Simulation of Autonomous Flight of UAVs for Search and Rescue Missions

Trabajo de Fin de Grado
Curso 2021–2022

Autor
Alejandro-Daniel Díaz Román

Director
Sandra Catalán Pallarés
Sergio Bernabé García

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Simulation of Autonomous Flight of UAVs for Search and Rescue Missions

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Arquitectura de Computadores y Automática

Autor
Alejandro-Daniel Díaz Román

Director
Sandra Catalán Pallarés
Sergio Bernabé García

Convocatoria: *Junio 2022*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

30 de mayo de 2022

Dedicatoria

A mis padres, por su apoyo y cariño infinito y por darme la inspiración y fuerza para ser quien soy.

A mi abuela, Omi, si alguien se merece esto eres tu, por tu garra y tu espíritu incansable.

A mis amigos, que no sé que sería de mi sin vosotros. A Pablo, porque Madrid sin ti no es tan Madrid. A Paula, por no parar de sorprenderme e inspirarme, navegaremos juntos hacia el abismo de Finisterre. A Nacho, por entenderme mejor que nadie, nos reímos solos, nos reímos con ganas, no nos da la gana de ponernos serios.

Y a todos los grupos que crean la banda sonora de mi vida y hacen que todo tenga algo más de sentido. A los Foo Fighters, Niña Polaca, Ginebras, Amaral, Estopa, Delaporte, ...

Agradecimientos

A todos los buenos profesores que nos enseñan a pensar, a ser críticos y a aprender, que nos transmiten no solo conocimientos si no más importante todavía, sabiduría. A todos ellos, mi eterno agradecimiento, por moldear el mundo a un lugar mejor a pesar de lo complicado que pueda llegar a ser. A Don Jesús, Mª Eugenia, Altamira, Lola, Pepe, Rafa, Carlos, Gonzalo, Jesús, Sandra.

A Sandra Catalán, por lanzarse a la piscina y dirigir este TFG sin ninguno de los dos saber prácticamente dodne nos estábamos metiendo y saber ver dónde yo no sabía encontrar el camino.

A Sergio Bernabé, por ser de inestimable ayuda dirigiendo el TFG y aportando soluciones y propuestas que han permitido poder realizar el mejor trabajo posible.

Y por último a todas aquellas personas que contribuyen al software libre, que prácticamente hacen que el mundo siga girando sin esperar nada a cambio.

Resumen

Uno nunca se da cuenta de lo que ha hecho; solo puede ver lo que queda por hacer.

Marie Curie

Los vehículos autónomos no tripulados (VANT) comúnmente conocidos como drones, han visto una gran evolución en los últimos años, principalmente orientados hacia un uso comercial, debido a sus características, ya que gozan de gran maniobrabilidad, tienen un coste asequible para cualquier persona y son relativamente fáciles de operar (salvo situaciones meteorológicas muy adversas). Como veremos a lo largo de este TFG (Trabajo Fin de Grado) también cuentan con herramientas desarrolladas para automatizar su vuelo, lo cual da pie a multitud de aplicaciones que antes eran o inviables o muy complicadas de desplegar.

Es por ello que están reinventando tareas como el envío de paquetes, asistencia en zonas catastróficas, videografía y fotografía aéreas, construcción, o en misiones de búsqueda y rescate. Siendo este último punto en el que se centrará el TFG, desarrollando un simulador para que múltiples drones sean capaces de encontrar una cabeza de ganado en un área determinada.

Palabras clave

UAV, Multi UAV, Visión por computador, Búsqueda y rescate, Simulación, Generación de rutas, FANET, Apache Kafka

Abstract

One never notices what has been done;
one can only see what remains to be
done.

Marie Curie

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have seen a great evolution in recent years, mainly oriented towards commercial use, due to their characteristics, since they are highly maneuverable, have an affordable cost for anyone and are relatively easy to operate (except in very adverse weather situations). As we will see throughout this work, tools have been developed to automate their flight, which gives rise to a multitude of applications that were previously unfeasible or very complicated to deploy.

This is why UAVs are reinventing tasks such as package delivery, assistance in catastrophic areas, aerial videography and photography, construction or search and rescue missions. The latter being the focus of this work, developing a simulator where multiple UAVs try to find a head of cattle in a determined area.

Keywords

UAV, Multi UAV, Computer Vision, Search and Rescue, Simulation, Path planning, FANET, Apache Kafka

Contents

Chapter 1

Introduction

UAVs have been under development for decades, mainly for military purposes. It was not until the beginning of the 21^{st} century when the development of projects aimed at building home-made drones started, it was then when the small community of hobbyists gathered to develop flight controllers capable of piloting various types of drones.

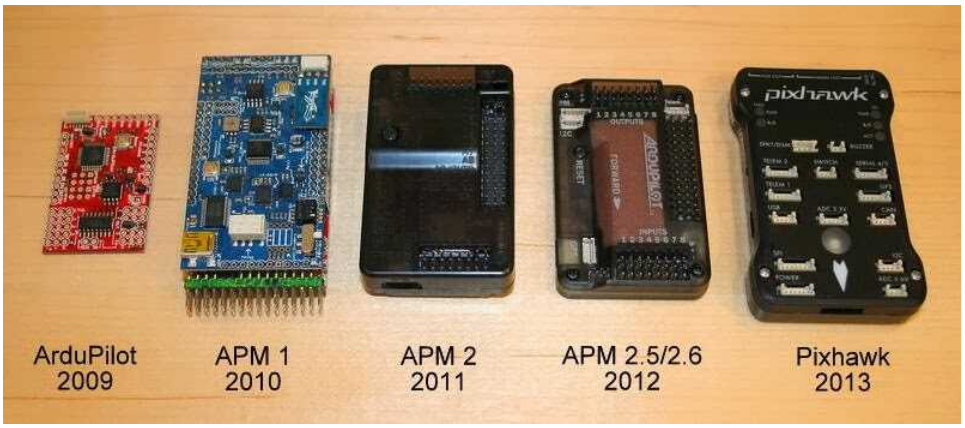

Figure 1.1: Evolution of different Ardupilot controllers.

In 2007 [Ardupilot contributors (2022a)] began the development of the first version of Ardupilot, a flight controller that currently supports a multitude of different vehicles like submarines, drones or traditional helicopter drones. Ardupilot started with the latter type of vehicles and over the last 15 years has grown in complexity to become one of the Open Source references for developing autopilots. Jointly with teams from PX4 and 3DRobotics, they developed Pixhawk which represented a breakthrough in autonomous pilots.

One of the biggest events was in 2014 when the Canberra UAV team took first place in the UAV Outback Challenge [Wikipedia contributors (2022)], a competition consisting of delivering a bottle of water to a person (represented by a mannequin) in the middle of a 24 km^2 area, as the first ever winners of the competition received a prize of 50,000 USD.

Another important mark was in 2015 when the 16-person team of Advanced Robotic Systems Engineering Laboratory (ARSENL) managed to fly 50 aircraft simultaneously with only two pilots supervising the operation.

With all this we come to today, where the future of drone use is already present in jobs such as inspection or maintenance, aerial photography and videography, critical utility transportation, assisting in weather prediction and a long etcetera. The near future of drones will probably involve the so-called FANETs (Flying Ad-hoc Networks), which consist of independent networks of other devices that communicate with each other, either to transmit information or to assist in multiple tasks.

Some of these points will be discussed in the state of the art, while others will also be discussed in the future work section, since this is a field where great advances are still to be made.

1.1. Motivation

The motivation for this work comes mainly from the curiosity of the drone industry and the impact they can have in important sectors such as agriculture or logistics.

With almost no prior knowledge, the research was started to find out the current state of the art of UAVs. As I was reading about autonomous drone flight and the possibilities they allow we decided to center the work of the TFG (Trabajo Fin de Grado) around the autonomous search of a target in a given area. This decision was quite appropriate as it covers several areas that are really interesting, such as computer vision, route generation or the different aspects of a simulation.

And it is in this last aspect where I have found more virtues in this project, since the simulation, not only serves to verify that a system works correctly but it can help immensely to find the optimal solution to a non-trivial problem. In this case it can help us to find the best way to cover the selected area or to know the number of drones to use and the resources they will consume.

1.2. Aim of the project

The main objective of this TFG is to simulate the autonomous flight of a set of UAVs to detect a target, in this case, a head of cattle (a cow,

for example) in a certain area. Trying to obtain as much information as possible from the simulation and developing a system which can adapt to as many different scenarios as possible and with the focus of scalability in mind. Further beyond the scope of this work, development will continue to physically implement this system.

The project will have different specific objectives regarding the different areas of work:

- Fully autonomous UAV flight, including battery recharging to be able to continue the mission.
- Route generation algorithm
- Object detection
- Visualization of the simulation

1.3. Work plan

The development this work it will be divided in different sections:

- Research of the UAV and Computer Vision state of the art.
- Selection of the right tools to use that include a way of having a simulated environment.
- Development of the route generation algorithms.
- Management of UAVs autonomous flight.
- Implemention of object detection.
- Visualization of the simulator.

The time division for each task is shown in the Gantt diagram of Figure [1.2]

The code developed for this work will be available at the public Github repository:

`https://github.com/Al3xFreeman/MultiUAV-SearchAndRescue-Simulation`

1.4. Structure of the document

The document has been divided into five chapters and two appendices, next we list a brief of each chapter:

- Chapter 2 - State Of The Art: The most important advances are highlighted both in the AUV applications for autonomous flight and in computer vision.
- Chapter 3 - Design Of The System: The algorithms to direct the UAVs and the control elements used to communicate UAVs and the ground station are described.
- Chapter 4 - Development Of The Simulator: Explanation of the different developed parts that allow the UAVs to perform the whole simulation.

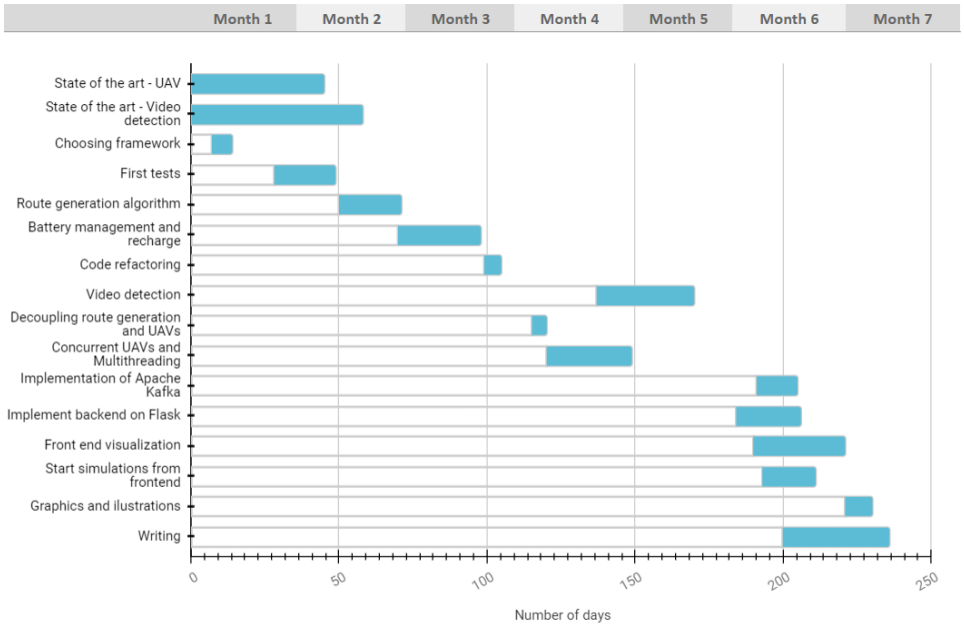

Figure 1.2: Gantt diagram.

- Chapter 5 - Conclussions And Future Work: The conclusions are presented and different improvements to perform are discussed.

- Appendix A - Using The Web: Instructions on how to use the web view are detailed.

- Appendix B - Tools And Programs: List of all the needed software to replicate the simulator on other machines.

Chapter 2

State Of The Art

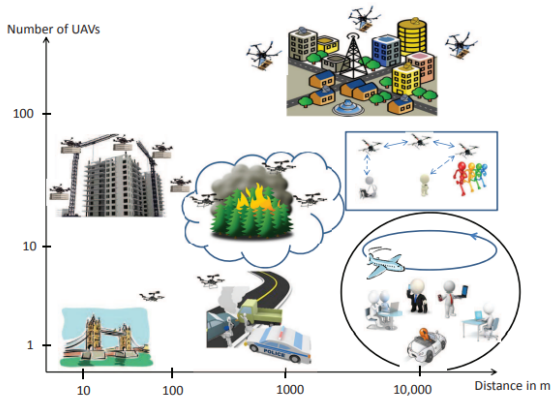

Figure 2.1: Different UAV applications and the UAV requirements to perform them[1].

In this section we will discuss the state of the art of the various areas that have been tackled on in this project. First the most advanced UAV applications will be discussed, as well as their limitations. Then, several computer vision techniques are described (image classification, object detection and image generation techniques).

2.1. UAVs

As already mentioned in the introduction, drones have had a long evolution, however it has not been until the last decades that their adoption has been possible by amateurs and by companies to try to find multiple uses (apart from the military field).

Improvements in both the batteries used and in the sensors available on board have led to greater availability and affordability. This massive adoption has allowed them to stand out for their great ability to adapt to a

[1] Hayat et al. (2016)

multitude of tasks, especially those aimed at gathering information or making small changes in the environment.

All this has led it to become an industry currently valued at about 27 billion USD in 2020 [Researchdive (2021)], which although small compared to others, is expected to grow at a very fast pace, reaching an estimated 58 billion USD by 2026.

The use of UAVs is usually limited to assisting a team located on the ground so that both collaborate in the resolution of the task. Therefore, although they are quite versatile, their limitations must be taken into account and several questions need to be asked [Alzahrani et al. (2020)]:

- Is it more beneficial to deploy a UAV system versus a ground-based one? For example, when inspecting an agricultural crop.

- What is the optimal number of UAVs and what will be their mobility?

- To what extent do UAVs improve the performance of the ground station?

2.1.1. Infrastructure maintenance and overhaul

In this paper [Hament and Oh (2018)], they explain the coordination between UAV and AGV (Autonomous Ground Vehicle) to be able to produce maps and track areas taking advantage of the virtues of both types of vehicles. In this case the use of drones is the differential factor, since thanks to its versatility it allows access to areas previously inaccessible to an AGV or modifications to the environment where required in order for the AGV to access.

Also in terms of photogrammetry, drones allow scanning various areas with great precision, in this paper [Daakir et al. (2017)] they go into detail of the different sensors and cameras that it is feasible to carry on board of a UAV.

There is also research [Monwar et al. (2018)] that focuses on optimizing the use of the drone battery, since it is one of the most limiting factors when performing any task with UAVs, in this case they propose a route planning algorithm to minimize the route to the target and the time.

2.1.2. Data collection

One of the most surprising use cases is to use UAVs as a data collection method in a network of IoT (Internet of Things) devices, usually used as simple sensors to gather data. Due to their simplicity, these devices may reduce the use of resources (energy or computation time) and have a moderate size, having to spare with ways of connecting to a network to send the information they have collected. That is why in this paper [Ebrahimi et al.

(2019)], they propose a generation of routes for a UAV to pass over all these devices to collect their information [Figure 2.2]. This saves the IoT devices a great amount of energy and the need to establish an entire infrastructure to access the collected information. Other approaches propose to generate this route with deep reinforcement learning [Bayerlein et al. (2021)] which can help over time find more efficient routes, without having to compute initially the best route, calculation that can consume a lot of resources.

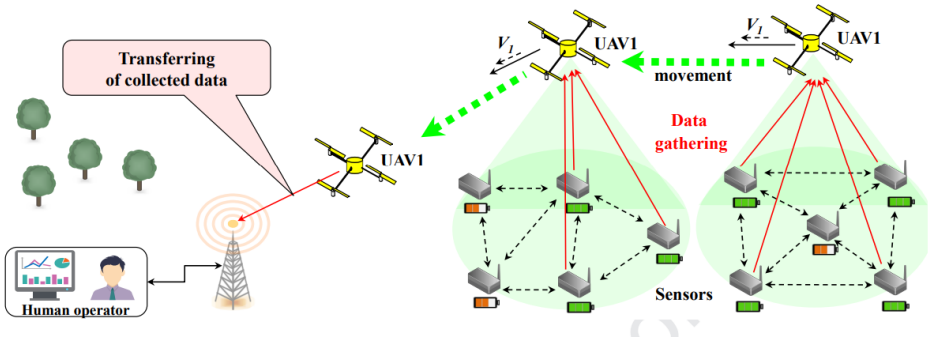

Figure 2.2: UAV used to gather information from certain sensors and deliver it to the ground station[2].

2.1.3. Package delivery

Regarding this topic there is a quite extend bibliography [Aurambout et al. (2019)][Goodchild and Toy (2018)][Pinto et al. (2020)][Muñoz et al. (2019)], due to the fact that large companies such as Amazon or Google have advertised these services quite a lot. Although currently their full implementation in our cities is rather far from a reality there are very interesting proposals, such as this paper [Sawadsitang et al. (2019)], which proposes a cooperation between aerial and ground vehicles to carry out the task. The most interesting idea of the paper is about introducing possible failures in the system (due to an accident, for example). This type of task is met with the complex problem of coordinating a huge fleet of UAVs and trying to optimize the movement of each part of it, that is why there are studies [Ham (2018)] that explore the best use of several drones and trucks working with several depots and destinations.

[2]Alzahrani et al. (2020)

2.1.4. Traffic monitoring

Another aplication for UAVs is that presented in [Kanistras et al. (2015)]. This work compares the use of drones to traditional ways of measuring and monitoring traffic. They propose and review different ways of detecting traffic with cameras from UAVs and then use this information to optimize the traffic flow and to be able to solve more efficiently possible accidents on the road [Figure 2.3].

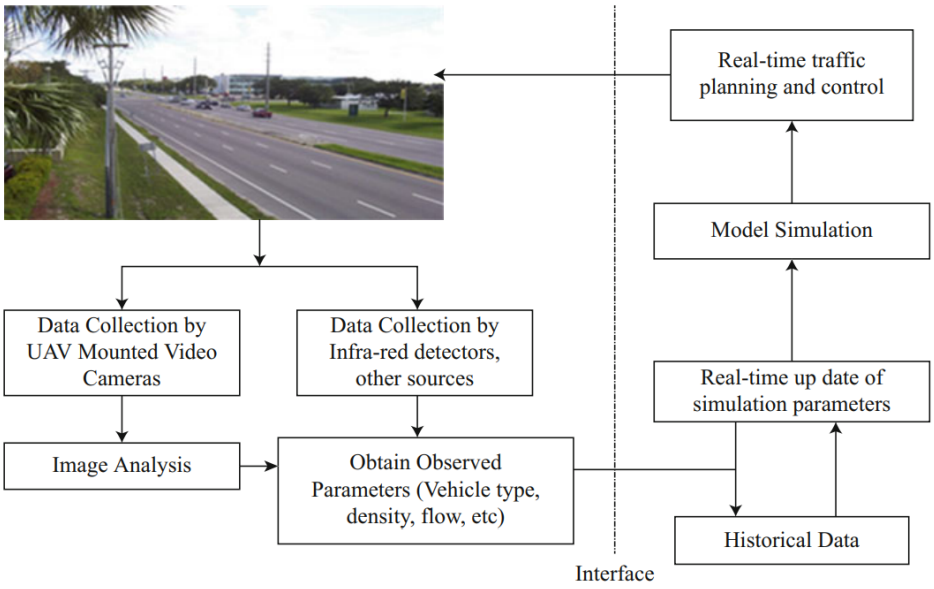

Figure 2.3: Data flow to use the information gathered by UAVs[3].

2.1.5. Disaster Management

In situations where a disaster has occurred and communications have been lost, or destroyed, UAVs can be crucial in re-establishing that communication and enabling rescue teams to operate as efficiently as possible by serving as communication links or providing information of the situation. As it is pointed in [Alzahrani et al. (2020)], this type of tasks can be divided in three stages: Pre-disaster supervision, Search and Rescue, and recovery (post-diasaster assistance) [Figure 2.4].

[3]Kanistras et al. (2015)

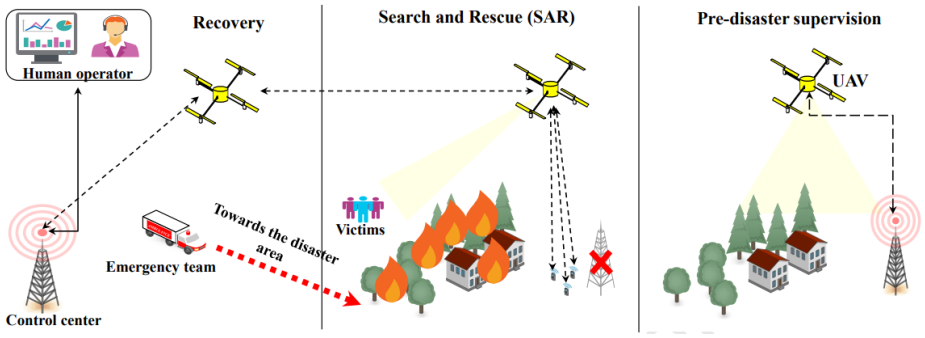

Figure 2.4: Different stages of disaster management[4].

2.1.5.1. Pre-disaster supervision

In the pre-disaster stage UAVs monitor and detect changes in the terrain in order to have an up to date report of the surveyed zone, to be able to achieve that, different proposals have been presented such as a perpetual flight of a UAV [Oettershagen et al. (2018)], or Wireless Sensor Networks (WSN) coordinated with Multi-UAV systems [Erdelj et al. (2017)].

2.1.5.2. Search and Rescue (SAR)

The efficiency of the first stage can be crucial, especially in those scenarios in which the disaster was predicted successfully, thanks to that emergency response teams can be more prepared and act more efficiently. Although there are scenarios where it can be impossible to be prepared against them, such as earthquakes. Nevertheless we enter the second stage of the disaster, the Search and Rescue operation begins. In situations where response time is vital, the use of UAVs, working together to cover a larger area faster can make a big difference. There are several ways to tackle this type of problem, for example, in this paper [Hayat et al. (2017)] they propose the use of a genetic algorithm to find multiple targets with a team of multiple UAVs and create a communication link between the target and the ground personnel. There are also other proposals [Alotaibi et al. (2019)] such as the creation of new algorithms for the most optimal location of possible survivors in an emergency situation.

2.1.5.3. Recovery

Finally, after the disaster happened one of the most vital aspects of it is to ensure communications are stable and reach every rescue team. Thanks to

[4] Alzahrani et al. (2020)

their versatility, UAVs, are the best way of reestablishing the infrastructure without the need to set up or repair ground systems. Works such as [Ahn et al. (2018)] propose communications between UAVs and IoT devices in order to ensure communication between rescue teams and the operations center.

2.1.6. FANETs

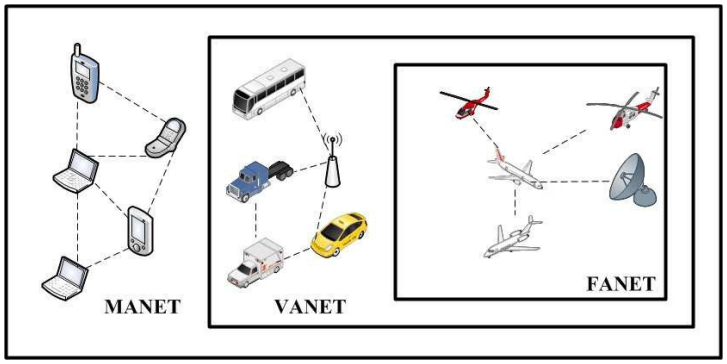

Figure 2.5: Clasifications of Ad-hoc networks[5].

Several of the projects mentioned above make use of FANETs (Flying Ad-hoc Network), which are a type of VANET (Vehicle Ad-hoc Network). This types of networks are usually connected to the rest of the world, but the main feature is that they are self-contained, meaning that they will be able to operate on their own. Some of the most interesting papers come from this technology, since it poses many problems for which it is difficult to choose an optimal solution. Its main function is to maintain a connected network of UAVs in which there can be fluid communication and minimize drones that either fail or run out of battery power. In this vein [Cumino et al. (2018)] propose models to ensure a stable connection by monitoring the battery status and managing the recharging of the drones. With the premise of keeping the network stable, they test transmitting video over it, which can be one of the most demanding tasks due to how fast the volume of data can increase if the resolution or the transmission rate is very high. There are also studies investigating the connectivity of this type of network [Guillen-Perez and Cano (2018)] and how different route generation and devices used can affect how they communicate with each other.

2.1.7. Simulators

Having reviewed a selection of the many projects and papers written on the different areas that concern this project, it is worth highlighting an

[5]Tareque et al. (2015)

important aspect that practically most of these researches have in common, the majority of them use simulated environments to test the theories they propose.

This is obviously largely due to the amount of costs saved by the use of a simulator and because of how advanced and useful they can be, in this TFG we will also base our work on simulated drones. However, we find it necessary to point out that the real implementation of these systems is often accompanied by new problems, since the simulated environments usually assume perfect conditions and may not be entirely in line with reality. Obviously, it is also complicated to implement these systems in real life as the cost increases rapidly, especially for those proposals that use a large number of drones.

There is a range of simulators to use, based on their main characteristics these are the best ones we have managed to find:

- SITL (Software In The Loop) [Ardupilot contributors (2022b)]: Built and maintained by the Ardupilot team, it is simple and has support for every vehicle type developed by Ardupilot.

- Gazebo [Koenig and Howard (2004)]: Simulation with 3D visualization with support for many types of sensors.

- AirSim [Shah et al. (2017)]: Simulation with support for drones, cars and more, built using Unreal Engine. Has support for a wide range of cameras and sensors in an aim to help experiment with deep learning, computer vision and reinforcement learning.

- SCRIMMAGE [DeMarco et al. (2018)]: used to simulate multi-agent flight and interactions.

Although out of the scope of this project, the implementation on the field is also a very complex area as the choice of the right equipment can be decisive for the success or failure of the missions and these aspects require in turn a great deal of research and knowledge of the subject.

2.2. Computer Vision

Computer vision is one of the disciplines that can be integrated together with UAVs, since being able to combine the versatility of drones with the power of the best image detection models can give rise to very relevant utilities such as those we have seen in the state of the art in terms of UAV applications, specifically in inspection and maintenance tasks, search and rescue, or traffic monitoring.

However, there are multiple techniques that can be used to try to improve the detection of various areas and targets. Some of the most prominent areas are image classification, segmentation, object detection, image generation, or data augmentation. Below there are some of the most relevant works within the scope of this work are reviewed.

2.2.1. Image Classification

One of the most important areas of computer vision is image classification. Most of the times it is used to classify images in which only one object is present but it can also become capable of analyzing very complex images to obtain a lot of information from a single image. For our case, in order to be able to implement image classification in UAVs, low power options will be needed. That is why we will make use of MobileNets [Howard et al. (2017)], a class of models trained with deep neural networks. MobileNets use convolutional networks and a great optimization of the parameters to reduce the time and resource consumption to classify the target image, making them key in the implementation of these techniques in UAVs.

2.2.2. Object Detection

Another area of considerable importance for this TFG is the detection of objects within an image. In this case, since the UAV may be quite far away from its target, it is necessary that in the context of an image that covers a relatively large area, it is able to detect a small element within it. Models such as YOLOv4 [Bochkovskiy et al. (2020)] manage to achieve very good results, or subsequent improvements to this paper such as Scaled-YOLOv4 [Wang et al. (2020)] that greatly improve the detection speed, reaching figures of about 1,700 frames per second.

2.2.3. Image Generation and Data Augmentation

One of the most important aspects when training a predictive model is to have a sufficiently complete dataset so that the model has enough examples to learn the characteristics of the different categories to be detected. That is why in scenarios tackled in this dissertation or even inspection/construction tasks it is possible that many elements are not correctly detected. Techniques such as GAN (Generative Adversarial Network) and data augmentation can both help the neural network to learn more about categories with few examples. In the case of GANs, in [Noguchi and Harada (2019)] is managed to go from thousands of images to train a GAN to less than 100 images using techniques such as Transfer Learning to achieve a high quality of generation for very small datasets.

Another very useful technique is data augmentation, in this way we can easily duplicate an existing dataset by applying transformations, rotations and modifications in the color of the available images. In this way, the trained model will learn more easily that an element may appear rotated and it will detect it without any problem thanks to the increased variability of the provided data. In the YOLOv4 paper [Bochkovskiy et al. (2020)], they discuss several techniques to perform this increase of information such as

CutOut [DeVries and Taylor (2017)], MixUp [Zhang et al. (2017)] or CutMix [Yun et al. (2019)]. These techniques that increase the dataset can also greatly favor the subsequent application of a GAN, as discussed in the MixUp paper.

Chapter 3

Design Of The System

The system has been built using the Python library "dronekit" an API available in Python that runs on the onboard computer of a UAV, which communicates with Ardupilot, an open source system for the development of automatic flight of different vehicles. It currently supports drones (with various propeller configurations), traditional helicopters, fixed-wing aerial vehicles (mainly airplanes), submarines, and tracking antennas used to improve communication with another device.

Ardupilot is developed in C++ and apart from being very versatile due to the number of different systems it supports, it is quite secure and predictable. It also has a very open development team, not only for the code, which is available to everyone, but also for its way of dealing with new developments proposed or requested by the community.

These developments are not only useful to be implemented in real hardware, but also incorporated in simulators (Software in the Loop), which allow to test and develop behaviors for different vehicles (in this case drones) and prepare them for different objectives or terrains in which they could operate.

It is thanks to this simulator that the development of this TFG has been possible. Since it has not been required the acquisition and assembly of any parts to be able to propose and develop autonomous systems that can solve real problems.

About the communication protocol used, it is worth mentioning MavLink, which is a protocol used for low latency connections between vehicles and ground stations. It is the one used by Ardupilot and has a large number of different messages to transmit information about the battery, the orientation of the drone, or send from the ground station the route of a particular mission to perform, among hundreds more. It is also extensible, in the sense that you can add your own messages to the protocol to send specific instructions to the vehicle.

For the system to work we propose the schema shown in [Figure 3.1].

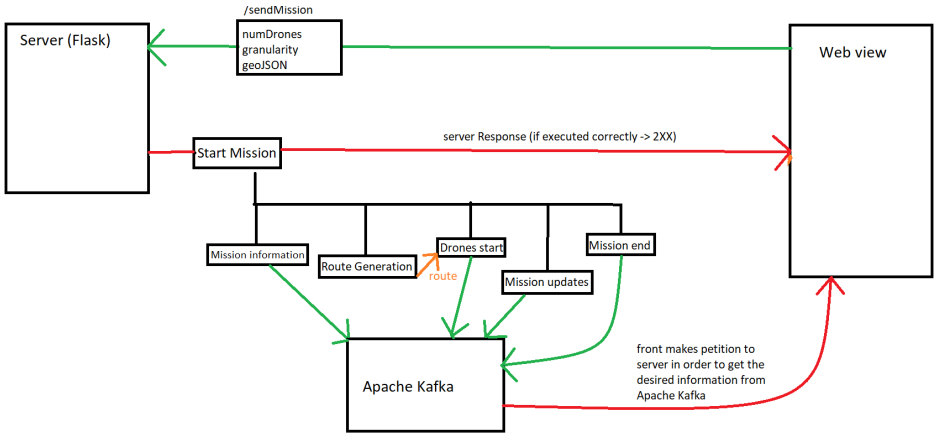

Figure 3.1: Sketch representing the data and execution flow of the application.

The general concept is that drones and the ground station are producers of information that is sent to Apache Kafka and the Web View is the consumer who extracts the desired information from Apache Kafka. It is built to be as scalable as possible.

3.1. Algorithms and Route Generation

In order for each UAV to know which area should it cover it will be necessary to generate it. In this section we will explain the different processes that have been developed to be able to determine what areas should the UAVs fly over. We will also be covering its limitations at the end of the section.

When developing this system, one of the main focus has been that it could suit other needs apart from this work. That is why, although certain algorithms have been used for this dissertation, it does not mean that they will be the most optimal for any task and with slight modifications to the code it will be able to accept other multitude of route generation algorithms. In the following, we will first explain the process that the program currently follows and the considerations that were taken in their implementation.

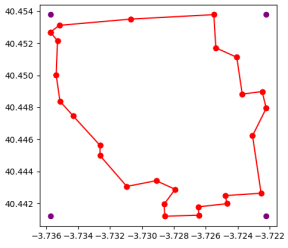

Figure 3.2: Polygon specified in GeoJSON file.

Taking into account the objective of the program (given a delimited area, to be covered by the drones and report whether or not the selected target

has been found) the first step will be to receive a series of geographical coordinates that delimit that area. In order not to complicate the task too much, it is assumed that this area does not interfere with any area restricted to drone flight. But as we will comment in the future work section, this feature is an intersection between both areas (area to fly and restricted area) and keep the left joint of this set, that is, the area that is only located in the area to fly and does not contain anything in the restricted area.

Once we have the right polygon [Figure 3.2] we perform the necessary data transformations from tuples of [latitude, longitude] so that it can be used as a series of dronekit points.

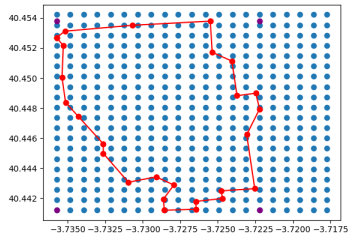

Figure 3.3: Dot matrix generated.

The next problem we have is to know where the drones have to pass through to travel as efficiently as possible in that area. To do this we will make use of a matrix. We will generate the matrix with a granularity (distance between points) specified by the user or defined by the program itself. This matrix will cover the entire specified polygon [Figure 3.3].

With the generated matrix we will then be able to detect which points are inside or outside the polygon, those outside will be discarded and those inside [Figure 3.4] will be used for the subsequent generation of the drone route. Like other sections, this one will also be discussed in the future work section, as many other algorithms could be implemented to obtain higher accuracy.

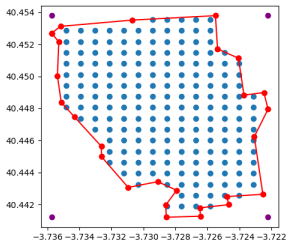

Figure 3.4: Matrix points inside the polygon.

Once we have the objective points to be covered by the drones, we have to calculate the path they have to follow. The simplest way to solve this problem would be to perform rectilinear movements from one end of the polygon to the other, moving laterally each time it reaches an edge and starting another rectilinear movement. However, this can lead to problems, since if the polygon has convex areas, the drone could fly through areas that are not allowed. That is why in this work, as an example we have opted for the use of the TSP (Travel Salesman Problem), by which we can generate a route that passes through each point (inside the polygon) only once [Figure 3.5]. TSP works by giving a list of vertices and the distances between each pair, compute the shortest possible route that visits each vertex exactly once and returns to the origin

vertex. It is important to note that currently we are not trimming those edges that intersect with the edges of the polygon to prevent possible paths that go outside of that polygon.

One of the problems we encounter then is the efficiency of the generation, since it is an NP-Hard algorithm, i.e., there is no computationally efficient way to find a solution to the problem, which is why we will have to use some heuristics to shorten the computation time. For this purpose we have chosen a Python library that implements the TSP which obtains an acceptable performance with a reduced number of points.

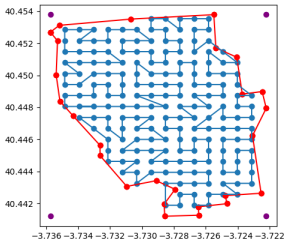

Figure 3.5: TSP generated route.

Once we have the route calculated, the only thing left to do is to send the drone the appropriate indications using the missions implemented in Ardupilot. In the case that there are several drones, we just divide the route into equal parts for each drone. These missions are a succession of instructions that the drone controller stores and proceeds to execute them in the specified order.

3.2. Drone Operation and control elements

In this case, since we have focused on the simulation of the drones, we will first start the SITL (Software in the Loop) of dronekit, which is the environment that is responsible for simulating all aspects of the drone. Then we will proceed to connect to the drone to have access to its state.

Each drone is identified with a unique *id* and will store the name of the connection it has with the ground station. It has several variables to control different aspects:

- It has finished the execution
- It must continue transmitting information about its status
- It has found the target and its position
- The video it is capturing
- The points it has to traverse
- The flight time
- The connection to Apache Kafka (for transmitting information)
- The batteries it has used

When the drone has successfully connected we proceed to launch several threads inside each drone. Each one will have threads to control different aspects:

- The execution of the mission
- The camera
- The detection of the target
- Apache Kafka

The mission execution has two main functions: the most important is the battery control and secondly to check if it has reached the end of the mission and the second one is to update the control variables to warn the rest of the drones and the rest of the processes.

The battery control is a crucial part of the program operation, since it allows to continuously monitor its level in order to decide if it has to return to the base location where the drone can recharge its battery and then resume the mission where it left off. When it has finished the execution of its route, it will set course to the base and, once there, disconnect.

This control of the execution is subject to one of the variables that control whether the drone has to finish or not, since another drone can communicate that it has found the target, this will cause the rest of the network to automatically finish its execution and return to the base.

The camera control has been implemented with the OpenCV [Bradski (2000)] library, which greatly facilitates the use and control of cameras. With the information captured by the camera we will proceed to adapt the image so that one of the pre-trained MobileNet models will analyze and detect if the target has been located. The model we have chosen is optimized for low-power devices, such as cell phones, or in this case, drones. Of course, any other model that could be trained to improve detection could be implemented without any problems. This aspect will also be discussed in the future work section, since currently one of the limitations is that the MobileNet model we use is relatively limited and, for example, if we are looking for a cow, it does not detect aerial photographs of cows well, since the model has not been trained with aerial photographs of cows.

Regarding the target detection control, it is used to update different control variables, but its operation can be used to send other types of messages, or in general extra processes that you want to do when the target has been found.

Finally, the last process is in charge of controlling the messages sent by Apache Kafka. This service, in very few words, allows to collect information from various sources (in this case all the drones and the ground station) and transmits them to the number of consumers of such information that request it (in this case it will only be the web display). In this way, with Kafka we will send messages containing information about the status of the drone for easy viewing or processing.

Chapter 4

Development Of The Simulator

When the algorithms and the communication schema was thought out it was time to develop the actual simulator and visualization. As there will be several components they were developed one by one, allowing enough time to check that each one worked as intended before starting the next part. As we can see in the Gantt diagram [Figure 1.2] the base ideas of the project like the route generation and battery management were first implemented. Next, the decoupling of the route generation from the drone execution was developed, which allowed later to support several UAVs simultaneously. In the meantime the video detection was researched and implemented. After that, improvements on the algorithms and better output of the information was developed. Finally, the web view was built in order to be able to execute missions fully withing it.

4.1. Simulation of one and several UAVs

Initially, the original idea was to have a single drone in charge of performing the generated mission, however, while working on its development, it was discovered that it would not be excessively complicated to perform the simulation of multiple drones simultaneously. This made it a much more interesting problem, in terms of route optimization and in terms of communication among the drones.

As we assume a continuous connection to the ground station with the drones we can ensure that there is an uninterrupted exchange of information. The limitations of this system lie mainly in its implementation in real life, because if a drone does not have a direct connection with the ground station or with another drone, it would not be able to communicate that it has found the target. That is why, as in projects like [Alotaibi et al. (2019)] that with tools like Dronemap Planner [Koubâa et al. (2017)] they would be able to

guarantee the constant connection of the UAV.

In this project, and following the ideas explained in the previous section, the drones share information with the ground station and the ground station sends the appropriate messages to each drone. At the end of the execution the information of the "costs" of the mission, time and battery consumption, are shown on the graphical interface.

4.2. Communication between UAVs and Ground Station

In the simulated environment that we have proposed the ground station sends all the necessary information and commands to the drones, without forgetting that much of the control of the vehicle is in the hands of the drone, the UAV is in charge of detecting its battery levels and changing the mods of how to operate according to what it detected.

One of the great advantages of the simulated environment is that it allows us to access the status of the drones in real time, for the system to work with an implementation in real life we should use the MavLink protocol, as we mentioned previously is a lightweight messaging protocol that has a multitude of messages to communicate the drone with the ground station. In addition to this, we can create our own messages, in this case we should create the appropriate messages to transmit the detection of the targets.

On the other hand we have the communication between the drones, in this case we enter the field of VANET/FANET, which are autonomous networks of vehicles between which information is shared. This type of communication has not been implemented in this work but it presents its own challenges and limitations that are also very interesting.

The MavLink messages and FANET implementations will be discussed in the future work section.

4.3. Object Detection

In order to be able to detect the selected target we decided to implement a neural network approach as they can achieve great results if correctly trained. For that reason we used a pre-trained model design for low consumption devices called MobileNet [Howard et al. (2017)]. This is perfect as the fewer energy consumption, the longer the UAV will be able to sustain its operation.

The model used was a caffe [chuanqi305 (2018)] implementation of the pre-trained MobileNet-SSD (Single Shot Multibox Detector), allowing to detect multiple targets in a single frame. According to the repository of the model, it was trained using VOC0712, which contains 20 classes separated in four main categories: Person, Animal, Vehicle and Indoor. We will be using

mainly the animal category as it includes cows, horses and sheep. One of the main limitations of this model in the context of our application is that is was trained using images not taken from a flying UAV perspective.

As the model can identify a target within the image, whenever that event happens, it is communicated to the drone which has encountered the target, thus, the ground station will be able to inform the rest of the UAVs that the mission is complete and order them to return to the base.

In order to see what the camera is detecting a window will pop-up while the simulation is running. Further improvements will depend on the camera, sensors and altitude at which the UAVs will fly, but that aspect regarding training of new models and applying image augmentation techniques will be discussed too in the future work section.

Bear in mind that as we are using a simulated environment, we simply use a camera that is connected to the computer where the program is running. Despite this limitation, we have built this part of the program so that each drone can independently control its camera to support a future physical implementation.

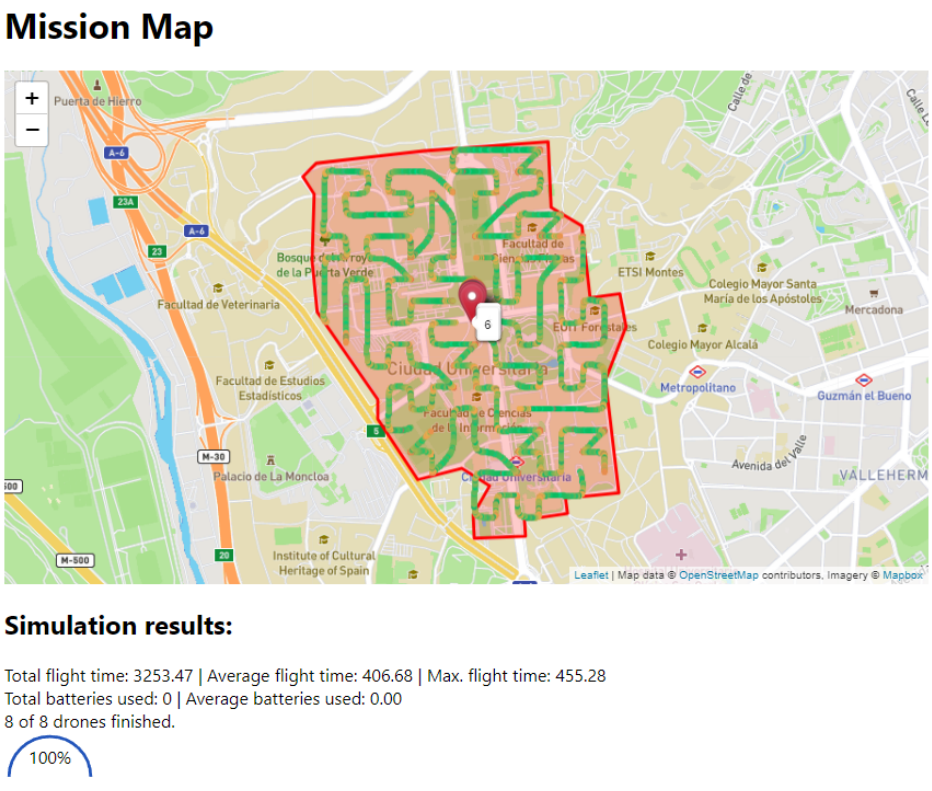

Figure 4.1: Example of a mission being executed from the web visualization.

4.4. Web Visualization

In order to make the program more useful we have chosen to implement a simple web interface from which we can visualize the status and position of the drones. Although the main use of it is starting the mission. In Figure 4.1 we can see the web layout and in Appendix A is explained how to deploy and use it.

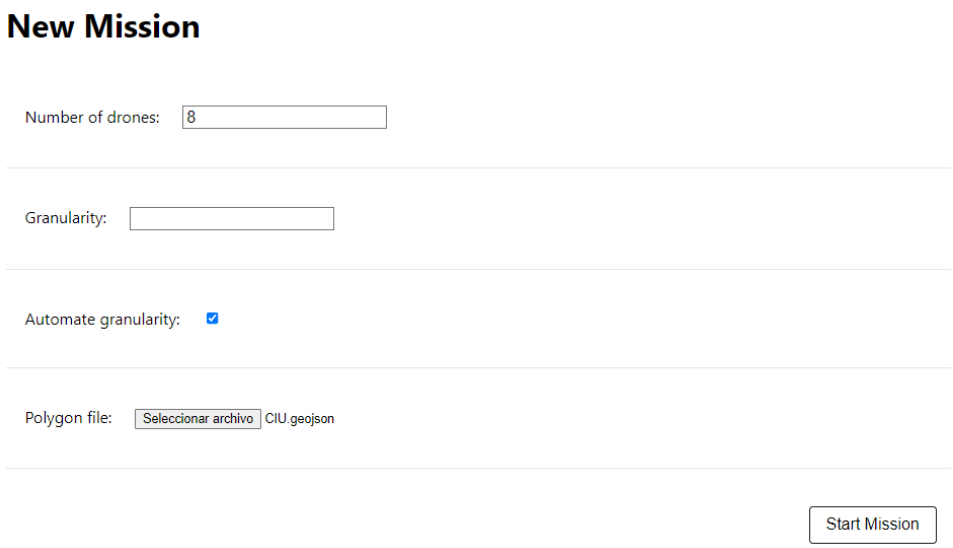

Figure 4.2: Available parameters to start a mission.

To start the mission, several parameters are available [Figure 4.2], such as the number of drones to use, the granularity (the space between each point the UAVs have to go through) and a GeoJSON file containing the area to be covered. In case the optimal value of the granularity is not, known it can be specified so that the application limits the generation of the route preventing too long computations. The reason for that is we are currently using a TSP algorithm to generate it and can incur in high computational costs if not bounded correctly.

Some of the current limitations are not being able to indicate the specifications of the drone that will perform the task, as parameters such as battery capacity, motor characteristics, or weight has to be further examined on how to indicate the simulator the correct settings. So for the time being it will be executed with the default Ardupilot simulator copter settings.

The web receives the different messages from Apache Kafka with information of the beginning or end of the mission and also updates of the information of each drone.

4.5. Use Cases

The application has been developed with the main objective of finding a target in a determined area. Thus, the main use case is simplifying the execution of said search and rescue mission from the web view, in the Table 4.1 are shown the steps to execute Use Case 01 and in the Appendix A there are detailed instructions to be able to execute the mission in the web view.

Other uses derived from the main function can be used, like letting the program optimize some of the parameters, such as calculating an optimal granularity or compute the different execution times with different number of drones.

In Use Case 02, shown in Table 4.2, we describe how to execute a mission where the granularity is specified by the application so it optimizes it to not taking too long to generate the route for the UAVs.

UC-01	Start Mission - Web view
Description	Starts the mission from the web view providing the appropriate parameters
Inputs	Number of drones Granularity GeoJSON File
Sequence	1. Enter the parameters 2. Click on Start Mission 3. Watch the simulation happen and wait for the results 4. You can use your webcam to test the detection algorithm while the mission is running

Table 4.1: Use case 01 - Execute mission from web view.

UC-02	Start Mission (Dynamic granularity) - Web view
Description	Starts the mission from the web view without providing a granularity as the application chooses an appropiate value
Inputs	Number of drones GeoJSON File
Sequence	1. Enter the parametrs 2. Check the "Automate granularity" checkbox 3. Click on Start Mission 4. Watch the simulation happen and wait for the results 5. You can use your webcam to test the detection algorithm while the mission is running

Table 4.2: Use case 02 - Execute mission with dynamic selection of granularity from web view.

Chapter 5

Conclusions and Future Work

5.1. Conclusions

After developing the presented system proposal, there are some key points that stand out. First of all, the importance of the simulation, as it can help on discovering pitfalls in the approach to the multiple challenges that surface when solving each individual part. Another important aspect is the information harnessed from the simulation, with it we can make a more informed decision of what exact equipment will be needed to perform the desired task. On the other hand one of the biggest drawbacks was the lack of information or documentation on certain areas of the project, this is especially troubling with the different versions of the libraries used, a long time was needed to research and testing how everything could work together.

This mainly shows the work still needed in the area, providing quality software that will help in the building of necessary tools which provide more safety and automation to the field. From improving the autopilots to easier use and simulation of sensors and better connections between UAVs and Ground Stations. Despite these setbacks, the UAV autonomous flight industry is a field with many areas to explore, with a myriad of applications and, in general, a promising future but with many challenges to solve.

We have also experienced the difficulty of creating a generalist autonomous flight system. For each different objective to be achieved, an evaluation of the objectives and available resources will be necessary, since there are a huge number of variables to take into account, from the number of different sensors that can be used, to the type of terrain where the mission is to be performed or the resources available. It is worth mentioning again the importance of being able to have a simulated environment where you can perform the desired amount of tests until you find the optimal configuration for each task.

With respect to the results that we have been able to extract from the

simulator we note the importance of being able to implement several UAVs simultaneously, and access their information in real time, as this allows to make modifications on the routes to be followed by the drones at any time from the ground station. Note that it is assumed that there is a connection with the ground station or with the route manager at all times, as in other proposals which depend on Dronemap Planner [Koubâa et al. (2017)].

5.2. Future Work

First we will talk about the more immediate future of the simulated environment and the process that would have to be performed to be able to implement it physically. We will also comment on the feasibility of incorporating some type of FANET to real projects, since it is one of the most promising technologies. We will then talk more specifically about various algorithms that we want to put forward that could vastly improve the performance of the simulator. We will also devote a section to review the work to be done in computer vision.

With respect to the simulated environment we have developed, one of the immediate improvements that could be made is to port it to another language such as C/C++ or Rust. Python has many advantages and rapid development is crucial to be able to develop prototypes or works of this style, but it would be essential in the future to implement it in languages that are compiled, since the very nature of Python, being an interpreted language can be orders of magnitude slower than compiled and lower level languages such as C/C++. Throughout the whole process we have tried to use libraries that were also available in C++, so that when such a change is made it will only be necessary to implement the same functionalities but adapted to the new language.

In order to physically implement this system, it would be necessary to implement the MavLink messages so that the drones can communicate their messages to the ground station and this in turn to the rest of the drones. Other aspects to bear in mind when implementing the system is to know where it is going to be deployed, as there can be obstacles that the UAVs will need to avoid such as building, trees, or power poles.

Another important aspect to comment are the FANET, they are a very powerful tool that allows a much more fluid communication among the different drones or components of the system, and also the extension of its functionalities to a great extent. As we have seen in the state of the art, there are some articles and research on the possible use of these systems in real life, mostly focused on autonomous vehicle driving, where a connection between different vehicles is proposed to avoid possible accidents or optimize traffic flow. However, few real applications have been implemented.

Although we have not implemented them, we have focused on talking

about FANETs as they can lead to some of the most interesting applications. We can take as an example MANETs (Mobile Ad-hoc Networks), which are a much broader definition, since any network of connected devices will constitute a MANET. It is worth mentioning them as they are one of the few examples that has been implemented to an almost worldwide scenario. A device called Tile which once paired to your phone could transmit its location to the user even if they were kilometers apart by communicating with other Tile devices, thus creating a global MANET. It is a simple but powerful idea that we can extrapolate to a VANET/FANET implementation.

With this in mind, being able to develop a good system that communicates to a network of UAVs can bring great benefits as it could take full advantage of the mobility and versatility of UAVs. One of the most prominent use case examples in other papers is the deployment in catastrophic situations, where the local network has been destroyed or did not exist in the first place. This type of network would allow temporary connection to rescue or medical teams to indicate the status of the area at all times. Also, if drones were equipped with the ability to analyze or collect information, they could also transmit it.

Without going into much more depth, we can see the potential they can have, but there is a lot of work to be done, as there are many factors to be taken into account, such as the weather, the characteristics of the drones (maximum flight time, speed, etc), the type of connection used, the terrain where they are deployed and a variety of problems that may arise when implementing systems of these characteristics.

Coming back to the more immediate future work, there are improvements that can be implemented for the route of the drones. Currently we have tried to assume the most restrictive case, and this is to avoid that drones leave as little as possible the delimited polygon, since in some airspaces many areas can be very restrictive for the flight of UAVs. In spite of this the implementation of the TSP is not the most optimal, since for a large number of points to be flown by drones it can become unfeasible to calculate the route. Also, with the current implementation, only the TSP route is generated for a single drone and in the case of multiple drones, the route is segmented and each drone is assigned to its corresponding part.

There are several ways we would like to propose to solve these problems. The first consists on the implementation of mTSP, which is the implementation of the TSP but for m vehicles. This could greatly reduce the distance traveled by the set of all drones, but it also increases the complexity when generating such a route. It would neither solve the problem of having too many target points that the drones have to pass through. Therefore, the computation time problem would not be solved.

Another option would be to randomly divide the points to be traversed into n grouped regions, in this way the TSP would only have to be computed

for areas with $\frac{p}{n}$ points (p being the total number of points), and one could algorithmically calculate what would be the appropriate number of regions so that the computation time would be within the margins to be established. The only problem then is to find an efficient way to divide the terrain into n regions.

The latter would be a quite ideal solution, since it allows assigning one or multiple drones (since having reduced the complexity we could apply the mTSP algorithm previously discussed) to each region and thus be able to discard entire sectors much faster than with the current implementation. Other solutions worth exploring are bio-inspired ones like the Ant Colony Optimization algorithm, which takes inspiration from how ants find paths between several points.

Also, one of the improvements required by the currently implemented TSP algorithm is that the paths between nodes that pass outside the marked polygon are not pruned. That is, there may be cases where the drone flies outside the perimeter. It would be necessary to indicate to the current algorithm that those connections that intersect with any of the edges of the polygon to not be included in the algorithm. This will need further research as a straightforward implementation would have an exponential cost of $O(n^2)$.

These algorithms would be implemented, as we have mentioned, to prevent drones from leaving the established perimeter, for security reasons, or any other. If this were not the case, another of the improvements that the simulator would benefit the most from is to have the ability to establish whether this is necessary or not. Since if the drone can fly outside the perimeter, route generation is completely trivial.

With respect to computer vision, the most immediate improvements that should be implemented would be to train a neural network, or any other way of target detection, with aerial photographs of livestock. To get the best possible results it would be better to apply Image Augmentation to increase the size of the dataset, since there is not much material available of this type of photographs. With this network created and tested with drones it would be turn to find the most appropriate parameters to use, such as adjusting the camera to be used (type of lens, zoom, etc) or the height at which to perform the missions. Also one of the best ideas that could be explored would be the use of other sensors such as temperature or infrared sensors.

Finally, it would also be interesting to work on flying drones in more complex environments such as forests or even urban environments. Especially those environments that require the drone to be constantly reevaluating its position depending on obstacles of various sizes or even mobile obstacles. As this could greatly help in the detection of endangered animals in jungles or difficult to access environments.

Last but not least, we focus on the issue of security. If a large-scale

adoption of drones is to be achieved, high standards must be adopted, in terms of reliability and efficiency. It has been seen in many occasions how several air shows have ended up with more than one drone crashing to the ground endangering people nearby. Also when dealing with networks such as FANETs, it is often possible that they are communicating sensitive information. That is why physical and communication security mechanisms must be implemented and in case of failure can be mitigated as quickly as possible. We do not want a drone in charge of monitoring forests for fires to cause one due to a battery failure.

The work ahead is quite exciting as it consists of small short-term improvements and large projects that could influence industries of all kinds. We have found some similarities to the evolution of Cobots (robots that collaborate in factory environments with humans). In their first decades humans were not even allowed to access to their immediacy, and it was with time that a multitude of sensors incorporated to allow great cooperation and exploit to the maximum the intelligence and adaptability of humans in conjunction with the precision, speed and strength of Cobots.

One last aspect that, due to lack of knowledge and time, we have not been able to iterate more on is web visualization. Although it is not vital to the operation of the application, it provides a great way to extract and visualize the status of the system. In the future it would be interesting to add more functionalities as it can massively simplify the process of deploying and managing complex applications.

Appendix A

Using the Website

In order to use the web interface firstly Apache Kafka and the the web server will have to be started.

With respect to Apache Kafka (in Appendix B there are the instructions on how to install it), it will be necessary to execute two commands. Firstly, we will open two consoles and head to the directory where Apache Kafka is located. Once there in one console we will execute this command:

```
bin\windows\zookeeper-server-start.bat config\zookeeper.
    properties
```

Around 20-30 seconds later, the service "zookeeper" will be initialized and in the other console we will execute the next command:

```
bin\windows\kafka-server-start.bat config\server.properties
```

The last step regarding kafka will be to create a topic, that is where the different producers of information will send their messages and the consumers will extract information from the different topics. They are conceptually similar to a folder in a file system. In a new console we will create the following topics:

```
bin\windows\kafka-topics.bat --create --topic mapaDronesTest --
    bootstrap-server localhost:9092
```

```
bin\windows\kafka-topics.bat --create --topic mapaDronesSetup
    --bootstrap-server localhost:9092
```

Once everything is correctly initialized we will be able to start the web server. In another console we will head to the project's directory and execute the next two commands:

```
set FLASK_APP = flaskServer

flask run
```

When everything has started, from the browser we will be able to access:

`http://localhost:5000/map`

Once there, we will simple have to provide the following parameters: number of drones, granularity (manually or letting the program choose an appropriate value), and area to cover in a GeoJSON file. To generate such file we can go the the next website:

`https://geojson.io`

and select the polygon tool to draw the desired area. A marker can be added to serve as a starting and recharge point for the drones, in case it is not specified it will be the centroid of the polygon. Once the desired area is drawn, on the top-left corner there is a save button which will display several formats to download the data. Just select GeoJSON and everything will be ready to be uploaded to the web view and start the mission.

Appendix B

Tools and Programs

In this appendix we list the various tools and their respective versions needed to execute the different systems that compose this project. The following instructions have been performed in Windows.

B.1. Apache Kafka

It requires Java 8+ to work, if Java is not installed on you environment, go to the following link and install java:

`https://www.oracle.com/java/technologies/downloads/#jdk18-windows`

It will be necessary to add java as a path variable, to do that, search for "path" and the system properties window will open [Figure B.1]. From there, click on "Environment Variables". Another window will open and on the section "User variables for [Username]" click on "New". The name should be: "JAVA_HOME" and the value: the route where Java JDK is installed (the default route is "C:\Program Files\Java\[your Java version]") [Figure B.2]. One last step is to edit the Path variable and add a new entry with this value: "%JAVA_HOME%\bin" [Figure B.3].

Now we can proceed and download kafka. In order to do that, go to the following link and download the Scala binaries (version 2.13):

`https://kafka.apache.org/downloads` (Tested with Apache Kafka version 3.1.0)

and extract it in the desired location (keep in mind where it is as it will be necessary later to access that directory).

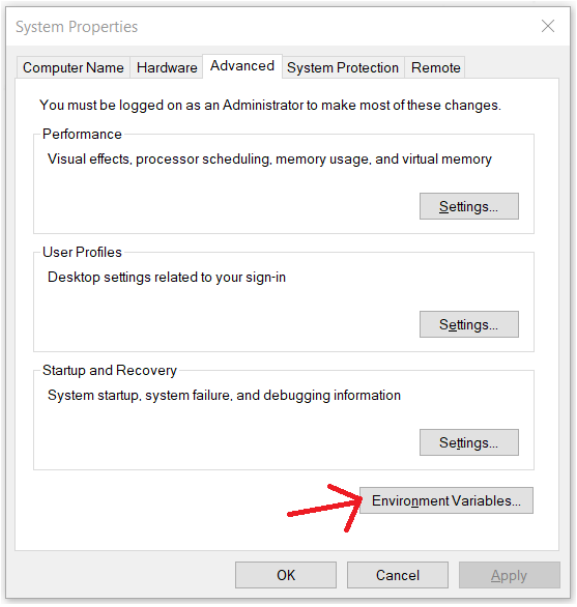

Figure B.1: System properties window.

B.2. Python

`https://www.python.org/downloads/` (version 3.8.2)

Before installing the required libraries, we recommend using a virtual environment, as it helps managing the installed libraries and prevents from having non-valid versions of different libraries in different projects. More information on how to operate with them can be found in the Python documentation:

`https://docs.python.org/3/tutorial/venv.html`

It is important to make sure to have all the package managers up to date, with this command:

```
pip install --upgrade pip setuptools wheel
```

In order to install the appropriate libraries we can use pip (a package installer for Python), using this command:

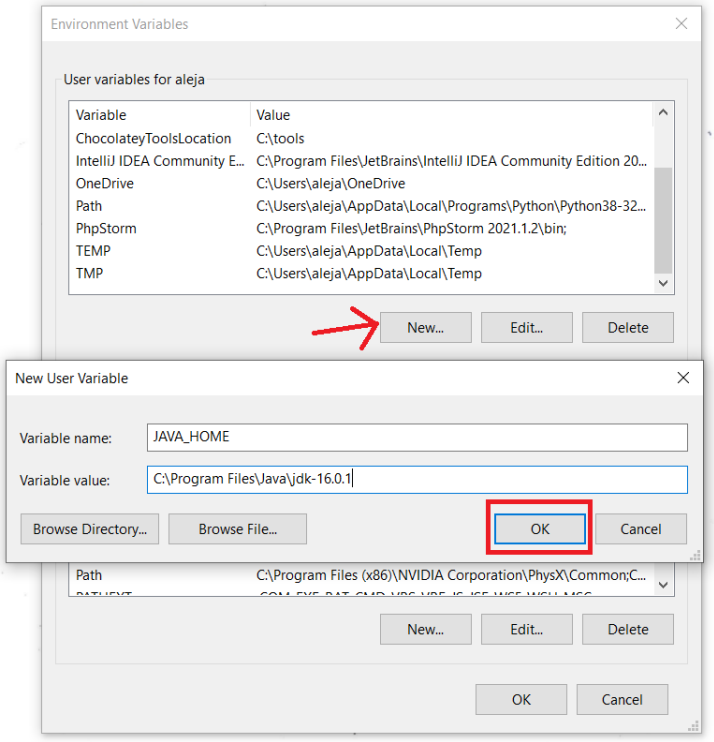

Figure B.2: Adding JAVA_HOME variable.

```
pip install -r requirements.txt
```

The Python libraries used are the following:

dronekit (version 2.9.2)

dronekit_sitl (version 3.3.0)

Flask (version 2.1.1)

imutils (version 0.5.4)

matplotlib (version 3.5.1)

numpy (version 1.21.4)

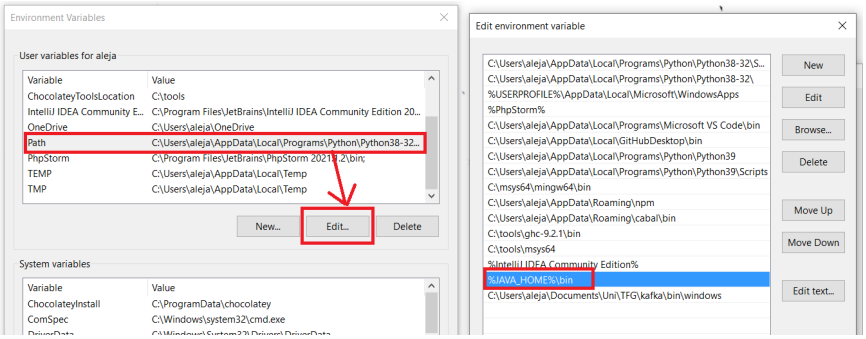

Figure B.3: Adding JAVA_HOME to Path.

opencv_python (version 4.5.5.64)

pykafka (version 2.8.0)

pymavlink (version 2.4.8)

pyproj (version 3.3.0)

python_tsp (version 0.2.1)

Shapely (version 1.8.0)

Bibliography

Ahn, T., Seok, J., Lee, I. and Han, J. Reliable flying iot networks for uav disaster rescue operations. *Mobile Information Systems*, Vol. 2018, 2572460, 2018. ISSN 1574-017X.

Alotaibi, E. T., Alqefari, S. S. and Koubaa, A. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, Vol. 7, 55817–55832, 2019.

Alzahrani, B., Oubbati, O. S., Barnawi, A., Atiquzzaman, M. and Alghazzawi, D. Uav assistance paradigm: State-of-the-art in applications and challenges. *Journal of Network and Computer Applications*, Vol. 166, 102706, 2020. ISSN 1084-8045.

Ardupilot contributors. Ardupilot history. `"https://ardupilot.org/planner2/docs/common-history-of-ardupilot.html"`, 2022a. [Online; accessed 16-May-2022].

Ardupilot contributors. Ardupilot history. `"https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html"`, 2022b.

Aurambout, J.-P., Gkoumas, K. and Ciuffo, B. Last mile delivery by drones: an estimation of viable market potential and access to citizens across european cities. *European Transport Research Review*, Vol. 11(1), 30, 2019. ISSN 1866-8887.

Bayerlein, H., Theile, M., Caccamo, M. and Gesbert, D. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open Journal of the Communications Society*, Vol. 2, 1171–1187, 2021.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. 2020.

Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

chuanqi305. Mobilenet-ssd. `https://github.com/chuanqi305/MobileNet-SSD`, 2018.

Cumino, P., Lobato Junior, W., Tavares, T., Santos, H., Rosário, D., Cerqueira, E., Villas, L. A. and Gerla, M. Cooperative uav scheme for enhancing video transmission and global network energy efficiency. *Sensors*, Vol. 18(12), 2018. ISSN 1424-8220.

Daakir, M., Deseilligny, M., Bosser, P., Pichard, F., Thom, C., Rabot, Y. and Martin, O. Lightweight uav with on-board photogrammetry and single-frequency gps positioning for metrology applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 127, 2017.

DeMarco, K., Squires, E., Day, M. and Pippin, C. Simulating collaborative robots in a massive multi-agent game environment (SCRIMMAGE). In *Int. Symp. on Distributed Autonomous Robotic Systems*. 2018.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. 2017.

Ebrahimi, D., Sharafeddine, S., Ho, P.-H. and Assi, C. Uav-aided projection-based compressive data gathering in wireless sensor networks. *IEEE Internet of Things Journal*, Vol. 6(2), 1893–1905, 2019.

Erdelj, M., Król, M. and Natalizio, E. Wireless sensor networks and multi-uav systems for natural disaster management. *Computer Networks*, Vol. 124, 72–86, 2017. ISSN 1389-1286.

Goodchild, A. and Toy, J. Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry. *Transportation Research Part D: Transport and Environment*, Vol. 61, 58–67, 2018. ISSN 1361-9209. Innovative Approaches to Improve the Environmental Performance of Supply Chains and Freight Transportation Systems.

Guillen-Perez, A. and Cano, M.-D. Flying ad hoc networks: A new domain for network communications. *Sensors*, Vol. 18(10), 2018. ISSN 1424-8220.

Ham, A. M. Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, Vol. 91, 1–14, 2018. ISSN 0968-090X.

Hament, B. and Oh, P. Unmanned aerial and ground vehicle (uav-ugv) system prototype for civil infrastructure missions. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 1–4. 2018.

Hayat, S., Yanmaz, E., Brown, T. X. and Bettstetter, C. Multi-objective uav path planning for search and rescue. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 5569–5574. 2017.

Hayat, S., Yanmaz, E. and Muzaffar, R. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys Tutorials*, Vol. 18(4), 2624–2661, 2016.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.

Kanistras, K., Martins, G., Rutherford, M. J. and Valavanis, K. P. *Survey of Unmanned Aerial Vehicles (UAVs) for Traffic Monitoring*, 2643–2666. Springer Netherlands, Dordrecht, 2015. ISBN 978-90-481-9707-1.

Koenig, N. and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3, 2149–2154 vol.3. 2004.

Koubâa, A., Qureshi, B., Sriti, M.-F., Javed, Y. and Tovar, E. A service-oriented cloud-based management system for the internet-of-drones. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 329–335. 2017.

Monwar, M., Semiari, O. and Saad, W. Optimized path planning for inspection by unmanned aerial vehicles swarm with energy constraints. In *2018 IEEE Global Communications Conference (GLOBECOM)*, 1–6. 2018.

Muñoz, G., Barrado, C., Çetin, E. and Salami, E. Deep reinforcement learning for drone delivery. *Drones*, Vol. 3(3), 2019. ISSN 2504-446X.

Noguchi, A. and Harada, T. Image generation from small datasets via batch statistics adaptation. 2019.

Oettershagen, P., Stastny, T., Hinzmann, T., Rudin, K., Mantel, T., Melzer, A., Wawrzacz, B., Hitz, G. and Siegwart, R. Robotic technologies for solar-powered uavs: Fully autonomous updraft-aware aerial sensing for multiday search-and-rescue missions. *Journal of Field Robotics*, Vol. 35(4), 612–640, 2018.

Pinto, R., Zambetti, M., Lagorio, A. and Pirola, F. A network design model for a meal delivery service using drones. *International Journal of Logistics Research and Applications*, Vol. 23(4), 354–374, 2020.

Researchdive. Uav drone market. 2021. [Online; accessed 16-May-2022].

Sawadsitang, S., Niyato, D., Tan, P.-S. and Wang, P. Joint ground and aerial package delivery services: A stochastic optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20(6), 2241–2254, 2019.

Shah, S., Dey, D., Lovett, C. and Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. 2017.

Tareque, H., Hossain, M. S. and Atiquzzaman, M. On the routing in flying ad hoc networks. 1–9. 2015.

Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y. M. Scaled-yolov4: Scaling cross stage partial network. 2020.

Wikipedia contributors. Uav outback challenge — Wikipedia, the free encyclopedia. 2022. [Online; accessed 16-May-2022].

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J. and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. 2019.

Zhang, H., Cisse, M., Dauphin, Y. N. and Lopez-Paz, D. mixup: Beyond empirical risk minimization. 2017.