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
Profesores Titulares

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

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 decentralizada. 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

Contents

1 Introduction

1.1 Motivation

Nowadays, *Unmanned Aerial Vehicles* (UAVs) has been continuously 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 light-weight sensors onboard, such as gimbal with RGB or thermal cameras, laser sensors, etc.

In principle, taking video shots with a drone requires a well-trained operator to fly the vehicle and handle camera movement and framing. The idea of conceiving drones 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 being 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.

Multiple UAVs 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

Figure 1.1 UAVs at inspection tasks of wind turbines.

Figure 1.2 DURABLE logo. Font: *durableproject.eu/*.

being used in applications: target search [4], exploration [5] or surveillance [6] where operational times can be significantly reduced using multiples UAVs.

This thesis focuses on a Multi-UAV formation that gives extra visual information to the operator that can be inspecting the structure itself or supervising the inspection operation to help and lead the process from the ground. Besides, those images could even be integrated with Augmented Reality (AR) in a 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 functionality or change the behaviour of the formation if it would be desired for monitoring the inspection operation. Augmented Reality (AR) 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-UAVs 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' buffer 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 to not overlap the field of view of others. 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 requirement of the aforementioned applications, since the path generated relies only on spatial information and may not actually be performed due to the dynamics of the robot or cannot take into account the smoothness requirement 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 the system dynamics and time-varying states are taken into account, trajectory generation is more complicated than path planning. In this thesis, we focus on the trajectory planning problem in multi-UAV wind turbine inspection where multiple cameras are involved.

This thesis is motivated by the objectives set forth in the scope of the EU-funded project DURABLE [1]. The objective of DURABLE is to develop of the renewable-technology mainly concerned with inspection tasks on renewable energy sources, as wind turbines or solar panels.

[1] The main webpage of DURABLE is the following: durableproject.eu/

1.2 Objectives

The global objective of this project 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 objectives are defined:

- Generate optimal trajectories by defining an optimization problem that minimizes the accelerations in order to achieve smooth trajectories.
- Run the system online and in real-time. For this purpose, we use a receding horizon approach and run the system in a decentralized way. This allows recalculating at a high rate to take into account disturbances.
- Develop a multi-UAV software architecture based on ROS.
- Perform software in the loop simulations (SITL) to prove that our system is able to generate trajectories for a formation of UAVs that is inspecting a wind turbine.

2 Preliminaries

2.1 Technology evolution

This chapter focus 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 project, i.e., inspection applications, trajectory planning in filming applications, trajectory planning with multi-UAV systems and Augmented Reality.

2.1.1 Renewable energy sources

It is known for a long time that renewable energy sources have existed to generate electric 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 electric 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 electric 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 technological improvement, it is possible to go without increasingly scarce fossil fuels and, in the following years, we could produce almost 100% green energy.

2.1.2 UAVs

Over the last decade, the use of UAVs has been continuously growing for a wide spectrum of applications [**?**, **?**]. In some of these applications, UAVs collect data by using cameras or sensors, e.g., for filming or remote sensing [8, 9, **?**, 10]; whereas in others, they are equipped with end-effectors to accomplish aerial manipulation tasks [**?**, **?**, **?**, **?**].

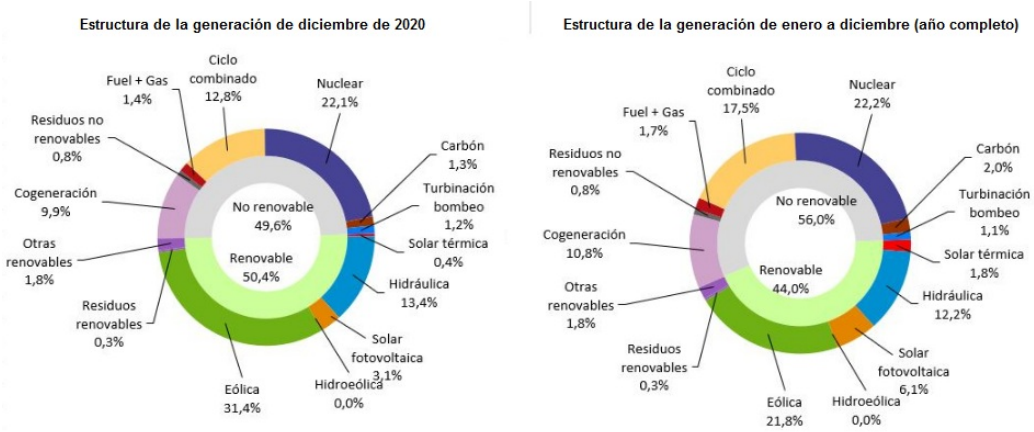


Figure 2.1 Total energy produced (in percent) in Spain (2020). Font: *energias-renovables.com/*.

Figure 2.2 Parrot Anafi 4K Quadrotor. Font: *parrot.com/*.

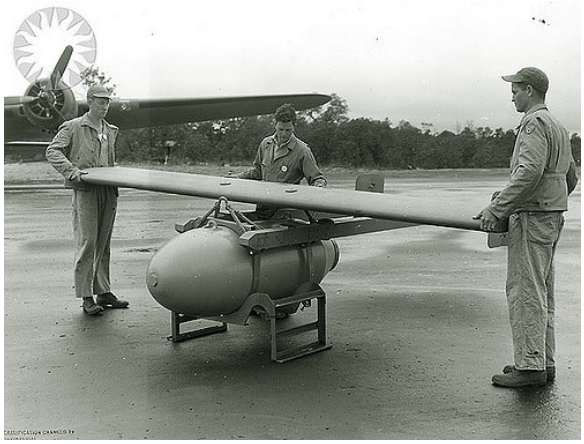


Figure 2.3 GB-1 Glide air bomb. Font: *eldrone.es/*.

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. 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 warlike, UAVs has shown their positive side. Until now, UAVs' architecture has evolved quite fast if we compare it to what UAVs were before. Being now relatively small and light, UAVs have become the most powerful tool in

Figure 2.4 Quadrotor as a extinguishing tool. Font: *futuristspeaker.com/*.

many applications.

UAVs are now used in research, rescue, conservation, infrastructure inspection, forest fire detections, among others.

2.2 Related work

Inspection tasks with UAVs are necessary to check the status of the object/building with the most points of view if possible. It has a huge variety of applications. We are going to make a breakdown of the different fields on inspection: Filming (Section 2.2.2), Inspection applications (Section 2.2.1), Augmented Reality (Section 2.2.4) and Trajectory planning (Section 2.2.3).

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 [11], 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 of the PV modules working and detecting broken down cells as soon as possible. For instance, [12] 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 (CCD) mounted on a single UAV that detects these faults and gives location information in terms of panel location by longitude and latitude. Also, in [13] 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, it needs 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 prevents catastrophic failures. UAVs have a remarkable potential for this work. In [14], the vibration of rotating wind turbines was detected using a single semi-autonomous. Surface damages in wind turbines are also detected using UAVs with cameras and deep learning [15]. Figure 2.5 shows an example of such detection. In [16], they inspect blades using a single UAV with LiDAR with minimal operator involvement. It is known that thunderstorms may affect in a bad way to all kind of electric equipment, especially to all of them that are exposed outdoors, for instance, voltage towers and electric 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 [17], this problem is contemplated and studied using tethered UAVs, but is still being investigated. Regarding localization, GPS is not always available close to wind turbines. In [18], they perform UAV localization around wind turbines using monocular model-based tracking . This is possible making a 3D skeleton of the wind turbine and doing matches to the image data with convolutional neural network (CNN) and, then, fusing this information with GPS and IMU to improve its accuracy.

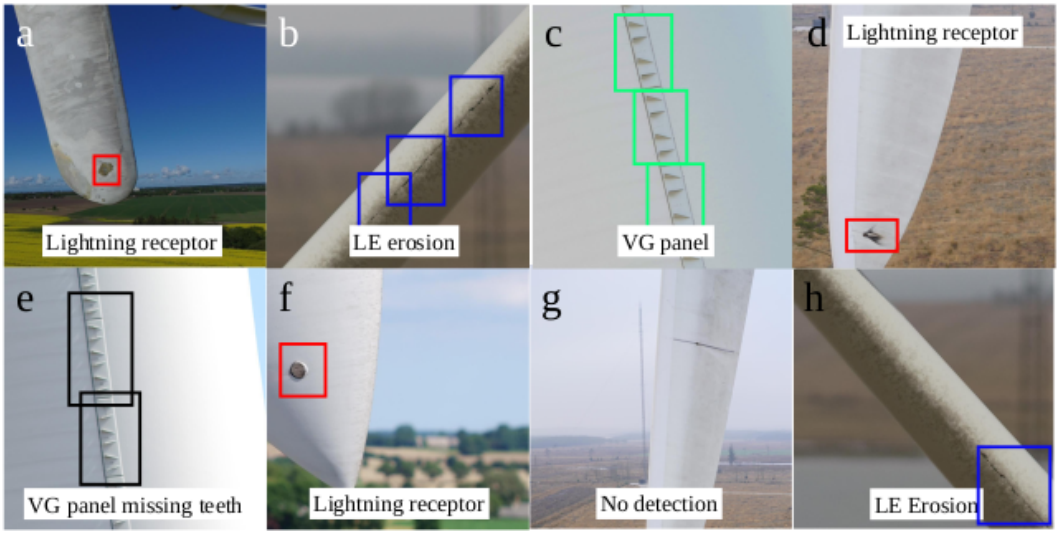


Figure 2.5 Detection of sensors and erosions of the wind turbine's blades. Font: [15]. It reveals the training sets done in this work that is able to detect the erosion of the blades, the VG panels even if there are missing teeths in them, the different sensors integrated in the wind turbine, among other things. All of this is detected with a high rate of success.

In many of these articles that were shown before, the UAV inspection was being done by itself in an autonomous/semi-autonomous way (lets call it direct-inspect UAV), without the necessity of having a ground operator supervising its flight, as for instance in [14, 16]. Moreover, UAVs can give extra information to a ground operator that is supervising some operation from ground [11, 15]. In such cases, the human action is necessary and the UAV can play a key role monitoring the operation and allowing the operation to be supervised from ground. In this project, we focus on that type of application.

Most of the applications 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 [19], they operate on a cinematography shot application with multiple UAV that grants the director to have multiple views at the same time. In this sense, teams with multiple UAVs expand upon these possibilities as they may be able to provide alternative points of view or even supplementary illumination. In [**?**], 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 [20]. 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 [21] 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 [22]. Timed reference trajectories are generated from 3D positions specified by the user, and the final timing of the shots is addressed designing easing curves that drives the UAV along the planned trajectory (i.e., curves that modify the UAV velocity profile). In [23], 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* [24] or interpolating polynomial curves [25, 22]. 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 [26] propose to detect limbs movement of a human for outdoor filming. Trajectory planning is performed online with polynomial curves that minimize snap. In [27, 28], 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 [29]. 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.

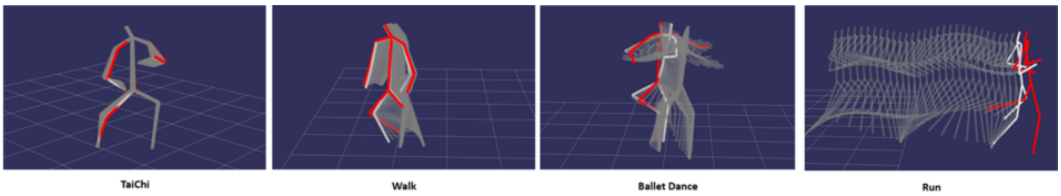


Figure 2.6 AR information in different movements. Font: [26].

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 [8] or reinforcement learning to achieve visually pleasant shots [9]. 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 [30], 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 [31] 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

Generating trajectories for a team of UAVs in real-time is challenging and still a problem to be solved. We differentiate multi-robot methods in centralized and decentralized; both can include formation purposes. In [32], they provide a survey of formation control presenting different strategies like leader-follower, virtual structure or behaviour-based formation control. [33] solves 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 [34] leading to faster computations but still not in real-time. In order to improve computational efficiency, decoupled planning methods have been proposed [35, 36, 37]. 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 multi-UAVs systems, taking advantage of receding horizon approaches to solve the problem in real-time. [38] presents a centralized leader-follower scheme, 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 [39]. They compute the largest collision-free convex polytope in a neighborhood of the robots, followed by a constrained optimization via sequential convex programming.

Decoupling path planning from trajectory planning has also been used for multi-UAVs recently.[40] first employs a graph-based multi-agent path planner to find an initial discrete solution and, then, refines 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. [41] also develops a two-stage system with long planning time horizons and real-time performance that uses 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. Distributed planning methods are computational efficient, but suffer from guaranteeing no deadlock and are poorly suited to problems in maze-like environments. Some works use potential fields algorithms to maintain the formation. In [42, 43], 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. [44] uses a potential function 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 algorithms is that they are susceptible to deadlocks. Optimal reciprocal collision avoidance (ORCA) has also been used to guarantee collision-free trajectories for non-holonomic agents [45]. [46] 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. [47] considers a set of motion primitives for the robot and solves an optimization problem in the space of control velocities with additional constraints. [48] introduces on-demand collision avoidance in a DMPC framework, where they detect and resolve only the first collision in the finite prediction horizon, modelling the collision as an ellipsoid and implementing soft constraints.

Some methods calculate a previous path geometrically to maintain the formation over a target and add inter-collision constraints in the problem formulation. In [49], they extend the centralized approach mentioned before [39] through distributed consensus to compute the convex hull of the robot's position and the intersection of convex regions. [50] also formulates a convex optimization problem constructing relative safe flight corridors.

Some methods maintain a formation with a decentralized leader-follower scheme. [51] maintains 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.

Some works solve a decentralized problem formulation taking into account the solved trajectories by others in a prioritized way. [29, 52] performs traditional priority planning where each robot would avoid only the robots of higher priority. In [29], they formulate non-convex constraints with slack variables to maintain the formation. [52] also include non-convex constraints for collision avoidance into an NMPC decentralized problem. Still, they only activate it if a potential field algorithm cannot maintain a minimum distance.

2.2.4 Augmented Reality

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 [53], which is an article published in 2005, multi-UAV formations were already integrated with AR to augment the sensor systems, using Decentralized Data Fusion (DDF), Simultaneous Localization And Mapping (SLAM) and picture compilation algorithms to develop this project. In [54], they present inspection with Mixed Reality (MR) headsets that tries to generate a 3D-2D correspondence from a 2D-2D image from a cloud of 3D points and the mission here is to get an image-based localization for efficient inspections. In [55], UAV with AR technology is integrated to do soil sampling in precision agriculture. This application is oriented to make proper decisions regarding the fertilization of fields.

In [56], an AR tool for the situational awareness improvement of UAV operators is presented. They add information that is quite important to an operator to the camera images (e.g. waypoints to reach).

In [57], a Mixed Reality (MR) system (which combines AR and VR data) is implemented onboard, making possible a command-detector by voice and gesture-detector (natural language). Depending on the command demanded (e.g "Take a photo of the green box"), the UAV does a particular task.

It is also possible to get a 3D map (rendered with AR) and demand some actions over it by head gaze and hand gesture [58]. This application is represented in Figure 2.7. 3D interactive map (and reconstructed) with operators is quite interesting to have the control and to interact with an autonomous UAV.

An interesting UAV company that is recently developing several inspection applications with AR is Skydio[1]. On its webpage, we can find a section of inspection tasks[2] that is interesting to get positioned on the area, which has autonomous roof inspection, autonomous asset inspection, among others.

[1] https://www.skydio.com/

[2] https://www.skydio.com/inspection

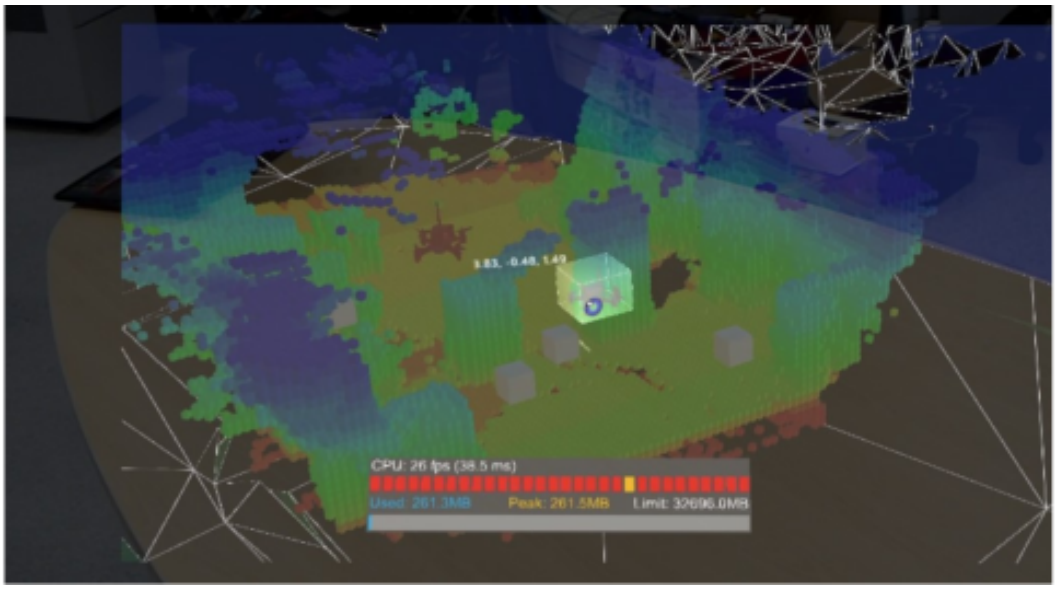


Figure 2.7 3D map construction with AR. Font: [58].

2.3 Previous study

In this chapter, we are going to introduce a little which programs and knowledge were necessary to develop the project, but just superficially.

2.3.1 Why ROS?

This project is developed in ROS[3] (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 of these services are useful to develop the whole robot system of this project, working with some helpful additional simulation programs: Gazebo and RViz, which are normally used in a ROS simulation; and lately implemented in real life as well.

Figure 2.8 ROS logo. Font: *ros.org/*.

[3] https://www.ros.org/

2.3.2 Gazebo

Talking now about Gazebo, it is an open-source 3D robotics simulator which is common to see it working with ROS. It allows testing the whole programming of the robot system: algorithms, robot designs, training AI systems, among others. In the case of this project, it is useful in order to watch the functionality of the formation over a simulator.

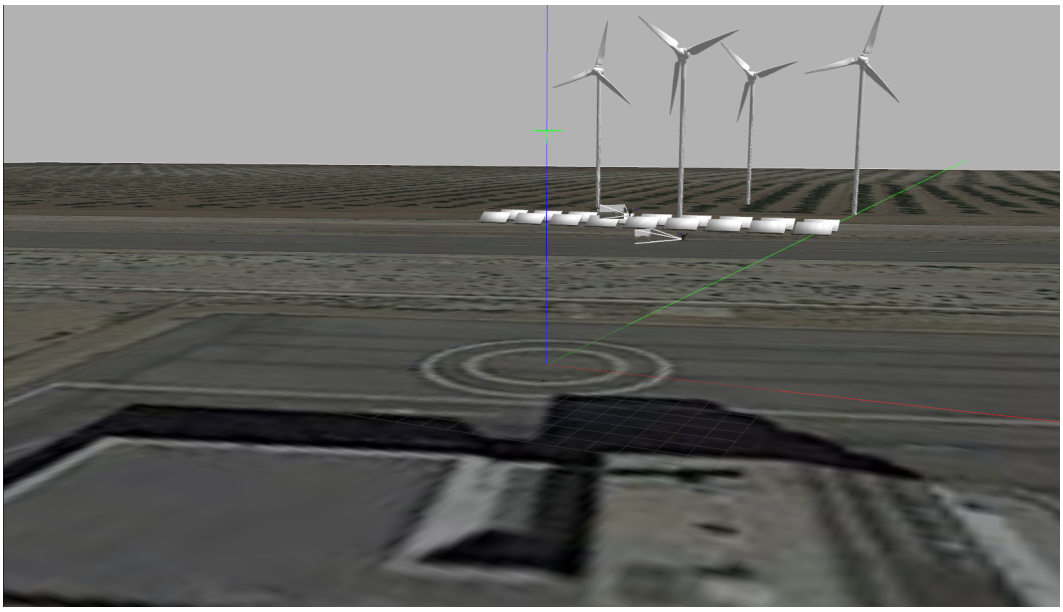


Figure 2.9 Gazebo simulator with the world implemented.

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 GPS measures received in order to know if this sensor is working properly, plotting interesting waypoints, showing the built map by a LiDAR sensor... For this project, it is interesting to use RViz in order to visualize the position and orientation of each drone, for plotting the waypoints and drawing as well the path to be done by each drone to reach the waypoints, to visualize the cylinder where the UAVs have to fly around, etc.

2.3.4 ACADO

ACADO[4] (Automatic Control And Dynamic Optimization) is a software environment and algorithm collection for automatic control and dynamic optimization. For this project, this is used to generate the optimal trajectories based on a given formulation that intends to minimize the accelerations in order to avoid jerky movements and increase the smoothness. This software environment is available on C++ and has a MATLAB interface as well.

2.3.5 MATLAB

MATLAB[5] is a versatile software very used in engineering because it is a easy language to learn and very powerful as well. It has access to many toolboxes and libraries that are also used in the framework of ROS, so it allows simpler debugging in all kinds of situations. For this project,

[4] For more information: https://acado.github.io/ or https://sourceforge.net/p/acado/wiki/MATLAB Interface/

[5] https://es.mathworks.com/products/matlab.html

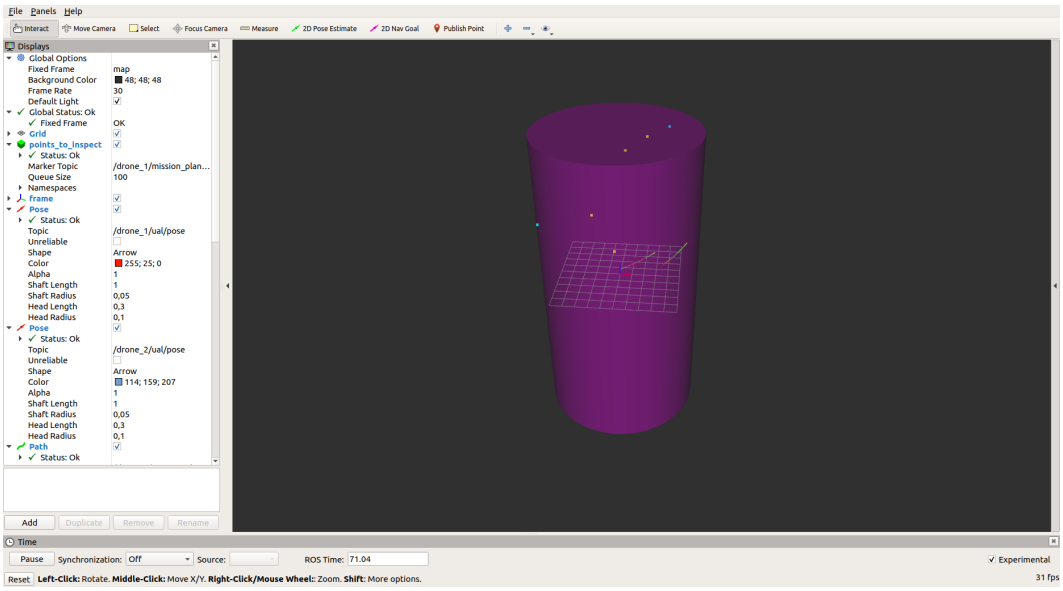


Figure 2.10 RViz visualization tool.

MATLAB was used at the beginning of the development in order to test some formulations to optimize trajectories with ACADO thought to be implemented afterwards in ROS.

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. Fig 3.1 shows a top view scheme with the following operators commands.

- Waypoint to inspect. The cameras on board the UAVs point to this waypoint. The operator can change this waypoint online to get images of different parts of the infrastructure.
- Formation angle. This angle is represented in Fig 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 UAVs team maintains depends on the parameters mentioned before. Our method generates trajectories for the UAV team maintaining those properties. As result, the trajectories fit a cylinder of radius equal to the inspection distance and centered on the waypoint to inspect. Several works have generated trajectories using the arcball principle for efficient virtual camera control. [59], [60], [61] and [62]. They define an arcball surface in polar coordinates centered on the target generating curved trajectories to improve visual properties.

Fig 3.2 depicts the architecture of the entire system. The leader UAV carries a main camera for filming while several 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 in Section 3.2.2. These initial trajectories do not consider obstacle avoidance, but only maintain the operator parameters. The leader attempts to compute a reference trajectory formulated in polar coordinates, whereas the followers maintain a surrounding formation with the desired configuration. The trajectory generation procedure of the leader is shown in Algorithm 1. On the other side, the followers calculate their trajectory from the leader trajectory

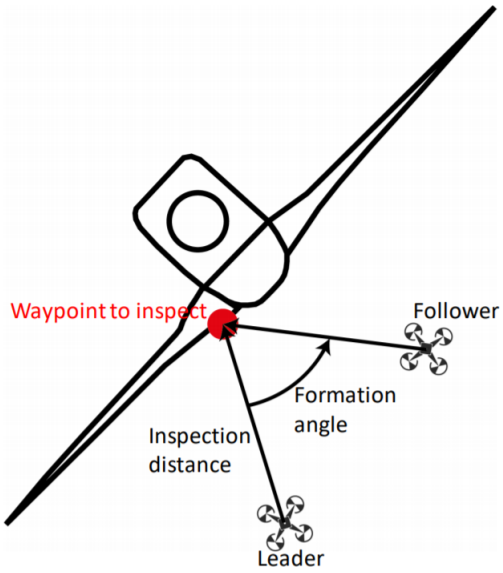


Figure 3.1 Leader-Follower scheme in 2D. Graphic representation of relative angle, inspection distance and inspection point.

Algorithm 1 Leader UAV

```
1: procedure LEADER(waypoints, inspected point, distance to inspect)
2:     System Initialization
3:     while Exist waypoints && Mission is started do          ▷ For each cycle
4:         Reference path ← Waypoint to reach, inspected point, distance to inspect
5:         Optimal path ← 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 ← Rotate ± formation angle Leader's reference path
6:         Optimal path ← Reference path
7:         Output: Follower trajectory
```

and operator commands in order to maintain the desired formation. This procedure is shown in Algorithm 2.

The entire pipeline shown in Fig 3.2 (except for the Human operator component) runs on board each UAV in a receding horizon manner. This enables the online planning to react properly to changes in the operator inputs, as well as to malfunctioning team-members or previously unseen obstacles. Note that either the leader trajectory generator or the follower trajectory generator 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 optimal trajectory, which is updated any time the Planner generates a new solution. The *UAV Abstraction Layer* (UAL) is a software component developed by our lab [**?**] 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 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.

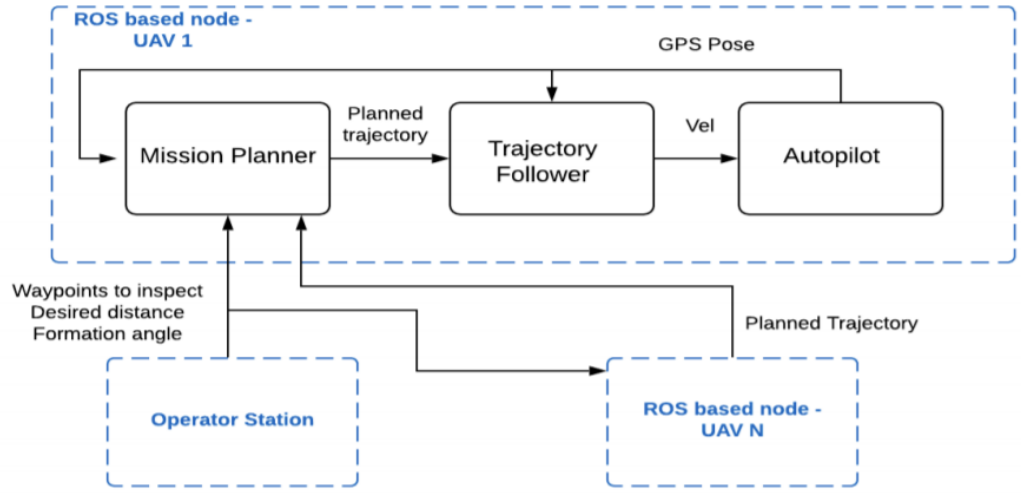


Figure 3.2 System overview scheme.

3.2 Problem formulation

3.2.1 Dynamic model

An independent trajectory tracker is used, which allows for planning with a kinematic UAV model. In addition, the camera's orientation needs to be modelled. 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 orientation of the camera may be modelled similarly:

$$\begin{aligned} \dot{\mathbf{o}} &= \omega, \\ \dot{\omega} &= \theta, \end{aligned} \tag{3.2}$$

where $\mathbf{o} = [\varphi\ \xi]^T$ represents an orientation with respect to a global frame given by its heading and pitch angles, $\omega \in \mathbb{R}^2$ are the corresponding angular rates, and $\theta \in \mathbb{R}^2$ 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 $\mathbf{x}_o = [\mathbf{o}^T\ \omega^T]^T \in \mathbb{R}^4$, and a vector of orientation control inputs $\mathbf{u}_o = \theta$.

3.2.2 Generation of reference paths

The leader reference paths are generated from the waypoint, the inspected point and the distance to inspect, as mentioned in Algorithm 1, step 4. As result, the path will fit a cylinder of radius equal to the distance to inspect and centered in the inspected point. Thus, the leader will orbit around the inspected point, passing through the waypoints commanded by the director and maintaining visual properties. For that purpose, the reference trajectory is parametrized in polar coordinates. Parametric interpolation 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.

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

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

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

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

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:

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

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

$$\mathbf{P}_{RF} = (\mathbf{P}_{RL} - \mathbf{p}_i) * R_Z\ (relative\_angle) \tag{3.7}$$

Where $\mathbf{P}_{RF}$ is the reference path of the follower, $\mathbf{P}_{RL}$ is the reference path of the leader, $\mathbf{p}_i$ is the point to inspect and R_Z is:

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

3.2.3 Trajectory optimization

Our main objective is to generate optimal trajectories taking into account smoothness, as mentioned in the 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.

$$\underset{\substack{\mathbf{x_0},\dots,\mathbf{x_N}\\ \mathbf{u_0},\dots,\mathbf{u_N}}}{\text{minimize}} \sum_{k=1}^{N}(||\mathbf{x}_{d,k}-\mathbf{x}_k||^2+\beta||\mathbf{u}_{k-1}||^2), \tag{3.9}$$

$$\begin{aligned}
\text{subject to } & \mathbf{x}_0=\mathbf{x}' && \text{(3.9.a)}\\
& \mathbf{x}_{k+1}=f(\mathbf{x}_k,\mathbf{u}_k) \quad k=0,\dots,N-1 && \text{(3.9.b)}\\
& \mathbf{u}_{min}\le\mathbf{u}_k\le\mathbf{u}_{max} && \text{(3.9.c)}\\
& \mathbf{v}_{min}\le\mathbf{v}_{Q,k}\le\mathbf{v}_{max} && \text{(3.9.d)}\\
& \mathbf{p}_{Q,k}\in\mathcal{F} && \text{(3.9.e)}\\
& ||\mathbf{p}_{Q,k}-\mathbf{p}_O||^2\ge r^2_{inspection}* && \text{(3.9.f)}
\end{aligned} \tag{3.10}$$

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. We could add the constraint 3.9.f of a no-fly zone where the wind turbine or any static obstacle is. However, this constraint is non-convex, which complicates quite a lot the optimization problem. This constraint has been considered implicitly in the generation of reference paths and erased from the optimization solver to make the trajectory planning quicker and safer.

In 3.9.a, **x'** is the current observed state value of the UAV, that is considered the initial state for the formulation $\mathbf{x}_0$. As expressed in 3.9.b, the system kinematics has some boundaries at the top and at the bottom for the velocity 3.9.c and for the acceleration 3.9.d, and this ensures the feasibility of the resultant trajectory.

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

$$\mathbf{o}_d=\left[\varphi_d\ \xi_d\right]^T=\left[\arctan(q_y,q_x)\ \sin\left(\frac{q_z}{||q||}\right)\right]^T. \tag{3.11}$$

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:

$$\underset{\mathbf{u_0},\dots,\mathbf{u_{N-1}}}{\text{minimize}} \sum_{k=1}^{N}(||\mathbf{o}_{d,k}-\mathbf{o}_k||^2+\gamma||\mathbf{u}_{k-1}||^2), \tag{3.12}$$

$$\begin{aligned}
\text{subject to } & \mathbf{x}_0=\mathbf{x}', && \text{(3.12.a)}\\
& \mathbf{x}_{k+1}=\mathrm{f}_o(\mathbf{x}_k,\mathbf{u}_k) \quad \forall k\in\{0,\dots,N-1\}, && \text{(3.12.b)}\\
& \boldsymbol{\omega}_{min}\le\boldsymbol{\omega}_k\le\boldsymbol{\omega}_{max} \ \forall k\in\{1,\dots,N\}, && \text{(3.12.c)}\\
& \xi_{min}\le\xi_k\le\xi_{max} \quad \forall k\in\{1,\dots,N\}, && \text{(3.12.d)}\\
& \mathbf{u}_{min}\le\mathbf{u}_k\le\mathbf{u}_{max} \ \forall k\in\{0,\dots,N-1\}, && \text{(3.12.e)}
\end{aligned}$$

where $\mathrm{f}_o(\cdot)$ represents the orientation aspect of the dynamic model defined in Section 3.2.1; $\boldsymbol{\omega}_{min}$, $\boldsymbol{\omega}_{max}$ 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 adjusting pitch angles. The heading and pitch angles of the camera or light can be controlled independently. Thus, Problem (3.12) 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*).

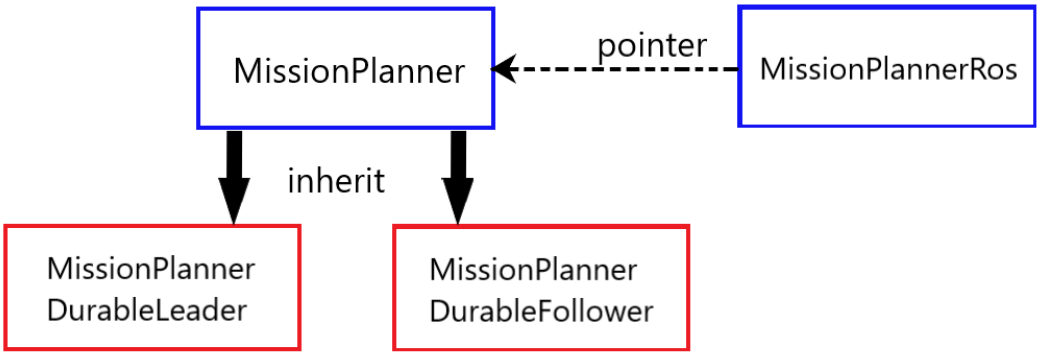


Figure 3.3 Class diagram.

The 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). Receding horizon lets the UAV to have a path to follow for the next 4 seconds and this benefits to have even smoother movements as the path is replanned each second.

Now, is also necessary to compute the follower's optimal path, as mentioned in Algorithm 2, step 6. To get it, we use the same optimization problem, but using the follower's reference path and execute it on the follower UAV.

3.3 Implementation

In this section, we are going to emphasize the programming in general: class diagram, pseudocodes and communications made on ROS.

This work is uploaded on a repository[1] and was developed with Ubuntu 18.04[2] and ROS Melodic[3].

3.3.1 Class diagram

In the Figure 3.3, it could be seen the class diagram done to develop the project. It has four classes in total: MissionPlanner, MissionPlannerDurableLeader (which inherits from MissionPlanner), MissionPlannerDurableFollower (which inherits from MissionPlanner), and MissionPlannerRos (which communicates with MissionPlanner class through a pointer). Now, we are going to explain more in depth what each class does.

The MissionPlanner class is where the main logic of the mission is located. It has several methods that mainly concerns the mission, as it could be the addition of a waypoint, getting the current relative angle or to set the solved trajectory path.

As mentioned before, both leader and follower UAV inherits from the MissionPlanner class.

In the case of the MissionPlannerDurableLeader, it instantiates a leader UAV object that has the logic of the MissionPlanner class. In this node is executed several methods that lets the leader UAV, if there are waypoints to reach and the mission is started, to generate a reference trajectory to follow (Section 3.2.2), optimizing it with ACADO solver (Section 3.2.3) and executing it sending the velocities got from the optimization.

[1] https://github.com/grvcTeam/inspection_trajectory_planning
[2] https://releases.ubuntu.com/18.04.5/
[3] http://wiki.ros.org/melodic

In the case of the MissionPlannerDurableFollower, it is also an inherited class from MissionPlanner, but it does not have as much computation to do as the leader UAV. The follower object does just a rotation of the previous reference path done by the leader UAV around the Z axis with relative_angle rad and executes the optimization problem. This makes the follower to have less computation charge regarding the leader UAV and, thus, have all the UAVs working at the same height, which is necessary to have a good augmented reality frame.

Last but not least, we have the MissionPlannerRos class which is, in a few words, a wrapper of ROS that lets transferring the information of topics and services to the MissionPlanner node. It has basically all the topics and services necessary to make the project work (which are going to be mentioned more in depth in Section 3.3.3) in a separate class in order to not have to make the UAVs interpret the ROS information. Finally, it calls MissionPlanner's methods referenced with a pointer to the class as a channel to transfer the information.

3.3.2 Pseudocodes

In this section, we are going to show some keys to make this project work of the code written in pseudocode.

First of all, it is important to mention how the code works from the beginning to the stationary functionality:

```
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 leader UAV's initial trajectory initial_
          traj_to_follow
       6.2.2. Calculate the leader UAV's optimal trajectory solved_
          trajectory
       6.2.3. Apply a rotation for getting follower UAV's path
    6.3. Wait until the mission is resumed
```

Code 3.1 Overall code functionality.

Then, we are going to mention the process of getting the optimal trajectory for the leader UAV mentioned in Section 3.2.2 path:

```
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
    7. Send initial_traj_to_follow to optimize the path
        optimal_traj = solved_traj
    8. Send solved_traj to autopilot
```

Code 3.2 Initial trajectory pseudocode.

As mentioned in Algorithm 2, the follower UAV gets the reference_trajectory path from the leader, but rotated relative_angle rad and afterwards it optimizes the path. The pseudocode in this case is the following:

```
    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 3.3 Optimal trajectory pseudocode for follower UAV.

Now, it is important to generate an optimal path for the UAVs 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.

```
    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 3.4 Optimal trajectory (ACADO solver).

3.3.3 ROS: possible interactions with the operator

The most important services in this work are:

- 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.

And the most important topics are:

- Distance to inspection point: 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.

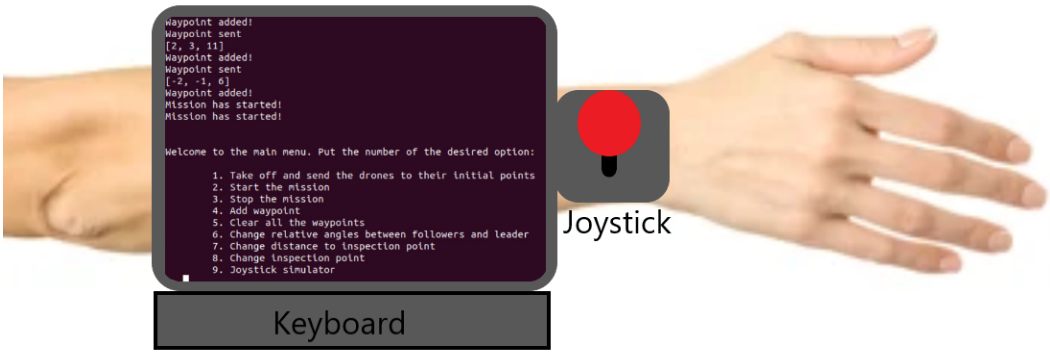


Figure 3.4 Operator interface aspect.

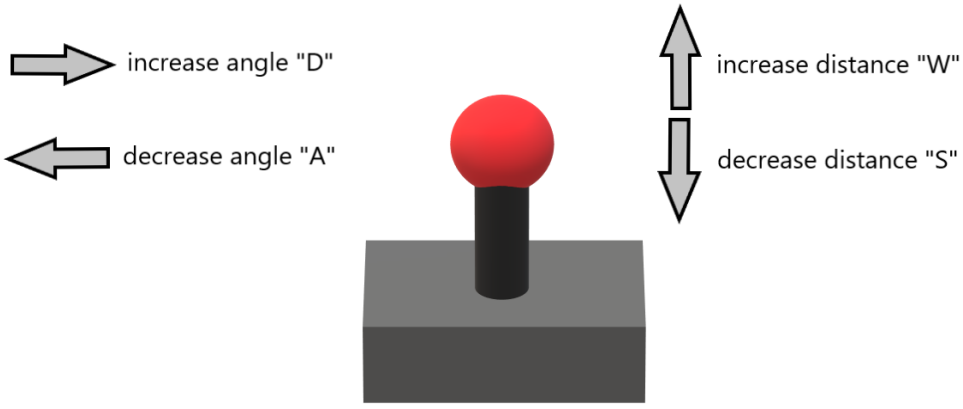


Figure 3.5 Joystick aspect.

4 Results

In this chapter, the results are going to be shown.

4.1 Environment

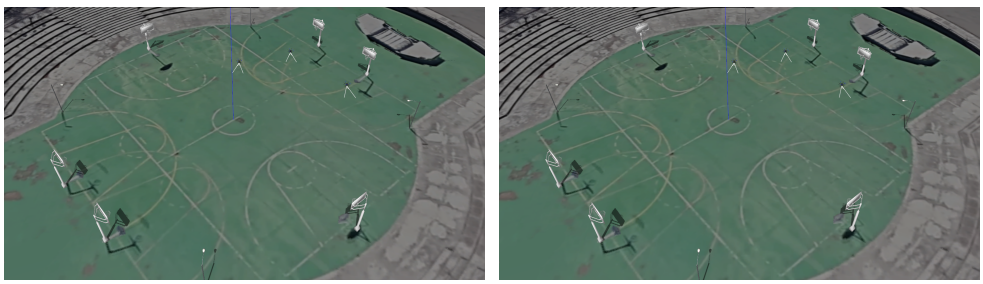
To develop the simulations, we used the simulator Gazebo which was mentioned in Section 2.3.2. We perform our experiments in a basketball court at "La Plaza del Agua" (ETSI, Seville), which has a mesh installed for safety. We created a simulation environment of this court to ease the real world experiment, testing the algorithms in this simulator before performing field experiments. Figure 4.1 represents the simulated environment and the real picture.

Cameras are quite important for this work as they provide the ground operator the most essential tool in inspection tasks: vision. The main aim of this project is to inspect a huge cylindrical-shaped structure and, with the formation carrying cameras with themselves, the operator can detect tears easier. The camera views' aspect are depicted in the Figure 4.2:

RViz interface provides information about what is going on with the mission: reference and real paths of each UAV, orientation of UAVs, waypoints to inspect, among other things. The appearance of the RViz interface can be seen in Figure 4.3:

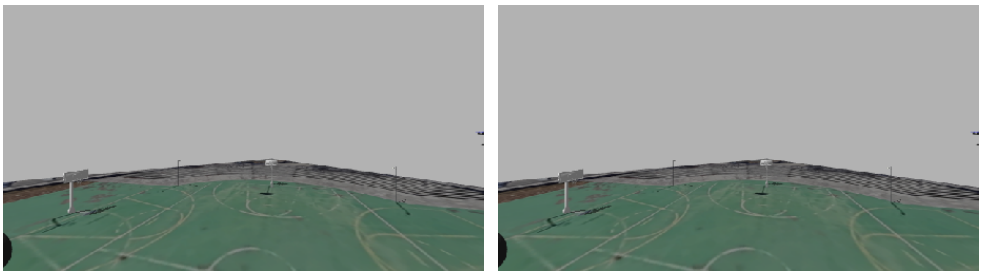
A demonstration video of this work is uploaded on the platform YouTube[1].

[1] https://www.youtube.com/watch?v=pXQHwkDVizw



(a) Gazebo (b) In real life

Figure 4.1 "Plaza del Agua".



(a) Gazebo (b) In real life

Figure 4.2 Leader UAV camera view.

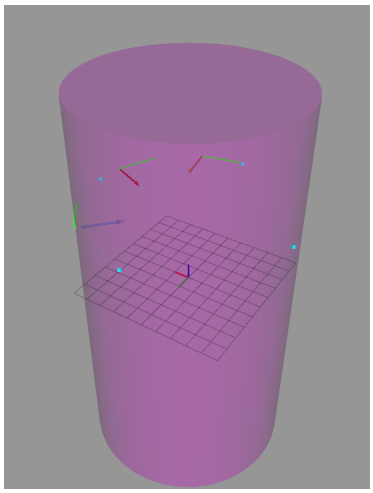


Figure 4.3 RViz in simulation.

4.2 Operator's interface

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, where is possible to interact with the formation.

```
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.

4.3 Simulation

In this simulation, we are going to plot the results of the experiments and also we are going to change the relative angle and the distance to inspection point with the joystick simulator shown in Figure 4.5.

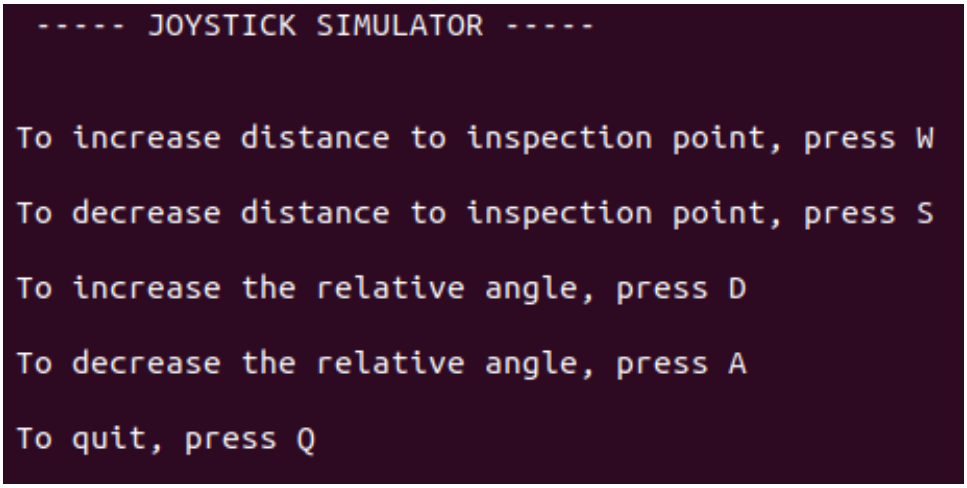


Figure 4.5 Operator's interface, joystick simulator.

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.

This way, the ground operator would get satisfied with the desired images that is looking for from the UAVs based on the desired waypoints that commanded before and the parameters of the formation.

Is important to know that in the results is considered the data obtained since the formation is stuck in the cylinder (already operating).

The results obtained in simulations are the following:

In Figure 4.6 is depicted the evolution in time of the described path of each UAV, where Drone 1 refers to the leader UAV and Drone 2 and 3 are the follower UAVs. Notice that from $\Delta t = 30\ s$ to $\Delta t = 40\ s$ 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 $\Delta t \approx 25\ s$), making the UAVs get away from the initial circumference that were describing and, in $\Delta t = 50\ s$, this change can be seen easier.

In Figure 4.7, is represented 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 $\sim 7.35\ m$ and the formation lasts to reach this distance approximately $15s$. 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 depends on how far is the next waypoint to inspect from the last reached (parameter t_k). The overoscillating behaviour comes from the necessity of detecting the desired waypoint when it is not close enough and, right after, the path to describe by the formation keeps being the same until a new desired waypoint to reach is able. In $t \approx 239\ s$, there is another change of the distance to the inspection point, going down from $\sim 7.35\ m$ to $\sim 6.7\ m$. The explanation done before is the same for this case.

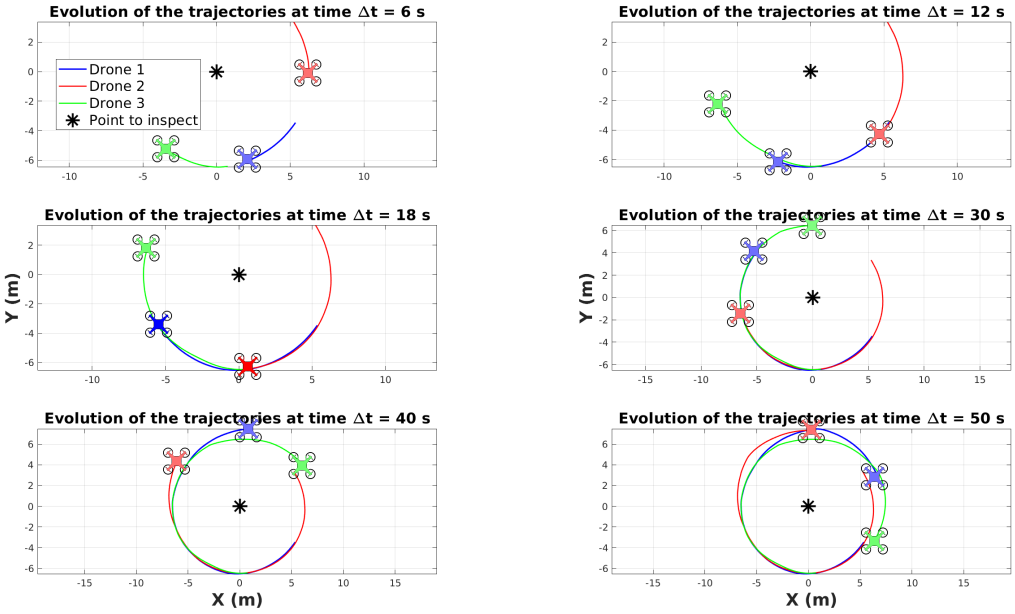


Figure 4.6 Simulation. UAVs path described mosaic. Notice that there is a change in the distance to the inspection point along the first turn over the inspection point. Drone 1 (leader) is between the other two UAVs which are followers (Drone 2, Drone 3) and they are describing a path always in the same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

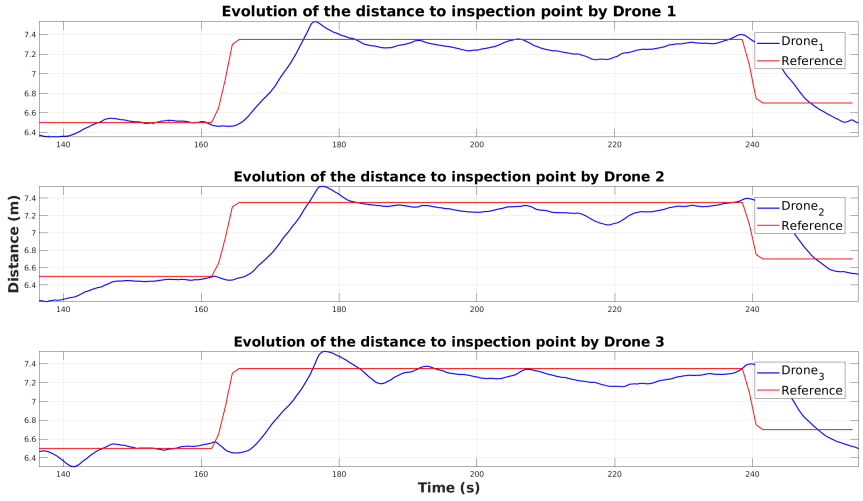


Figure 4.7 Simulation. UAVs distance to inspection tracking. Notice that in every change of reference, the UAVs seems to have an overoscillating behaviour. This is due to the tolerance given that could be a bit demanding and forces to pass quite close to the desired waypoints when a change of reference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

In Figure 4.8 is shown 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 synchronization of the followers trajectories. Regarding the relative angle tracking, it could be seen that there is an unavoidable offset of some degrees (∼6-8º) 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. With the mentioned upgrades, the UAV should be always quite close to the reference.

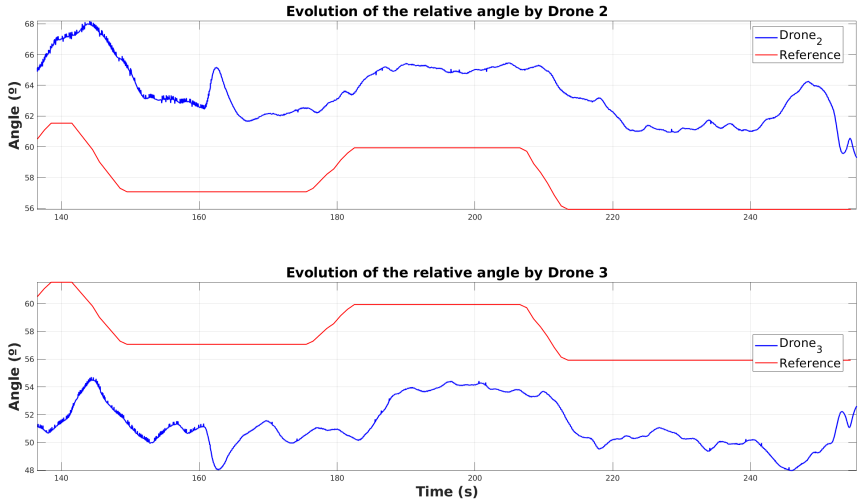


Figure 4.8 Simulation. UAVs relative angle tracking. In the top plot, it is depicted the evolution of the relative angle between the leader UAV (Drone 1) and the follower 1 UAV (Drone 2); while in the bottom plot, it is shown the evolution of the relative angle between leader UAV and the follower 2 (Drone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

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 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 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.

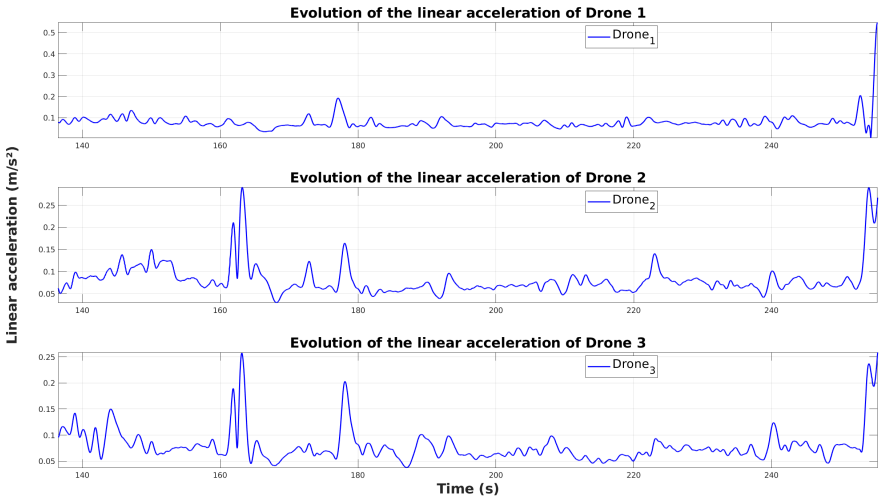


Figure 4.9 Simulation. UAVs linear acceleration. While the references are not changed, each UAV's acceleration should tend to be the minimum possible due to there is a solver generating the trajectories that minimizes the accelerations. When there are significant peaks they could be due to: a change on the height of the waypoints (not the case); a change on the distance to inspection point (it is normally smooth); or a change on the relative angles for the follower UAVs, that only concerns to themselves accelerations. Because of the bad synchronization of the reference trajectories of follower UAVs, there could be a worse behaviours. As said before, when the upgrades are implemented, those behaviours should disappear.

In Figure 4.10 and in Figure 4.11, 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.

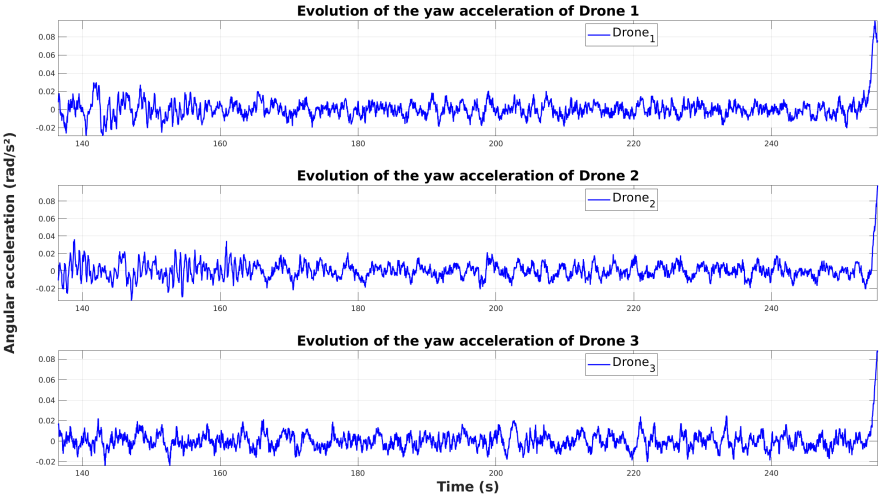


Figure 4.10 Simulation. UAVs yaw acceleration. The solver minimizes this acceleration in order to avoid shakiness and have smooth views on the UAVs' cameras. As could be seen, the accelerations on yaw are quite small.

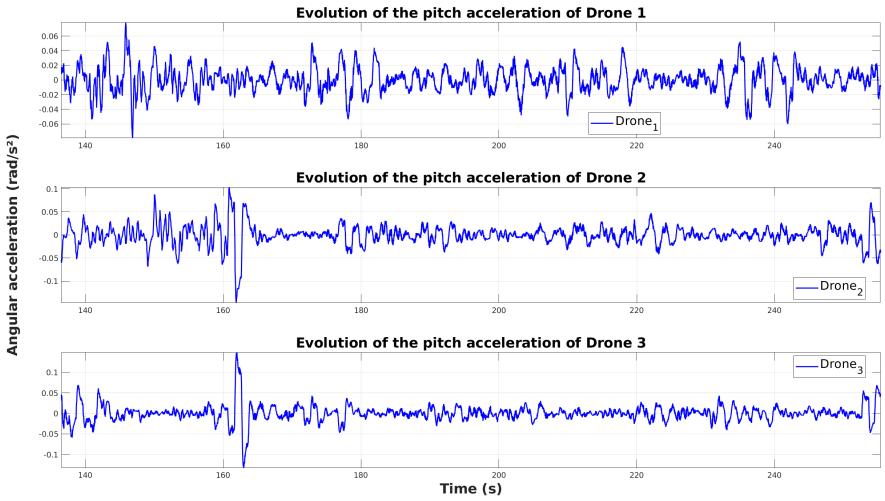


Figure 4.11 Simulation. UAVs pitch acceleration. This acceleration could vary principally due to the max velocity and max acceleration that is established on the configuration of the UAVs and on the solver.

4.4 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 $1.80 \times 1.80 \times 0.70\ m$ 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 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 4.12). 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 generate its own reference path that will be shared to the follower UAVs and make a rotation of it.

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 inspection point or relative angle from the previous value.

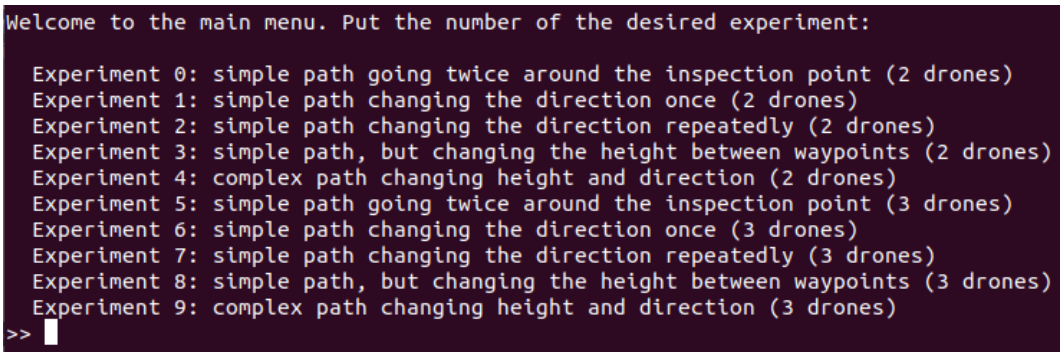


Figure 4.12 Operator's interface, selection of the desired experiment.

5 Conclusions and future work

Having now the notion of the project and the results in mind, it is evident to realize the huge potential that this tool has for inspection tasks where the human perspective is not enough to check the whole structure, or even if it was enough. The mobility, the variety of sensors that can be installed and the size that UAVs have are a point in favor in almost every task application compared with other kind of robots. The possibility of having an autonomous multi-UAV formation that can be interactive with a ground operator in a simple way that also provides image views makes the inspection tasks much easier than the current tools. Thus, this work demonstrates that it would be very useful for any structure inspection with cylinder shape in the current point of the project.

Regarding the future work, it is important to mention that the project is still in progress, which means that everything that is going to be written here are pending tasks that are in development.

First of all, in the results shown in this work, we mentioned that there is a bug in the synchronization of the trajectories, making the relative angle of the drones vary a little, but this may affect quite a lot to the AR image to build the spatial-sense visualization. It is essential to fix this bug to avoid dragging some other future bugs.

Also, it is quite interesting to implement the obstacle avoidance to the UAVs in order to dodge the blades (among other things) of the turbine when the operator wants to have a closer view of the wind turbine. If this is not implemented with regard to a future integration in real life, the UAVs are vulnerable to collisions and fatal crashes may happen, up to the point of smashing the UAV into pieces. This will be the next point to develop.

Straightaway, simulations with disturbances are necessary to make them trustworthy since, in real life, huge irregular gusts are on the wind turbines power plants. This way, it is necessary to implement a wind topic that follows a wind velocity profile or a constant value that alters the UAVs behaviour and check how the wind could affect the UAVs' path tracking. It would be also interesting to add noise to the sensors, mainly to the GPS and IMU, in order to know how the UAVs would act in the real scenarios.

It is also important to develop an AR interface in order to give extra information to the operator about the mission. This information could be represented on a screen or with AR glasses to give the operator the possibility of being on the UAVs' eyes.

Regarding to the experiments in real life, it is thought to develop a real experiment connecting the system via IP with the Ecole Supérieure des Technologies Industrielles Avancées (ESTIA) at Bidart, France. In that place, everything needed to make a real experiment is available and, remotely, we

are going to connect with their formation.

Finally, this project could be used in an early future to do other kind of inspection tasks, not only static and vertical structures, by making a variable inspection point in time and having a relative distance between leader UAV and follower UAVs instead of having a relative angle. 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:

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

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

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

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

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.

$$\theta_{pass} = \frac{Vxy * steptime}{2\pi * R_{inspection}} * 2\pi \tag{A.4}$$

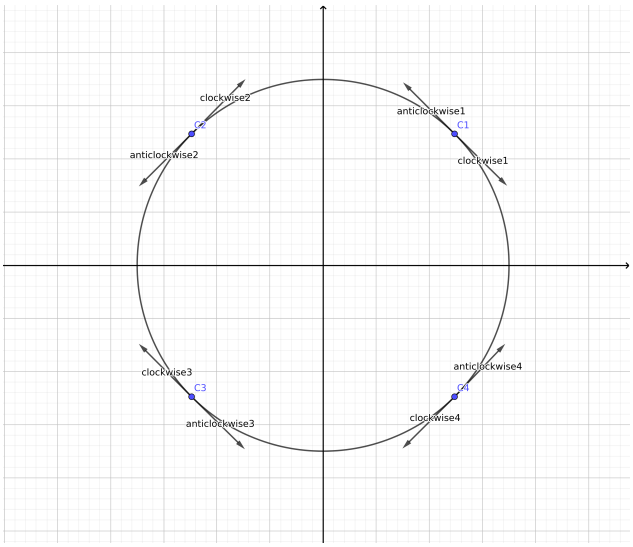


Figure A.1 Clockwise/anticlockwise graphic.

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.

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

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:

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

- 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

List of Codes

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] 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.

[9] 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.

[10] 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

[11] J. Seo, L. Duque, and J. Wacker, "Drone-enabled bridge inspection methodology and application," *Automation in Construction*, vol. 94, pp. 112–126, 2018.

[12] 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.

[13] 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.

[14] 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.

[15] A. Shihavuddin, X. Chen, V. Fedorov, A. Nymark Christensen, N. Andre Brogaard Riis, K. Branner, A. Bjorholm 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] M. Christie, P. Olivier, and J. M. Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008.

[21] 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

[22] 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.

[23] 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

[24] 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.

[25] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated Cinematography with Unmanned Aerial Vehicles," *Eurographics Workshop on Intelligent Cinematography and Editing*, 2016.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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

[31] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F.-l. Tariolle, and P. Guillotel, "Directing cinematographic drones," *ACM Trans. Graph.*, vol. 37, no. 3, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3181975

[32] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, pp. 1019 – 1047, 2018.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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

[39] 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

[40] 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.

[41] C. Ho, A. Jong, H. Freeman, R. Rao, R. Bonatti, and S. Scherer, "3D Human Reconstruction in the Wild with Collaborative Aerial Cameras," 2021.

[42] 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.

[43] 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/

[44] 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{%}2FJSYST.2017.2778063{&}partnerID=40{&}md5=e40bf3aca4c1659f494cb6e394b3be32

[45] 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

[46] 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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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

[53] A. H. Göktoğan and S. Sukkarieh, "An augmented reality system for multi-uav missions," *SimTect'05*, 2005.

[54] 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.

[55] 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

[56] 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.

[57] 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.

[58] C. Liu and S. Shen, "An augmented reality interaction interface for autonomous drone," *arXiv preprint arXiv:2008.02234*, 2020.

[59] K. Shoemake, "Arcball: A user interface for specifying three-dimensional orientation using a mouse," in *Graphics interface*, vol. 92, 1992, pp. 151–156.

[60] 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.

[61] 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.

[62] 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.