

Trabajo Fin de Máster

Máster en Ingeniería Electrónica, Robótica y Automática

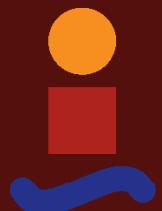
Multi-UAV system for human support in inspection operations with multiple views

Autor: Damián Jesús Pérez Morales

Tutor: Jesús Capitán Fernández, Alfonso Ángel Alcántara Marín

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021



Trabajo Fin de Máster
Máster en Ingeniería Electrónica, Robótica y Automática

Multi-UAV system for human support in inspection operations with multiple views

Autor:
Damián Jesús Pérez Morales

Tutor:
Jesús Capitán Fernández, Alfonso Ángel Alcántara Marín
Professor Contratado Doctor, Estudiante de Doctorado

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster: Multi-UAV system for human support in inspection operations with multiple views

Autor: Damián Jesús Pérez Morales

Tutor: Jesús Capitán Fernández, Alfonso Ángel Alcántara Marín

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Acknowledgement

Before anyone else, I would like to thank some members of the GRVC. Dr. Jesús Capitán Fernández, my tutor, for his assistance, for his kindness and for his valuable contribution to this project. I would also like to thank Alfonso Alcántara Marín, my teammate project, for his boundless patience, for his attention and for helping me to learn much more about this world. And lastly, I would like to thank Álvaro Calvo Matos, my schoolmate and my friend, for his endurance to stand my most difficult times with the project and along the school year.

I do not forget to express my gratitude to my whole family. Their incredible hope for me has encouraged me to be where I am now and I will be eternally thankful for it.

Last but not least, I have to show my infinite happiness and luck for having such good friends, to all of them.

Thanks for everything

*Damián Jesús Pérez Morales
Sevilla, 2021*

Abstract

This thesis presents a trajectory generation method for a multi-UAV team performing inspection operations in a wind turbine. The main objective is to provide different views of the wind turbine to a ground operator, giving better perspectives of what is being supervised. To make this possible, a formation is used that follows a leader-follower scheme in a decentralized way. Each UAV calculates an initial trajectory according to the operator's commands, maintaining visual properties. Afterwards, these paths are automatically optimized, minimizing accelerations in order to obtain smooth paths avoiding jerky movements. The presented method is evaluated in simulations and field experiments.

Resumen

Se presenta un generador de trayectorias para formaciones multi-UAV que monitorizan la misión de la inspección de una turbina eólica. El objetivo principal es ofrecer diferentes puntos de vista de la turbina de viento a un operador de tierra, ofreciendo mejores perspectivas de lo que se está supervisando. Para hacer esto posible, se emplea una formación que sigue un esquema líder-seguidor de forma descentralizada. Cada UAV calcula una trayectoria inicial a partir de los comandos del operador que mantienen las propiedades visuales. Seguidamente, estas trayectorias son optimizadas, minimizando aceleraciones para obtener caminos suaves y, de esta manera, evitar movimientos bruscos. Este método es evaluado con simulaciones y con experimentos de campo.

Short Outline

<i>Abstract</i>	III
<i>Resumen</i>	V
<i>Short Outline</i>	VII
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
2 Preliminaries	5
2.1 Technology evolution	5
2.2 Related work	6
2.3 Used tools	12
3 Methodology	15
3.1 System overview	15
3.2 Problem formulation	17
3.3 Implementation	20
4 Results	25
4.1 Simulation Environment	25
4.2 Simulation	26
4.3 Experiments in real life	31
5 Conclusions and future work	33
Appendix A Other generation of paths	35
A.1 V_{XY} cruising speed	35
<i>List of Figures</i>	39
<i>List of Tables</i>	41
<i>List of Codes</i>	43
<i>Bibliography</i>	45

Contents

<i>Abstract</i>	III
<i>Resumen</i>	V
<i>Short Outline</i>	VII
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
2 Preliminaries	5
2.1 Technology evolution	5
2.1.1 Renewable energy sources	5
2.1.2 UAVs	5
2.2 Related work	6
2.2.1 Inspection applications with UAVs	7
2.2.2 Trajectory planning in filming applications	8
2.2.3 Trajectory planning with multi-UAV systems	10
2.2.4 Augmented Reality	11
2.3 Used tools	12
2.3.1 Why ROS?	12
2.3.2 Gazebo	13
2.3.3 RViz	13
2.3.4 ACADO	13
2.3.5 MATLAB	14
3 Methodology	15
3.1 System overview	15
3.2 Problem formulation	17
3.2.1 Dynamic model	17
3.2.2 Generation of reference paths	18
3.2.3 Trajectory optimization	19
3.3 Implementation	20
3.3.1 Class diagram	20
3.3.2 Pseudocodes	21
3.3.3 ROS: possible interactions with the operator	23
4 Results	25

4.1	Simulation Environment	25
4.2	Simulation	26
4.3	Experiments in real life	31
5	Conclusions and future work	33
Appendix A Other generation of paths		35
A.1	V_{XY} cruising speed	35
A.1.1	Explanation of advantages and disadvantages	36
<i>List of Figures</i>		39
<i>List of Tables</i>		41
<i>List of Codes</i>		43
<i>Bibliography</i>		45

1 Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) have been recently growing for a wide spectrum of applications such as cinematographic shots [1], forest fire sighting [2], breakdown detection or inspection tasks [3]. They stand out for their reduced size and weight, allowing them to be used under challenging scenarios that require high manoeuvrability. Besides, UAVs can integrate different sensors onboard, such as a gimbal with RGB or thermal cameras, laser sensors, etc.

In principle, taking video shots with a UAV requires a well-trained operator to fly the vehicle and handle the camera movement and framing. The idea of conceiving UAVs that can accomplish these tasks autonomously is interesting to reduce human operators' burden. Besides, structural repair activities in complex areas (wind turbines, electrical towers) are carried out by humans. They usually need a supervisor on the ground who monitors the mission and commands high-level decisions. The images provided by UAVs could be of great use to the supervisor, for instance, in an inspection performed in a complex access area.

Multi-UAV teams can play an essential role in that application, having more advantages than a single UAV. A formation could increase the covering scene, providing images from different angles and reducing mission time, or increasing the robustness to robot failures. Teams of UAVs are already being used in multiple applications like target search [4], exploration [5], or surveillance [6], where operational times can be significantly reduced using multiples UAVs.



Figure 1.1 UAVs performing inspection tasks of wind turbines.



Figure 1.2 DURABLE logo.

This thesis focuses on a multi-UAV formation that gives extra visual information to operators inspecting the structure themselves or supervising the inspection operation to help and lead the process from the ground. The obtained images could even be integrated with Augmented Reality (AR) technologies in the future, creating a user interface that gives relevant information about the inspection operation to the ground operator. Through that interface, the operator could handle some functionalities or change the behaviour of the formation if needed during the inspection operation. Augmented Reality is being used in robotics as a new medium for interaction and information exchange with autonomous systems, increasing the Human-Robot Interaction (HRI) efficiency [7].

However, autonomous multi-UAV systems present several issues. One of them is collision-avoidance, with several vehicles in the air, it is necessary that they do not go into each others' safe zones. Besides, most coordination strategies are centralized, making them computationally intractable in real-time, so a decentralized approach is often necessary. In applications with cameras, it is also necessary not to overlap others' field of view. Due to these problems, there is still a need for multi-UAV autonomous systems that are intelligent enough to execute tasks coordinately, for instance, filming multiple action points in real-time.

If we add the problems mentioned above to the fact that UAVs with cameras need to navigate smoothly, the trajectory planning problem becomes complex. Traditional path planning techniques are not sufficient to satisfy the requirements of the aforementioned applications, since the paths generated only rely on spatial information, and they may not actually be performed due to the dynamics of the robot, or may not achieve the smoothness requirements of applications with cameras. A trajectory is a time parameterized function of robot states that contains both spatial and temporal information. Trajectories recover the full states of a dynamic system such that they are adequate to guarantee the system's feasibility, safety, and smoothness if it is required. Since system dynamics and time-varying states are taken into account, trajectory generation is more complicated than path planning. This thesis focuses on the trajectory planning problem for multi-UAV wind turbine inspection, where multiple cameras are involved.

The thesis is motivated by the objectives set forth in the scope of the EU-funded project DURABLE¹. The objective of DURABLE is the development of autonomous technologies concerned with inspection tasks on renewable energy sources, as wind turbines or solar panels.

¹ The main webpage of DURABLE is the following: www.durableproject.eu

1.2 Objectives

The global objective of this thesis is to develop a multi-UAV system which supports a wind turbine inspection operation providing different views of the wind turbine to an operator who is on the ground. The operator can send high-level commands to the formation in order to film different views of the inspected structure. For this purpose, the following sub-objectives are defined:

- Generating reference trajectories according to the high-level commands commanded by an operator. These trajectories should maintain certain formation and visual properties for the camera images.
- Generating optimal trajectories by defining an optimization problem that minimizes the accelerations in order to achieve smooth trajectories.
- Running the system online in real time. For this purpose, a receding horizon approach is used, and the system runs in a decentralized way. This allows recalculating trajectories at a high rate, to take into account disturbances.
- Developing a multi-UAV software architecture based on ROS.
- Performing *Software In The Loop* (SITL) simulations in a realistic environment, to prove that the method is able to generate trajectories for a formation of UAVs inspecting a wind turbine.

2 Preliminaries

This chapter focuses on the evolution of UAVs and renewable energy, two key technologies in this project. Both technologies have evolved quite fast over the last years and have significantly brought several benefits, especially in inspection applications. Besides, we present a related work citing publications about the technologies used in this thesis, i.e., inspection applications, trajectory planning in filming applications, trajectory planning with multi-UAV systems and Augmented Reality.

2.1 Technology evolution

2.1.1 Renewable energy sources

It is known that renewable energy sources have existed for a long time to generate electrical energy from natural resources (not fossil fuels): solar energy, wind energy, potential energy, biomass, among others. One of the most symbolic renewable energy sources in history could be the windmill or the watermill. These sources generate electrical energy from making rotations around the mill, either with wind energy or potential energy. With the evolution of technology and the concern about climate change, renewable energy sources have improved very fast, being the primary energy source in many countries. Nowadays, the most used sources are:

- Solar energy: solar panels
- Wind energy: wind turbines
- Potential energy: hydroelectric power plant
- Biomass: biomass plant

In the last year, 2020, wind energy has got the record of renewable energy production in Spain according to Red Eléctrica Española (REE). This entity manages the electrical energy in Spain, producing 21'7% of the total energy. In Figure 2.1, a graph with the energy produced in Spain is presented.

Due to recent technological improvements, it is possible to increasingly reduce the use of fossil fuels and, in the following years, green energy could cover almost the 100% of the whole consumption.

2.1.2 UAVs

Over the last decade, the use of UAVs has been continuously growing for a wide spectrum of applications [8, 9]. In some of these applications, UAVs collect data by using cameras or sensors,

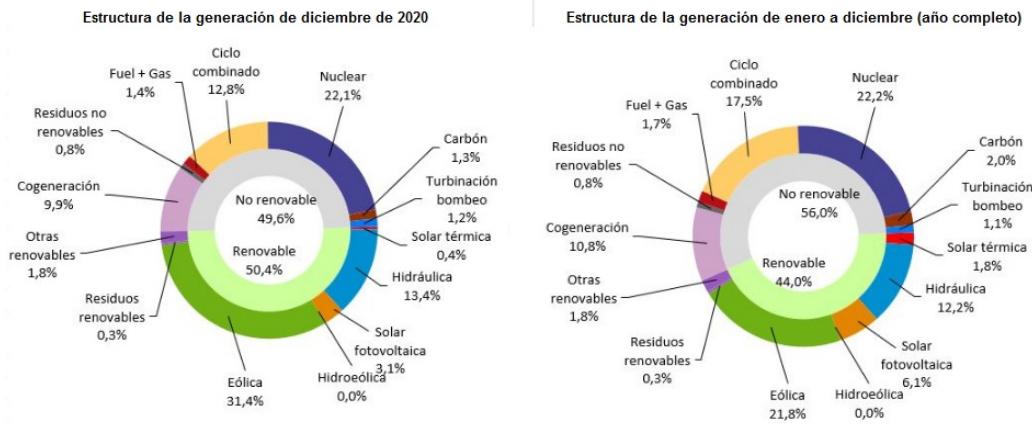


Figure 2.1 Total energy produced (in percent) in Spain (2020). Source: *energias-renovables.com*.



Figure 2.2 GB-1 Glide air bomb. Source: *eldrone.es*.

e.g., for filming or remote sensing [10, 11, 12, 13]; whereas in others, they are equipped with end-effectors to accomplish aerial manipulation tasks [14, 15, 16, 17].

The beginning of UAVs dates back to the XIX century, where UAVs were mainly used for military tasks, for instance, to bombard a target spot or for vigilance (see Figure 2.2). Basically, as a weapon. When Nikola Tesla discovered and created radio control in 1898, the applications diversified and got bigger, due to the interaction with the UAV via radio control.

However, even though the main uses throughout history were military, UAVs have also shown their positive side. Nowadays, UAVs' technology has evolved quite fast compared to what UAVs were before. Being now relatively small and light, UAVs have become the most powerful tool in many varied applications (see Figure 2.3). For instance, UAVs are now used in research [9], rescue [4], conservation [3], infrastructure inspection [18, 19], forest fire detections [2], among others.

2.2 Related work

Inspection tasks with UAVs can help to check the status of the object/building with the maximum number of points of view possible. This has a wide variety of applications. This section surveys different aspects related to UAV inspection technologies: inspection applications (Section 2.2.1), UAV filming (Section 2.2.2), trajectory planning (Section 2.2.3) and Augmented Reality (Section 2.2.4).



Figure 2.3 Parrot Anafi 4K Quadrotor. Source: parrot.com.



Figure 2.4 Quadrotor as a extinguishing tool. Source: futuristspeaker.com.

2.2.1 Inspection applications with UAVs

In the construction field, inspection of bridges is essential to keep its structural health and check wears in general. In [18], UAVs at bridge inspection perform a very important role as they can check every corner and every spot due to its manoeuvrability. UAVs also develop an important role in renewable energy sources. As mentioned before, solar panels are one of the most important renewable sources that we have, at least for now, and it is crucial to keep all PV modules working and detecting broken down cells as soon as possible. For instance, [19] developed a fault detection system using a single UAV and machine vision, which is basically working with a thermal camera and a Charge-Coupled Device mounted on a single UAV, that detects these faults and gives location information in terms of panel location by longitude and latitude. Also, in [20] faults on solar panels are detected with thermal infrared hitched on a single UAV that generates an orthographic image to make the inspection easier and distinguish accurately between normal and abnormal panel points.

As wind turbines are a large-sized structure, they need to be periodically checked, and the vibrations of the blades can determine the structural health. These vibrations are detected using Digital Image Correlation (DIC) and prevent catastrophic failures. UAVs have a remarkable potential for this work. In [21], the vibration of rotating wind turbines was detected using a single semi-autonomous vehicle. Surface damages in wind turbines have also been detected using UAVs with cameras and deep learning [22]. Figure 2.5 shows an example of such detection. In [23], they inspect blades using a single UAV with a LiDAR with minimal operator involvement. It is known that thunderstorms may affect in a bad way to all kind of electrical equipment, especially to those exposed outdoors, for instance, voltage towers and electrical transport elements in general. Wind turbines can

collect a huge load of electrostatic discharge (ESD), making the turbines not work right because of the induced electromotive force acting on the electrical system. In [24], this problem is contemplated and studied using tethered UAVs, but it is still being investigated. Regarding localization, GPS is not always available close to wind turbines. In [25], they perform UAV localization around wind turbines using monocular model-based tracking . This is possible by making a 3D skeleton of the wind turbine and matching image data by means of Convolutional Neural Network (CNN) and, then, fusing this information with GPS and IMU data to improve its accuracy.

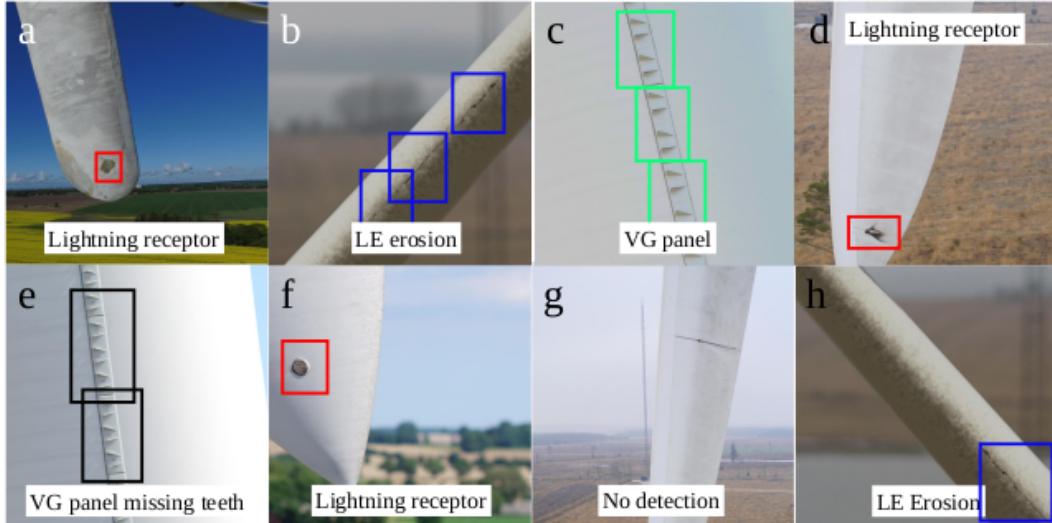


Figure 2.5 Detection of sensors and erosions of wind turbine’s blades. Source: [22]. They are able to detect the erosion of the blades, the VG panels even if there are missing teeth in them, the different sensors integrated in the wind turbine, etc. All this is detected with a high rate of success.

In many of the aforementioned works, the UAV inspection is done by the vehicle itself in an autonomous/semi-autonomous way (let us call that direct-inspect UAV), without the necessity of having a ground operator supervising its flight, as for instance in [21, 23]. However, UAVs could also give extra information to a ground operator that is supervising the operation from the ground [18, 22]. In such cases, human intervention is necessary and the UAV can play a key role monitoring the operation and allowing the operation to be supervised from ground. In this thesis, we focus on that type of application.

Most of the applications already mentioned are operated with a single UAV. However, a multi-UAV formation can offer multiple images from different points of view at the same time, giving extra information to a ground operator. In [26], they operate on a cinematography shot application with multiple UAVs that grants the director to have multiple views at the same time. Teams with multiple UAVs can provide alternative points of view, but also supplementary illumination. For instance, in [27], efficient variable illumination plays a key role for documentation of historical buildings interiors.

2.2.2 Trajectory planning in filming applications

In the computer animation community, there are several works related with trajectory planning for the motion of virtual cameras [28]. They typically use offline optimization to generate smooth trajectories that are visually pleasant and comply with certain cinematographic aspects, like the *rule of thirds*. However, many of them do not ensure physical feasibility to comply with UAV dynamic constraints and they assume full knowledge of the environment map. In terms of optimization

functions, several works consider similar terms to achieve smoothness. For instance, authors in [29] model trajectories as polynomial curves whose coefficients are computed to minimize snap (fourth derivative). They also check dynamic feasibility along the planned trajectories, and the user is allowed to adjust the UAV velocity at execution time. A similar application to design UAV trajectories for outdoor filming is proposed in [30]. Timed reference trajectories are generated from 3D positions specified by the user, and the final timing of the shots is addressed by designing easing curves that drive the UAV along the planned trajectory (i.e., curves that modify the UAV velocity profile). In [31], aesthetically pleasant footage is achieved by penalizing the snap of the UAV trajectory and the jerk (third derivative) of the camera motion. An iterative quadratic optimization problem is formulated to compute trajectories for the camera and the look-at point (i.e., the place where the camera is pointing at). They also include collision avoidance constraints, but the method is only tested indoors.

Although these articles on computer graphics approach the problem mainly through offline optimization, some of them have proposed options to achieve real-time performance, like planning in a *toric space* [32] or interpolating polynomial curves [33, 30]. In general, these works present interesting theoretical properties, but they are restricted to offline optimization with a fully known map of the scenario and static or close-to-static guided tour scenes, i.e., without moving actors.

In the robotics literature, there are works focusing more on filming dynamic scenes and complying with physical UAV constraints. For example, authors in [34] propose to detect limbs movement of a human for outdoor filming. Trajectory planning is performed online with polynomial curves that minimize snap. In [35, 12], they present an integrated system for outdoor cinematography, combining vision-based target localization with trajectory planning and collision avoidance. For optimal trajectory planning, they apply gradient descent with differentiable cost functions. Smoothness is achieved minimizing trajectory jerk; and shot quality by defining objective curves fulfilling cinematographic constraints associated with relative angles w.r.t. the actor and shot scale. Cinematography optimal trajectories have also been computed in real time through receding horizon with non-linear constraints [36]. The user inputs framing objectives for one or several targets on the image, and errors of the image target projections, sizes and relative viewing angles are minimized; satisfying collision avoidance constraints and target visibility. The method behaves well for online numerical optimization, but it is only tested in indoor settings.

Some of the aforementioned authors from robotics have also approached UAV cinematography applying machine learning techniques. In particular, learning from demonstration to imitate professional cameraman's behaviors [10] or reinforcement learning to achieve visually pleasant shots [11]. In general, most of these cited works on robotics present results quite interesting in terms of operation outdoors or online trajectory planning, but they always restrict to a single UAV. Regarding methods for multiple UAVs, there is some related work which is worth mentioning. In [37], a non-linear optimization problem is solved in a receding horizon fashion, taking into account collision avoidance constraints with the filmed actors and between the UAVs. Aesthetic objectives are introduced by the user as virtual reference trails. Then, UAVs receive current plans from all others at each planning iteration and compute collision-free trajectories sequentially. A UAV toric space is proposed in [38] to ensure that cinematographic properties and dynamic constraints are ensured along the trajectories. Non-linear optimization is applied to generate polynomial curves with minimum curvature variation, accounting for target visibility and collision avoidance. The motion of multiple UAVs around dynamic targets is coordinated by means of a centralized master-slave approach to solve conflicts. Even though these works present promising results for multi-UAV teams, they are only demonstrated at indoor scenarios where a *Vicon* motion capture system provides accurate positioning for all targets and UAVs. These works present quite valuable contributions for cinematography with multiple UAVs, but they are evaluated in indoor settings. The specifics of the outdoor scenarios considered in our work are different in several aspects, as the environment is less controlled: UAVs require more payload to carry onboard cameras with

better lenses and equipment for larger range communication; achieving smooth trajectories is more complex due to external factors such as wind gusts or communication delays; UAV positioning is less accurate in general; and so on.

2.2.3 Trajectory planning with multi-UAV systems

Multi-UAV trajectory planning aims to provide guidance information such as the optimized trajectories for the formation to benefit the coordination of multiple vehicles [?]. In addition, when computing the plan, apart from the costs considered in conventional planning, constraints specifically related to the formation need to be considered to facilitate the formation control. Thus, internal collision avoidance has to be considered to ensure the safety of the formation, and some formation behaviours like shape keeping is often desired. Moreover, it is usually necessary to achieve cooperative behaviour depending on the application. In [39], they provide a survey of multi-UAV trajectory planning for formation control and cooperative planning.

Regarding the formation control strategies, works can be classified in leader-follower formations, virtual formations or behaviour-based formations [39]. In the leader-follower approach, one vehicle works as the reference vehicle in the formation. Authors in [40] maintain a formation with a leader that is filming and the rest of the formation lighting the leader. They solve a special problem formulation for the leader and another formulation to maintain the formation and lighting accordingly. Virtual formations maintain a rigid geometric relationship to each other and to a frame of reference. The formation is maintained by minimizing the position error between the virtual structure and the actual formation position. In [?], a decentralized cooperative control scheme is presented allowing a team of UAVs to asymptotically converge to the desired formation of a particular shape and orientation from almost any initial conditions. Many multi-UAV MPC-based (Model Predictive Control) works minimize the distance to a reference path, enabling the possibility of maintaining precomputed virtual formations [41, 42]. A formation controlled by the behaviour-based approach has a more flexible formation shape. It generates the control commands based on various aspects of the mission. For instance, in applications with cameras, multi-UAV teams maintain a formation to film a scene from multiple perspectives, without entering the camera field of view of others and minimizing other costs related to the smoothness of the camera [36, ?]. In [40], the formation is generated and maintained by the leader-follower scheme, while the behaviour-based scheme specifically focused on the motion planning of individual vehicles.

Regarding the distribution of the computation, methods can be classified as centralized and distributed. Centralized methods solve a single problem for the whole team running the computation on a single computer. There is a central node that can access all the information from the vehicles [?]. In distributed methods, some algorithms require that some global information is available for all the UAVs, whereas, in others, UAVs are just required to access local information. The computation is distributed among the computers of each UAV. A centralized approach will be possible if the computational capabilities are compatible with the amount of information to process and the problem to solve, considering that the difficulty of solving scales poorly with the number of UAVs. Centralized approaches are usually better for stability. A distributed approach will be possible if the available knowledge within each distributed vehicle is sufficient to perform coherent decisions, and if this required amount of knowledge does not endow the distributed components with the inconveniences of a centralized system (in terms of computation power and communication bandwidth requirements) [?].

Authors in [43] solve the trajectory generation problem in a centralized way, formulating a *Mixed-Integer Quadratic Programming* (MIQP) problem. However, it lacks real-time performance due to the computational cost. Sequential Convex Programming (SQP) has been used in [44], leading to faster computations but still not in real-time. In order to improve computational efficiency, decoupled planning methods have been proposed [45, 46, 47]. These methods solved a centralized problem sequentially by avoiding the previously planned robots in a prioritized way, improving

computational efficiency. MPC-based approaches have been used for trajectory generation in centralized multi-UAVs architectures, taking advantage of receding horizon approaches to solve the problem in real time. In [48], a centralized leader-follower scheme is presented, where the leader executes the centralized NMPC approach and communicates with all followers to obtain their local sensor information and deliver propeller speed commands to them. A centralized method is also proposed in [49]. They compute the largest collision-free convex polytope in a neighborhood of the robots, followed by a constrained optimization via sequential convex programming. Authors in [50] first employ a graph-based multi-agent path planner to find an initial discrete solution, and then refine this solution into smooth trajectories using non-linear optimization. Although they compute trajectories in a centralized way, they divide the robot team into small groups and propose a prioritized trajectory optimization method to improve the algorithm's scalability. In [51], it is developed a two-stage system with long planning time horizons and real-time performance, using a centralized planner for formation control and a decentralized trajectory optimizer that runs on each robot.

The multi-robot motion coordination problem can also be solved in a decentralized way distributing the computational effort among the agents. Trajectory generation for distributed multi-UAV teams can generally be classified into (i) optimization-based and (ii) reactive. Many optimization-based works avoid non-convex constraints to enable real-time performance and scale with the number of UAVs. In [41, 42], each robot executes a local motion planning algorithm based on model predictive control and includes these non-linear potential field functions as constraints within a convex optimization framework. In [52], a potential function is also used to ensure inter-agent collision avoidance and connectivity, and the concept of consensus to acquire and maintain the desired formation pattern with velocity agreement among agents. The drawback in these potential-field based constraints is that they are more susceptible to deadlocks.

Some methods calculate a previous path geometrically to maintain the formation over a target and add inter-collision constraints in the problem formulation. In [53], they extend the centralized approach in [49] by means of a distributed consensus to compute the convex hull of the robot's position and the intersection of convex regions. In [54], they consider a set of motion primitives for the robot and solve an optimization problem in the space of control velocities with additional convex constraints. In [55], a convex optimization problem is formulated constructing relative safe flight corridors.

Some methods include non-convex constraints for collision avoidance. In [56], they include them into an NMPC decentralized problem. Still, they only activate them if a potential field algorithm cannot maintain a minimum distance. In [36], they formulate non-convex constraints with slack variables to maintain the formation, since the performance is enough for a few UAVs in the team. Authors in [57] introduce on-demand collision avoidance in a DMPC framework, where they detect and resolve only the first collision in the finite prediction horizon using sequential convex programming, modelling the collision as an ellipsoid and implementing soft constraints.

Many reactive approaches are based on the concept of *Velocity Obstacle* (VO). *Optimal Reciprocal Collision Avoidance* (ORCA) has been used to guarantee collision-free trajectories for non-holonomic agents [58]. The work in [59] proposes the concept of chance constraints to derive PRVO, a probabilistic variant of RVO (*Reciprocal Velocity Obstacle*). They take into account the uncertainty associated with both state estimation as well as the actuation of each robot. Reactive planning methods are computationally efficient, but suffer from guaranteeing no deadlock and are poorly suited to problems in maze-like environments.

2.2.4 Augmented Reality

Now, we are going to give a brief explanation of what is Augmented Reality (AR), Virtual Reality (VR) and Mixed Reality (MR).

Augmented Reality consists of superimposing information in real-time over real images through artificial graphics; whereas Virtual Reality recreates a virtual environment in every sense.

Mixed Reality is a novel technology that fuses both technologies. The user can see in the same frame real objects and virtual objects, and have the possibility of interacting with the last mentioned.

For this work, Augmented Reality can play a key role in providing information and spatial-sense in a supervised operation. Besides, thanks to Augmented Reality, people inspecting structures can view task instructions, checklists, troubleshooting procedures and get real-time video assistance from remote experts, all while keeping their eyes on the job and their hands free.

In [60], they present inspection with Mixed Reality (MR) headsets that tries to generate a 3D-2D correspondence from a 2D-2D image using a 3D point cloud. They obtain an image-based localization for efficient inspections using ground vehicles. In [61], UAV and AR technology are integrated to do soil sampling in precision agriculture. This application is oriented to make proper decisions regarding the fertilization of fields providing an user interface (UI) with virtual reality. In the AR headset, the user can see GNSS information, the soil colour of the region, the name of the field, etc. In [62], an AR tool for situational awareness improvement of UAV operators is presented. They add information to the camera images (e.g., waypoints to reach). In [63], a Mixed Reality (MR) system is implemented. The mixed reality interface allows people to provide landmarks that they can then refer to by using the natural language interface, which enables operators to command UAVs with higher flexibility. It is also possible to get a 3D map (rendered with AR) and demand some actions over it by head gaze and hand gestures [64]. This application is represented in Figure 2.6. A 3D interactive map is quite interesting for operators so that they can have the control and interact with autonomous UAVs. In [65], they use AR so that a human can see through obstacles to observe the robots' current states and intentions, and provide feedback. Then, the robots' behaviors are adjusted through human-multi-robot teamwork. They have developed a novel simulation environment using Unity (for AR and human simulation) and Gazebo (for robot simulation). Their simulation platform is open-source under the name of SUGAR2. The commercial product Skydio is recently developing several inspection applications with AR¹. On its webpage, there is an inspection task section² that provides extra information of the area using AR for autonomous roof inspection or autonomous asset inspection.

2.3 Used tools

In this section, they main tools and software frameworks used throughout the thesis are introduced.

2.3.1 Why ROS?

This project is developed in ROS³ (Robot Operating System), which is a collection of software frameworks (not a operating system itself) focused on robot software development. It provides services designed for a heterogeneous computer cluster, for instance: hardware abstraction, low-level device control, message-passing between different processes, among others. All these services are useful to develop the whole UAV system of this project, working with some helpful tools for simulation and visualization: Gazebo and RViz, which are normally used in a ROS simulation; and lately implemented in real life as well.

¹ <https://www.skydio.com>

² <https://www.skydio.com/inspection>

³ <https://www.ros.org>

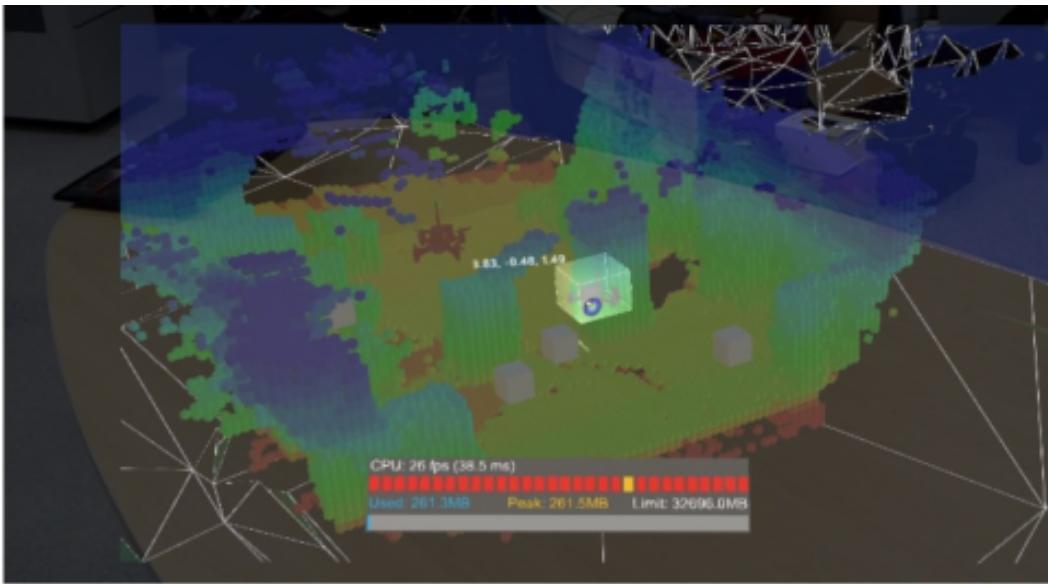


Figure 2.6 3D map construction with AR. Source: [64].



Figure 2.7 ROS logo. Source: ros.org.

2.3.2 Gazebo

Regarding Gazebo, it is an open-source 3D robotics simulator which is commonly used with ROS. It allows testing the whole programming of a robot system: algorithms, robot designs, training AI systems, etc. In the case of this project, it is useful in order to watch the functionality of the formation over a simulator.

2.3.3 RViz

RViz is a 3D visualization tool for ROS applications that provides many useful possibilities for debugging out robot systems. Thanks to RViz, it is possible to graphically visualize the information that the robot system is receiving, for instance, drawing the last 10 position measures received, in order to know if this sensor is working properly, plotting interesting waypoints, showing the map built by a LiDAR sensor, etc. For this project, it is interesting to use RViz in order to visualize the position and orientation of each UAV, to plot the waypoints of the path to be done by each UAV, to visualize the cylinder where the UAVs have to fly around, etc.

2.3.4 ACADO

ACADO⁴ (Automatic Control And Dynamic Optimization) is a software environment and algorithm collection for automatic control and dynamic optimization which is used as a linear MPC (*Model*

⁴ For more information: <https://acado.github.io> or [https://sourceforge.net/p/acado/wiki/MATLAB Interface](https://sourceforge.net/p/acado/wiki/MATLAB%20Interface)

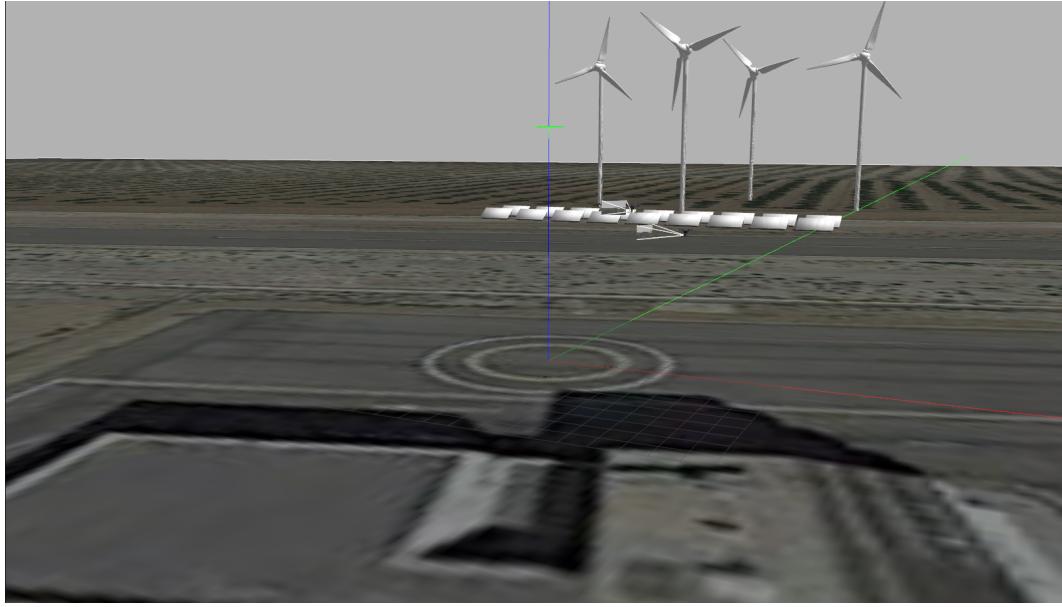


Figure 2.8 Gazebo simulator with an inspection scenario.

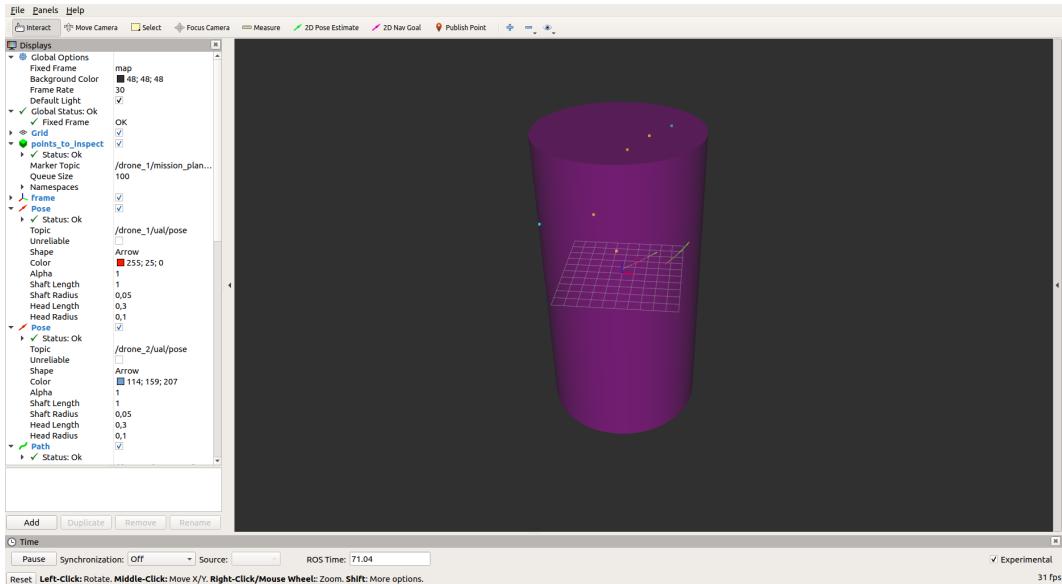


Figure 2.9 RViz visualization tool.

Predictive Control). For this project, it has been used to generate optimal trajectories given a problem formulation that intends to minimize accelerations in order to avoid jerky movements and increase smoothness. ACADO software is available on C++ and has a MATLAB interface as well.

2.3.5 MATLAB

MATLAB⁵ is a versatile software very used in engineering due to its quick learning curve and its potential. It has access to many toolboxes and libraries that are also used in the ROS framework, so it allows simpler debugging in all kinds of situations. For this project, MATLAB has been used at the beginning of the algorithm development, in order to test some problem formulations to optimize trajectories, before being implemented with ACADO and ROS.

⁵ <https://es.mathworks.com/products/matlab.html>

3 Methodology

In this chapter, we begin by explaining our system overview (Section 3.1). Then, our trajectory planning problem formulation is described. First, we detail the UAV dynamic model (Section 3.2.1). Afterwards, we describe our procedure to generate reference trajectories (Section 3.2.2). Lastly, we explain the generation of optimal trajectories (Section 3.2.3).

3.1 System overview

As mentioned in Chapter 1, this work aims to develop a system that covers a wind turbine inspection operation offering real-time images to a ground operator. The ground operator is able to send high-level commands to the UAV formation in order to get more convenient images. Figure 3.1 shows a top view scheme with the following operators commands:

- Next waypoint to inspect. The UAVs should go to this point that is commanded by the ground operator and the cameras on board that each UAV should point to the inspected point (see Figure 3.1). The operator can add new waypoints online to get images of different parts of the infrastructure and are stored on a list of waypoints that must be tracked in the same order that the operator commanded. When the leader UAV reaches a waypoint, this one is popped from the list.
- Formation angle. This angle is represented in Figure 3.1. The increment of this angle allows the operator to cover a wider scene.
- Inspection distance. The operator can also indicate the distance that the UAV team has to keep with respect to the point to inspect. This allows the operator to get images closer, more detailed, or further, covering a wider perspective.

The configuration that the multi-UAV team maintains depends on the parameters mentioned before. Our method generates trajectories for the multi-UAV team maintaining those properties. As result, the trajectories fit a cylinder of radius equal to the inspection distance and centered on the inspected point. Several works have generated trajectories using the arcball principle for efficient virtual camera control [66, 67, 68, 69]. These works define an arcball surface in polar coordinates centered on the target generating curved trajectories to improve visual properties.

Figure 3.2 depicts the architecture of the entire system. The leader UAV carries the main camera for filming, while the others carry cameras to provide supplementary images. A human operator specifies the inspection parameters explained above. This information is used to generate reference trajectories for the UAVs (see Section 3.2.2). These initial trajectories do not consider obstacle avoidance, but only consider the geometrical formation parameters indicated by the operator. The

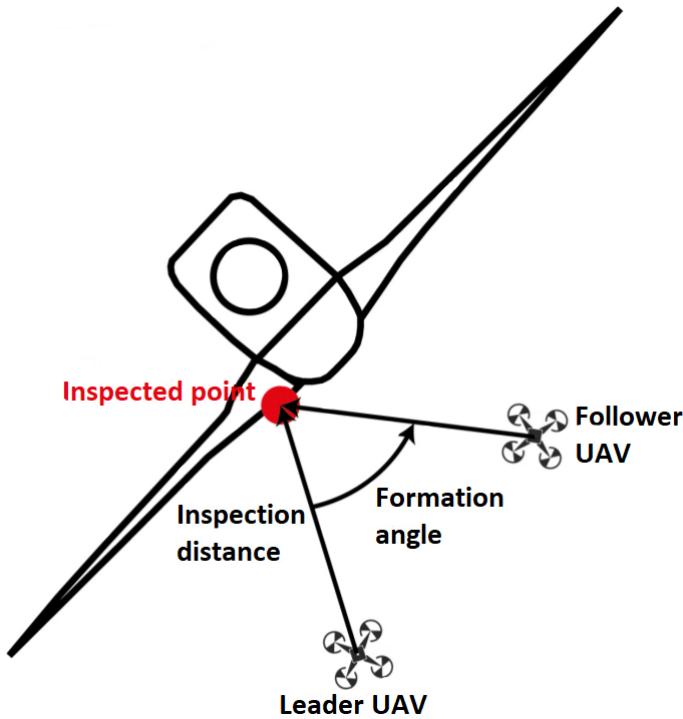


Figure 3.1 Top view of the leader-Follower scheme. Graphical representation of the relative angle, inspection distance and inspection point..

leader attempts to compute a reference trajectory formulated in polar coordinates, whereas the followers maintain a surrounding formation with the desired configuration. On the one hand, the trajectory generation procedure of the leader is shown in Algorithm 1. On the other side, the followers calculate their trajectories taking into account the leader trajectory and the formation parameters, in order to maintain the desired formation. This procedure is shown in Algorithm 2.

Algorithm 1 Leader UAV

```

1: procedure LEADER(Waypoint to reach, inspected point, inspection distance)
2:   System Initialization
3:   while Exist waypoint to reach && Mission is started do           ▷ For each cycle
4:     Reference path  $\leftarrow$  Waypoint to reach, inspected point, inspection distance
5:     Optimal path  $\leftarrow$  Reference path
6:   Output: Leader trajectory

```

Algorithm 2 Follower UAV

```

1: procedure FOLLOWER(Leader's reference path, formation angle)
2:   System Initialization
3:   while Exist leader's reference path do           ▷ For each cycle
4:     Input: Leader's reference path, formation angle
5:     Follower's reference path  $\leftarrow$  Rotate  $\pm$  formation angle Leader's reference path
6:     Optimal path  $\leftarrow$  Reference path
7:   Output: Follower trajectory

```

The entire pipeline shown in Figure 3.2 (except for the human operator interface) runs on board each UAV in a receding horizon manner. This enables online planning to react properly to changes

in the operator inputs, as well as to malfunctioning team-members or previously unseen obstacles. *Trajectory Planner* infers a planned trajectory that either the leader trajectory generator mentioned in Algorithm 1 or the follower trajectory generator mentioned in Algorithm 2 is activated on each UAV, depending on its role. The *Trajectory Follower* calculates 3D velocity commands at a higher rate so that the UAV follows the planned optimal trajectory, which is updated any time the *Trajectory Planner* generates a new solution. The *UAV Abstraction Layer* (UAL) is a software component developed by our lab [70] to interface with the position and velocity controllers of the UAV autopilot. It provides a common interface abstracting the user from the protocol of each specific autopilot hardware. Finally, recall that each UAV has a communication link with other teammates in order to share their current computed trajectories, which are used for multi-UAV coordination by the *Trajectory Planner*.

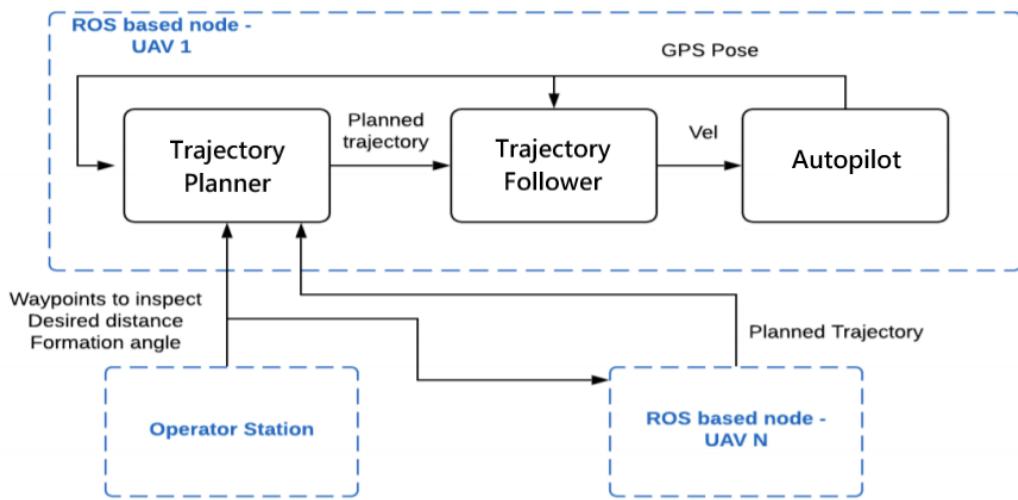


Figure 3.2 System overview scheme.

3.2 Problem formulation

This section describes the formulation of the optimization problems that are solved for trajectory generation, including the cost functions to be minimized and the involved constraints.

3.2.1 Dynamic model

Since we have a trajectory follower that allows us to follow the demanded paths with accuracy, we formulate a simple kinematic UAV model (using position, velocity and acceleration) to do not increase the computational cost. In addition, the camera's orientation needs to be modelled. For that, we assume the existence of a gimbal mechanism to compensate angle deviations due to changes in UAV attitude. Therefore, it is assumed that camera roll is negligible and we only control pitch and heading. Since the heading of a multi-rotor vehicle can be controlled independently of its position, we command the gimbal's pitch control and the UAV heading to point the camera to the inspected point.

The positional part of the dynamic model is defined as a linear double integrator:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= \mathbf{a}, \end{aligned} \tag{3.1}$$

where $\mathbf{p} = [p_x \ p_y \ p_z]^T \in \mathbb{R}^3$ is the UAV position, $\mathbf{v} = [v_x \ v_y \ v_z]^T \in \mathbb{R}^3$ the linear velocity, and $\mathbf{a} = [a_x \ a_y \ a_z]^T \in \mathbb{R}^3$ the linear acceleration. The camera position is assumed to have the same position as the UAV. The orientation of the camera may be modelled similarly:

$$\begin{aligned}\dot{\mathbf{o}} &= \boldsymbol{\omega}, \\ \dot{\boldsymbol{\omega}} &= \boldsymbol{\theta},\end{aligned}\quad (3.2)$$

where $\mathbf{o} = [\varphi \ \xi]^T$ represents an orientation with respect to a global frame given by its heading/yaw and pitch angles, $\boldsymbol{\omega} \in \mathbb{R}^2$ are the corresponding angular rates, and $\boldsymbol{\theta} \in \mathbb{R}^2$ the angular accelerations. For the description of the proposed method, we define a full positional state of the UAV $\mathbf{x}_p = [\mathbf{p}^T \ \mathbf{v}^T]^T \in \mathbb{R}^6$, a vector of positional control inputs $\mathbf{u}_p = \mathbf{a}$, an orientation state $\mathbf{x}_o = [\mathbf{o}^T \ \boldsymbol{\omega}^T]^T \in \mathbb{R}^4$, and a vector of orientation control inputs $\mathbf{u}_o = \boldsymbol{\theta}$.

3.2.2 Generation of reference paths

The leader reference paths are generated by means of a geometric procedure given the next waypoint to reach, the inspected point and the inspection distance, as mentioned in Algorithm 1, step 4. As a result, the path will fit a cylinder of radius equal to the inspection distance and centered in the inspected point. Thus, the leader will orbit around the inspected point, passing through the desired waypoints commanded by the human operator and maintaining visual properties: having the most stable camera vision possible, focusing on the inspected point. For that purpose, the reference trajectory is parametrized in cylindrical coordinates. A parametric interpolation is computed given a waypoint $w_0 = [r_0 \ \theta_0 \ z_0]$ and $w_1 = [r_1 \ \theta_1 \ z_1]$ and a parameter $t \in [0, 1]$. w_0 represents the current position and w_1 the desired waypoint to reach, both in cylindrical coordinates. The parametrized equations of the curve are:

$$\begin{aligned}r &= r_0 + (r_1 - r_0) * t \\ \theta &= \theta_0 + (\theta_{total}) * t \\ z &= z_0 + (z_1 - z_0) * t\end{aligned}\quad (3.3)$$

$$\theta_{total} = \begin{cases} \theta_2 - \theta_1 - 2\pi, & \text{if } \theta_2 - \theta_1 > \pi \\ \theta_2 - \theta_1, & \text{if } \theta_2 - \theta_1 < \pi \text{ and } \theta_2 - \theta_1 < 0 \\ \theta_2 - \theta_1 + 2\pi, & \text{if } \theta_2 - \theta_1 < -\pi \\ \theta_2 - \theta_1, & \text{if } \theta_2 - \theta_1 < \pi \text{ and } \theta_2 - \theta_1 > 0 \end{cases}\quad (3.4)$$

We know that if $t = 1$, the curve length L is:

$$L = \sqrt{(R^2(\theta_{total})^2 + (z_1 - z_0)^2)}\quad (3.5)$$

If we want to interpolate N points separated by t_{step} seconds at a cruising velocity V_c , applying the rule of three, for each point k we have its corresponding t_k :

$$t_k = \frac{V_c * t_{step} * k}{\sqrt{(R\theta_{total})^2 + (z_1 - z_0)^2}}\quad (3.6)$$

In the case of the followers, their reference path is a rotation of *relative_angle* radians around the Z axis of the leader UAV's path, as mentioned in Algorithm 2, step 5:

$$\mathbf{P}_{RF} = (\mathbf{P}_{RL} - \mathbf{p}_i) * R_Z(\text{relative_angle}),\quad (3.7)$$

where \mathbf{P}_{RF} is the reference path of the follower, which is a list of the reference points that the follower has to reach; \mathbf{P}_{RL} is the reference path (list) of the leader, \mathbf{p}_i is the inspected point and R_Z is:

$$R_Z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.8)$$

The reason of doing the subtraction of the \mathbf{p}_i to the \mathbf{P}_{RL} is to do a rotation of each point from the leader's reference path around the \mathbf{p}_i by relative_angle radians around the Z axis to obtain the \mathbf{P}_{RF} . In other words, is a translated list of points that is rotated by relative_angle radians around the Z axis.

3.2.3 Trajectory optimization

Our main objective is to generate optimal trajectories taking into account smoothness, as mentioned in Algorithm 1, step 5. Thus, we formulate an optimization problem minimizing the deviations with respect to the reference trajectory and accelerations in order to avoid jerky movements. Moreover, we add the UAV kinematic model and dynamic limitations as constraints. The whole problem formulation is the following that, for simplicity of description, we denoted $\mathbf{x} := \mathbf{x}_p$ and $\mathbf{u} := \mathbf{u}_p$:

$$\underset{\mathbf{x}_0, \dots, \mathbf{x}_N, \mathbf{u}_0, \dots, \mathbf{u}_N}{\text{minimize}} \sum_{k=1}^N (\|\mathbf{x}_{d,k} - \mathbf{x}_k\|^2 + \beta \|\mathbf{u}_{k-1}\|^2), \quad (3.9)$$

$$\text{subject to } \mathbf{x}_0 = \mathbf{x}' \quad (3.9.a)$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad k = 0, \dots, N-1 \quad (3.9.b)$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \quad (3.9.c)$$

$$\mathbf{v}_{min} \leq \mathbf{v}_{Q,k} \leq \mathbf{v}_{max} \quad (3.9.d)$$

$$\mathbf{p}_{Q,k} \in \mathcal{F} \quad (3.9.e)$$

where the optimization variables $\mathbf{x}_k = [\mathbf{p}_{Q,k} \ \mathbf{v}_{Q,k}]$ and $\mathbf{u}_k = \mathbf{a}_k$ are the discretized states (position and velocity) and the control inputs (acceleration) of the system in each t_k . $\mathbf{x}_{d,k}$ is a list of the desired reference points to reach by the UAV that comes from Section 3.2.2 and its size depends on the receding horizon length.

In case there is a predefined no-fly zone (e.g., around the wind turbine), a new constraint could be added to the optimization problem. However, this constraint is non-convex, which complicates quite a lot the optimization problem. Fortunately, we have an external module that generates reference paths (see Section 3.2.2) and this constraint can be considered implicitly in the generation of the reference paths. This way, the optimization solver does not consider a non-convex constraint and makes the optimization problem quicker and more robust to find solutions.

In 3.9.a, \mathbf{x}' is the current observed state value of the UAV, that is considered the initial state for the formulation \mathbf{x}_0 . The system dynamics are expressed in 3.9.b, while 3.9.c and 3.9.d indicate bounds for the UAV velocity and acceleration, respectively.

In this application, cameras need to always be pointing at the filmed target. Hence, their desired orientation is given by:

$$\mathbf{o}_d = [\varphi_d \ \xi_d]^T = \left[\arctan(q_y, q_x) \ \sin\left(\frac{q_z}{\|q\|}\right) \right]^T. \quad (3.10)$$

where $\mathbf{q} = [q_x \ q_y \ q_z] = \mathbf{p}_{UAV} - \mathbf{p}_i$, that is the relative position between the position of the camera (assuming that is the same of the UAV) and the inspected point, being q_x the relative position in x axis, q_y the relative position in y axis and q_z the relative position in z axis. With this information, the desired heading/yaw (φ_d) and pitch (ξ_d) orientation can be inferred.

Orientation control is also formulated as a constrained quadratic optimization problem in receding horizon in order to achieve smoother orientation changes. For simplicity of description, $\mathbf{x} := \mathbf{x}_o$ and $\mathbf{u} := \mathbf{u}_o$ in the following problem formulation:

$$\underset{\mathbf{u}_0, \dots, \mathbf{u}_{N-1}}{\text{minimize}} \sum_{k=1}^N (\|\mathbf{o}_{d,k} - \mathbf{o}_k\|^2 + \gamma \|\mathbf{u}_{k-1}\|^2), \quad (3.11)$$

$$\text{subject to } \mathbf{x}_0 = \mathbf{x}', \quad (3.11.a)$$

$$\mathbf{x}_{k+1} = f_o(\mathbf{x}_k, \mathbf{u}_k) \quad \forall k \in \{0, \dots, N-1\}, \quad (3.11.b)$$

$$\omega_{min} \leq \omega_k \leq \omega_{max} \quad \forall k \in \{1, \dots, N\}, \quad (3.11.c)$$

$$\xi_{min} \leq \xi_k \leq \xi_{max} \quad \forall k \in \{1, \dots, N\}, \quad (3.11.d)$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \quad \forall k \in \{0, \dots, N-1\}, \quad (3.11.e)$$

where $f_o(\cdot)$ represents the orientation aspect of the dynamic model defined in Section 3.2.1; ω_{min} , ω_{max} are limitations on the angular velocities; \mathbf{u}_{min} , \mathbf{u}_{max} control inputs limitations; and ξ_{min} , ξ_{max} represent hardware limitations of the gimbal to adjust pitch angles. The heading and pitch angles of the camera can be controlled independently. Thus, Problem (3.11) was decoupled into two simpler problems. The optimal solution for each problem can be found analytically with a standard framework for linear MPC (*Model Predictive Control*), proposed in [71].

The Trajectory Follower has a rate of 100 Hz which is the responsible of sending velocity commands to the Autopilot (see Figure 3.2) and Trajectory Planning is executed with a rate of 1 Hz, having a planned path anyway for 4 seconds along due to the receding horizon implemented (40 steps as receding horizon, N , with each step as 0.1 seconds). The receding horizon approach allows the UAV to have a path to follow for the next 4 seconds, which leads to smoother movements, as the path is replanned each second.

The followers' optimal paths, as mentioned in Algorithm 2, step 6, are computed in a similar manner. We use the same optimization problem, but using the follower's reference paths instead.

3.3 Implementation

In this section, we provide more details about the software implementation of the method: class diagram, pseudocodes and communications between modules. All code is open-source and available online¹, and was developed under Ubuntu 18.04 operating system and ROS Melodic.

3.3.1 Class diagram

Figure 3.3 depicts the class diagram for the software architecture developed in this project. It has four classes in total: Trajectory Planner, Trajectory Planner Leader (which inherits from Trajectory Planner), Trajectory Planner Follower (which inherits from Trajectory Planner), and Trajectory Planner Ros, which wraps the code adding ROS communications.

The idea behind defining the base class Trajectory Planner is that every UAV can inherit from it and implement its own version of the initial trajectory calculation and, if necessary, its own version of the optimization phase. Most of the methods regarding to the planning are similar and implemented in the Trajectory Planner class, therefore, we do not duplicate code with this implementation.

¹ https://github.com/grvcTeam/inspection_trajectory_planning

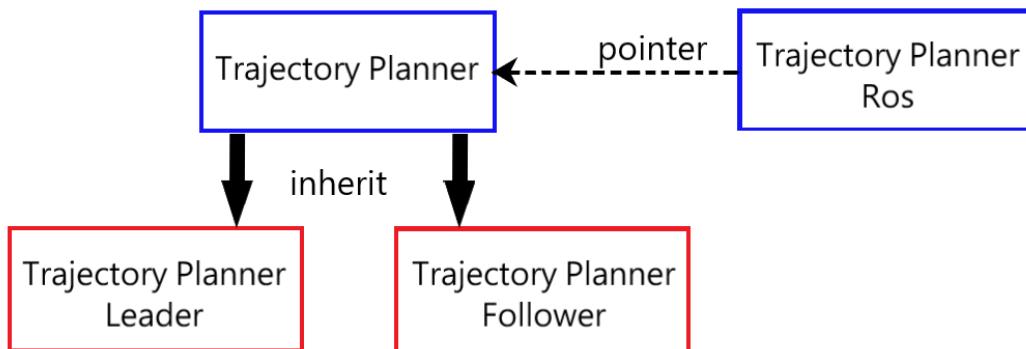


Figure 3.3 Class diagram. The blue colour on the edge symbolizes the base classes and the red colour symbolizes inherited classes.

The leader node instantiates a `Trajectory Planner Leader` object. This node executes several methods that check if there are waypoints to reach and the mission is started to generate a reference trajectory and later optimize it, sending optimized trajectories to the follower.

In the case of the follower node, it instantiates a `Trajectory Planner Follower` object which has its own implementation of the initial trajectory calculation method. The follower object calculates a rotation of the previous reference path done by the leader UAV around the Z axis to maintain the formation angle, and then, executes the optimization problem.

We have implemented the `Trajectory Planner Ros` class which wraps the code explained before adding ROS communication properties to each node (topics and services). This class contains a `Trajectory Planner` member that use virtual functions to wrap a leader or a follower depends on the node executed.

3.3.2 Pseudocodes

In this section, we show some pseudocodes describing the global mission process (Code 3.1), how the initial trajectories are inferred (Code 3.2), the leader node (Code 3.3), the follower node (Code 3.4) and the steps that the solver does to get the optimal path (Code 3.5).

1. Launch the UAV nodes and simulation (.launch)
2. Setup the configuration of the mission (parameters)
3. Wait until UAVs are armed and ready to start the mission
4. Take off all the UAVs
5. Start the mission
6. While (True)
 - 6.1. Read the waypoints not reached yet
 - 6.2. While there are waypoints yet to reach and the mission is started
 - 6.2.1. Calculate the UAVs' initial trajectory `initial_traj_to_follow`
 - 6.2.2. Calculate the UAVs' optimal trajectory `solved_trajectory`
 - 6.3. Wait until the mission is resumed

Code 3.1 Overall code functionality.

Then, we are going to mention in Code 3.2 the process of getting the initial trajectory mentioned in Section 3.2.2 path that is inferred in the leader UAV node.

```
1. Get the current position pc_xyz (Xc, Yc, Zc)
2. Get the desired position pd_xyz (Xd, Yd, Zd)
3. Transform both current and desired position in cylindrical
coordinates: pc_rtz (rho_c, theta_c, z_c), pd_rtz
(rho_d, theta_d, z_d)
4. Get the total theta_angle to go down from current position to
desired position
5. Get the total curve length L
6. For loop from parameter k = 0 to horizon_length - 1:
   6.1. Get parameter t_k in each iteration
   6.2. Get pk_rtz = (rho_k, theta_k, z_k)
   6.3. Transform pk_rtz to pk_xyz (Xk, Yk, Zk)
   6.4. Stack/push to the back pk_xyz to the vector initial_traj_to_
follow
```

Code 3.2 Initial trajectory pseudocode.

Once the initial trajectory is calculated, the leader UAV node only has left to optimize the path and send it to the autopilot (see Code 3.3).

```
1. Infer the initial trajectory
2. Send initial_traj_to_follow to optimize the path (solved_traj)
   optimal_traj = solved_traj
3. Send solved_traj to autopilot
```

Code 3.3 Optimal trajectory pseudocode for leader UAV.

As mentioned in Algorithm 2, the follower UAV's node gets the initial_traj_to_follow path (Code 3.2), rotates it relative_angle radians and, afterwards, it optimizes the path. The pseudocode is represented in Code 3.4.

```
1. Get leader's UAV reference_trajectory
2. Get the current relative_angle
3. Rotate relative_angle rad around Z axis leader's UAV reference_
trajectory
4. Send reference_trajectory to optimize the path
   optimal_traj = solved_traj
5. Send solved_traj to autopilot
```

Code 3.4 Optimal trajectory pseudocode for follower UAV.

It is important to insist that the UAVs' nodes are run in a decentralized way.

Now, it is important to mention how the optimal paths for the UAVs are generated in order to avoid the jerky movements that the reference trajectory path could have, minimizing the accelerations and generating velocities to send to each autopilot. The steps of the optimization problem are shown in Code 3.5.

```
1. Declaration of DifferentialStates, Controls and the
   DifferentialEquation
2. Define the model
```

3. Add constraints (maximum and minimum velocities/accelerations)
4. Define a start position, velocity and acceleration (got from the first element from initial_traj_to_follow)
5. Setup the reference trajectory to follow
6. Define the weights of the costs to minimize
7. Minimize LSQ
8. Setup the parameters of the solver
9. Solve the optimization problem
10. Get the solved_traj and send it back to the UAV

Code 3.5 Optimal trajectory (ACADO solver).

3.3.3 ROS: possible interactions with the operator

The ROS nodes implement several services to interact with the ground operator:

- Take off: it lets the UAV take off, giving a desired height to be set.
- Add waypoint: it adds a waypoint given by the console with (X, Y, Z) coordinates and is sent to a queue of goals to reach orderly.
- Clear waypoints: it cleans the whole queue of goals mentioned before.
- Changing the point to inspect: it lets you change the point to inspect. It could be useful if there are more than one wind turbine on the powerplant and need to check one by one.

Besides, we implement topics to publish commands with the joystick at high rate:

- Distance to inspection point (inspection distance): it lets you increase/decrease the distance to the inspection point.
- Relative angle: it lets you increase/decrease the relative angle between the leader UAV and the followers.

These two topics are controlled by a virtual joystick that has been implemented in the operator interface.

These services, in real life, are thought to be implemented with some kind of PAD or tablet set on the arm with a real joystick to operate those topics of distance and relative angles as well. The idea of the implementation is depicted in Figures 3.4 and 3.5.

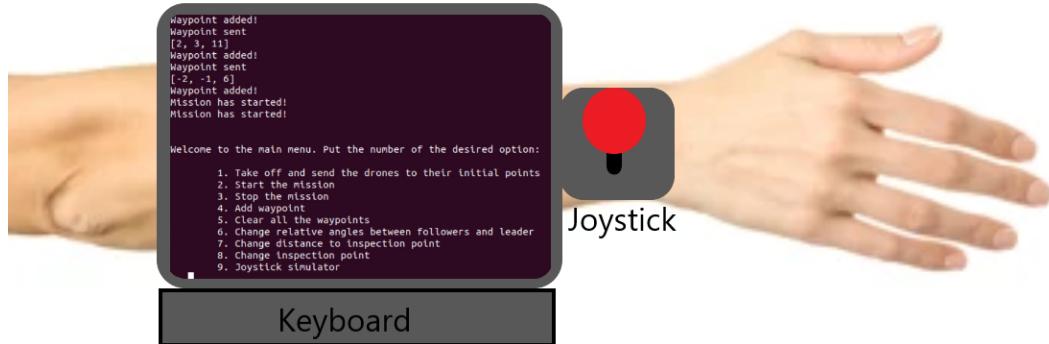


Figure 3.4 Operator interface aspect.

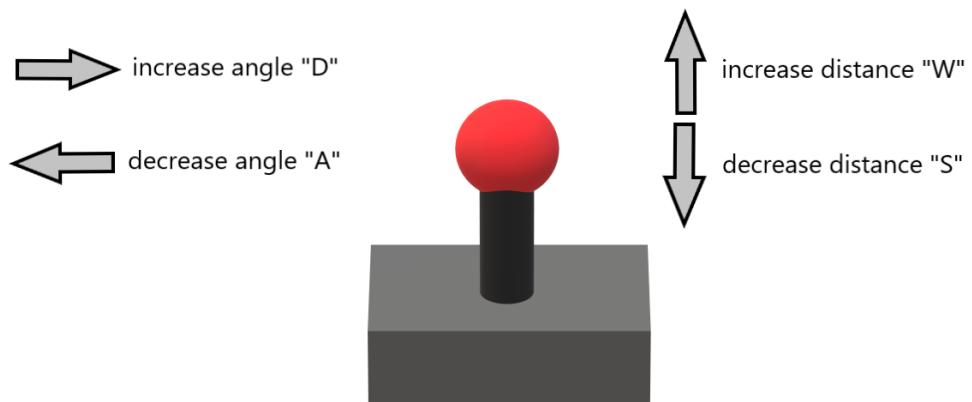


Figure 3.5 Joystick aspect.

4 Results

4.1 Simulation Environment

We used the simulator Gazebo which was mentioned in Section 2.3.2. We perform our real experiments in a basketball court at "La Plaza del Agua" (ETSI, Seville), which has a mesh installed for safety. We created a simulation environment of this court to ease the real world experiment, testing the algorithms in this simulator before performing field experiments. Figure 4.1 shows the simulated environment. We have also implemented a realistic environment of a wind turbine that can be seen in Figure 4.2.

We use RViz to visualize the reference and optimized trajectories of each UAV, pose of the UAVs, waypoints to inspect or the inspection structure. The appearance of the RViz interface can be seen in Figure 4.3:

The operator's interface has the possibility to run a determined auto-configuration file with some default waypoints to make either simulations or experiments quicker, without the necessity to adjust the parameters at the beginning.

In addition, in Figure 4.4 is represented the operator's interface menu, which can be used to interact with the formation.



Figure 4.1 Our field experiment facility simulated in Gazebo.



Figure 4.2 Scenario of the uploaded video.

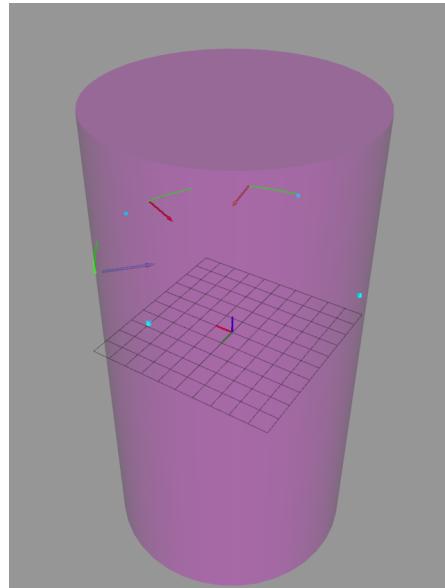


Figure 4.3 RViz in simulation.

4.2 Simulation

In this simulation, we inspect an hypothetical wind turbine on "Plaza del Agua". The ground operator will change the relative angle and the distance to the inspection point with the joystick simulator shown in Figure 4.5 in order to have different points of view, getting a closer view of the structure or moving away from it.

The ground operator has the control of the formation. Initially, some desired waypoints to reach are sent to the mission, a distance to the inspection point is demanded and a relative angle between the leader and the followers is commanded. After that, the UAVs take off and the mission starts. The parameters of the simulation are shown in Table 4.1, and the waypoints commanded are shown in Table 4.2. Is important to mention that the waypoints in Table 4.2 are automatically fitted in the

```
Welcome to the main menu. Put the number of the desired option:
0. Take off the drones
1. Start the mission
2. Stop the mission
3. Add waypoint
4. Clear all the waypoints
5. Change relative angles between followers and leader
6. Change distance to inspection point
7. Change inspection point
8. Joystick simulator
9. Land the drones
>> [ ]
```

Figure 4.4 Operator's interface, selection of the menu.

```
----- JOYSTICK SIMULATOR -----
To increase distance to inspection point, press W
To decrease distance to inspection point, press S
To increase the relative angle, press D
To decrease the relative angle, press A
To quit, press Q
```

Figure 4.5 Operator's interface, joystick simulator.

cylinder.

Table 4.1 Parameters of the simulation.

Parameter	Value	Unit
Horizon length	40	steps
Step size	0.1	seconds
Planning rate	1	seconds
Cruising speed	0.7	m/s
Max acceleration	1	m/s ²
Inspection distance	6.5	m
Relative angle	1	radian
Inspection point	[0, 0, 3.5]	m

In Figure 4.6, we show the evolution in time of the followed trajectories of each UAV, where Drone 1 refers to the leader UAV and Drone 2 and 3 are the follower UAVs. Notice that from $\Delta t = 30 \text{ s}$ to $\Delta t = 40 \text{ s}$ there is a change of the distance to the inspected point (also see Figure 4.7, where we can see a change of the distance to the inspected point in $\Delta t \approx 25 \text{ s}$), making the UAVs get away from the initial circumference that were describing and, in $\Delta t = 50 \text{ s}$, this change can be seen easier.

Table 4.2 Waypoints of the simulation.

Waypoint	Position [x, y, z] (m)
Waypoint 1	[0, 2, 5]
Waypoint 2	[2, 0, 5]
Waypoint 3	[0, -2, 5]
Waypoint 4	[-2, 0, 5]
Waypoint 1	[0, 2, 5]
Waypoint 2	[2, 0, 5]
Waypoint 3	[0, -2, 5]
Waypoint 4	[-2, 0, 5]

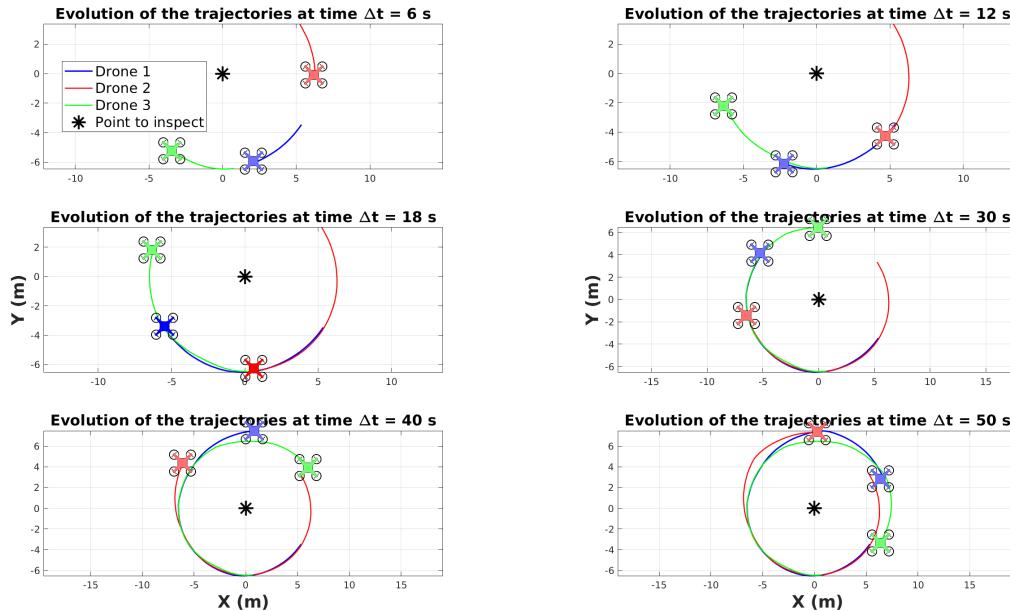


Figure 4.6 Simulation. UAVs path described mosaic. Notice that there is a change in the distance to the inspection point along the first turn over the inspection point. Drone 1 (leader) is between the other two UAVs which are followers (Drone 2, Drone 3) and they are describing a path always in the same direction.

Figure 4.7 represents the evolution of the distance to the inspection point of each UAV. Notice that there is a increasing change in $t \approx 161$ s from 6.5 m to ~ 7.35 m and the formation lasts to reach this distance approximately 15 s. Due to the formulation used in this work (see Section 3.2.2), the time that the formation needs to reach the desired distance to the inspection point is variable and depends on how far is the next waypoint to inspect from the last reached (parameter t_k in the formulation, see Section 3.2.2) and, in order to reach the next waypoint to inspect, the UAVs' tend to describe a spiral. The overoscillating behaviour comes from the necessity of detecting the desired waypoint when the UAV is not close enough (tolerance margin) and, when it is finally detected, the path to describe by the formation keeps being the same as the last one until a new desired waypoint to reach is able (no more publications of solved trajectories). In $t \approx 239$ s, there is another change of the distance to the inspection point, going down from ~ 7.35 m to ~ 6.7 m. The explanation done before is the same for this case.

In Figure 4.8, we show the evolution of the relative angles of the UAVs. In this case, there are still pending upgrades to polish the results: the implementation of a virtual target and refine the

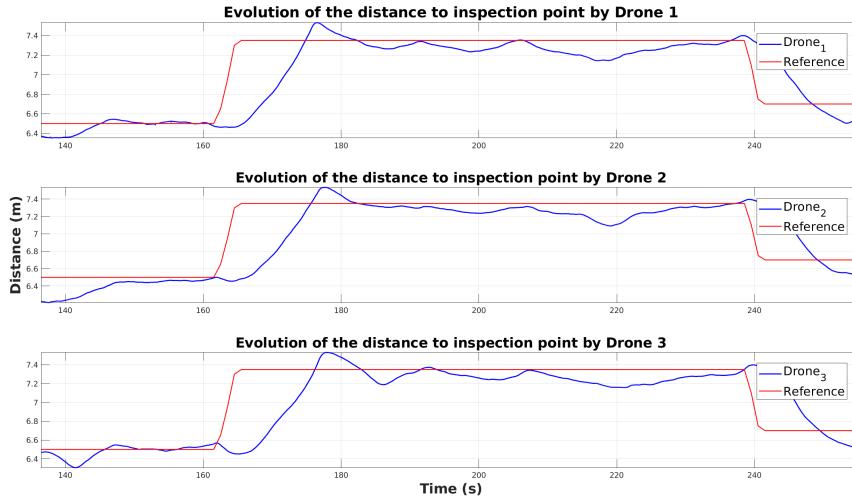


Figure 4.7 Simulation. UAVs distance to the inspection point tracking. Notice that in every change of reference, the UAVs seems to have an overoscillating behaviour. This is due to the tolerance given that could be a bit demanding and forces to pass quite close to the desired waypoints when a change of reference is given.

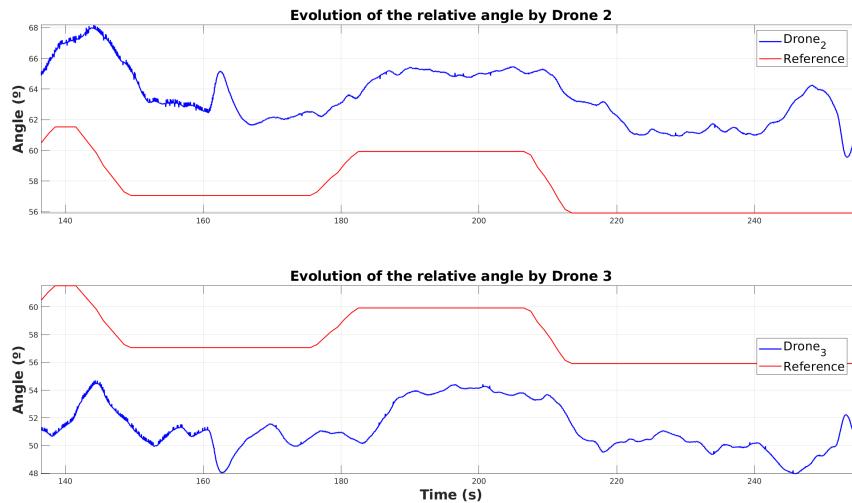


Figure 4.8 Simulation. UAVs relative angle tracking. In the top plot, it is depicted the evolution of the relative angle between the leader UAV (Drone 1) and the follower 1 UAV (Drone 2); while in the bottom plot, it is shown the evolution of the relative angle between leader UAV and the follower 2 (Drone 3). Both of them show the effect of having a lack synchronization of trajectories.

synchronization of the followers trajectories. Regarding the relative angle tracking, it could be seen that there is an unavoidable offset of some degrees ($\sim 6\text{-}8^\circ$) due to the leak synchronization made at this moment. When the follower UAV is behind the leader UAV, this offset is added to the relative angle, being quite behind from where it should be; whereas if the follower UAV is ahead of the leader UAV, this offset is subtracted from the real reference angle, being behind from where it should be. In any case, we demonstrate that the followers can maintain a formation with the formation angle command. With the mentioned upgrades, the UAV will avoid the mentioned offset.

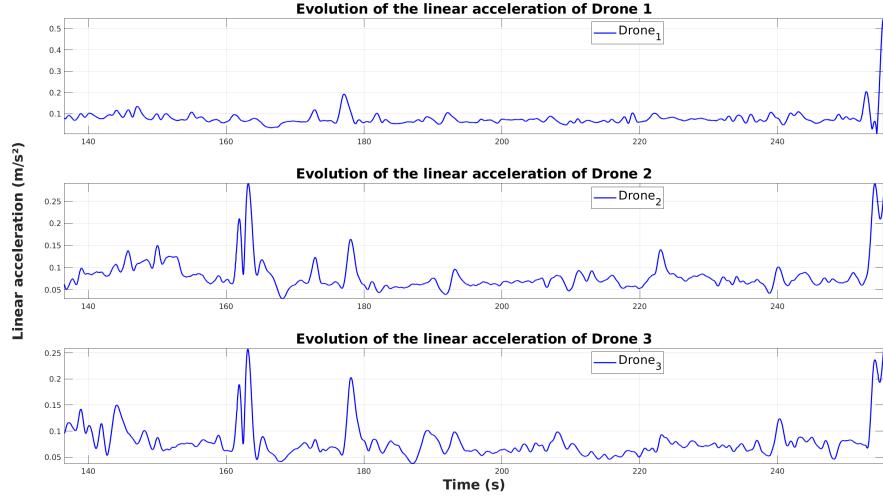


Figure 4.9 Simulation. UAVs linear acceleration of the optimized trajectories. While the operator commands are not changed, each UAV’s acceleration should tend to be the minimum possible due to there is a solver generating the trajectories that minimizes the accelerations. When there are significant peaks they could be due to: a change on the height of the waypoints (not the case); a change on the distance to the inspection point (it is normally smooth); or a change on the relative angles for the follower UAVs, that only concerns to themselves accelerations..

Linear acceleration of the UAVs are shown in Figure 4.9. As expected, due to the optimization of the path that the UAVs have to follow, these accelerations may tend to be close to 0 as it is describing a circumference, which means that some acceleration is needed to not follow a straight line. When there is a change of the relative angle, the accelerations of the followers (Drone 2 and Drone 3) may have some slight peaks to get closer or to move away from the leader UAV. Even having no variations on the relative angles or in the distance to the inspection points, the followers’ accelerations may be a little more irregular than the leader UAV as the followers’ path depends on the leader’s trajectory and they have to adapt it in order to maintain the relative angle with the leader UAV.

In the Figure 4.10 is shown the linear accelerations of the UAVs from another experiment where the trajectories were not optimized and have similar changes of reference in the inspection distance and in the relative angle. In these results, we can see that linear acceleration for the leader UAV does not vary too much in relation to the experiment with the optimized path. Nevertheless, there is a significant variation with the follower UAVs, having big peaks when there is a change whether in the inspection distance or in the relative angle, even when there is no changes in the references as the followers’ path depends on the leader’s path because, in this case, there is no optimization problem that minimizes the accelerations.

In Figure 4.11 and in Figure 4.12, yaw and pitch accelerations are depicted respectively. They both are thought to be close to 0 due to the solver that is minimizing the accelerations of the UAVs. This way, the camera views tends to be smooth.

A demonstration video of this work is uploaded on the platform YouTube¹. This video is developed in our wind turbine scenario (see Figure 4.2).

¹ <https://www.youtube.com/watch?v=pXQHwkDVizw>

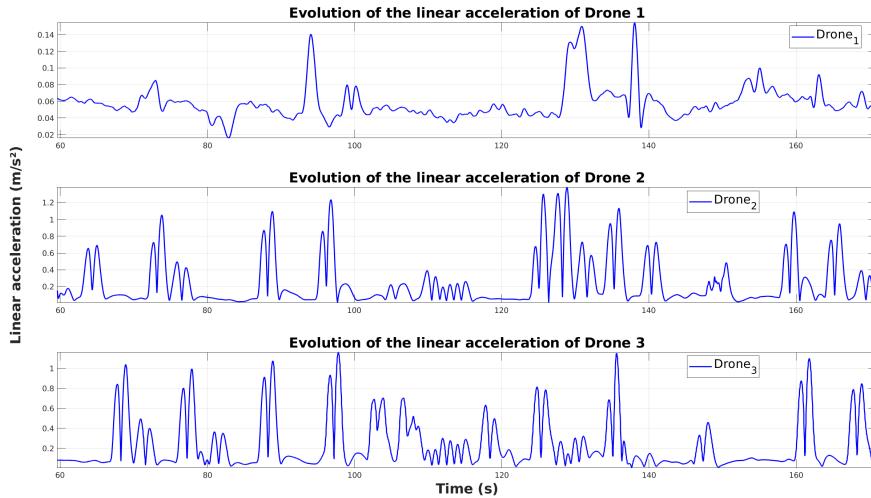


Figure 4.10 Simulation. UAVs linear acceleration without MPC. Comparing with the experiment with the optimized path (see Figure 4.9), there are significant peaks on the follower UAVs.

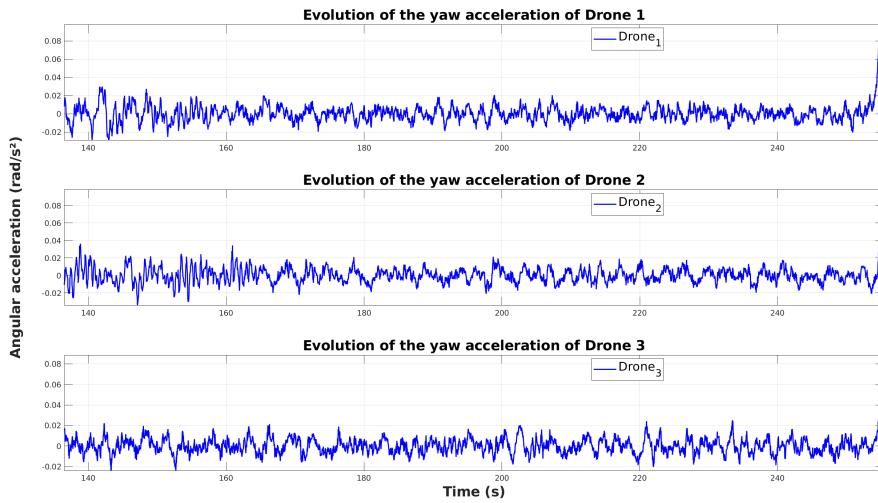


Figure 4.11 Simulation. UAVs yaw acceleration. The solver minimizes this acceleration in order to avoid shakiness and have smooth views on the UAVs' cameras. As could be seen, the accelerations on yaw are quite small.

4.3 Experiments in real life

NOT FINISHED The UAVs used in our experiments were like the one in Figure ???. They were custom-designed hexacopters made of carbon fiber with a size of $1.80 \times 1.80 \times 0.70\text{ m}$ and had the following equipment: a PixHawk 2 autopilot running PX4 for flight control; an RTK-GPS for precise localization; a 3-axis gimbal controlled by a BaseCam (AlexMos) controller receiving angle rate commands; a Blackmagic Micro Cinema camera; and an Intel NUC i7 computer to run our software for shot execution. The UAVs used Wi-Fi technology to share among them their plans and communicate with our Ground Station. Moreover, as explained in Section ???, our target carried a GPS-based device during the experiments, to provide positioning measures to the Target

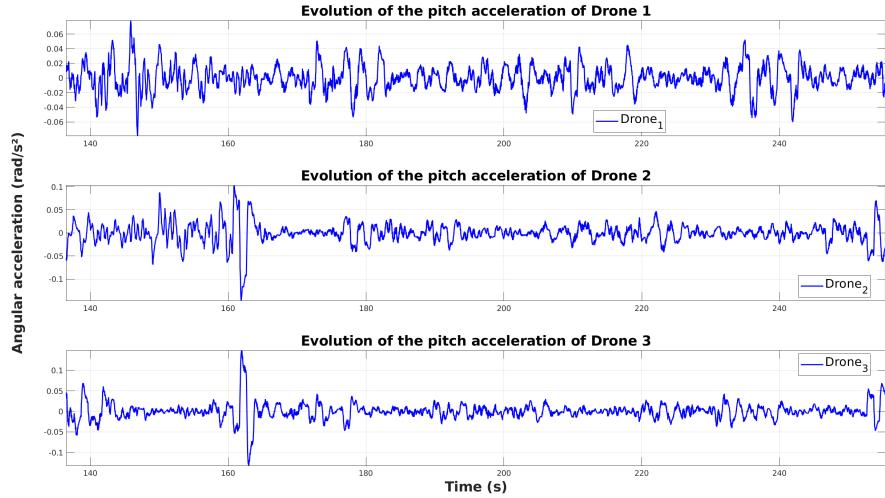


Figure 4.12 Simulation. UAVs pitch acceleration. This acceleration could vary principally due to the max velocity and max acceleration that is established on the configuration of the UAVs and on the solver.

```
Welcome to the main menu. Put the number of the desired experiment:

Experiment 0: simple path going twice around the inspection point (2 drones)
Experiment 1: simple path changing the direction once (2 drones)
Experiment 2: simple path changing the direction repeatedly (2 drones)
Experiment 3: simple path, but changing the height between waypoints (2 drones)
Experiment 4: complex path changing height and direction (2 drones)
Experiment 5: simple path going twice around the inspection point (3 drones)
Experiment 6: simple path changing the direction once (3 drones)
Experiment 7: simple path changing the direction repeatedly (3 drones)
Experiment 8: simple path, but changing the height between waypoints (3 drones)
Experiment 9: complex path changing height and direction (3 drones)

>> █
```

Figure 4.13 Operator's interface, selection of the desired experiment.

Tracker component on board the UAVs. The device weighted around 400 grams and consisted of an RTK-GPS receiver with a Pixhawk, a radio link and a small battery. This target provided 3D measurements with a delay below *100 ms*, that were filtered by the Kalman Filter on the Target Tracker to achieve centimeter accuracy. These errors were compensated by our gimbal controller to track the target on the image.

In the operator's interface there are multiple scenarios considered, but only one of these will be shown in this work in order to not overload it, that will be the simpler and most clarifying one (Exp 5 at the operator's interface, see it in Figure 4.13). In this case, it consists of three UAVs (one leader, two followers) that must follow a circular path that never changes its direction and also has a constant height. The leader will be located between the follower UAVs and will generate its own reference path that will be shared to the follower UAVs that will do a rotation of it (explained in Section 3.2.2).

In order to clarify it, this is the same experiment that has been run in the simulation.

As may be seen in Figure 4.4, there are a huge variety of options that the operator can select to interact with the formation. They are all self explanatory apart from the joystick simulator. It has been explained before, but not how is its interface on the terminal, so on the Figure 4.5 is depicted its aspect, which increments/decrements an amount of distance to the inspection point or relative angle from the previous value.

5 Conclusions and future work

This work has presented a method for autonomous multi-UAV trajectory planning for human support in inspection operations. We have proposed a novel methodology for multi-UAV trajectory planning, maintaining a desired formation depending on the operator commands and taking into account visual properties in order to get images from different points of view. We have demonstrated that the method minimizes the accelerations, obtaining smooth camera movements. Besides, we have performed Software In The Loop simulations, showing that we can plan trajectories in a distributed and online manner.

As future work, we plan to address synchronization of the trajectories received by the followers in order to approximate better to the formation angle reference. Besides, we plan to implement obstacle avoidance dodging the blades (among other things) of the turbine when the operator wants to have a closer view of the wind turbine.

We think that this method can be integrated with an Augmented Reality interface in order to give extra information to the operator about the mission. Thus, we plan to connect our system with the headset and user interface provided by the Ecole Supérieure des Technologies Industrielles Avancées (ESTIA) at Bidart, France. They have designed a virtual reality scenario where the operator can visualize several aspects of the mission, such as UAVs positions, camera images, inspected positions etc.

Finally, this project could be used to do other kind of inspection tasks, not only cylindrical structures, by making slight changes on the code. This last proposal is pending approval.

Appendix A

Other generation of paths

In this appendix, we are going to explain the V_{XY} path generation that was not included in this work, as mentioned in Section 3.2.2.

A.1 V_{XY} cruising speed

It is good to remember that the formulation is working with a constant-speed model and, in this case, keeps the direction of the previous path unless the drone is stopped. This means that each drone has to fly at a cruising speed (max velocity) V_{XY} that can be adjusted by the operator.

To know which sense is the drone following, see the equation A.1, A.2 and check it in case of doubts with the Figure A.1:

$$clockwise = \begin{cases} \text{quadrant 1} & \text{if } v_x > 0 \text{ and } v_y < 0 \\ \text{quadrant 2} & \text{if } v_x > 0 \text{ and } v_y > 0 \\ \text{quadrant 3} & \text{if } v_x < 0 \text{ and } v_y > 0 \\ \text{quadrant 4} & \text{if } v_x < 0 \text{ and } v_y < 0 \end{cases} \quad (\text{A.1})$$

$$anticlockwise = \begin{cases} \text{quadrant 1} & \text{if } v_x < 0 \text{ and } v_y > 0 \\ \text{quadrant 2} & \text{if } v_x < 0 \text{ and } v_y < 0 \\ \text{quadrant 3} & \text{if } v_x > 0 \text{ and } v_y < 0 \\ \text{quadrant 4} & \text{if } v_x > 0 \text{ and } v_y > 0 \end{cases} \quad (\text{A.2})$$

The problem is going to be focused to work with cylindrical coordinates:

$$\begin{aligned} x &= \rho \cos(\theta) \\ y &= \rho \sin(\theta) \\ z &= z \end{aligned} \quad (\text{A.3})$$

It is very important to mention before the explanation of each model of constant-speed that, if the velocity on XY is not constant, the angle of pass will not be either. In this case, as we are working with V_{XY} cruising speed, θ_{pass} would be constant.

$$\theta_{pass} = \frac{V_{xy} * steptime}{2\pi * R_{inspection}} * 2\pi \quad (\text{A.4})$$

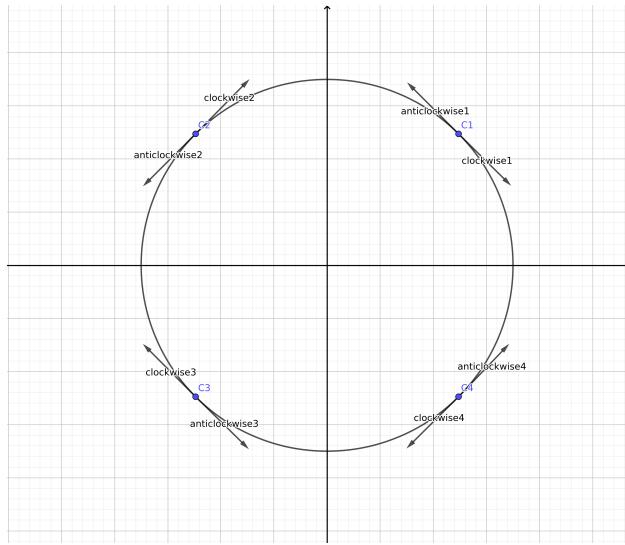


Figure A.1 Clockwise/anticlockwise graphic.

Thus, if the way the drone is going down is clockwise, θ_{pass} has to be subtracted to the current angle and, if not, θ_{pass} has to be subtracted to the current angle.

$$\theta_{next} = \begin{cases} \theta_{current} + \theta_{pass} & \text{if } \text{anticlockwise} \\ \theta_{current} - \theta_{pass} & \text{if } \text{clockwise} \end{cases} \quad (\text{A.5})$$

A.1.1 Explanation of advantages and disadvantages

There are two possibilities to focus this problem: considering on the max velocity parameter the XYZ components or break it down into a max velocity XY and other max velocity for the Z component.

The main problems of taking independently the velocity of XY and Z are:

$$\begin{aligned} V_{drone} &= \sqrt{V_{xy}^2 + V_z^2} \\ V_{xy} &= V_{max_{xy}} \end{aligned} \quad (\text{A.6})$$

- The drone is being forced to be flying at a constant speed on the XY plane, which means that in each step time it will have the same angle of pass. This may be a controversial point as it seems to be a good advantage because the drone claims stability, allowing smooth trajectories.
- However, the drone may have some difficulties if the next waypoint to reach is close on XY plane from the previous waypoint (or even in the same point on XY, but having different Z coordinate) and in the same direction (clockwise/anticlockwise) that the drone has been tracing along the previous path. It will make the drone go on a spiral path until it reaches the waypoint (which is not a bad solution though).
- If that were not enough, the velocity on Z cannot go at the max speed of its component in almost any cases and this, considering the velocity on XYZ from the module of XY and Z velocity, does not approve the requisite of going through the different points with a cruising speed.

Nevertheless, not everything is bad for the problem set out on independent velocity of XY and Z:

- The fact that the drone has a constant angle of pass due to the XY constant speed, causes throughout the drone's flight quite smooth changes between waypoints, as mentioned before.
- In some cases, it would be interesting to be going across the waypoints on spirals to have a full view of the whole inspection structure.

List of Figures

1.1	UAVs performing inspection tasks of wind turbines	1
1.2	DURABLE logo	2
2.1	Total energy produced (in percent) in Spain (2020). Source: energias-renovables.com	6
2.2	GB-1 Glide air bomb. Source: eldrone.es	6
2.3	Parrot Anafi 4K Quadrotor. Source: parrot.com	7
2.4	Quadrotor as a extinguishing tool. Source: futuristspeaker.com	7
2.5	Detection of sensors and erosions of wind turbine's blades. Source: [22]. They are able to detect the erosion of the blades, the VG panels even if there are missing teeths in them, the different sensors integrated in the wind turbine, etc. All this is detected with a high rate of success	8
2.6	3D map construction with AR. Source: [64]	13
2.7	ROS logo. Source: ros.org	13
2.8	Gazebo simulator with an inspection scenario	14
2.9	RViz visualization tool	14
3.1	Top view of the leader-Follower scheme. Graphical representation of the relative angle, inspection distance and inspection point.	16
3.2	System overview scheme	17
3.3	Class diagram. The blue colour on the edge symbolizes the base classes and the red colour symbolizes inherited classes	21
3.4	Operator interface aspect	24
3.5	Joystick aspect	24
4.1	Our field experiment facility simulated in Gazebo	25
4.2	Scenario of the uploaded video	26
4.3	RViz in simulation	26
4.4	Operator's interface, selection of the menu	27
4.5	Operator's interface, joystick simulator	27
4.6	Simulation. UAVs path described mosaic. Notice that there is a change in the distance to the inspection point along the first turn over the inspection point. Drone 1 (leader) is between the other two UAVs which are followers (Drone 2, Drone 3) and they are describing a path always in the same direction	28

4.7	Simulation. UAVs distance to the inspection point tracking. Notice that in every change of reference, the UAVs seems to have an overoscillating behaviour. This is due to the tolerance given that could be a bit demanding and forces to pass quite close to the desired waypoints when a change of reference is given	29
4.8	Simulation. UAVs relative angle tracking. In the top plot, it is depicted the evolution of the relative angle between the leader UAV (Drone 1) and the follower 1 UAV (Drone 2); while in the bottom plot, it is shown the evolution of the relative angle between leader UAV and the follower 2 (Drone 3). Both of them show the effect of having a lack synchronization of trajectories	29
4.9	Simulation. UAVs linear acceleration of the optimized trajectories. While the operator commands are not changed, each UAV's acceleration should tend to be the minimum possible due to there is a solver generating the trajectories that minimizes the accelerations. When there are significant peaks they could be due to: a change on the height of the waypoints (not the case); a change on the distance to the inspection point (it is normally smooth); or a change on the relative angles for the follower UAVs, that only concerns to themselves accelerations.	30
4.10	Simulation. UAVs linear acceleration without MPC. Comparing with the experiment with the optimized path (see Figure 4.9), there are significant peaks on the follower UAVs	31
4.11	Simulation. UAVs yaw acceleration. The solver minimizes this acceleration in order to avoid shakiness and have smooth views on the UAVs' cameras. As could be seen, the accelerations on yaw are quite small	31
4.12	Simulation. UAVs pitch acceleration. This acceleration could vary principally due to the max velocity and max acceleration that is established on the configuration of the UAVs and on the solver	32
4.13	Operator's interface, selection of the desired experiment	32
A.1	Clockwise/anticlockwise graphic	36

List of Tables

4.1	Parameters of the simulation	27
4.2	Waypoints of the simulation	28

List of Codes

3.1	Overall code functionality	21
3.2	Initial trajectory pseudocode	22
3.3	Optimal trajectory pseudocode for leader UAV	22
3.4	Optimal trajectory pseudocode for follower UAV	22
3.5	Optimal trajectory (ACADO solver)	22

Bibliography

- [1] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, “High-level multiple-uav cinematography tools for covering outdoor events,” *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.
- [2] C. Yuan, Z. Liu, and Y. Zhang, “Uav-based forest fire detection and tracking using image processing techniques,” in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2015, pp. 639–643.
- [3] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, “A uav system for inspection of industrial facilities,” in *2013 IEEE Aerospace Conference*. IEEE, 2013, pp. 1–8.
- [4] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, “Lsar: Multi-uav collaboration for search and rescue missions,” *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.
- [5] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, “Multi-uav exploration with limited communication and battery,” pp. 2230–2235, 2015.
- [6] J. Scherer and B. Rinner, “Multi-uav surveillance with minimum information idleness and latency constraints,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4812–4819, 2020.
- [7] A. van Wynsberghe and T. Comes, “Drones in humanitarian contexts, robot ethics, and the human–robot interaction,” *Ethics and Information Technology*, vol. 22, no. 1, pp. 43–53, 2020.
- [8] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of Unmanned Aerial Vehicles*. Springer Publishing Company, Incorporated, 2014.
- [9] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges,” *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.
- [10] C. Huang, Z. Yang, Y. Kong, P. Chen, X. Yang, and K.-T. Cheng, “Learning to capture a film-look video with a camera drone,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1871–1877.
- [11] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, “Can a robot become a movie director? learning artistic principles for aerial cinematography,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1107–1114.

- [12] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, “Autonomous aerial cinematography in unstructured environments with learned artistic decision-making,” *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, 2020.
- [13] L.-E. Caraballo, Á. Montes-Romero, J.-M. Díaz-Báñez, J. Capitán, A. Torres-González, and A. Ollero, “Autonomous planning for multiple aerial cinematographers,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2020. [Online]. Available: <https://arxiv.org/abs/2005.07237>
- [14] F. Ruggiero, V. Lippiello, and A. Ollero, “Aerial manipulation: A literature review,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.
- [15] A. Ollero and B. Siciliano, *Aerial Robotic Manipulators Research, Development and Applications*. Springer International Publishing, 2019.
- [16] A. Suarez, V. M. Vega, M. Fernandez, G. Heredia, and A. Ollero, “Benchmarks for aerial manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2650–2657, 2020.
- [17] A. Mohiuddin, T. Tarek, Y. Zweiri, and D. Gan, “A survey of single and multi-uav aerial manipulation,” *Unmanned Systems*, vol. 08, no. 02, pp. 119–147, 2020.
- [18] J. Seo, L. Duque, and J. Wacker, “Drone-enabled bridge inspection methodology and application,” *Automation in Construction*, vol. 94, pp. 112–126, 2018.
- [19] M. Alsafasfeh, I. Abdel-Qader, B. Bazuin, Q. Alsafasfeh, and W. Su, “Unsupervised fault detection and analysis for large photovoltaic systems using drones and machine vision,” *Energies*, vol. 11, no. 9, p. 2252, 2018.
- [20] J. Park and D. Lee, “Precise inspection method of solar photovoltaic panel using optical and thermal infrared sensor image taken by drones,” in *IOP Conference Series: Materials Science and Engineering*, vol. 611, no. 1. IOP Publishing, 2019, p. 012089.
- [21] A. Khadka, B. Fick, A. Afshar, M. Tavakoli, and J. Baqersad, “Non-contact vibration monitoring of rotating wind turbines using a semi-autonomous uav,” *Mechanical Systems and Signal Processing*, vol. 138, p. 106446, 2020.
- [22] A. Shihavuddin, X. Chen, V. Fedorov, A. Nymark Christensen, N. Andre Brogaard Riis, K. Branner, A. Bjørholm Dahl, and R. Reinhold Paulsen, “Wind turbine surface damage detection by deep learning aided drone inspection analysis,” *Energies*, vol. 12, no. 4, p. 676, 2019.
- [23] M. Car, L. Markovic, A. Ivanovic, M. Orsag, and S. Bogdan, “Autonomous wind-turbine blade inspection using lidar-equipped unmanned aerial vehicle,” *IEEE Access*, vol. 8, pp. 131 380–131 387, 2020.
- [24] J. Montanya, J. Lopez, P. Fontanes, M. Urbani, O. Van Der Velde, and D. Romero, “Using tethered drones to investigate esd in wind turbine blades during fair and thunderstorm weather,” in *2018 34th International Conference on Lightning Protection (ICLP)*. IEEE, 2018, pp. 1–4.
- [25] O. Moolan-Feroze, K. Karachalios, D. N. Nikolaidis, and A. Calway, “Improving drone localisation around wind turbines using monocular model-based tracking,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7713–7719.

- [26] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, “Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming,” in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 119–129.
- [27] P. Petracek, V. Kratky, and M. Saska, “Dronument: System for reliable deployment of micro aerial vehicles in dark areas of large historical monuments,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2078–2085, April 2020.
- [28] M. Christie, P. Olivier, and J. M. Normand, “Camera control in computer graphics,” *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008.
- [29] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, “An interactive tool for designing quadrotor camera shots,” *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 1–11, oct 2015. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2816795.2818106>
- [30] N. Joubert, J. L. E. D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, and P. Hanrahan, “Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles,” *ArXiv e-prints*, 2016.
- [31] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, “Airways: Optimization-Based Planning of Quadrotor Trajectories according to High-Level User Goals,” in *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, ACM. New York, New York, USA: ACM Press, 2016, pp. 2508–2519. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2858036.2858353>
- [32] C. Lino and M. Christie, “Intuitive and efficient camera control with the toric space,” *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 82:1–82:12, jul 2015.
- [33] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, “Automated Cinematography with Unmanned Aerial Vehicles,” *Eurographics Workshop on Intelligent Cinematography and Editing*, 2016.
- [34] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K. T. Cheng, “Act: An autonomous drone cinematography system for action scenes,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7039–7046.
- [35] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. Scherer, “Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments,” in *International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 229–236.
- [36] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, “Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, July 2017.
- [37] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, “Real-time planning for automated multi-view drone cinematography,” *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–10, jul 2017. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3072959.3073712>
- [38] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F.-l. Tariolle, and P. Guillotel, “Directing cinematic drones,” *ACM Trans. Graph.*, vol. 37, no. 3, Jul. 2018. [Online]. Available: <https://doi.org/10.1145/3181975>
- [39] Y. Liu and R. Bucknall, “A survey of formation control and motion planning of multiple unmanned vehicles,” *Robotica*, vol. 36, pp. 1019 – 1047, 2018.

- [40] V. Krátký, P. Petráček, V. Spurný, and M. Saska, “Autonomous reflectance transformation imaging by a team of unmanned aerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2302–2309, 2020.
- [41] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, and A. Ahmad, “Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios,” *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–8, 2018.
- [42] E. Price, G. Lawless, R. Ludwig, I. Martinovic, M. Black, and A. Ahmad, “Deep Neural Network-based Cooperative Visual Tracking through Multiple Micro Aerial Vehicles,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3193–3200, Oct. 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8394622/>
- [43] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.
- [44] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1917–1922.
- [45] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, “Prioritized planning algorithms for trajectory coordination of multiple mobile robots,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.
- [46] C. Yu, J. Wang, J. Shan, and M. Xin, “Multi-uav uwa video surveillance system,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016, pp. 1–6.
- [47] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, “An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1215–1222, 2018.
- [48] I. K. Erunsal, R. Ventura, and A. Martinoli, “Nonlinear Model Predictive Control for 3D Formation of Multirotor Micro Aerial Vehicles with Relative Sensing in Local Coordinates,” *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019. [Online]. Available: <http://arxiv.org/abs/1904.03742>
- [49] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, “Collision avoidance for aerial vehicles in multi-agent scenarios,” *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, jun 2015. [Online]. Available: <https://doi.org/10.1007/s10514-015-9429-0>
- [50] J. Li, M. Ran, and L. Xie, “Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 405–412, 2021.
- [51] C. Ho, A. Jong, H. Freeman, R. Rao, R. Bonatti, and S. Scherer, “3D Human Reconstruction in the Wild with Collaborative Aerial Cameras,” 2021.
- [52] A. Mondal, C. Bhowmick, L. Behera, and M. Jamshidi, “Trajectory Tracking by Multiple Agents in Formation With Collision Avoidance and Connectivity Assurance,” *IEEE Systems Journal*, 2017. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85040082928&doi=10.1109%2FJST.2017.2778063&partnerID=40&md5=e40bf3aca4c1659f494cb6e394b3be32>

- [53] J. Alonso-Mora, S. Baker, and D. Rus, “Multi-robot navigation in formation via sequential convex programming,” *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, pp. 4634–4641, 2015.
- [54] J. Alonso-Mora, P. Beardsley, and R. Siegwart, “Cooperative Collision Avoidance for Non-holonomic Robots,” *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.
- [55] J. Park and H. J. Kim, “Online Trajectory Planning for Multiple Quadrotors in Dynamic Environments Using Relative Safe Flight Corridor,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 659–666, 2021.
- [56] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, “Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, 2017, pp. 236–243. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041954247&doi=10.1109%2FIROS.2017.8202163&partnerID=40&md5=5ad0219879fbdbd1dbb39c702613232b>
- [57] C. E. Luis and A. P. Schoellig, “Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.
- [58] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 203–216. [Online]. Available: https://doi.org/10.1007/978-3-642-32723-0_15
- [59] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, “PRVO: Probabilistic Reciprocal Velocity Obstacle for multi robot navigation under uncertainty,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, 2017, pp. 1089–1096. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041949927&doi=10.1109%2FIROS.2017.8202279&partnerID=40&md5=20e7bcc5599c8ce06191ee67796b551e>
- [60] M. P. Das, Z. Dong, and S. Scherer, “Joint point cloud and image based localization for efficient inspection in mixed reality,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6357–6363.
- [61] J. Huuskonen and T. Oksanen, “Soil sampling with drones and augmented reality in precision agriculture,” *Computers and Electronics in Agriculture*, vol. 154, pp. 25–35, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168169918301650>
- [62] S. Ruano, C. Cuevas, G. Gallego, and N. García, “Augmented reality tool for the situational awareness improvement of uav operators,” *Sensors*, vol. 17, no. 2, p. 297, 2017.
- [63] B. Huang, D. Bayazit, D. Ullman, N. Gopalan, and S. Tellex, “Flight, camera, action! using natural language and mixed reality to control a drone,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6949–6956.
- [64] C. Liu and S. Shen, “An augmented reality interaction interface for autonomous drone,” *arXiv preprint arXiv:2008.02234*, 2020.
- [65] K. Chandan, V. Kudalkar, X. Li, and S. Zhang, “ARROCH : Augmented Reality for Robots Collaborating with a Human,” no. Icra, pp. 3787–3793, 2021.

- [66] K. Shoemake, “Arcball: A user interface for specifying three-dimensional orientation using a mouse,” in *Graphics interface*, vol. 92, 1992, pp. 151–156.
- [67] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F. o.-l. Tariolle, and P. Guillotel, “Directing cinematographic drones,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 3, pp. 1–18, 2018.
- [68] A. Bucker, R. Bonatti, and S. Scherer, “Do you see what i see? coordinating multiple aerial cameras for robot cinematography,” *arXiv preprint arXiv:2011.05437*, 2020.
- [69] C. Lino and M. Christie, “Intuitive and efficient camera control with the toric space,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–12, 2015.
- [70] F. Real, A. Torres-González, P. R. Soria, J. Capitán, and A. Ollero, “Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 4, pp. 1–13, 2020.
- [71] V. Krátký, A. Alcántara, J. Capitán, P. Štěpán, M. Saska, and A. Ollero, “Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7580–7587, 2021.