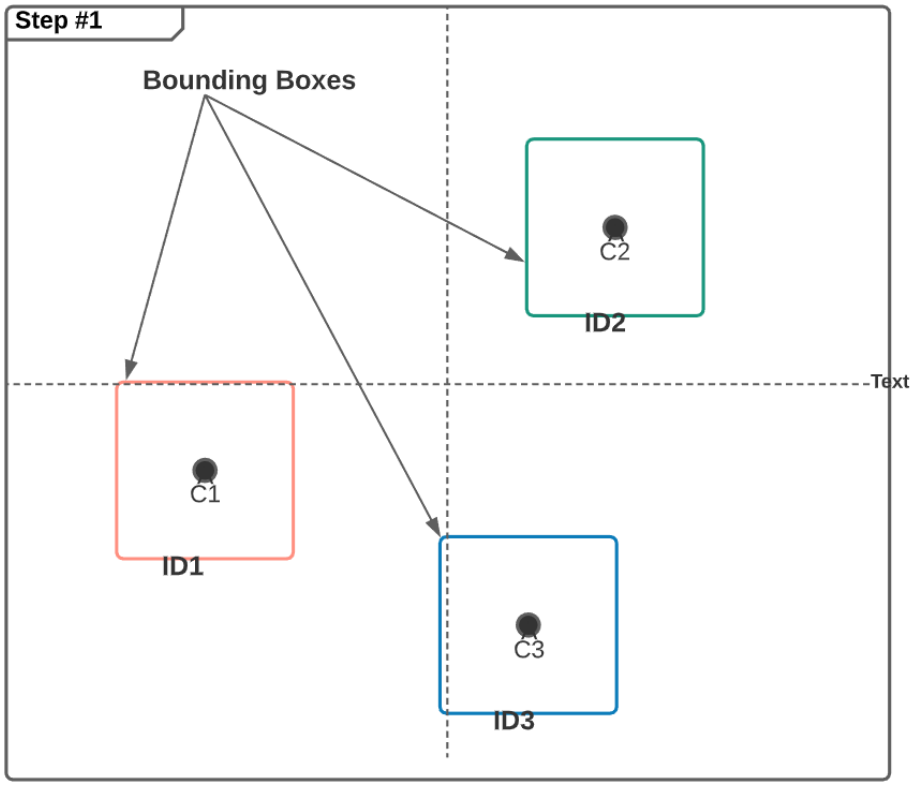

Figure 3.2: *Centroids algorithm: step1*

In the next step, given the new centres of the input objects, it calculates for each old centroid the Euclidean distance to all the new centres.
Therefore, if you choose to associate the centroids with minimum distances between subsequent frames, you can build the object tracker.

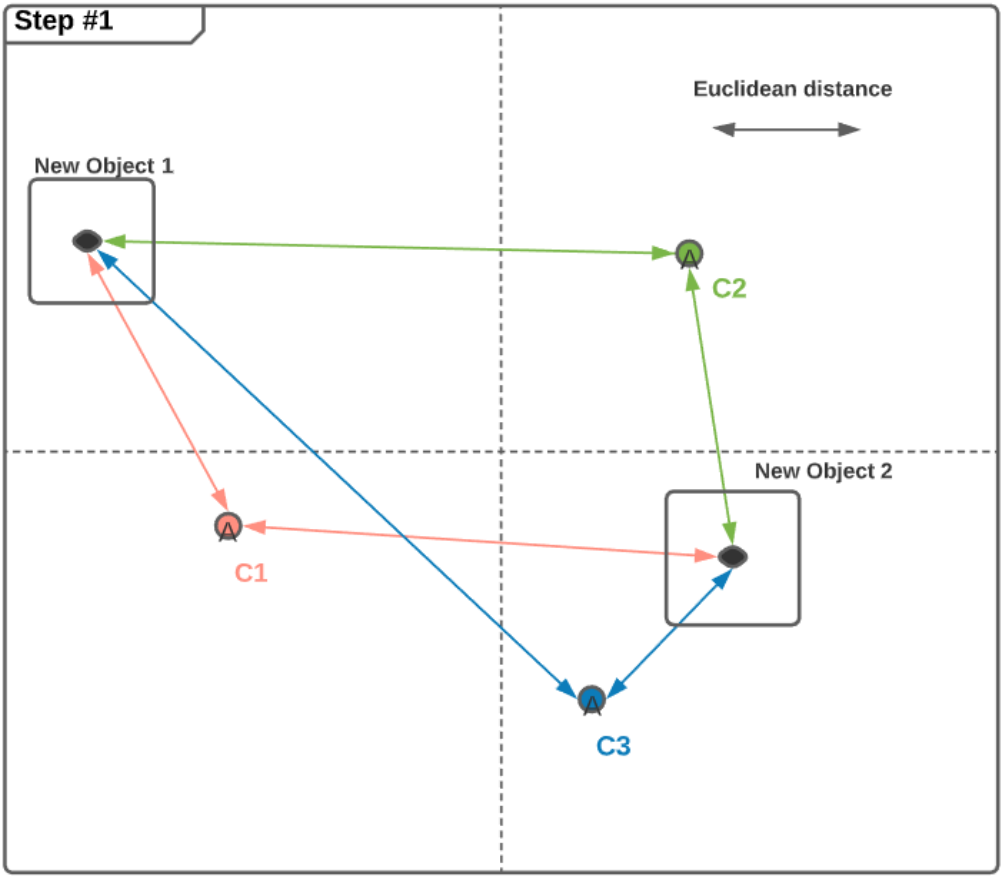

Figure 3.3: *Centroid algorithm: step 2*

The algorithm finally associates the new centers with the nearest centroid, to maintain the identity of the object, creating, if necessary, new pictures and/or eliminating those that in too many frames no longer appear in the image.

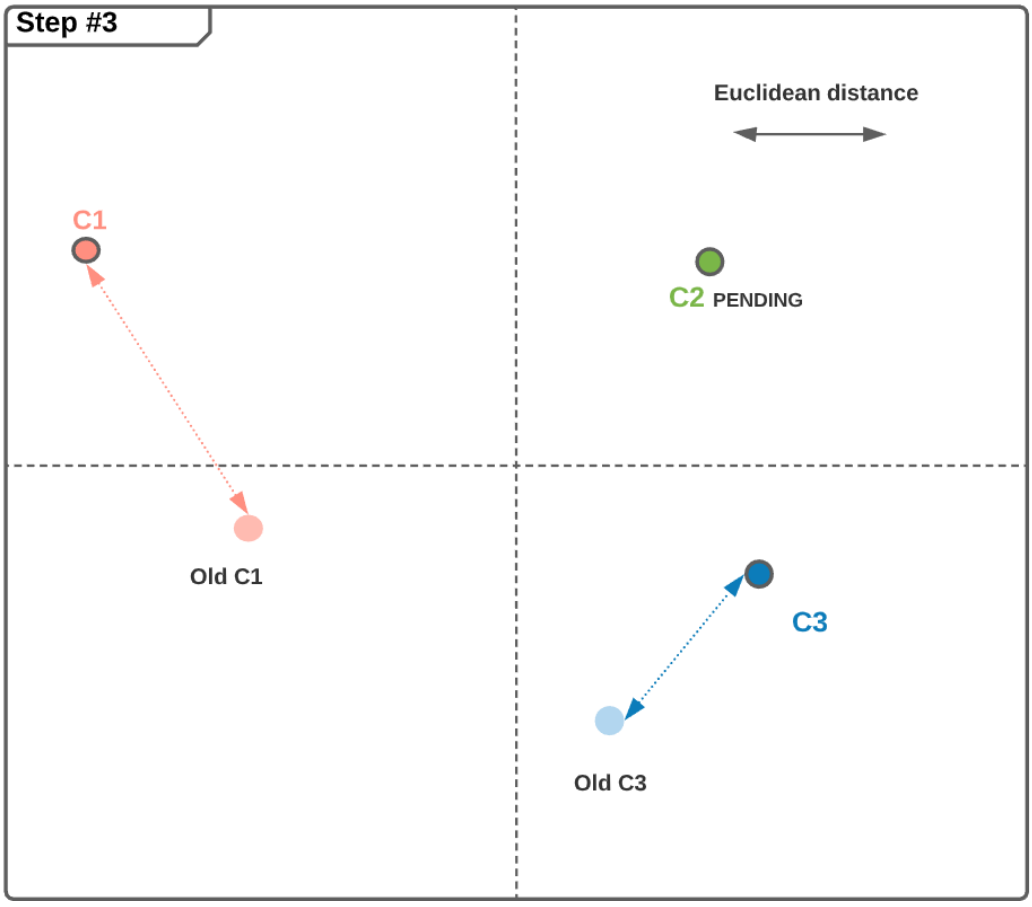

Figura 3.3: *Centroid algorithm: step 3*

3.1.2 Algorithm implementation

The implemented algorithm follows the following steps:

1) *(#old_centroids x #new_centroids*) of Euclidean distances between the centroid *i*, and the new centroid *j*. In the example, with *C1-C2-C3* old centroids and *N1-N2* new object centers.

2) After creating the matrix, for each old centroid the distance plus and a growing sorting out among the old centers to have a sort of a list of the "best" old centroids. Basically the "best" is the old centroid that gets the shortest distance from any new centroid.
Resultant vector *V1: [C2,C3,C1]*
3) Then another vector is created containing the nearest new centroid to each old centroid, always keeping the previously used sorting.

Then *V2: [N1,N2,N1]* always with index *(C2,C3,C1)*

- Finally, in a cycle, each old centroid (starting with the best old centroid) is associated with the nearest new centroid.
- loop 1: associate *C2* to *N1*
- loop 2: *C3->N2*
- loop 3: *C1* makes a miss, because N1 is already associated.

(Iter1) *C2->N1*, ***(Iter2)*** *C3->N2*, ***(Iter3)*** *C1->miss!*

Clearly the perfect case of use is when there is an NxN matrix, but there also may have more old or new centroids. In the example case, there is in fact an old centroid that does not find any coincidence, C1 is classified as "missing". After several frames, the centroid "falls" and is then removed. If there are more new ones than old ones, then the new ones simply do not associate with any of the old ones and are saved with new identifications incrementally.

4

Additional modules and GUI

This chapter will explain the modules, i.e. the GUI, the secure module and the Follow Me module. These modules serve to improve the user experience of the system. They also solve certain edge cases that are essential for the proper functioning of the system.

4.1 Identity switch recognizer

Among the functions implemented in the security module, the identity switch recognizer plays a central role. In the context of multiple objects tracking, one of the most common situations is in fact the overlapping of the target object with other objects in the image. It is therefore necessary to have a security function that is aware of this behavior and warns the system of a possible ID.

4.1.1 Identity switch

Computer vision systems have difficulty in maintaining the identity of the individuals over long periods of time; many modern vision systems multiple objects are based on tracking identification, which means they propagate the identities along the track. This can lead to the spread of identity changes as individuals approach and then disperse again. In the example examined, there may be an overlap in the image of the lens with another moving object:

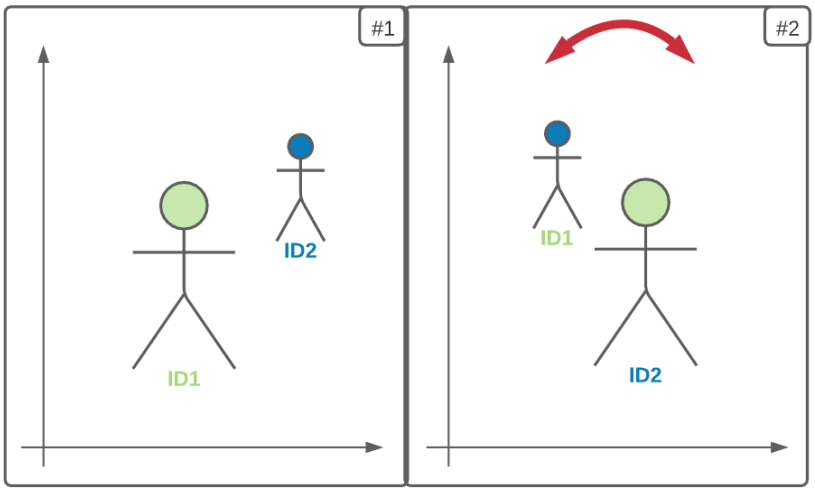

Figure 4.1: *Identity switch*

In the case of the centroid algorithm, based only on the displacement of the centers of the objects and not on the use of other characteristics that distinguish them, it happens that in case of overlapping there will be a random assignment of identifications, in fact the centre of a new object, previously assigned to a centroid, can be erroneously associated to another centroid that is closer than its associated object:

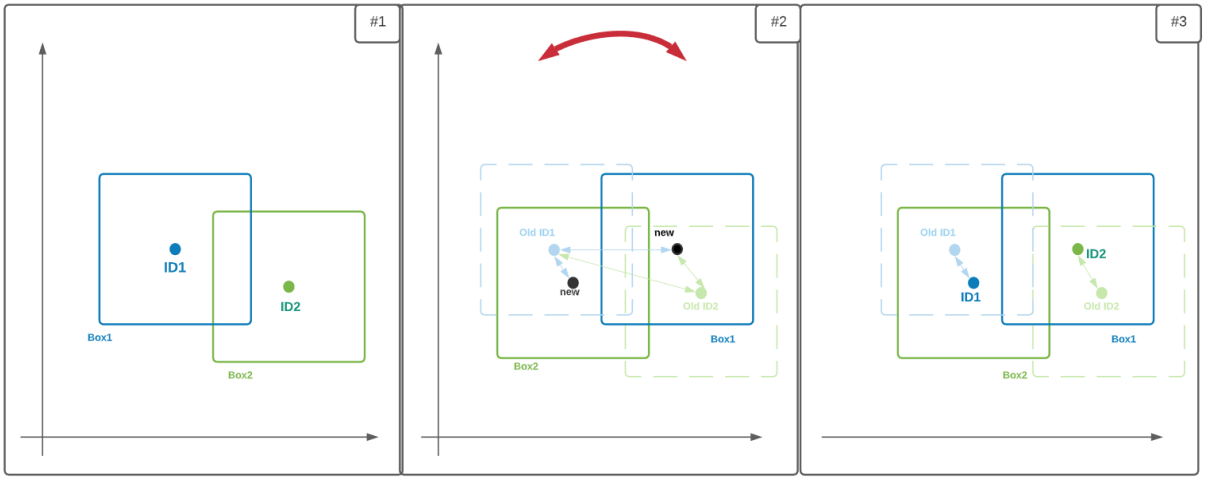

Figure 4.2: *Example of incorrect centroid assignment*

4.1.2 Behaviour description (identity switch)

The IS case, on a practical level, can be seen through the tests to behave according to this series of events:

1) The two centroids x,y are progressively approached until they reach a minimum distance selected with the variable *limit_swapping*.

2) Once this minimum distance is exceeded, the two objects overlap and only a new object with its center will be detected, and therefore one of the two IDs will be pending for *n* frames.

3) After *n* frames, the assignment of the IDs shall be random.

4.2 Follow Me Module

The Follow Me module allows any person detected in a certain frame to activate and, vice versa, to voluntarily deactivate the monitoring function of the system. The idea of this algorithm is to detect, within a certain number of frames the movement of all people and, based on a certain movement, of the which is repeated during a predetermined time interval, allow the tracking function to be activated and deactivated.

4.2.1 Behaviour description (Follow Me)

It was decided to detect as movement the opening and closing of the arms, this movement is related to the bounding boxes, more specifically to the size in the x-axis. The size of the last and the current bounding box is then checked for a fixed number of consecutive frames, if they are one time larger and one time smaller with respect to the saved dimension for a predefined number of times the target is set (or released).

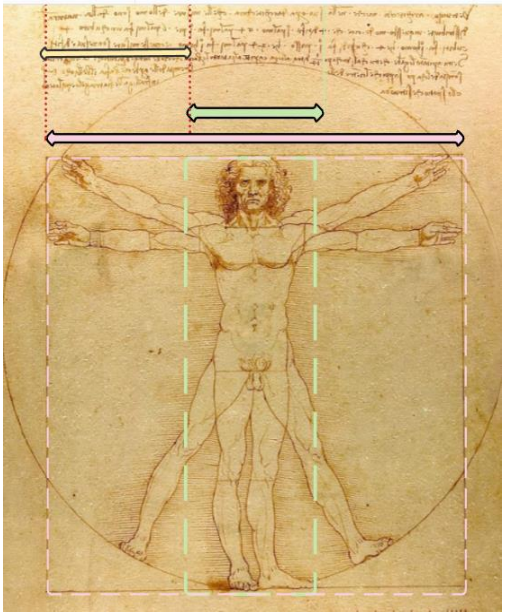

Figure 4.3: *Example of bounding box width variation*

Through the tests, and therefore knowing the frame rate of the system, we calculate the time in which the object's box has changed by a certain percentage of width, compared to the previous one.
If *dim1* and *dim2* are the box widths at moment one and moment two respectively, then the counter is increased according to this condition:

$$dim2 > dim1 * K$$

if *dim2* is larger than *dim1*.

$$dim2 * K < dim1$$

if *dim1* is larger than *dim2*.

K is the variable that indicates how much the box size should vary before increasing the counter. Through the tests, it can be seen that the net is not able to accurately capture the image in the opening situation of the arms, so it is decided to establish the variable K with a lower value than it should have in reality to respect the real size of the person. This will result in less sensitivity to unwanted variations but will allow the subject to activate the tracking.

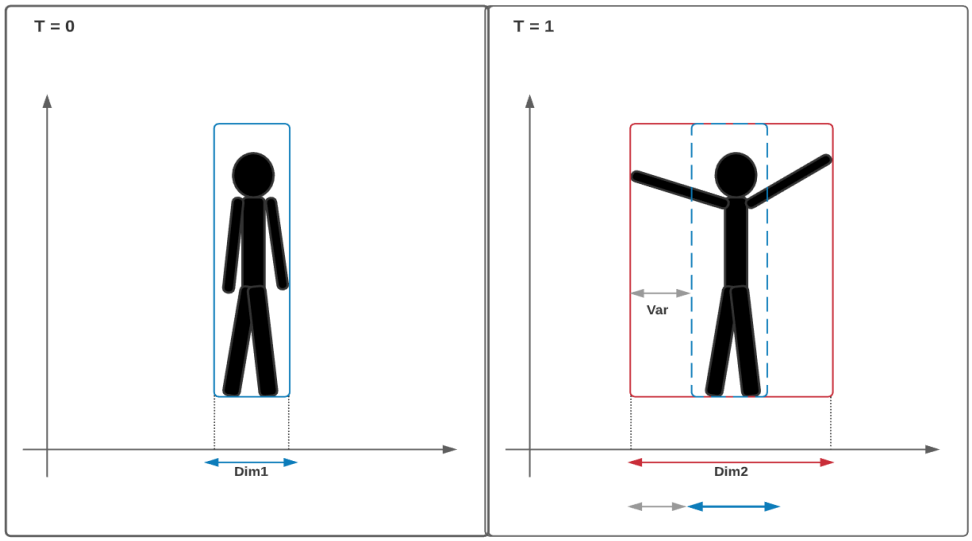

Figure 4.4: *Example 2 of bounding box width variation*

The limit value of the size change (for which the counter is increased) is directly related to the size of the bounding box. This is necessary for prevent the counter from rising for small variations.

4.2.2 Parameters of the Follow Me Module

- ***max_reset_counter*** *:* parameter that indicates when a counter related to the activation of the tracking must be reset.

- ***max_start_or_stop*** *:* parameter that compares to the counter associated to each ID to understand if a particular user has requested to be tracked or not.

- ***max_sleep***: parameter that indicates for how many seconds the algorithm should not detect any movement.

- ***count_sleep:*** parameter that is incremented to be compared with *max_sleep.*

4.2.3 Implementation of the algorithm

4.2.3.1 Follow Me function

1. In the first step, the algorithm is called by the Object Detection module, that for each frame it passes in input to the algorithm all the returned centroids by the Tracking algorithm and all the bounding boxes associated with people detected in the current frame.

2. If the length of the bounding boxes passed as input is zero, the centroids are registered as missing.

3. Calculate the centre of all bounding boxes in input and compare these centroids with those returned by the Tracking algorithm, to associate each centroid with the correct bounding box.

    a. In case there is a coincidence between the centroids, if the centroid does not was saved is recorded. Otherwise, if it already exists, take the bounding box saved from the previous frame and check its size with the current one by incrementing a counter (only if respectively in the frame $t=n$ once the size is larger than the saved size multiplied by a certain sensitivity parameter and in the table $t=n+1$ if the current size multiplied by the sensitivity is less than the previous one (i.e. the size current table $t=n$).

    b. In case there is no coincidence between the centroids, the new centroid is recorded in case it is not already saved and related to a new object or establish that centroid as disappear.

4. In the last step all updated counters are checked and if there are any that has a counter above a certain limit parameter is returned this objective.

4.2.3.2 Unfollow Me function

1. In the first step, the algorithm is called by the Object Detection module, that for each frame it passes in input to the algorithm all the returned centroids by the tracking algorithm, all the bounding boxes associated with the people detected in the current frame and the target ID.

2. If the length of the bounding boxes passed as input zero. the centroids are registered as missing.

3. Calculate the centre of all bounding boxes in input and compare these centroids with those returned by the Tracking algorithm, to associate the bounding box correct.

    i. In case there is a match between the centroids if the centroid is not was saved an error is returned because the target is not activated, from another way if it already exists takes the saved bounding box from the previous frame and checks its size with the current one by increasing a counter only if respectively in the frame *t=n* once the size is larger than the size stored multiplied by a certain sensitivity parameter and in the table *t=n+1* if the current size multiplied by the sensitivity is less than the previous one (i.e. the current size of the table *t=n*).

4. In the last step the updated counter of the target is checked and if it is higher than a certain limit parameter the target is returned with the value -1.

    i. Thanks to the implementation of this algorithm, it is possible to precisely parameterize the opening and closing movement of any person's arms.

    ii. In addition, to prevent a user from opening and closing the arms more times than necessary and activating and deactivating the AI, the *sleep_time* which allows us to overcome this problem by pausing on algorithm for a stable number of frames.

4.2.4 Algorithm limitations

The limitations of the algorithm are:

- If two or more people wanted to "call" the system at the same time, the first person with the meter above the threshold would be targeted, which would depends on the order in which the object detection algorithm returns the bounding boxes.

- If two or more people at the same time wanted to "call" the system and were too close together, object detection for some paintings could create a single bounding box or return the wrong size of one of the two without allow the meter to increase.

4.3 GUI

It was decided to structure the GUI in 4 windows organized as follows:
The first one has two buttons that refer respectively to the window that manages the static functionalities and it handles the dynamic functionalities.
Finally, a window in which the user can enter his chosen class.

4.3.1 First window

In the first window there are two buttons:

- **Dynamics Modes** : calls the Dynamic_Modes() function that handles the GUI of all dynamic functionalities.

- **Static Modes** : calls the Static_Modes() function that handles the GUI of all static functionalities.

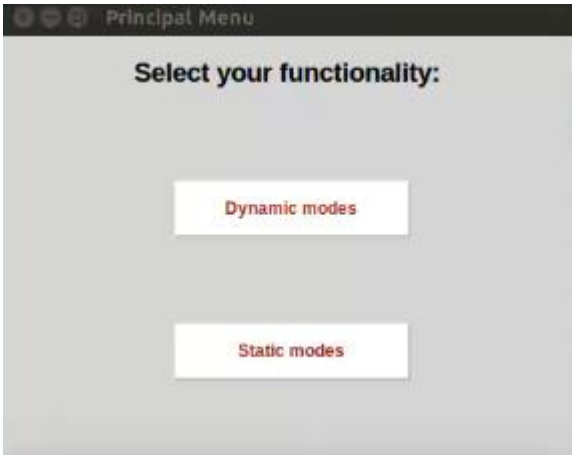

Figure 4.5: *First window*

4.3.2 Second Window (Static Modes)

In the second window there are three buttons:

- **Most Central Object**: it calls the *Central_Mode()* function that handles the MOB functionality.

- **Farthest Object**: it calls the *Farthest_Mode()* function which handles the FM functionality

- **Select detection class**: it calls the *Select_Class()* function that handles the GUI so that the user can choose their own object class..

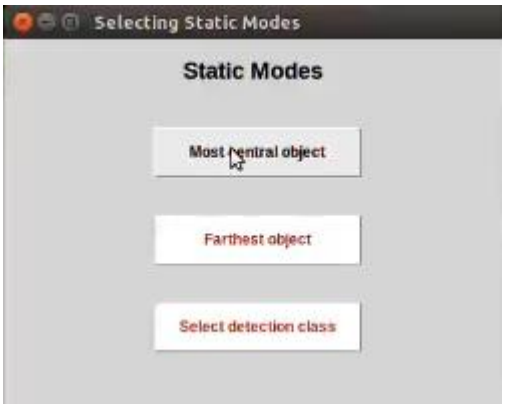

Figura 4.6: *Second window*

4.3.3 Third window (Dynamic Mode)

In the second window there are three buttons:

- **Follow Me:** it calls the Follow_Me() function that handles the Follow Me functionality.

- **Reach Target:** it calls the Select_Class_Dyn() function that handles the Follow Me functionality.

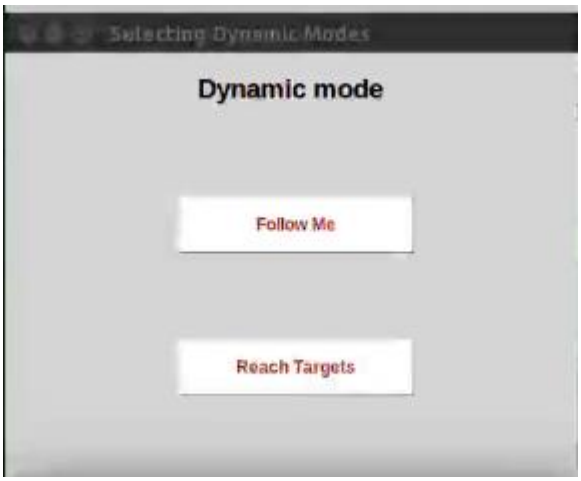

Figura 4.7: *Third window*

4.3.4 Fourth window (Class Selection)

In the second window there is an input entry to call the function relative to the Reach Target if the tracking is dynamic or Select Class if it is static tracking.

- **Input:** it calls the *reach_target()*/*personalized_mode()* function that handles the Reach Target/Select Class functionality.

- **Get Target:** input box where the user can put his chosen class.

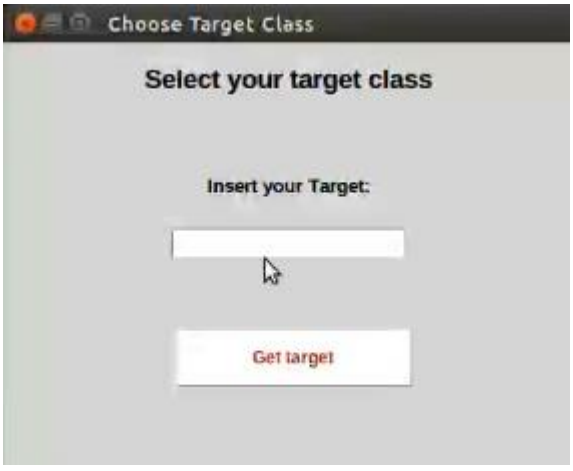

Figure 4.8: *Fourth window*

5

Test and results

Once the various improvements introduced have been explained in detail, as well as the objectives to be achieved, a testing process should be carried out to determine whether these objectives have been achieved. In this way, it could be determined what the limits are and what future work should be done to improve them.
In turn, within each of these sections, the study of specific situations of interest is carried out.
The degree of complexity of the tests will increase due to the subsequent aggregation of different modules and, therefore, of the different variables present in them.

5.1 Tests

5.1.1 Centroid Tracking algorithm experimentation

Table 5.1: Centroid Tracking algorithm experimentation

Title: Centroid tracking algorithm experimentation
Description: The algorithm is tested for correct operation and its ability to create identities and track them in frames. It is desired to study the behaviour of the algorithm in the presence of different objects, both in static and moving positions.
Video: Tracking Testing **Seconds:** 0:00 - 1:08 **Link:**https://www.youtube.com/watch?v=T7RIFbr1Ybs&ab_channel=ProyectoIntegradorCali%C3%B2-Forese

Place: living room **Light**: artificial	**Camera position**: 27 above the ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**: between 1.5 and 4 m
max_disapp_frames: 30 frames	**Number of objects**: 5 **Object class**: person

In the first frame the algorithm can correctly assign five different centroids created in order of detection. The stativity of the subjects allows the algorithm to have a stable behaviour; this is lost when the subjects start to move. In fact, in the second image ID5 and ID3 are pending and they do not have any box associated with them.
In the third picture due to an IS, the subject with ID5 is now assigned ID3, which was located on the opposite side of ID5 in the first frame.

Figure 5.1: *Centroid tracking algorithm experimentation*

5.1.2 Identity switch experimentation

Table 5.2: Identity switches experimentation

Title: Identity switches experimentation	
Description: The case of a possible change of identity is analyzed. We study whether the network is able to recognize this event and then report it correctly. We look for the minimum distance considered to establish the overlap and the number of frames in which one of the identifications is pending to assert the change.	
Place: living room	**Camera position**: 27 above the

Light: artificial	ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**: between 2 and 2.5 m
limit_swapping: 200 px **abort_limit**: 4 frames	**Number of objects**: 2 **Object class**: person

The experiment shows an example of control over the identity switch event. It can be seen, in *Figure 5.2*, how in the second frame the possibility of a switch occurring in the following frames is correctly signaled, as the centroids approach the 200px boundary. In the next frame, one of the two centroids in question remains pending, thus without an associated object for more than 8 frames (a little more than a second). This means that the situation of a possible change of identity has occurred.
In the third picture, in addition, one can see the creation of a false identification: the superimposition of the two subjects in the picture has created a third ID4, due to an error in the detection of the person class objects in the picture.

Figure 5.2: *Identity switch experimentation*

5.1.3 Follow Me mode experimentation

5.1.3.1 Follow Me mode experimentation *(1)*

Table 5.3: Follow Me mode experimentation (1)

Title: Follow Me mode experimentation (1)
Description: Selection, through a predetermined movement of the body of a person or object class, of the target. We experience the ability of the network to track the object in several frames, even when the target moves out of the frame. Deselection of the target by the same movement as before.

Video: Follow Me Testing **Seconds:** 0:06 - 0:18 (Activation) 0:40 - 0:50 (Disact.) **Link:**https://www.youtube.com/watch?v=-i_stxDzrNo&ab_channel=ProyectoIntegradorCali%C3%B2-Forese	
Place: living room **Light**: artificial	**Camera position**: 27 above the ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**: between 1.5 and 4 m
max_reset_counter: 4 frames **max_start_or_stop**: 4 repetitions **max_sleep**: 4 frames **sensitivity**: 1.30	**Número de objetos**: 2 **Clase de los objetos**: person

Figure 5.3 shows how the follow-me mode activation function works correctly. The enable sequence, that ID0 performs to allow it to self-stabilize as a target, is highlighted. In the fourth picture the arrow shows the white line that determines that the tracking has been triggered.
In Figure 5.4, on the other hand, the target deactivation sequence is correctly implemented. Here also the yellow arrow in the last frame indicates the correct disabling of the target tracking mode.
The two sequences show as main difference the ability of the network to detect the subject on the left of the image; in fact, the subject is not detected in the first sequence, but in the second sequence, when approaching the camera. It is correctly detected and identified.

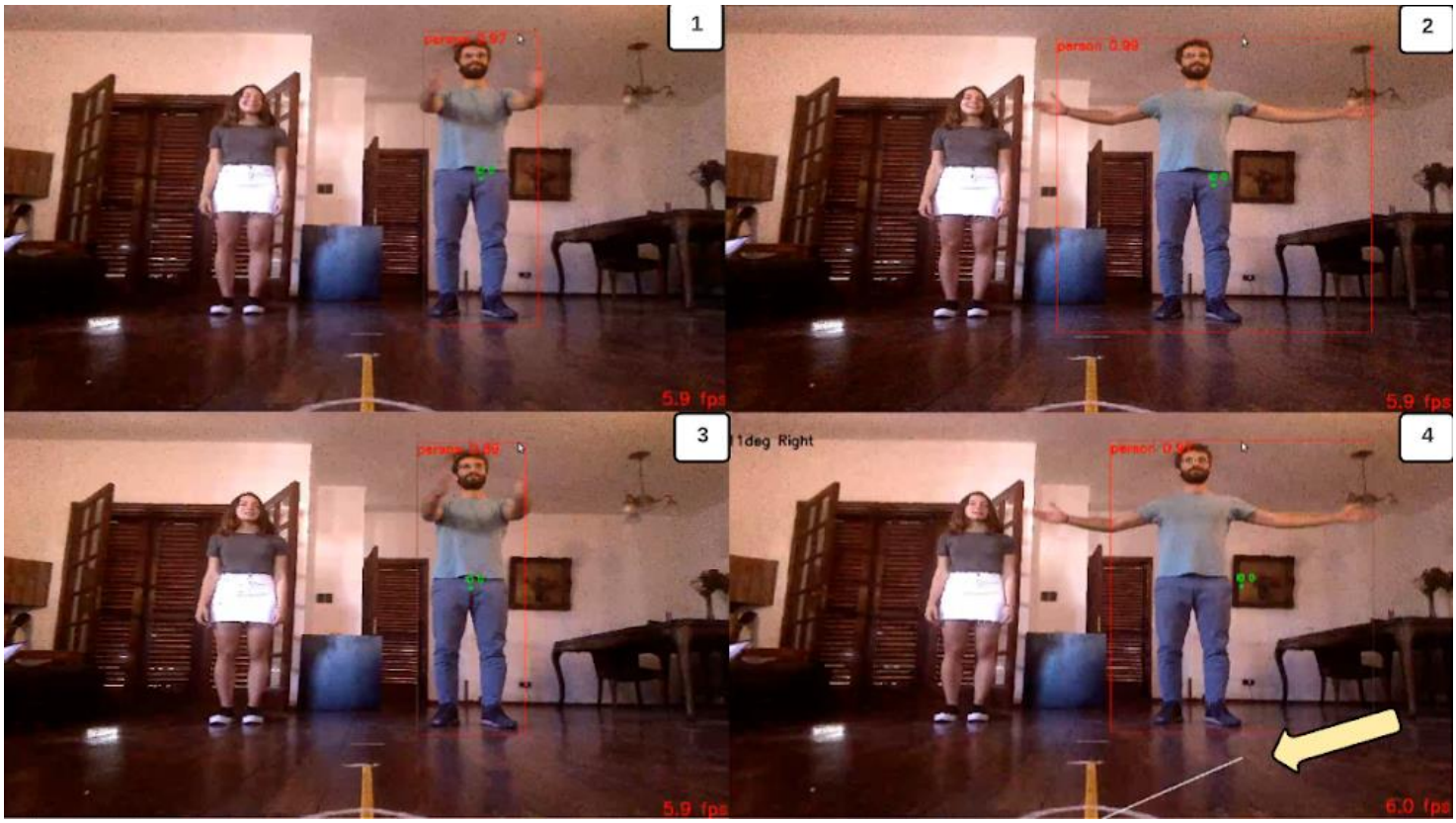

Figure 5.3: *Functionality activation*

Figure 5.4: *Functionality deactivation*

5.1.3.2 Follow Me mode experimentation *(2)*

Table 5.4: Follow Me experimentation (2)

Title: Follow me experimentation (2)
Description: Selection, through a predetermined movement of the body of a person or object class, of the target. The ability of the network to track the object in several frames is experimented. Deselection of the target by the same movement

as before. The case where the sensitivity of the motion change control is low is studied..	
Place: living room **Light**: artificial	**Camera position**: 27 above the ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**: between 2 and 3 m
max_reset_counter: 4 frames **max_start_or_stop**: 4 repetitions **max_sleep**: 4 frames **sensitivity**: 1.15	**Number of objects**: 1 **Object class**: person

By setting the limit of the variation of the Follow Me activation movement to a low value, small variations in the size of the box, due to the simple movement of the subject, incorrectly activate the function. Therefore, the expected behavior actually occurs, both on activation (*Figure 5.5*) and deactivation (*Figure 5.6*), randomly.

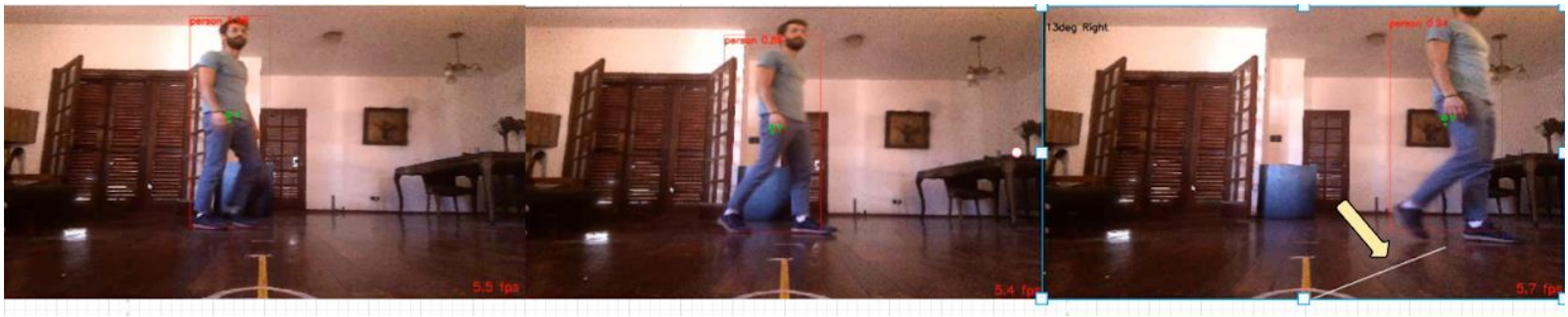

Figure 5.5: *Activation of the system with low sensitivity*

Figure 5.6: *Deactivation of the system with low sensitivity*

5.1.3.3 Follow Me mode experimentation *(3)*

Table 5.5: Follow Me mode experimentation (3)

Title: Follow Me mode experimentation (3)	
Description: Selection, through a predetermined movement of the body of a person or object class, of the target. The ability of the network to track the object in several frames is experimented. Deselection of the target by the same movement as before. We examine the case in which the variable that determines the number of frames before deleting the Id is too low.	
Place: living room **Light**: artificial	**Camera position**: 27 above the ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**: between 1.5 and 4 m
max_reset_counter:4 frames **max_start_or_stop**:4 repetitions **max_sleep**: 4 frames **sensitivity**: 1.30 **max_disapp_frames**:8 frames	**Number of objects**: 2 **Object class**: person

The experiment shows how the *max_disapp_limit* variable set too low affects the correct functioning of the Follow Me module. The network, in fact, erases the target centroid too quickly, so that it is identified as lost event though the target was only out of the image for a few moments.

Figure 5.7: *Follow Me experimentation with a low max_disapp_frames variable*

5.1.4 Reach Target mode experimentation

Table 5.6: Reach Target mode experimentation

Title: Reach Target mode experimentation	
Description: The case in which the interface object class is selected is analyzed. After checking the correct selection of the class, we study the case in which there are objects of the selected class and the opposite case, evaluating the correct emission of messages over the network. Finally the correct message is evaluated when it reaches the target and therefore the end of the algorithm.	
Video: Static functionalities testing **Seconds:** **Link:**https://www.youtube.com/watch?v=o9XpidIo7RU&ab_channel=ProyectoIntegradorCali%C3%B2-Forese	
Place: living room **Light**: artificial	**Camera position**: 27 above the ground **Angle of view**: 62.2 x 48.8 deg **Objects position with respect to the camera**:

	between 0.8 and 2 m
	Number of objects: 1 **Object class**: person/chair

The experiment of pre-selecting the class and therefore reaching the object relative to the selected class is successful. It can be seen in Figures *Figure 5.8* and *Figure 5.9,* how the success message is displayed correctly.
Also, it can be seen how, in the second sequence, the chair is first detected and then assigned as a target to be reached.

Figure 5.8: *Reach Target experimentation with a person*

Figure 5.9: *Reach Target experimentation with a chair*

5.2 Results

5.2.1 Best parameters based on the experimentation

As shown in the tests, both functional and non-functional requirements were met, as well as the objectives set at the beginning of the project.

All the best system parameters that allowed an acceptable stability condition and error tolerance to be achieved are listed in detail.

- **max_disapp_frames =** 30

Running the algorithm at an average of 8 fps, setting this parameter to 80 means that the AI waits 10 seconds before deleting a centroid.
If this parameter were higher it would have problems in swapping the id because centroids would remain pending and overlap with others, and vice versa if it were too low the system would not be able to resume the target id once lost.

- **sens =** 1.3

By setting the sensitivity in the "Follow Me" mode to 1.3 it is possible to have a good accuracy in the movement of the accuracy in the movement of any person's hands.
Any higher would risk inadvertently activating the system, whereas the user would have to try to activate the system.

- **max_reset_count** = 4

The number 4 means that if the target is lost for more than 4 frames its counter to activate the movement is reset to zero.
Again, thanks to this parameter, if a target is pending when it reappears it must make the move again rather than risk being activated on the first move.

- **sleep_time** = 24

Thanks to this parameter the algorithm avoids detecting movements for 3 seconds after the target has been set. In this way there is no risk that if the user makes more consecutive movements than necessary with his arms once the Follow Me is activated; it is not immediately deactivated.

- **limit_swap** = 200

By setting the distance limit for the switch to 200 pixels, the swap can be recognised with an acceptable margin of error.

- **abort_limit_count** = 8

By setting the abort mission counter for the swap to one second you can have a very good reaction to the switch identity problem.

5.2.2 Requirements Traceability Matrix

As shown in the tests and in the table, both functional and non-functional requirements were met, as well as the objectives established at the beginning of the project.

Tabla 5.7 : Requirements traceability matrix

ID Requirement	***Requirement declaration***	***Id Test***	Status
R01	The system shall be able to detect different kinds of objects.	All tests	Past
R02	The system shall parameterize the viewing space to assign a position to the object.	All tests	Past
R03	The system shall be able to select only one class of objects.	5.1.4	Past
R04	The system shall provide an interface for communication with the AI.	All tests	Past
R05	The system shall assign different IDs to each object of the selected class.	All tests	Past
R06	The system shall dynamically select and track the target object.	5.1.3	Past
R07	The system shall reach a stationary or moving object.	5.1.4	Past
R08	The system shall allow an object to activate and deactivate the tracking function by a specific body movement.	5.1.3.1, 5.1.3.2	Past
R09	Python 3.6 will be used as programming language.	All tests	Past
R10	The system will have to achieve low latency between image production and algorithm output.	All tests	Past
R11	The system shall apply the safeguards for borderline cases in tracking	5.1.2	Past

5.2.3 Limit parameters for Non-Functional Requirements

- **Object detection capabily:**

  ❖ The IA can perfectly detect objects up to 3 meters distance from the camera.

  ❖ Beyond 3 meters it can detect motion every 8 frames on average.

  ❖ Beyond 4 meters from the tests, it can be seen the system detects motion on average every 12 frames, but if the target remains stationary it is lost.

- **Limiting Parameters for Non-Functional Requirements:**

This non-functional requirement is met thanks to the choice of the Mobilenet v2 SSD, which achieves 8fps which guarantees a real-time implementation of the system.

The tests and the explanation of the work can be found at the following page:

https://www.youtube.com/channel/UCU2Crx70ouY-IAGcFeVRv6g?view_as=subscriber

6

Conclusion

This paper suggests that it is possible, using a neural network, to implement a series of functionalities that make an autonomous system independent. The initial objective of the project was to allow a ground vehicle, equipped with a series of basic and essential components, to perceive its surroundings and, through a graphical interface, allow the user to select between different modes of use of the system.

It was therefore necessary to implement a tracking algorithm that is able to assign an identification to each object of interest and that subsequently allows one of these to be selected as a target. It was decided to implement the centroid algorithm, which is light and versatile and considered ideal for the hardware used. This has made it easier to detect and resolve a larger number of events, such as the temporal output of the image target, and to further extend the dynamism of the system.

Subsequently, the study continued with the development of an additional module of Follow Me, a technology frequently used in various autonomous devices, which allows the user without using the GUI to capture the interest of the vehicle, so as to allow the latter to follow it until the deactivation of this mode.
Subsequently, the problem of identity switches, a strong limitation of all multiple objects tracking algorithms, has been addressed, creating a functionality to search for this problem and then reporting it to the system.
Finally, the paper aims to create a graphical interface that allows the user to interact with the system, in a simple and fast way, selecting between different modes of use.

Several problems were encountered during the testing phase, mainly for two reasons: the low speed of the algorithm, due in part to the limited number of boards used, and

the inadequacy of the test site, which does not guarantee adequate exposure of the objects under test.
All the functional and non-functional requirements set as an objective at the beginning of the project were met by means of precise tests that made it necessary to study the various parameters used up to the best configuration explained in the Testing section.

6.1 Recommendation for further research

The use cases proposed in the objective of the work and implemented in this project serve as an example to create a basis for extending the functionality of user use and behavior of the autonomous system.
As far as tracking is concerned, the next step is to implement different advanced visual tracking algorithms, which allow a better identification of the different objects in the image and thus a better recognition of these especially in borderline cases. An example of an algorithm to be implemented would be MOSSE. The MOSSE filter is a stable correlation filter which can be initialized on a single frame of a video. The MOSSE filter adapts with the changes in the appearance of the object while tracking. Tracking using the MOSSE filter is not dependent on changes in lighting, non-rigid transformations, pose, and scale. [7]

Among the improvements that can be added to the project, the improvement of hardware and simulation conditions certainly plays an important role. The implementation of the intelligence in the Jatson-Nano is adequate for the case study, but one must consider the fact that adding new features will slow down the algorithm, considerably decreasing the fps value.
To achieve an environment with sufficient power and therefore computing speed, we recommend using an NVIDIA Jetson TX2, which, compared to the Nano, has twice as many GPUs (256 instead of 128 cores) and twice as much memory (8 GB).
Even a camera with a better resolution would improve system performance, especially considering that some of the problems in the test phase relate to poor image quality.

APPENDIX

A

Selection of the camera

The camera market is very wide and varied in terms of features and prices. For the purposes of the implementation of this IP, the Pi Camera Module, from the manufacturers of Raspberry, was chosen as the camera of choice. There are two versions of this camera, whose characteristics are compared in table A.1.1 . The Pi Camera Module V2 (see figure 3.5.1) was chosen because it has better features for the same price.

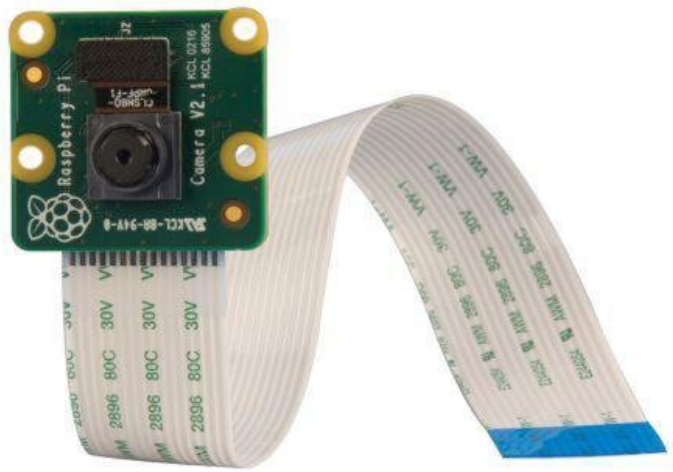

Figure A.1: *Pi Camera v2*

Table A.1.1: *Hardware specifications*

	Camera Module v1	Camera Module v2	HQ Camera
Net price	$25	$25	$50
Size	Around 25 × 24 × 9 mm		38 x 38 x 18.4mm (excluding lens)
Weight	3g	3g	
Still resolution	5 Megapixels	8 Megapixels	12.3 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available	V4L2 driver available	V4L2 driver available
C programming API	OpenMAX IL and others available	OpenMAX IL and others available	
Sensor	OmniVision OV5647	Sony IMX219	Sony IMX477
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels	4056 x 3040 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm diagonal)	6.287mm x 4.712 mm (7.9mm diagonal)
Pixel size	1.4 µm × 1.4 µm	1.12 µm x 1.12 µm	1.55 µm x 1.55 µm
Optical size	1/4"	1/4"	
Full-frame SLR lens equivalent	35 mm		
S/N ratio	36 dB		
Dynamic range	67 dB @ 8x gain		
Sensitivity	680 mV/lux-sec		
Dark current	16 mV/sec @ 60 C		
Well capacity	4.3 Ke-		
Fixed focus	1 m to infinity		N/A
Focal length	3.60 mm +/- 0.01	3.04 mm	Depends on lens
Horizontal field of view	53.50 +/- 0.13 degrees	62.2 degrees	Depends on lens
Vertical field of view	41.41 +/- 0.11 degrees	48.8 degrees	Depends on lens
Focal ratio (F-Stop)	2.9	2.0	Depends on lens

B

Why choose Python for ML?

For starters, a language with good machine learning libraries is needed. It also needs good runtime performance, good tool support, a large community of programmers and a healthy ecosystem of support packages.

Figure A.2: *The best programming languages for ML*

Python is known for its concise and easy-to-read code, earning great respect for its ease of use and simplicity. The same cannot be said for C, which is considered a lower-level code, meaning it is easier for the computer to read (hence its higher performance), but harder for humans to read.

Given the complexity of machine learning algorithms, the less a developer has to worry about the complexities of coding, the more they can focus on what really matters: finding solutions to problems and achieving project goals.

Python's simple syntax also allows for a more natural and intuitive ETL (Extract, Transform, Load) process, and means it is faster for development compared to C++, allowing developers to quickly test machine learning algorithms without having to implement them.

Python is an open source programming language and is supported by many high-quality resources and documentation. It also has a large and active community of developers ready to provide advice and assistance at all stages of the development process.

C

Jetson Nano installation

For installation requires:

- A microSD card (16GB UHS-1 minimum)
- USB keyboard and mouse
- Computer screen (either HDMI or DP)
- Micro-USB power supply (5V⎓2A)

The Jetson Nano Developer Kit uses a microSD card as the boot device and for primary storage. It is important to have a card that is fast and large enough for our projects; the recommended minimum is a 16GB UHS-1 card.

The first thing to do is to write the operating system image to the microSD card.

For this we use the Jetson Nano Developer Kit SD Card Image.

Once the operating system has been written, the microSD card can be placed on the Jetson Nano[8].

It was decided to use Ubuntu 18.04 as the operating system, developing in Python (see annexes A.3 to understand the choice of Python) with CUDA version 10.0.

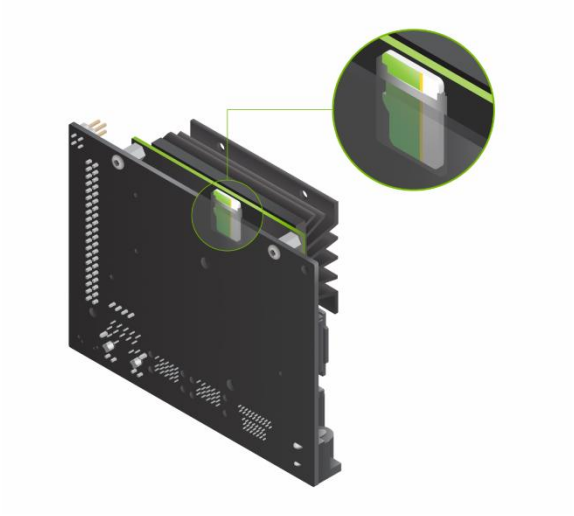

Figure A.3: *Placing the microSD in the Jetson Nano*

Using the microUSB with a power bank or 5V⎓2A power supply, when booting, it is advised to set the power mode of the Jetson Nano to 5W so that it does not crash, to do this, open a terminal and type:

- sudo nvpmodel -m 1

To return to 10W power mode (default):

- sudo nvpmodel -m 0

Given the number of peripherals connected (monitor, camera, ..) the 10W setting causes the system to crash because it cannot guarantee the necessary power.

To reduce memory pressure (and crashes), it is a good idea to set up a 6GB swap partition (Nano only has 4GB RAM).

- git clone https://github.com/JetsonHacksNano/installSwapfile
- cd installSwapfile
- chmod 777 installSwapfile.sh
- ./intallSwapfile.sh

References

- [1] “Metodo Incremental” https://obsbusiness.school/es/blog-project-management/metodologias-agiles/caracteristicas-y-fases-del-modeloincremental#:~:text=El%20modelo%20incremental%20de%20gesti%C3%B3n,por%20el%20cliente%20o%20destinatario (October 2020)

- [2] https://missinglink.ai/guides/computer-vision/object-tracking-deep-learning/

- [3] “Tracking Object” https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/ (October 2020)

- [4] “Comparacion Modules Intel” https://developer.nvidia.com/embedded/jetson-modules (October 2020)

- [5] “Bounding Box” https://d2l.ai/chapter_computer-vision/bounding-box.html (ultima visita septiembre 2020)

- [6] https://github.com/prat96/Centroid-Object-Tracking (Sept 2020)

- [7] “Tutorial on Minimum Output Sum of Squared Error Filter” https://mountainscholar.org/bitstream/handle/10217/173486/Sidhu_colostate_0053N_13486.pdf

- [8] ”Jetson Nano Developer Kit” https://developer.nvidia.com/embedded/jetson-nano-developer-kit (2020)

Image References

- **Figure 1.1:** *Abstraction of the tracking issue(https://www.move-lab.com/blog/tracking-things-in-object-detection-videos)*
- **Figure 1.2:** *NVIDIA Jetson Nano (*https://developer.nvidia.com/embedded/jetson-nano-developer-kit)
- **Figure 3.1:** *Bounding boxes example (*https://d2l.ai/chapter_computer-vision/bounding-box.html)
- **Figure A.1:** *Pi Camera v2 (*https://www.amazon.co.uk/Raspberry-Pi-Camera-Module-Filter/dp/B01ER4FA9U*)*
- **Figure A.2:** *The best programming languages for ML(https://twitter.com/gp_pulipaka/status/1108360695479857152)*
- **Figure A.3:** *Placing the microSD in the Jetson Nano((*https://developer.nvidia.com/embedded/jetson-nano-developer-kit)