DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
UNIVERSIDAD DE SEVILLA

OPTIMAL MULTI-UAV TRAJECTORY PLANNING FOR FILMING APPLICATIONS

por
Alfonso Ángel Alcántara Marín

Graduado en Ingeniería de Tecnologías Industriales / Máster en Ingeniería Electrónica, Robótica y Automática

PROPUESTA DE TESIS DOCTORAL PARA LA OBTENCIÓN DEL TÍTULO DE
DOCTOR POR LA UNIVERSIDAD DE SEVILLA
SEVILLA, 2022

Director
Dr. Ing. Jesús Capitán Fernández

Agradecimientos

En mis inicios como doctorando, comenzaba a leer algunas tesis doctorales y siempre me paraba atentamente en la sección de agradecimientos, sorprendido del sacrificio, pasión y trabajo que transmitían las palabras de aquellos doctores. Ahora, escribiendo esta sección, empatizo con el esfuerzo, en ocasiones frustración y con el orgullo por el resultado conseguido. Para mí y para muchos, una tesis doctoral es de los mayores desafíos a los que se puede enfrentar una persona a lo largo de su carrera.

Porque en esa dificultad las personas que acompañan al doctorando son también protagonistas, esta carta de agradecimiento cobra especial importancia.

En primer lugar me gustaría agradecer a Jesús Capitán por haber creido en mí para este proyecto, gracias a él empecé este bonito camino. Su implicación y compromiso con mi tesis ha sido insuperable, allanando el camino y facilitando discusiones enriquecedoras cuando encontraba dificultades. La forma en la que concibe la investigación siempre me ha resultado inspiradora.

A los miembros del GRVC y a su director Aníbal Ollero, por crear un ambiente de trabajo en el que siempre me sentí como en casa. Aportando ideas y arrimando el hombro con cualquier cosa que necesitaba. Especial mención a Arturo, Rafa, Héctor y Ángel Montes, compañeros de proyecto. También al Multi-Robot System Lab de la Czech Technical University, y en especial a Vit Krátký, por su trabajo y dedicación durante nuestra colaboración. La que ha me enriquecido enormemente.

Gracias a mis amigos, por prestarme su escucha cuando lo necesitaba. Su habilidad para empatizar me permitió desahogarme en periodos difíciles. En especial a mi amigo Edu, aunque de diferentes disciplinas, compartimos fatigas de doctorando.

A mis padres, porque aunque a veces se pierdan en los tecnicismos de la robótica, siempre he percibido en cada una de sus palabras y acciones que mis dudas son sus dudas, mis miedos sus miedos y mis alegrías las suyas. Sin su apoyo esto no hubiera sido posible.

Por último a Irene, por estar en el día a día. Aprendí de ti a confiar en mí mismo y siempre he admirado tu capacidad de trabajo. Mis esfuerzos son también los tuyos.

Muchas gracias.

Resumen

Los equipos de vehículos aéreos no tripulados (UAV) son sistemas prometedores para grabar eventos cinematográficos, en escenarios exteriores de grandes dimensiones difíciles de cubrir o para tomar vistas complementarias de diferentes puntos de acción. La generación de trayectorias para este tipo de vehículos desempeña un papel fundamental, ya que debe garantizarse que se cumplan requisitos dinámicos, de suavidad y de seguridad.

Los enfoques basados en la optimización para la planificación de trayectorias de múltiples UAVs se pueden ver beneficiados por el auge de los métodos numéricos para la resolución de problemas de optimización no lineales. En particular, estos métodos son bastante prometedores para las aplicaciones de grabación de vídeo, ya que permiten formular múltiples restricciones y objetivos, como la suavidad de la trayectoria, el cumplimiento de la dinámica del UAV y de la cámara, la evitación de obstáculos y de conflictos entre UAVs, y la visibilidad mutua. El objetivo principal de esta tesis es planificar trayectorias para equipos multi-UAV en aplicaciones de vídeo, formulando novedosos problemas de optimización y resolviéndolos en tiempo real.

La tesis comienza presentando un marco de trabajo para la realización de misiones cinematográficas autónomas con un equipo de UAVs. Este marco permite a los directores de medios de comunicación diseñar misiones que incluyan diferentes tipos de tomas con una o varias cámaras, ejecutadas de forma secuencial o concurrente. En segundo lugar, la tesis propone una novedosa formulación no lineal para el difícil problema de calcular las trayectorias óptimas de los vehículos aéreos no tripulados en cinematografía, integrando en el problema la dinámica de los UAVs y las restricciones para evitar colisiones, junto con aspectos cinematográficos como la suavidad, los límites mecánicos del cardán y la visibilidad mutua de las cámaras. Por último, la tesis describe un método de grabación aérea autónoma con iluminación distribuida por un equipo de UAVs. El problema de optimización de trayectorias se desacopla en dos pasos para abordar los aspectos cinematográficos no lineales y la evitación de obstáculos en etapas separadas. Esto permite al planificador de trayectorias actuar en tiempo real y reaccionar en línea a los cambios en los entornos dinámicos.

Es importante señalar que todos los métodos de la tesis han sido validados mediante extensas simulaciones y experimentos de campo. Además, todos los componentes del software se han desarrollado como código abierto.

Abstract

Teams of multiple Unmanned Aerial Vehicles (UAVs) can be used to record large-scale outdoor scenarios and complementary views of several action points as a promising system for cinematic video recording. Generating the trajectories of the UAVs plays a key role, as it should be ensured that they comply with requirements for system dynamics, smoothness, and safety. The rise of numerical methods for nonlinear optimization is finding a flourishing field in optimization-based approaches to multi-UAV trajectory planning. In particular, these methods are rather promising for video recording applications, as they enable multiple constraints and objectives to be formulated, such as trajectory smoothness, compliance with UAV and camera dynamics, avoidance of obstacles and inter-UAV conflicts, and mutual UAV visibility. The main objective of this thesis is to plan online trajectories for multi-UAV teams in video applications, formulating novel optimization problems and solving them in real time.

The thesis begins by presenting a framework for carrying out autonomous cinematography missions with a team of UAVs. This framework enables media directors to design missions involving different types of shots with one or multiple cameras, running sequentially or concurrently. Second, the thesis proposes a novel non-linear formulation for the challenging problem of computing optimal multi-UAV trajectories for cinematography, integrating UAV dynamics and collision avoidance constraints, together with cinematographic aspects such as smoothness, gimbal mechanical limits, and mutual camera visibility. Lastly, the thesis describes a method for autonomous aerial recording with distributed lighting by a team of UAVs. The multi-UAV trajectory optimization problem is decoupled into two steps in order to tackle non-linear

cinematographic aspects and obstacle avoidance at separate stages. This allows the trajectory planner to perform in real time and to react online to changes in dynamic environments.

It is important to note that all the methods in the thesis have been validated by means of extensive simulations and field experiments. Moreover, all the software components have been developed as open source.

Contents

Agradecimientos **3**

Resumen **5**

Abstract **7**

1 Introduction **19**

1.1 Motivation . . . 19

1.2 Thesis objectives . . . 22

1.3 Thesis contributions . . . 23

1.4 Thesis framework . . . 26

2 Background **29**

2.1 Optimal trajectory planning . . . 29

2.1.1 Problem formulation . . . 30

2.1.2 Solving approaches . . . 32

2.1.3 Solvers for specific types of problems . . . 34

2.2 Related work on UAV trajectory planning . . . 37

2.2.1 Path planning . . . 38

2.2.2 Trajectory planning . . . 39

2.2.3 Multi-UAV trajectory planning . . . 42

3 Scheduling and execution of cinematography missions with multiple UAVs **47**

3.1 Introduction 48
3.2 Related work 49
3.3 System architecture 51
3.3.1 System overview 53
3.3.2 Shot description 56
3.3.3 Canonical shots 58
3.4 Distributed mission execution 61
3.4.1 Scheduler 61
3.4.2 Shot executor 64
3.5 Field experiments 67
3.5.1 System integration 68
3.5.2 Results 69
3.5.3 System usability evaluation 75
3.5.4 Lessons learned 77
3.6 Conclusions 79

4 Optimal trajectory planning for cinematography with multi-UAV coordination 81
4.1 Introduction 82
4.2 Related work 84
4.3 Dynamic models 87
4.3.1 UAV model 87
4.3.2 Gimbal angles 89
4.4 Optimal trajectory planning 91
4.4.1 Trajectory planning 91
4.4.2 Multi-UAV coordination 94
4.4.3 Trajectory execution 97
4.4.4 Discussion 98
4.5 System architecture 100
4.5.1 Cinematography shots 102
4.6 Performance evaluation 103

4.6.1 Cinematographic aspects 104
4.6.2 Time horizon 107
4.6.3 Multi-UAV coordination 110
4.7 Field experiments 111
4.8 Conclusions 115

5 Aerial filming with distributed lighting by a team of UAVs 117
5.1 Introduction 118
5.2 Related work 120
5.3 System overview 121
5.4 Autonomous aerial cinematography 122
5.4.1 Multirotor aerial vehicle dynamic model 122
5.4.2 Generation of reference trajectories 123
5.4.3 Generation of safe corridors 126
5.4.4 Trajectory optimization 128
5.4.5 Orientation control 130
5.5 Experimental evaluation 131
5.5.1 Experimental setup 132
5.5.2 Simulation experiment – cinematography trajectories 132
5.5.3 Simulation experiment – cluttered environment 135
5.6 Conclusion 137

6 Conclusions and future work 139
6.1 Conclusions 139
6.2 Future work 142

A Open-source code repositories 145
A.1 Multidrone Planning 147
A.2 Cinematography Trajectory Planner 148
A.3 Shot Executor 148
A.4 Safe Corridor Generator 149
A.5 Trajectory Follower 149

References **151**

List of Figures

1.1 Different views of our field experiments with multiple UAVs filming sports activities. 20

2.1 OCPs can be addressed using dynamic programming, indirect methods, and direct methods. In particular, the software frameworks used in this thesis for trajectory optimization apply direct methods to discretize and solve OCPs. 32

3.1 System architecture for multi-UAV cinematography. Mission design and planning components run on a Ground Station, but components for mission execution run on board the UAVs. 53

3.2 Snapshots of the Dashboard graphical interface. Upper: several events with different associated missions. Lower: an example of the window to design a specific Shooting Action within a mission, indicating the Reference Target trajectory. 54

3.3 Scheme describing the shot geometry for the different shot types. Shot parameters refer to the RT frame. Lighter figures represent the start of the shot, and solid figures the shot end. 60

3.4 Example of the event-based procedure for a distributed execution of a mission with two UAVs. 63

3.5 Flow diagram for gimbal control. Vision-based or GPS-based measurements can be used to feed a attitude controller. 65

3.6 One of the UAVs used during our field experiments, with the cinematographic camera mounted on the gimbal. 69
3.7 Parkour mission with two UAVs and five different shots. Blue indicates UAV 1 and red UAV 2. 70
3.8 Images from a mission with two UAVs filming a parkour activity. . . 71
3.9 Rowing race mission with two UAVs and three different shots. Blue indicates UAV 1 and red UAV 2. 73
3.10 Images from a mission with two UAVs filming a rowing race. 74
3.11 Timeline of a simulated mission where re-planning is carried out. The UAVs were executing their initial plans when the battery alarm of UAV 1 went off. A new plan was computed assigning the missing shot from UAV 1 to others. UAV 2 covered part of the lateral shot initially assigned to UAV 1. 75

4.1 Cinematography application with two UAVs filming a cycling event. Bottom, aerial view of the experiment with two moving cyclists. Top, images taken from the cameras on board each UAV. 82
4.2 Definition of the reference frames used. The origins of the camera and multirotor frames coincide. The camera points to the target. 87
4.3 Mutual visibility constraint for two UAVs. The UAV on the right (blue) is filming an action point at the same time that it keeps the UAV on top (red) out of its angle of view α. 95
4.4 System architecture on board each UAV. A Scheduler initiates the shot and continuously updates the desired state for trajectory planning. The Shot Executor plans optimal trajectories to carry out the shot. UAVs exchange their plans for coordination. 101

4.5 At the top, top view of the resulting trajectories for different solver configurations of the cost weights. *High-yaw* is not shown as it was too similar to *low-yaw*. The target follows a straight path and the UAV has to execute a flyby shot (10 s) starting 20 m behind and ending up 15 m ahead of the target. The predefined no-fly zone simulates the existence of a tree. At the bottom, temporal evolution of the jerk of the camera angles and the norm of its 3D acceleration. We compare the *full-cinematography* configuration against *no-cinematography*. . . 105
4.6 Receding horizon comparison for a lateral shot. Top view of the trajectories resulting from different time horizons. The target follows a straight path and the UAV has to execute a lateral shot (10 s) at a distance of 8 m. 108
4.7 Top view of different time instants of an experiment with three UAVs filming a target in a coordinated manner. The target (black) follows an 8-shaped path. UAV 1 (blue) performs a chase shot 2 m behind the target, UAV 2 (green) a lateral shot 2 m aside the target, and UAV 3 (magenta) an orbit shot with a 4 m radius. A no-fly zone (red) is placed in the middle of the target trajectory for multi-UAV avoidance. Each UAV is represented with a 2 m circle around to show the collision avoidance constraint. 109
4.8 Temporal evolution of the UAV altitudes during the multi-UAV coordination experiment. The vertical lines mark the time instants of the snapshots depicted in Figure 4.7. 110
4.9 One of the UAVs used during the field experiments. 112
4.10 Trajectories followed by the UAVs and the human target during the field experiment. UAV 1 (blue) does a lateral shot, UAV 2 (green) a flyby, and UAV 3 (red) another lateral. 113
4.11 Examples of images from the cameras on board the UAVs during the experiment: top left, UAV 1 (blue); top right, UAV 2 (green); and bottom left, UAV 3 (red). Bottom right, an overall view of the experiment showing the three UAVs and the target. 114

5.1 UAV filming applications to provide external lighting, to capture smooth shots outdoors, and to monitor dangerous maintenance operations for power lines. Pictures were obtained within AERIAL-CORE and DRONUMENT projects, for which the proposed technology is being developed. 118
5.2 The architecture of the proposed system. C_s and C_l represent the desired cinematographic shot type and lighting configuration specified by a human director; T_T is the target estimated trajectory; D_L, D_F are reference trajectories for the leader UAV and the follower UAVs, respectively; P_L, P_F are collision-free paths generated along the desired trajectories; S_L, S_F are safe corridors along the collision-free paths; and T_L, T_F are optimized trajectories for the camera and lighting UAVs, respectively. The modules enclosed in the blue rectangle run on both types of UAVs. 121
5.3 The camera shooting angle, the leader UAV position, the target position, and their relative position. 125
5.4 The leader–follower scheme defined by Equation (5.6). 127
5.5 The safe corridor–generating process. The initial reference trajectory (dark green) is converted into a collision-free path (purple), and the obstacle-free polyhedrons are generated along this path. The final optimized trajectory within the safe corridor is also shown (blue). We inflate the obstacles for safety purposes (light red). 128
5.6 Illustration of the proposed system filming a worker under the required illumination. 131

5.7 Trajectories for the camera-carrying UAV while monitoring a worker on a transmission tower. For simplicity, only the lateral shot and the first flyby shot are shown. We compare the trajectories generated by our method (blue) with those from a baseline approach without cinematographic costs (green). The upper image displays a top view of the UAV's and target's trajectories. The small white dots on the trajectories depict transition points sampled every 5 s to give a notion of the speed. The bottom image depicts the evolution of the camera angular rates. 133
5.8 Illustration of the experiment where an operator is filmed working on a transmission tower. The trajectories of the camera-carrying leader (orange), both followers carrying lights (blue and green), and the human worker (red) are shown. The obstacle map is represented by a point cloud, including the power lines and tower. The worker is tracked with a lateral shot as he walks to the tower and then with a sequence of flyby shots while he climbs up. Several onboard images taken during the experiment are also shown. 134
5.9 A top view of the trajectories generated in the cluttered forest scenario. The trajectories of the target (red), the leader (orange), and the two light-carrying UAVs (blue and green) are shown. The trees are represented by black dots. 136

A.1 Interconnections between the different repositories. Each module represents a code repository. Solid lines imply connections based on ROS topics and services, whereas dashed lines refer to connections made by library headers. In Chapter 3 and Chapter 4, UAL is used as UAV navigation framework, while the MRS UAV System is used in Chapter 5. 146

Chapter 1

Introduction

This chapter introduces the motivation, main objectives, and scope of this thesis. The main contributions are then outlined, as well as the research framework in which it was developed.

1.1 Motivation

The use of teams of multiple unmanned aerial vehicles (UAVs) that cooperate to perform autonomous tasks is becoming mainstream (Real et al., 2021b; Spurný et al., 2019). These multi-UAV teams are possible due to recent advancements in UAV- and communication-related technologies, and they hold remarkable advantages over single-vehicle systems; increased efficiency, reduced mission time, robustness to UAV failures, and scalability for larger scenarios among others. For instance, multi-UAV systems have recently been used for a wide spectrum of applications including package delivery (Dorling et al., 2017), search and rescue (Alotaibi et al., 2019; Real et al., 2021a), exploration (Cesare et al., 2015), and surveillance (Scherer and Rinner, 2020). In these tasks, the use of multiple UAVs is key to reducing operational time, which increases performance efficiency. Efficacy is also improved since, for example, the failure of one UAV would not necessarily imply mission termination.

Particularly in filming applications, teams of multiple UAVs can also broaden the range of possibilities, as they could film several action points concurrently, or obtain

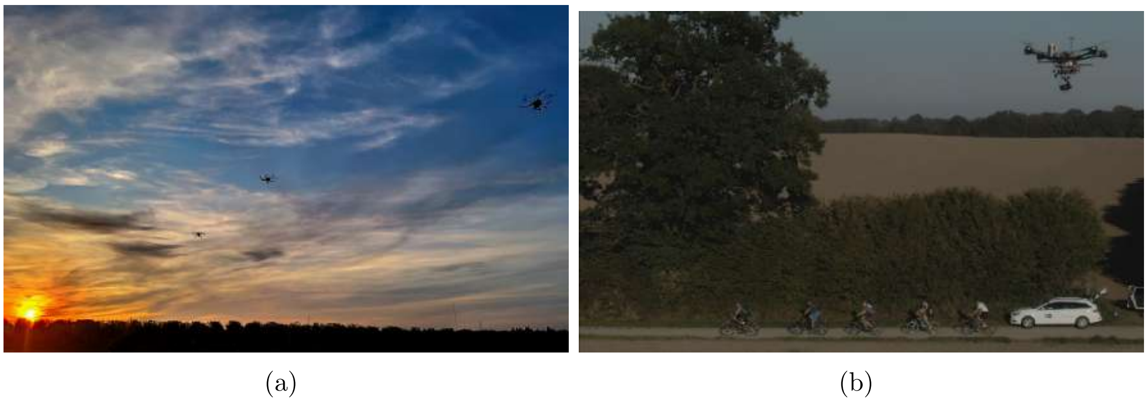


Figure 1.1: Different views of our field experiments with multiple UAVs filming sports activities.

alternative perspectives of the same subject (Nägeli et al., 2017b; Galvane et al., 2018). This becomes especially interesting in outdoor settings (see Figure 1.1), where UAVs may need to cover large-scale scenarios with multiple action points. The use of UAVs as flying cameras presents not only a remarkable potential for recreational cinematography but also for surveillance and monitoring in inspection operations. Even if the footage is not to be used for entertainment purposes; for example, for monitoring inspection operations, high-quality videos could speed up the assessment by human operators on the ground. For instance, the recent commercial platform Skydio (Skydio, 2019) integrates a novel data capturing system with a UAV that flies around large structures for better inspection. Multi-UAV teams can expand upon these possibilities, as they may provide alternative points of view or even supplementary illumination (Krátký et al., 2020).

However, autonomous multi-UAV systems also present some additional complexities. One of them is inter-vehicle collision avoidance: with several vehicles operating in the same airspace, they must coordinate to resolve potential conflicts and stay a safe distance apart. Further planning is also needed in complex missions in order to, for example, schedule different tasks that may happen sequentially or in parallel, predict and anticipate the temporal evolution of the scenario, or address possible failures and contingencies. On the one hand, if these coordination and planning approaches are centralized, they could suffer from scalability issues, mainly in highly dynamic scenarios with real-time restrictions. On the other hand, if decentralized strategies are used, inter-UAV communication plays a crucial role, with UAVs sharing information to perform their tasks more efficiently. Due to these issues, despite recent advances, there is still room for new multi-UAV autonomous systems that are intelligent enough to execute tasks cooperatively in real-time.

In addition to the aforementioned problems, multi-UAV systems for filming applications have specific constraints: UAVs with cameras need to navigate smoothly to produce aesthetic footage, take into account certain cinematographic principles, and not occlude the camera field of view of others, to name a few. Therefore, planning UAV trajectories becomes even more complex in this context. Traditional path planning techniques are not sufficient to satisfy these application requirements, since the paths generated rely only on spatial information, and they may not be dynamically feasible nor consider smoothness constraints for camera motion. Instead, trajectories are used; a trajectory is a time-parameterized function of UAV states containing both spatial and temporal information. Trajectory planning is usually required to ensure compliance with system dynamics, smoothness, and safety. Of course, trajectory planning is more complicated than just computing paths, especially in multi-UAV settings. This thesis aims to find solutions for optimal trajectory planning in filming applications with multiple UAVs. Our methods enable UAVs to cooperate in a distributed manner to autonomously film a common target, either for recreational (e.g., cinematography) or industrial purposes (e.g., filming inspection operations).

Many authors have used optimization-based techniques for UAV trajectory planning; e.g., using search-based methods or numerical optimization. This thesis focuses on optimization problems which could be non-linear and non-convex, in order to comply with diverse constraints such as trajectory smoothness, UAV and camera dynamics, obstacle avoidance, inter-UAV collisions, or mutual UAV visibility. The main challenge is to balance these constraints while considering computational requirements, in order to achieve the real-time performance required by filming applications in dynamic environments.

1.2 Thesis objectives

The main objective of this thesis is to plan online optimal trajectories for multi-UAV teams, formulating novel optimization problems related to filming applications. For this purpose, we have defined the following goals:

- **Building a framework for task scheduling and execution of cinematographic missions with cooperative teams of UAVs**. Within this framework, a human operator or *director* would provide cinematography missions to be executed by the autonomous multi-UAV team. The framework should be able to deal with parallel and sequential shots, and with different types of camera motion. In addition, once the shots are assigned to each UAV, the system should be able to execute them in a distributed manner, having the ability to accommodate certain mission delays or uncertainties. For instance, models for target motion prediction should be used, and the system should be able to react to possible UAV failures (e.g., a UAV running out of battery or losing GPS coverage).

- **Planning optimal trajectories for aerial filming with a cooperative team of UAVs**. In order to produce aesthetically pleasing footage, these UAV trajectories should take into account specific cinematographic aspects such as UAV dynamics, trajectory smoothness, gimbal mechanical limits, and mutual camera visibility.

- **Planning trajectories that are safe, even in dynamic scenarios**. This implies collision avoidance both with other UAVs and with obstacles in the environment. Considering collision avoidance for optimal trajectory planning is challenging, since the presence of multiple and possibly dynamic obstacles can lead to more complex problem formulations, which are usually non-convex and hard to solve in real time. The objective is to integrate obstacle avoidance into the problem formulation in a scalable manner, so that UAVs can plan safe trajectories for cluttered scenarios in real time.

- **Planning UAV trajectories online**. Since we target dynamic scenarios, we pursue methods that can solve optimization problems in real time, adapting to varying conditions during flight. For this, we will favor methods that are scalable with the number of UAVs and in terms of the planning time. In particular, our approach is to achieve real-time performance by applying distributed multi-UAV trajectory planning and receding horizon optimization. Thus, UAV trajectories could be recalculated online at a suitable frequency to cope with external uncertainties and disturbances.

- **Validating the methods in realistic conditions**. New methods that work well in simulation may break when they are applied to real systems. A key objective in this thesis is to evaluate all methods, apart from simulation, in field experiments where the multi-UAV system needs to cope with imperfect communication, onboard computational resources, less controlled dynamic actors, and so on.

1.3 Thesis contributions

This thesis makes several contributions in the field of optimal trajectory planning for multi-UAV aerial filming. First, in **Chapter 2**, we include a thorough review of related work, to introduce the state of the art on trajectory planning. This chapter also gives background information on optimal trajectory computation, providing insight into the mathematical formulation, analyzing the problem complexity, and introducing some numerical tools to solve non-linear optimization.

Chapter 3 presents a complete architecture for autonomous scheduling and execution of cinematography missions with a team of UAVs. The architecture integrates components for target tracking, UAV motion planning, and gimbal control. Even though we implement a representative set of canonical shots, we also generalize to describe parametric shots, making the system easily extensible. We propose distributed schedulers that trigger the execution of the different shots, based on starting events that may be generated manually or autonomously (e.g., a certain actor reaching an action point). This procedure works as a synchronizing mechanism for multi-UAV shots but also makes the system robust to uncertainties in the planning phase (e.g., if the planned starting time of some action is delayed). This chapter includes the results of extensive field experiments to assess the performance of our framework in aerial filming of outdoor sport activities. The results of this chapter produced the following publications:

- Alcántara, A., Capitán, J., Torres-González, A., Cunha, R., and Ollero, A. (2020). Autonomous execution of cinematographic shots with multiple drones. *IEEE Access*, 8:201300–201316

- Torres-Gonzalez, A., Alcantara, A., Sampaio, V., Capitan, J., Guerreiro, B., Cunha, R., and Ollero, A. (2019). Distributed Mission Execution for Aerial Cinematography with Multiple Drones. In *EUSIPCO. Satellite Workshop on Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, La Coruña, Spain

Chapter 4 presents an optimal trajectory planning method for UAV cinematography with multi-UAV coordination. We derive a non-linear, optimization-based problem formulation for trajectory planning. Trajectories are planned and executed by a team of UAVs in a distributed manner using a receding horizon scheme, providing multiple views of the same scene. The method considers UAV dynamic constraints, and it imposes constraints on UAVs such that they avoid predefined no-fly zones and collisions with others. In addition, cinematographic aspects are addressed, such as shot definition, mutual camera visibility, physical gimbal bounds, trajectory smoothness, and gimbal motion. The method is integrated within the architecture explained in Chapter 3, to run complete cinematography missions with aesthetically pleasing footage. This chapter includes outdoor experiments to validate our optimal trajectory planning method for cinematography. The results of this chapter produced the following publications:

- Alcántara, A., Capitán, J., Cunha, R., and Ollero, A. (2021). Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems*, 140:103778

- Sabetghadam, B., Alcántara, A., Capitán, J., Cunha, R., Ollero, A., and Pascoal, A. (2019). Optimal trajectory planning for autonomous drone cinematography. In *European Conference on Mobile Robots (ECMR)*, pages 1–7

Chapter 5 formulates a novel optimization problem for trajectory planning in aerial filming with distributed lighting. Using a leader–follower scheme, our system computes smooth trajectories with pleasing footage for UAV filming (the leader), which takes shots of a dynamic target indicated by an external user. The followers compute their trajectories to maintain a formation with predefined lighting angles on the target. In addition, we propose a new method to tackle non-convex trajectory optimization with obstacle avoidance in real time by decomposing the problem into two parts: non-linear cinematographic aspects are formulated without obstacle avoidance to generate reference trajectories, and then these are used to generate collision-free regions that are convex and transform the problem into a QP (quadratic programming) optimization task. This obstacle avoidance method meets our objective of integrating collision avoidance with UAV trajectory planning in a scalable way. This chapter presents experimental results for aerial filming to monitor inspection activities, both in simulated and real scenarios. These results produced the following joint publication, where the main contribution of the author of this thesis is to the leader UAV problem formulation and trajectory planning:

- Krátký, V., Alcántara, A., Capitán, J., Štěpán, P., Saska, M., and Ollero, A. (2021). Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 6(4):7580–7587

Chapter 6 summarizes the conclusions of this thesis, and also discusses directions for future research. Additionally, this thesis makes an extensive contribution in terms of open-source software. All the methods developed here have been implemented as open-source and are available online. **Appendix A** compiles all these code repositories and gives an overview of the software generated throughout the thesis. The overall software architecture, as well as the functionalities implemented by each module and their relations, are detailed.

Finally, apart from the aforementioned contributions, throughout this thesis the author participated in some collaborations that led to the following additional publications. Even though these are not part of the thesis core, they are worth mentioning as secondary contributions in (i) UAV autonomous navigation for inspection, and (ii) multi-UAV reactive conflict resolution.

- Benjumea, D., Alcántara, A., Ramos, A., Torres-Gonzalez, A., Sánchez-Cuevas, P., Capitan, J., Heredia, G., and Ollero, A. (2021). Localization system for lightweight unmanned aerial vehicles in inspection tasks. *Sensors*, 21(17)
- Ferrera, E., Alcántara, A., Capitán, J., Castaño, A. R., Marrón, P. J., and Ollero, A. (2018). Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors*, 18(12)

1.4 Thesis framework

This thesis was developed within the framework of several research projects. The main part of the thesis was carried out within the framework of the *MultiDrone* project [1] (*Multiple drone platform for media production*). This project was funded by the European Commission (H2020-EU.2.1.1-731667), and its main objective was to develop a multi-UAV system for media production of outdoor sport events. Most of the work in this thesis was developed within Workpackage 3 of MultiDrone. This workpackage was devoted to multi-UAV planning and control techniques for aerial cinematography, in order to achieve enhanced levels of autonomy, safety and robustness in the system, which would allow the production crew to focus on the creative part of their work. The specific objectives of Workpackage 3 were:

1. Development of tools for high-level mission planning in aerial cinematography applications.
2. Multi-UAV formation control and online planning during mission execution.

[1] `https://multidrone.eu`.

3. Producing a multi-UAV system in which safety, robustness and autonomy are ensured.
4. Development of an autonomous and reliable multi-UAV communication infrastructure.

More specifically, the work in Chapter 3 and Chapter 4 of this thesis was carried out to comply with objectives 2 and 3 of Workpackage 3 in the MultiDrone project.

Additionally, the work in this thesis was applied to aerial filming for inspection activities in large infrastructures. These research activities were carried out within the framework of the following projects funded by the European Commission: AERIAL-CORE [2] (H2020-EU.2.1.1-871479), DURABLE [3] (Interreg-EAPA-986/2018), and PILOTING [4] (H2020-EU.2.1.1-871542). One of the objectives of AERIAL-CORE (*Aerial cognitive integrated multi-task robotic system with extended operation range and safety*) is to develop a multi-UAV system for monitoring inspection activities performed by human operators working on high-voltage electric power lines. Having a fleet of UAVs offering complementary views for the inspection of wind turbines is one of the objectives of DURABLE (*Maintenance drones and robots to enhance renewable energy systems in the Atlantic area*), and a similar multi-UAV system is being developed to inspect large viaducts in PILOTING (*Pilots for robotic inspection and maintenance grounded on advanced intelligent platforms and prototype applications*).

Finally, this thesis was also partially supported by the regional project MULTICOP (*Autonomous multi-aerial systems for cooperative maneuvers with physical interaction*), funded by the Andalusian regional government in Spain (FEDER-US-1265072).

[2]`https://aerial-core.eu`.
[3]`https://www.durableproject.eu`.
[4]`https://piloting-project.eu`.

Chapter 2

Background

In recent years, there has been a trend toward using receding horizon optimization approaches to generate trajectories for UAVS online. This increasing interest is mainly due to the development of new tools including efficient and reliable algorithms for non-linear programming, enhanced computational resources, and more flexible optimal control frameworks to encode diverse constraints, among others. This chapter introduces the main theoretical concepts for trajectory optimization, describing the problem formulation, the main approaches and algorithms to find optimal solutions, and some widely available software tools to implement these solvers. A summary of the state of the art in UAV optimal trajectory generation is then presented.

2.1 Optimal trajectory planning

The use of optimization frameworks for trajectory planning requires an analysis of the characteristics of the formulated problem, since the nature of the optimization function or the problem constraints can generate different types of optimization problems, with different solving complexities that should guide the selection of the proper algorithms to find optimal solutions.

2.1.1 Problem formulation

Optimal UAV trajectory planning can be seen as an optimal control problem (OCP), which deals with finding a control law for a given system to achieve a specific optimization criterion. OCPs include a cost function to be minimized, defined over a set of state and control variables that are subject to constraints. A generic problem formulation can be written as follows [1]:

$$\begin{aligned}
&\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} && \int_{0}^{T} l(\mathbf{x}(t), \mathbf{u}(t))dt + m(\mathbf{x}(T)) && (2.1)\\
&\text{subject to} && \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), && (2.2)\\
& && g(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, && (2.3)\\
& && \mathbf{x}(0) = \overline{x_0}. && (2.4)
\end{aligned}$$

In this notation, $\mathbf{x}$ denotes the state variables of the system and $\mathbf{u}$ the control inputs. The *Lagrange term* or running cost is defined by l, whilst the *Mayer term* or terminal cost is m. System dynamics are modeled with a set of differential equations (2.2), g represents the inequality constraints (2.3), and $\overline{x_0}$ the initial state value (2.4). The horizon time T is assumed to be fixed.

A major issue when modeling motion planning problems as OCPs is that the resulting problem may become a challenging non-convex optimization problem. Because of this, it is crucial to analyze the different parts of the problem. A critical part is the dynamics (2.2), which is a set of differential equations that represents the dynamic behavior of the vehicle. Non-linear models usually lead to more computationally costly problems, so, depending on the application, simplified linear versions are sometimes more appropriate. On the one hand, kinematic models that do not consider vehicle dynamics can yield problems that are easier to solve, and are often used for trajectory planning. On the other hand, non-linear dynamic models are also interesting in certain

[1]Throughout the thesis, matrices are denoted by bold uppercase symbols, vectors by bold lowercase symbols and scalars by plain symbols.

applications, such as trajectory tracking, where disturbances and forces on the vehicle should be taken into account.

Another part relevant to determining the difficulty of the problem is the inequality constraints (2.3), which may be *hard* or *soft*. Constraints that are allowed to be violated are called soft constraints, while constraints that always must hold are called hard constraints. In general, soft constraints imply a problem relaxation in order to alleviate complexity, and they usually have associated *slack* variables and a penalty function that is added to the objective function. Thus, the effect of minimizing the cost function will be a trade-off between minimizing the original objective and minimizing these slack variables associated with the soft constraints. Under a sufficiently high penalization of (a linear norm of) the slack variables, the solution to the hard-constrained problem is recovered when it exists; otherwise, a *soft* solution with minimum deviation will be computed by the optimizer. This approach has been widely used in trajectory planning methods (Castillo-Lopez et al., 2018; Nägeli et al., 2017b,a).

OCPs are formulated for a given time horizon T; i.e., the planning horizon. The solution of an OCP will provide a trajectory of the control and state variables from the current state to a final state at the end of the horizon. Due to inaccuracies in the system model, there may be discrepancies between the state values in the solution and the actual ones followed by the system. In order to avoid the drawbacks associated with applying the computed solution in an open-loop fashion, a receding horizon approach is usually applied. This means that only the first time steps of the solution are applied to the vehicle, and then the optimization problem is solved again, starting from the new current state. This technique obtains a closed-loop effect in the system and provides a certain degree of robustness against external perturbances or model inaccuracies due to, for instance, wind gusts or localization errors.

However, an issue with this receding horizon approach is that the OCP needs to be solved quickly, requiring high-frequency replanning. In general, the computational cost can be regulated by tuning the time horizon and the number of discrete time steps in each solution trajectory.

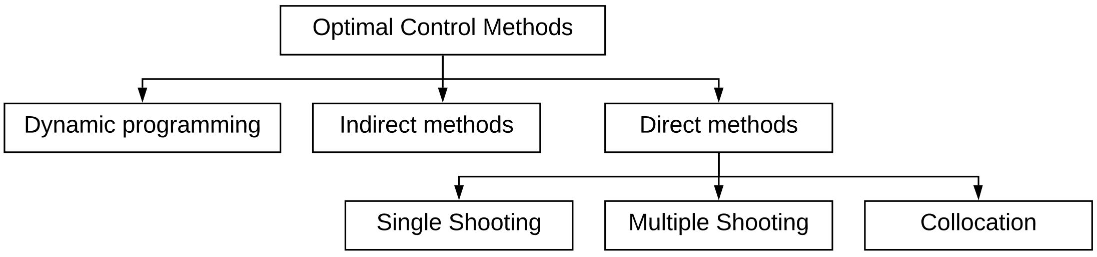


Figure 2.1: OCPs can be addressed using dynamic programming, indirect methods, and direct methods. In particular, the software frameworks used in this thesis for trajectory optimization apply direct methods to discretize and solve OCPs.

2.1.2 Solving approaches

A generic OCP involves differential equations in a continuous space, so numerical methods are typically used to address the problem. Figure 2.1 shows a scheme with the main approaches to tackling OCPs (Diehl et al., 2006). Dynamic programming has been used since OCPs emerged, solving Hamilton-Jacobi-Bellman equations over the entire state space. There are numerical methods to compute approximate solutions to these equations, and hence dynamic programming is an excellent choice for solving OCPs in unconstrained low-dimensional systems. However, this approach does not scale well to high-dimensional systems, so it is limited to simple OCPs. Indirect methods determine the necessary conditions of optimality of the infinite problem to transform the OCP into a boundary-value problem (BVP). Then this BVP must be numerically solved; i.e., the idea is to “first optimize, then discretize”. Indirect methods also include the well-known calculus of variations and the Euler-Lagrange differential equations. The main problem with indirect methods is that they tend to be numerically unstable, hard to implement and initialize, and have difficulty dealing with inequality constraints. Thus, direct methods have become the most popular option to address OCPs (Bianco et al., 2018).

Direct methods transform the initial infinite OCP into a finite non-linear programming (NLP) problem, which is the process of solving optimization problems where some of the constraints or the objective function are non-linear. First, the control and state variables are discretized over a time grid to form a finite-dimensional problem. Then, the differential equations (2.2) are converted into a finite set of equality constraints (Bazaraa, 2013), which results in a problem with a scalar algebraic function and an algebraic vector of constraints. Moreover, the remaining trajectory constraints (2.3) are transformed into inequality constraints (Zhou and Yan, 2014). Last, the resulting NLP is solved with any numerical optimization method, following a scheme of "first discretize, then optimize". In this thesis, direct methods are used to solve non-linear trajectory optimization problems.

There are three different direct methods commonly used: single shooting, multiple shooting, and collocation. The specific details for each of these techniques can be consulted in multiple books in the literature (Diehl et al., 2006; Betts, 2009; Topputo and Zhang, 2014). Most commercial software frameworks for OCPs offer the possibility of choosing among multiple algorithms, depending on the problem formulation. All these direct methods can be considered robust, even if high non-linearities are present. Multiple shooting results in a higher dimensional non-linear program, but it is sparse and more linear than the program produced by single shooting methods. The advantage of collocation methods over multiple shooting approaches is that the former manage the discontinuities of the control functions better. However, collocation methods have a higher number of variables (Diehl et al., 2006).

Numerical integration is an essential part of implementing both direct single and multiple shooting methods for trajectory optimization. It consists of computing numerical simulations and partial derivatives for the non-linear system of differential-algebraic equations that represent the vehicle's dynamic model. Explicit and implicit schemes are used in numerical analysis for obtaining numerical approximations to the solutions of time-dependent differential equations (Hairer et al., 1993). Explicit methods calculate the state of a system at a later time from the state of the system at the current time; Runge-Kutta methods are well-known examples, such as the explicit Euler or the Runge-Kutta method of order 4 (RK4). Implicit methods find a solution by solving an equation involving both the current state of the system and a later one (Ascher et al., 1997). Implicit integration schemes are more difficult to implement, as they usually need to be solved numerically using an iterative procedure, such as a

Newton-type method. On the other hand, they offer better numerical stability and a higher order of accuracy (Verschueren et al., 2022). In the case of differential-algebraic equations, implicit integration methods can fit better (Hairer et al., 1993). In any case, whether one should use an explicit or implicit method depends on the problem at hand, and most optimization software frameworks allow users to choose among several options.

2.1.3 Solvers for specific types of problems

Depending on the shape of the objective function and the constraints involved, optimization problems can be divided into several categories, which are solved with different approaches. If the objective function is linear and the constrained space is a polytope, the problem becomes a linear programming problem, which can be solved using well-known linear programming techniques such as the simplex method (Karloff, 1991). For non-linear programs, there are different solving methods depending on the problem structure.

A function is called convex if the line segment between any two points on the graph of the function lies above the graph between those two points. If the objective function is convex and the constraints define a convex set (i.e., given any two points in the set, the set contains the whole line segment that joins them), the program is called convex (Bubeck, 2015). A convex optimization problem can only have a globally optimal solution, which greatly facilitates its resolution. Several methods can be used to solve these problems, and they will either find the globally optimal solution or prove that there is no feasible solution to the problem (Bubeck, 2015).

Quadratic programming (QP) is a special case of non-linear optimization that consists of optimizing a quadratic objective function of several variables subject to linear constraints on these variables. A QP problem has the following structure:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i,\ i = 1, \ldots, m. \end{aligned}$$

QP problems are convex if the matrix $\mathbf{Q}$ is positive definite. Moreover, if there are only equality constraints, the QP becomes particularly simple, as the solution process is linear.

When the objective function or some of the constraints are non-convex, we have a non-convex program. For non-convex programs, sequential convex programming (SCP), sequential quadratic programming (SQP), and the interior-point method are frequently used. SCP is a local optimization method for non-convex problems that leverages convex optimization by solving a sequence of convex subproblems (Schittkowski and Zillober, 1995). SQP methods also iteratively solve a sequence of optimization subproblems, each of which optimizes a quadratic version of the objective function subject to a linearization of the constraints (Boggs and Tolle, 1995). Interior-point methods are a certain class of algorithms to solve linear and non-linear optimization problems (Potra and Wright, 2000). They pursue the optimal solution by traversing the interior of the feasible region; i.e., the region whose points fulfill the problem constraints.

The implementation of efficient software packages plays a significant role in bringing non-linear optimization problems into real-time applications. In Moulard et al. (2014), they review the most common non-linear optimization suites used in robotics. In the following, we summarize some of these frameworks for non-linear programming, which implement some of the solving methods explained in this chapter.

CVXGEN (Mattingley and Boyd, 2012) is a tool for embedded convex optimization created by Stanford University. It works for problems which can be transformed into QP, generating fast custom C code for small quadratic convex optimization problems using an online interface requiring no software installation. CVXGEN is available under commercial license and has been used for UAV trajectory generation (Tallamraju et al., 2018; Baca et al., 2018; Saska et al., 2017). OOQP is another software package designed for QP problems. It is an object-oriented C++ package based on a primal-dual interior-point method (Vanderbei and Shanno, 1999). Its design allows easy substitution of the linear algebra modules, enabling the use of different standard linear algebra packages. Thus, OOQP can also be used as a framework to design efficient solvers for new classes of structured QPs. It has also been used for UAV trajectory planning (Luis and Schoellig, 2019).

SNOPT (Gill et al., 2005) (Sparse Nonlinear OPTimizer) is a software package provided by Stanford University for solving large-scale optimization problems. It employs a sparse SQP algorithm with limited-memory quasi-Newton approximations to the Lagrangian Hessian. Although it is not open-source, they have evaluation versions with an academic license. SNOPT has been used for UAV trajectory planning (Alonso-Mora et al., 2015a; Joubert et al., 2016). ACADO (Houska et al., 2011) is another widespread solver for UAV trajectory planning and Model Predictive Control (Kamel et al., 2017; Kostadinov and Scaramuzza, 2020). ACADO is an open-source software environment that offers friendly interfaces to a collection of algorithms for non-linear optimization. In particular, ACADO includes single and multiple shooting methods for discretization, as well as SQP and SCP solvers for NLPs. FORCES PRO (Zanelli et al., 2017) is another software package for non-convex finite-time NLPs. FORCES PRO requires external functions to evaluate the cost function terms and their gradients, the system dynamics and their Jacobians, and the inequality constraints and their Jacobians. These external functions can be supplied by either the CASADI library or hand-coded C-functions. The academic license is available for free but it can only be used on one machine. FORCES PRO has also been used by the UAV trajectory planning community (Nägeli et al., 2017b; Zhu and Alonso-Mora, 2019; Cheng et al., 2017). NLopt is an open-source library for non-linear optimization, providing a common interface for a number of different free optimization routines available online, and original implementations of various other algorithms. It is callable from C, C++, Fortran, Matlab, GNU Octave, Python, GNU Guile, Julia, GNU R, Lua, OCaml and Rust. NLopt has been recently used for UAV trajectory optimization (Krishnan et al., 2020). Acados (Verschueren et al., 2022) is a software package with a collection of solvers for fast embedded optimization. Its interfaces to higher-level languages make it useful for quick design of optimization-based control algorithms by integrating different algorithmic components that can be readily connected and interchanged. Acados is open-source and has been applied to UAV trajectory generation (Torrente et al., 2021; Carlos et al., 2020). Finally, OpEn (Optimization Engine) is a framework for fast and accurate embedded non-convex optimization. It is free, open-source and well documented with abundant examples. Problems are formulated with Rust and the generated code can also be called from C/C++. OpEn has been used to solve non-convex NMPC (non-linear model predictive control) problems for UAV obstacle avoidance and control (Lindqvist et al., 2020).

2.2 Related work on UAV trajectory planning

The objective of this section is to provide a general overview of the state of the art for UAV optimal trajectory planning. We survey recent works that formulate the trajectory generation problem as variants of (2.1), considering UAV dynamics and collision avoidance as core constraints, and then use the solvers described in Section 2.1. We place special emphasis on multi-UAV settings for trajectory planning, where inter-UAV constraints need to be integrated into the formulation.

In the literature, there is a distinction between trajectory and path planning: while the former results in time-indexed trajectories for UAV states (e.g., positions, veloctities, accelerations, etc.) that minimize a cost function typically related with energy consumption or flight time, the latter merely consists of finding a collision-free path from an initial UAV configuration to a goal which is a target configuration, given a workspace with obstacles. Although different, path and trajectory planning are closely related problems. Some methods for trajectory planning initiate their optimization procedure by using a piecewise path previously computed by a path planner (Tordesillas et al., 2021; Zhou et al., 2021; Liu et al., 2017; Oleynikova et al., 2020). Others also carry out a similar initialization process by means of precomputed paths that fulfill application requirements. For instance, this has been commonly done in UAV cinematography (Nägeli et al., 2017a; Joubert et al., 2016; Gebhardt et al., 2016), where initial paths can be calculated using well-established visual composition principles for cameras, and these can then be used to feed a trajectory optimization algorithm. As path planning is interconnected with trajectory optimization in many cases, we first overview the main approaches for UAV path planning in the next section.

2.2.1 Path planning

Most approaches for UAV path planning can be categorized into search-based or sampling-based methods. Search-based algorithms model the workspace as a graph and define a cost function over each vertex, in order to search for minimum-cost paths connecting the starting and goal vertices. A classic search-based algorithm such as Dijkstra can ensure that a solution is found, if one exists, but its computation time may not be short enough for real-time performance, mainly in 3D path planning scenarios. Therefore, heuristic algorithms such as A* (Hart et al., 1968) are typically used, as they can find nearly optimal solutions more efficiently. A drawback imposed by A* and other discrete search algorithms is that they constrain paths to a grid, so their solutions are indirect paths that sometimes poorly approximate the actual shortest path in continuous space. This issue is addressed by *any-angle* path planners, which allow any angle in the turns of the path, yielding paths with fewer turns that head toward the goal more directly. This is the case of the Theta* planner (Nash et al., 2007), built upon A*. Unfortunately, the number of line-of-sight checks that Theta* performs in 3D space is still rather high, increasing computational requirements and making it invalid for some real-time UAV applications. Lazy Theta* (Nash et al., 2010) was introduced as a simple improvement of the Theta* algorithm, reducing the number of line-of-sight checks. Thus, it finds paths faster than the original algorithm and enables search-based algorithms in real-time UAV applications (Faria et al., 2019; Wu et al., 2020). Perez-Grau et al. (2018) proposed a variant called Weighted Lazy Theta*, which can reduce the planning time by weighting the heuristic distance to the goal and avoiding the symmetric 3D expansion of vertices while searching for the solution path. Jump Point Search (JPS) (Harabor and Grastien, 2011) is another variant of A* that provides the same completeness and optimality guarantees as A*. Under the assumption of a uniform cost grid, JPS significantly speeds up the running time in 3D planning scenarios, which makes it suitable to be combined with trajectory optimization methods in receding horizon (Liu et al., 2017; Tordesillas et al., 2021). Ding et al. (2019) have also recently used a search-based algorithm to generate paths that are later combined with a UAV trajectory planner. Their Efficient B-spline-based

Kinodynamic (EBK) search algorithm finds B-spline control points on a spatial grid, then generates lowest-cost dynamically feasible trajectories in real time.

Sampling-based approaches have also become popular for path planning, as they work quite efficiently in high-dimensional configuration spaces. The most famous sampling-based algorithms are the Rapidly exploring Random Tree (RRT) and the Probabilistic Road-Map (PRM). Karaman and Frazzoli (2011) analyze these sampling-based algorithms more deeply to study the asymptotic behavior of their solution cost as the number of samples increases. RRT (LaValle and Kuffner, 2000) samples random points in the planning space and iteratively builds a tree connecting the samples with collision-free segments. Due to their efficiency in complex planning problems, RRT-like algorithms are even popular for kinodynamic planning, where trajectories holding to dynamic constraints are generated (Gammell et al., 2015; Karaman and Frazzoli, 2011; Janson et al., 2015). Richter et al. (2016) use the RRT* algorithm (Karaman and Frazzoli, 2011) to find a collision-free path through the environment, initially considering only the kinematics of the vehicle and ignoring the dynamics. The resulting path is then optimized to join its waypoints through a smooth minimum-snap trajectory. If a particular trajectory segment is found to intersect an obstacle after optimization, some works also use RRT algorithms to add additional waypoints between the two segment ends (Mellinger and Kumar, 2011; Richter et al., 2016; Loianno et al., 2017). Nonetheless, an issue with RRT algorithms is that they usually suffer when applied to complex kinodynamic systems, as they typically require solving a computationally expensive non-linear two-point boundary value problem (BVP) (Xie et al., 2015).

2.2.2 Trajectory planning

Due to the increase of computational resources for onboard computers and recent advances in efficient numerical methods, constrained optimization approaches are becoming commonplace for UAV trajectory planning. Some methods solve the trajectory planning problem using a two-stage pipeline (Ding et al., 2019), generating initial reference paths that later turn into optimized trajectories. The most common options are to calculate these initial references running a global path planning algorithm, or referring to specific requirements of the application at hand. For instance, Nägeli et al. (2017b) present a UAV cinematography system that generates reference paths as "virtual rails" in analogy to physical camera cranes and dollies. And Galvane et al. (2018) calculate the reference curve to be fit into a spherical space that ensures cinematographic properties. Once a reference is computed, the final optimized trajectory is usually obtained through a quadratic problem, simpler to solve than the original trajectory planning problem. Regarding this posterior optimization process, most state-of-the-art methods exploit the differential flatness of multirotor vehicles and, using an integrator model, minimize the squared norm of a derivative of the UAV position in order to obtain a dynamically feasible and smooth trajectory from the initial reference path (Mellinger and Kumar, 2011; Richter et al., 2016).

Other methods formulate the trajectory optimization problem without calculating a reference path, using cost functions and constraints that are specific for each application. For instance, cost terms to prevent UAV collisions are commonly defined using the distance to obstacles. Smoothness is usually achieved by penalizing jerky motions (i.e., accelerations and higher derivatives) that may lead to unstable flight. Some approaches maintain a formation between UAVs, formulating costs that, e.g., maximize the angular distance between multiple UAVs to cover the whole scene (Ho et al., 2021). In applications with cameras, occlusions can be handled by defining costs that measure the actor's environmental occlusion (Bucker et al., 2021; Bonatti et al., 2020). Others include costs of following cinematographic guidelines, such as Huang et al. (2019), where they formulate a cost to increase the camera-to-subject distance if the subject moves fast and vice-versa, ensuring smooth displacement of the subject.

Vehicle dynamics and bounds on the control and state variables are common problem constraints. Connectivity constraints are also typical in the case of multi-UAV formations (Mondal et al., 2018). Moreover, obstacle avoidance is of uppermost importance in UAV trajectory planning. Some approaches deal with this after solving the optimization problem (Mellinger and Kumar, 2011; Richter et al., 2016; Loianno et al., 2017), but many others integrate it into the problem formulation. In the following, we describe various options to address collision avoidance in UAV trajectory optimization problems.

A common approach is a cost function that penalizes distance to the obstacles (Oleynikova et al., 2016, 2018), usually requiring computationally expensive distance field representations. Bonatti et al. (2020) obtain the truncated signed distance transform (TSDT) map function, which linearly penalizes intersections with obstacles and decays quadratically with distance. Another option is to add obstacles as constraints in the optimization problem. Liu et al. (2017) does a convex decomposition of the known free space as a series of overlapping polyhedrons. They include the safe corridor, created as a set of linear inequality constraints in the QP problem. The drawback of this approach is that it can sometimes be very conservative, since the solver can only place the two end points of each interval in the overlapping area of two consecutive polyhedrons. A possible way to solve this problem, but usually with higher computation requirements, is to use binary variables (Landry et al., 2016; Deits and Tedrake, 2015) to allow the solver to choose the specific interval allocation. Tordesillas et al. (2021) propose a MIQP (mixed-integer quadratic programming) formulation that allows the solver to choose the trajectory interval allocation. Moreover, they solve the problem at high frequency in real time. Another approach is to encode the shape of the obstacles in the constraints using successive convexification (Augugliaro et al., 2012). However, successive convexification depends heavily on the initial guess and is usually unsuitable for real-time planning in unknown cluttered environments. In Luis and Schoellig (2019), collision avoidance constraints are linearized using a Taylor series expansion to transform the problem into a convex optimization. Zhu and Alonso-Mora (2019) transform collision constraints into deterministic constraints on the mean and covariance of the robot's state, and then linearize those deterministic constraints. Alonso-Mora et al. (2015b) linearize non-convex constraints leading to a convex optimization with quadratic cost and linear and quadratic constraints. Each non-convex constraint is approximated by five linear constraints, representing avoidance to the right, to the left, over and under the obstacle, and a head-on maneuver.

Most of the above methods try to transform non-convex constraints into a convex problem to facilitate their resolution. However, some authors have also addressed non-convex problems directly, using the frameworks mentioned in Section 2.1.3, which apply SQP, SCP, or interior-point methods. For example, (Nägeli et al., 2017a) formulate a non-convex problem to not enter zones with obstacles and use FORCES PRO to solve it. They use slack variables to relax the constraints guaranteeing feasible solutions in tight situations. It can be shown that under sufficiently high penalization of (a linear norm of) the slack variables, the solution of the hard-constrained problem is found when it exists; otherwise, a plan with minimum deviation is computed by the optimizer. This approach is also used in Castillo-Lopez et al. (2018). Joubert et al. (2016) formulate a non-convex problem to capture well-composed images and solve it using SNOPT. Alonso-Mora et al. (2015a) also use SNOPT to navigate a multi-UAV formation using SCP. Lindqvist et al. (2020) solve a non-convex NMPC using OpEn, for multi-UAV control with dynamic obstacles.

2.2.3 Multi-UAV trajectory planning

Multi-UAV trajectory planning consists of simultaneously computing trajectories for multiple vehicles that should be coordinated. Apart from the costs and constraints considered in conventional trajectory planning, multi-UAV settings impose additional requirements to enable team operation. On the one hand, inter-UAV distances need to be ensured, for safety or connectivity reasons. On the other hand, cooperative or formation behaviors (e.g., keeping a certain shape) are often required, depending on the application. Liu and Bucknall (2018) provide a survey on multi-UAV trajectory planning and formation control.

Methods for formation control strategies can be classified into leader–follower formations, virtual formations, or behavior-based formations (Liu and Bucknall, 2018). In the leader–follower approach, one vehicle functions as the leader, and the others act as followers, adapting their motion to the leader. Krátký et al. (2020) propose a multi-UAV formation with a leader that is filming a scene while the others provide proper lighting. They solve a special optimization problem for the leader, and use a different formulation for the followers, so that they can illuminate adequately from different angles. Virtual formations aim to keep a geometric shape with respect to a moving reference frame which is virtual; i.e., they use a virtual leader. The idea is to maintain a formation by minimizing position errors between the virtual structure and the actual UAV positions. Tanner and Kumar (2005) present a decentralized cooperative control scheme that allows a team of UAVs to asymptotically converge to the desired formation of a particular shape and orientation from almost any initial conditions. Also, many multi-UAV MPC-based methods minimize the distance to a reference path, making it possible to maintain precomputed virtual formations (Tallamraju et al., 2018; Price et al., 2018). Instead of a rigid geometric shape, behavior-based formations follow a more flexible approach, generating control commands that are based on various aspects of the mission. For instance, in applications with cameras, multi-UAV teams could maintain a formation to film a scene from multiple perspectives, without entering each other's camera fields of view and minimizing other costs related to the smoothness of the camera motion (Bucker et al., 2021).

In terms of distribution of the computation, methods can be classified as centralized or decentralized. Centralized methods solve a single problem for the whole team, concentrating all computation on a single computer. A central node accesses information from all vehicles and computes the multi-UAV plan (Maza et al., 2015). In decentralized methods, computation is distributed among the computers of each UAV in the team. Some decentralized approaches require that some global information is available for all UAVs, whereas in others, UAVs are only required to access local information. Centralized methods are usually better for stability and could reach better solutions in terms of efficacy, as they consider global information, but they scale poorly with the number of UAVs. On the other hand, decentralized methods alleviate computational and communication bandwidth requirements (Maza et al., 2015), but they are only possible when local information is good enough to achieve coherent and cooperative behaviors.

Mellinger and Kumar (2011) solve the trajectory generation problem in a centralized manner, formulating a MIQP. However, they do not achieve real-time performance due to their high computational cost. Sequential convex programming has been used in Augugliaro et al. (2012), leading to faster computation but still not in real time. In order to improve computational efficiency, decoupled planning methods have been proposed (Čáp et al., 2015; Yu et al., 2016; Robinson et al., 2018). These methods solve a centralized problem sequentially by avoiding the previously planned robots in a prioritized manner, improving computational efficiency. MPC-based approaches have also been used for trajectory computation in centralized multi-UAV architectures, taking advantage of receding horizon approaches to solve the problem in real time. For instance, Erunsal et al. (2019) present a centralized leader–follower scheme, where the leader executes a centralized NMPC approach and communicates with the followers to obtain their local sensor information and deliver velocity commands to them. Another centralized method is proposed in Alonso-Mora et al. (2015a). They compute the largest collision-free convex polytope in a neighborhood of the UAVs, followed by a constrained optimization via SCP. Li et al. (2021) first apply a graph-based multi-agent path planner to find an initial discrete solution, and then this solution is transformed into smooth trajectories using non-linear optimization. Although they compute trajectories in a centralized way, they divide the robot team into small sub-groups and propose a prioritized trajectory optimization method to improve the algorithm scalability. Ho et al. (2021) also develop a two-stage approach with long planning time horizons and real-time performance. They use a centralized planner for formation control, but also a decentralized trajectory optimizer that runs on board each UAV.

Optimization methods to tackle multi-UAV trajectory planning in a decentralized fashion also exist. Many of these attempt to avoid non-convex constraints in order to enable real-time performance and scale with the number of UAVs. For instance, the authors in Tallamraju et al. (2018); Price et al. (2018) present a method where each UAV executes a local motion planning algorithm based on MPC, and use repulsive potential field functions for obstacle avoidance. They propose a novel mechanism to convexify these non-linear potential field functions, embedding them in the optimization framework. Mondal et al. (2018) also use potential fields to ensure inter-agent collision avoidance and connectivity, and a consensus strategy to keep a desired formation via velocity agreement among the agents. The main problem of these methods is that due to the use of constraints based on potential fields, they are prone to getting trapped in local minima. In Alonso-Mora et al. (2016), they extend their previous centralized approach (Alonso-Mora et al., 2015a) to achieve a decentralized version. Via distributed consensus, UAVs compute the convex hull of their positions and the largest convex region within free space. Then they apply SCP to compute locally optimal formation parameters within this convex neighborhood. Alonso-Mora et al. (2018) consider a set of motion primitives for the robots and solve a multi-robot optimization problem in the space of control velocities with additional convex constraints. Park and Kim (2021) formulate another distributed convex optimization problem constructing relative safe flight corridors. Kamel et al. (2017) consider non-convex constraints for collision avoidance, incorporating them into an NMPC decentralized problem. Still, they only activate them if a potential field algorithm cannot maintain a minimum distance. In Nägeli et al. (2017b), they formulate non-convex constraints with slack variables to maintain the formation, since the performance is enough for a few UAVs in the team. Luis and Schoellig (2019) introduce on-demand collision avoidance in a decentralized MPC framework, where they detect and resolve only the first collision within a finite prediction horizon using SCP, modeling each collision as an ellipsoid and implementing soft constraints.

Even though they might not be considered trajectory planning, it is worth mentioning that there are many distributed approaches based on reactive methods, where UAVs use only local information to continuously react to possible collisions. Many of these reactive approaches are based on the concept of *velocity obstacles*. For example, optimal reciprocal collision avoidance (ORCA) has been used to guarantee collision-free trajectories for non-holonomic agents (Alonso-Mora et al., 2013). Gopalakrishnan et al. (2017) propose the concept of chance constraints to derive PRVO, a probabilistic variant of RVO (reciprocal velocity obstacle) (van den Berg et al., 2008). They take into account the uncertainty associated with both the state estimation and the actuation of each robot. Reactive planning methods are computationally efficient, but they have no guarantees about deadlock avoidance and are poorly suited to problems in maze-like environments.

Chapter 3

Scheduling and execution of cinematography missions with multiple UAVs

This chapter presents a framework for autonomous scheduling and execution of cinematography missions with multiple UAVs. We devise a methodology for parametric description of generic aerial shots, so that a human operator or *director* can design the cinematography missions to be executed by the autonomous multi-UAV team. Our framework integrates components to plan and schedule the shots, and to execute them in parallel or sequentially with multiple UAVs. Even though there are also components for target tracking, UAV motion planning, and gimbal control, the chapter focuses on those related to multi-UAV shot execution. For this, we propose a distributed scheduler that runs on board each UAV and activates different shot controllers depending on the shot type. These controllers are in charge of both UAV and gimbal motion. Then, an event-based system is used to synchronize shot execution among the UAVs and ensure proper coordination. Furthermore, we increase the system robustness by considering contingency plans. In particular, our system is able to react to possible UAV failures (e.g., lack of battery or GPS signal), re-planning the remaining shots with the available UAVs, and leaving the failed ones to perform emergency maneuvers.

3.1 Introduction

The interest in multi-UAV cinematography is growing rapidly due to the capacity of a multi-UAV mission to film several action points at the same time, different perspectives of the same target, or to cover large-scale scenarios. However, there is still a need for autonomous systems that are intelligent enough to execute cinematography shots with multiple action points and multiple UAVs. This implies, for instance, predicting how the scene will evolve and being able to schedule shots that may be happening sequentially or in parallel, as well as coping with possible failures and contingencies.

In this chapter, we focus on the high-level assignment and execution of the aerial shots belonging to a multi-UAV cinematography mission. In Section 3.2, we review similar systems to design and execute cinematography missions with one or multiple UAVs. The main contributions of this chapter are the following:

- We present a complete architecture for autonomous execution of cinematography missions with a team of UAVs (Section 3.3). We formulate the problem of autonomous cinematography as two steps: mission planning and execution. In particular, this chapter describes our novel solution for mission execution. Even though we implement a representative set of canonical shots, we also generalize the way to describe parametric shots, making the system easily extendible. In this way, we allow for different camera motion modes, including actual target tracking and predefined virtual rails.

- We describe our method for cinematography mission execution (Section 3.4), which is agnostic to the planner used to schedule and assign shots to the available UAVs. We propose distributed schedulers that trigger the execution of the different shots based on starting events that may be generated manually or autonomously (e.g., a certain actor reaching an action point). This works as a synchronizing mechanism for multi-UAV shots but also makes the system robust to uncertainties in the planning phase (e.g., the planned starting time of some action becoming delayed). Moreover, our onboard controllers implement shots autonomously decoupling gimbal and UAV motion, which improves robustness to noisy actor measurements (compensating through gimbal control).

- We provide an open-source implementation of our system [1] using off-the-shelf hardware, and validate it for outdoor media production with multiple UAVs (Section 3.5). In particular, we show our field experiments filming several sport activities (including a real regatta), with the system running all components on board in real time. We also report on lessons learnt after our experimental campaigns within the framework of the MultiDrone project, which are supported by feedback provided by the media experts involved in the project.

3.2 Related work

There are multiple commercial products for UAV cinematography in outdoor settings. On the one hand, aerial platforms such as DJI Mavic (DJI, 2018), Skydio (Skydio, 2019), 3DR SOLO (3DR, 2015) or Yuneec Typhoon (Yuneec International, 2018) offer good performance, including some semi-autonomous functionalities for tracking moving targets visually or by GPS, as well as simplistic collision avoidance. However, the set of shots is predefined and not easily extensible, as their software suites are not open-source. Besides, they do not consider multi-UAV systems nor multi-shot scheduling. On the other hand, there are commercial applications to enhance the user experience. Skywand (Garage, 2019) is a virtual reality system that allows the user to explore the scene and select desired key-frames within the virtual environment. Then the system computes a UAV trajectory for a smooth shot containing these key-frames. Freeskies CoPilot (FreeSkies, 2019) is a mobile software suite that offers similar functionality but with a simple 3D map instead of a virtual reality interface. In both cases, the resulting UAV autonomy and environment perception are minimal; the cinematography plans consist of example key-frames and they cannot be adjusted online.

A complete system for UAV cinematography in unstructured environments is presented in Bonatti et al. (2020). They combine vision-based target tracking with a real-time motion planner that avoids collisions and fulfills artistic guidelines. They show impressive field experiments, but their focus is mainly on mapping and obstacle

[1] `https://github.com/grvcTeam/multidrone_planning`

avoidance rather than multi-shot scheduling. Moreover, only a single UAV is considered, as well as a simplified set of shots: left, right, front, back. The same authors also extended their work to multi-UAV systems (Bucker et al., 2021), focusing on filming in unstructured cluttered environments.

An approach for cinematography with multiple UAVs is described in Nägeli et al. (2017b). They resolve a non-linear optimization problem to generate 3D trajectories for the UAVs. Aesthetic objectives and collision avoidance between the UAVs and with the filmed actors are considered. The problem is solved on each UAV in a distributed fashion, after exchanging planned trajectories. The authors extend their own previous work (Nägeli et al., 2017a) by including multiple UAVs and preference trajectories from the user as virtual trails. Although the approach is quite promising for autonomous cinematography, it is only tested in indoor settings and does not consider the scheduling of multiple shots as we do. The work in Galvane et al. (2018) is closer to ours, as the authors also propose a complete architecture for cinematography with multiple UAVs. The motion of the multiple UAVs around dynamic targets is coordinated by means of a master–slave approach that resolves conflicts: only one master UAV is supposed to be shooting the scene at a time, while the slaves offer alternative viewpoints or act as replacements. Moreover, the user can only select among different framing types. In contrast, our system adds more flexibility, as we define framing and shot types, as well as introducing multi-view shots more explicitly, allowing different types of shots to take place concurrently. In addition, the system in Galvane et al. (2018) is only tested in indoor settings, with a Vicon motion capture system that provides accurate positioning for all targets and UAVs.

Recently, the MultiDrone project, of which this work is part, has concluded successfully, producing an integrated system for autonomous cinematography with multi-UAV teams for outdoor sport events. In the MultiDrone project, a new taxonomy for cinematographic shots with UAVs was proposed (Mademlis et al., 2019; Mademlis et al., 2019a), and with the support of experts from the media production companies involved in the project, a set of representative shots was selected to be implemented autonomously by the system. These shots can be defined by the *media director* through a high-level graphical interface with a novel language that was created for cinematography mission description (Montes-Romero et al., 2020). The director indicates the desired shot types, starting times/positions and durations; but does not assign specific UAV cinematographers to them. Instead, the system autonomously computes feasible plans for the UAVs (Caraballo et al., 2020), considering constraints such as their remaining battery, no-fly zones, collision avoidance, and more. Each UAV is scheduled to take one or several shots, together with the *events* that will trigger each shot. These shots may be sequential, filming different action points along time (or the same point with different views); or they may be concurrent shots with multiple UAVs filming one or several action points. The focus of this chapter is on mission execution, so we assume these planned schedules for each UAV as a starting point. Various different planning techniques may be used to compute those schedules (Torres-González et al., 2017; Caraballo et al., 2020). Based on those plans, we propose a general architecture to execute the scheduled shots in a cooperative manner with the multi-UAV team.

3.3 System architecture

In this section, we present the complete architecture of our autonomous system for multi-UAV cinematography. We assume that there is a media director in charge of designing the mission by describing multiple shots from a high-level and artistic point of view. This director is then supported by autonomous components that are able to compute plans to execute the designed shots and carry out the mission with a team of UAV cinematographers. Our system separates the whole cinematography problem into two sub-tasks: mission planning and mission execution.

Mission planning: *Given an input cinematography mission, this sub-task consists of deciding which UAV should execute each of the shots. For each shot, the director specifies (among other parameters) a starting position and time for the action to be filmed, and the desired duration and type. Taking into account the initial position and remaining flight time of the UAVs, a schedule with the shots assigned to each UAV must be computed.*

This problem can be solved with scheduling and task allocation algorithms. Each shot represents a task with a duration and an estimated starting time and position, and it must be ensured that each UAV has enough flight time to cover all its assigned shots. After every shot, a path to reach the starting position of the next shot is necessary. For this, an estimation of the ending position of the UAV after the shot is required. For certain sports events such as those in our work (e.g., rowing or cycling races), this can be assumed, as the targets move along a predefined route with an approximate known speed. We integrated an algorithm for optimal mission planning with time constraints and avoiding inter-UAV conflicts (Caraballo et al., 2020), although our architecture is more general and could accommodate alternative methods (Natalizio et al., 2019). The integrated planning algorithm maximizes the percentage of shots covered by the multi-UAV team and it provides as output a list of actions for each UAV in the team. We consider two types of actions: *Navigation Actions* (without filming); taking off, landing, and navigating from one shot to the next; and *Shooting Actions*; executing a specific shot. Shooting Actions involve concurrent UAV and gimbal control, and they can have an associated starting *Event* which triggers execution. The planner computes the plan in a centralized fashion, with all available information from the UAV's states and shots designed by the director. It considers as constraints the remaining battery (i.e., flight time) of each UAV, and the fact that they need to cross (when navigating between shots) at no less than a certain minimum distance from each other in order to prevent them from colliding. Also, it avoids flying UAVs above predefined no-fly areas which are specified by the director (buildings, the audience of a sports event, etc). Thus, due to all these constraints, the algorithm may not find a valid plan, or it may return a plan where the original shots are only partially covered. We leave mission planning out of the scope of this thesis and concentrate on the problem of mission execution.

Mission execution: *Given a plan for a cinematography mission; i.e., the list of Shooting and Navigation Actions assigned to each UAV, this sub-task consists of executing the shots in a synchronized manner with a multi-UAV team. This means triggering gimbal and UAV controllers depending on the shot type, avoiding UAV collisions, and performing target tracking.*

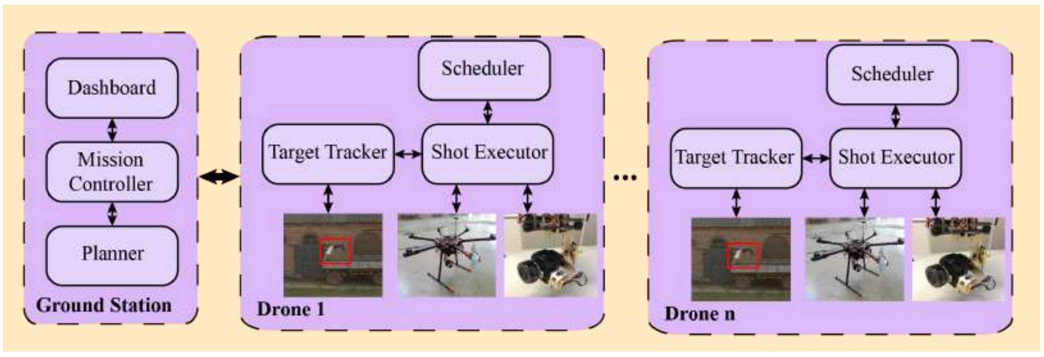
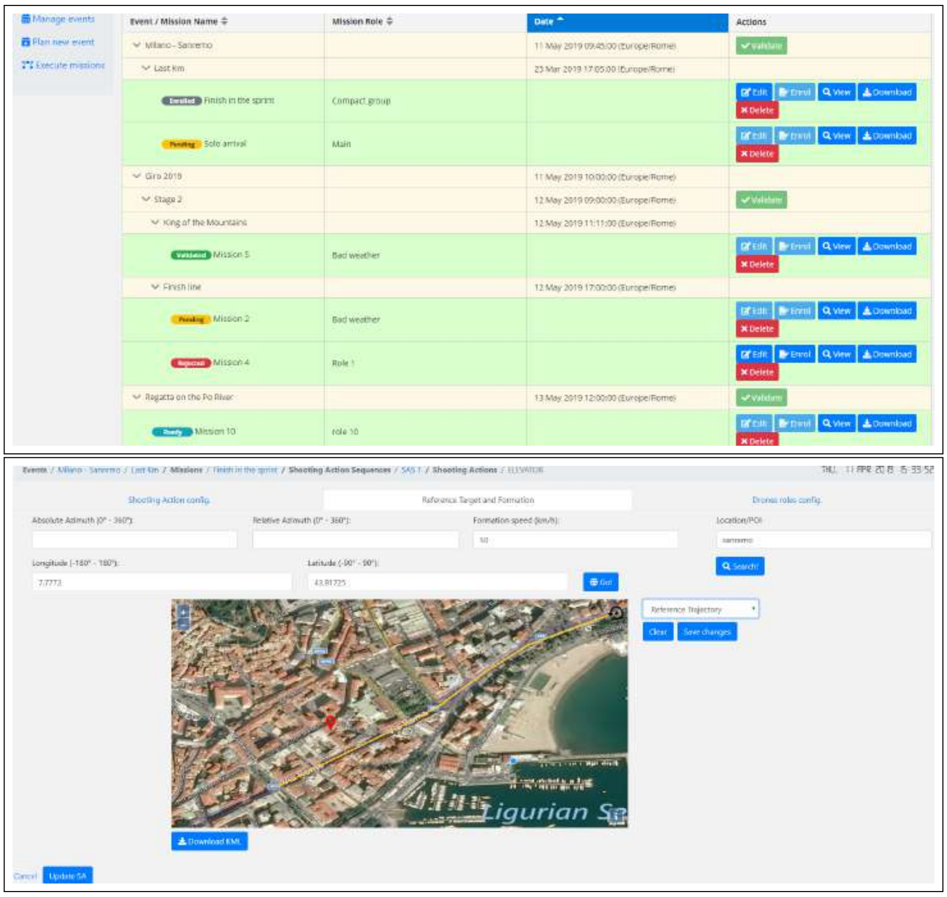


Figure 3.1: System architecture for multi-UAV cinematography. Mission design and planning components run on a Ground Station, but components for mission execution run on board the UAVs.

We solve this problem by means of a set of distributed shot schedulers and executors that run on board the UAVs. For mission planning, we assume that the director can estimate the occurrence time for the Events triggering Shooting Actions. We also assume that target trajectories can be predicted approximately. However, the system tolerates errors in these estimations to a certain extent, since it reacts online during mission execution in two ways: (i) UAVs can wait at shot starting positions until the shot execution is triggered, to account for delays in the action to be filmed; and (ii) UAVs can track actual target trajectories instead of planned ones during shot execution, to account for possible deviations.

3.3.1 System overview

Figure 3.1 shows the complete architecture of our system. Components related to mission planning are executed on a Ground Station that interfaces with the director, while components related to mission execution run mainly on board the UAVs. The *Dashboard* is a graphical tool for human-computer interaction between media end-users and the rest of the system. Figure 3.2 shows snapshots of the most representative windows. Further details about the Dashboard and mission design can be seen in Montes-Romero et al. (2020). In summary, this component enables the director to design cinematography missions, including all shot descriptions and their triggering Events, when needed. For instance, a director could design a mission to film a rowing



Event / Mission Name	Mission Role	Date	Actions
Milano - Sanremo		11 May 2019 09:45:00 (Europe/Rome)	Validate
Last Km		23 Mar 2019 17:05:00 (Europe/Rome)	
Enrolled Finish in the sprint	Compact group		Edit Enrol View Download Delete
Pending Solo arrival	Main		Edit Enrol View Download Delete
Giro 2019		11 May 2019 10:00:00 (Europe/Rome)	
Stage 2		12 May 2019 09:00:00 (Europe/Rome)	Validate
King of the Mountains		12 May 2019 11:11:00 (Europe/Rome)	
Validated Mission 5	Bad weather		Edit Enrol View Download Delete
Finish line		12 May 2019 17:00:00 (Europe/Rome)	
Pending Mission 2	Bad weather		Edit Enrol View Download Delete
Rejected Mission 4	Role 1		Edit Enrol View Download Delete
Regatta on the Po River		13 May 2019 12:00:00 (Europe/Rome)	Validate
Ready Mission 10	role 10		Edit Enrol View Download Delete

Events / Milano - Sanremo / Last Km / Missions / Finish in the sprint / Shooting Action Sequences / SAS 1 / Shooting Actions / ELEVATOR

Shooting Action config. | Reference Target and Formation | Drones roles config.

Absolute Azimuth (0° - 360°):
Relative Azimuth (0° - 360°):
Formation speed (km/h): 50
Location/POI: sanremo
Longitude (-180° - 180°): 7.7773
Latitude (-90° - 90°): 43.81725
Reference Trajectory

Figure 3.2: Snapshots of the Dashboard graphical interface. Upper: several events with different associated missions. Lower: an example of the window to design a specific Shooting Action within a mission, indicating the Reference Target trajectory.

race, specifying a lateral shot from the `START_RACE` Event to the end of the race, and an orbital shot starting with the `FINISH_LINE` Event; i.e., when the boats reach the finish line. A novel cinematography language (Montes-Romero et al., 2020) was proposed in MultiDrone so that the director's input is written with a specific syntax that is later understandable for our planning components.

At the Ground Station, there is another central component called *Mission Controller*, which manages the whole planning and execution process for a mission. This module receives the director's input through the Dashboard and it uses the *Planner* component to compute feasible plans in order to execute the mission. Then the

Mission Controller sends each UAV its plan, which basically consists of a list of Shooting Actions to execute the assigned shots, with interleaved Navigation Actions to fly between shots. During the mission execution, the Mission Controller monitors the UAV status for possible contingencies and sends out the triggering Events as they actually occur. Depending on the Event, this occurrence may be automatically detected by the Mission Controller or manually indicated by the director.

Components on board the UAVs manage mission execution. Communication with the Ground Station is done by means of an LTE (Long Term Evolution) link (Mademlis et al., 2019b) through the *Scheduler* components, which are the ones receiving plans and Events from the Mission Controller. They are in charge of executing shots in a distributed manner with multiple UAVs. Each Scheduler listens to Events and starts/stops the execution of Shooting Actions as required. These Events act as a synchronizing mechanism for multi-camera shots, since all the involved UAVs wait for the same Event to start. Shooting Actions are carried out by calling the *Shot Executor* component, which implements UAV and gimbal controllers. Depending on the shot parameters, the Shot Executor adapts its controllers to execute the corresponding shot. A *Target Tracker* module is necessary to provide positioning of the target, which is used by the Shot Executor to point the gimbal and move the UAV accordingly. Navigation Actions are also managed by the Shot Executor, but with different controllers that do not consider gimbal motion or cinematographic constraints.

Our system is flexible enough to adapt to upcoming situations during execution. In particular, we allow for mission re-planning due to a director's choice or in case of unexpected situations. The former is manually triggered, but the latter is autonomously managed as follows. Schedulers report back the status of the mission execution to the Mission Controller; i.e., which action each UAV is executing or waiting for. In the event of an emergency in a UAV; e.g., low battery or loss of GPS, the corresponding Scheduler is able to trigger an emergency maneuver (landing safely), but at the same time, it informs the Ground Station about the situation. Then the Mission Controller starts a re-planning procedure through the Planner component, involving only the available UAVs and the remaining shots to be executed. Once those new plans are sent to the UAVs, each of them will finish its current ongoing action, and will append the new list of actions behind the current one. The other safety mechanism that is considered in our architecture is collision avoidance. This is integrated at planning level within the Planner, and at execution level within the Shot Executor. First, our Planner uses a high-level map of the environment (including no-fly zones due to obstacles, audience, etc.) to provide collision-free paths. It also resolves inter-UAV conflicts when their paths become too close. Second, our Shot Executor runs collision avoidance online to react to unexpected situations and maintain safe inter-UAV distances.

3.3.2 Shot description

Our multi-UAV system autonomously takes a series of shots that are represented by Shooting Actions. All the properties for each shot are encoded through the attributes of its corresponding Shooting Action. Table 3.1 gives the definitions of a shot, with the multiple properties that can be specified when designing the shot.

Attribute	**Data type**	**Description**
Shot type	Discrete value	Chase, lateral, orbit, etc.
Framing type	Discrete value	Long shot, medium shot, close-up shot, etc.
Start Event	String	Event that triggers this action
Duration	Time	Duration of the shot
RT path	List of global positions	Estimated path of the RT
RT speed	Float	Speed along the RT path
RT mode	Discrete value	virtual-traj, virtual-path or actual-target
RT ID	String	Identifier of the RT to follow
ST type	Discrete value	Virtual, real, or none
ST ID	String	Identifier of the ST to follow
Shooting parameters	Set of parameters	E.g., relative distance to RT, angular velocity in an orbit, etc.

Table 3.1: Attributes of a Shooting Action for shot definition.

The shot type describes the kind of movement of the camera with respect to the action; i.e., chasing, orbiting around, etc. We will define all shot types in Section 3.3.3, together with the shooting parameters defining their geometry. Apart from the shot type, we need to specify the framing type (this indicates how close the action will appear in the image; i.e., the zoom level), the duration, and the starting Event. This latter is optional; if not specified, the shot would start right after the previous one. In addition, we create two relevant concepts to describe shots: the Reference Target (RT) and the Shooting Target (ST). The RT is used to guide UAV motion, as the UAV should follow this target describing its corresponding type of shot. The ST is used to guide gimbal motion, as the camera should point at this target when filming. The two targets could coincide, but not necessarily. For instance, we may want a camera moving along a lateral rail but filming a static scene or an actor moving in a different direction.

We specify the RT path as a list of waypoints expressed in global coordinates, and depending on the RT mode, we define three different kinds of motion for the UAV:

- Mode *virtual-traj*: A virtual UAV trajectory is specified. The UAV should move along the rail indicated by the RT path and at the velocity specified by the RT speed.
- Mode *virtual-path*: A virtual UAV path is specified but no speed is provided. The UAV should move along the rail indicated by the RT path but at the speed of an actual target, which would be indicated by the RT ID.
- Mode *actual-target*: No virtual path is indicated for the UAV, which should move following an actual target specified by the ST.

The above modes widen the spectrum of possibilities for the director and were actually recommended by media experts from our end-user partners in the MultiDrone project. On top of that, we can track different targets with the UAV and on the image; i.e., having non-coincident RT and ST. We consider three types of ST: (i) *virtual*, if the target is specified as a virtual point or path; i.e., through the RT path; (ii) *real*, if it is an actual physical target (e.g., a cyclist, a runner, etc.) whose position can be estimated, for instance through visual detection or with a mounted GPS; and (iii) *none*, if the camera is simply fixed or following a predefined motion. In the case of a real ST, an ST ID can be indicated to identify the specific target to track visually or the corresponding GPS transmitter. The RT ID plays a similar role when we use the virtual-path RT mode to track an actual target with the UAV.

Finally, notice that our shot description only requires a starting Event for particular shots. The director may want to take a series of sequential shots after a given Event, to take several views along the line of action. For that, she/he would only need to specify the starting Event for the first shot, and the others would happen consecutively. Furthermore, it is important to highlight how multi-UAV shots are addressed within this framework. The director could design multi-camera shots to be done by a formation of multiple UAVs simultaneously. For that, she/he could assign the same starting Event and RT to several Shooting Actions. Thus, all the UAVs involved would track a common reference trajectory together, implementing complementary shots of the same or different types. The shooting parameters for each Shooting Action would determine the geometry of the formation, and the starting Event would synchronize the motion so that they all start shooting simultaneously.

3.3.3 Canonical shots

In this section, we describe the set of shots that have been implemented for our system. In the cinematography literature there is a lot of information about cinematographic rules and canonical types of shots (Smith, 2016). Within the context of the MultiDrone project, a wide spectrum of shots was studied (Mademlis et al., 2019; Mademlis et al., 2019a). Following the recommendations of the media experts in the project, we selected our canonical list of representative shots for the autonomous system. In the following, we describe the shot types and the specific shooting parameters considered for each of them. Table 3.2 summarizes all the parameters, and Figure 3.3 depicts the shot geometry. Note that the parameters indicating the shot geometry are coordinates expressed in the RT frame. Moreover, unless the stated otherwise, the UAV yaw is such that it points forward toward the direction of movement.

Static: The UAV remains stationary above a fixed RT location at a height indicated by the parameter z_0. Since the RT represents a static position, the only RT mode that

Shot type	**Shooting parameters**
Static	pan_s, $tilt_s$, pan_e, $tilt_e$, z_0
Fly-through	pan_s, $tilt_s$, pan_e, $tilt_e$, z_0
Elevator	z_s, z_e
Chase/lead	x_s, x_e, z_0
Flyby	x_s, x_e, y_0, z_0
Lateral	y_0, z_0
Establish	x_s, x_e, z_s, z_e
Orbit	r_0, $azimuth_s$, $angular\_speed$, z_0

Table 3.2: Shooting parameters for each shot type.

makes sense is virtual-traj. Depending on what the gimbal tracks, the ST type can be real or virtual. The ST type "none" can be used to implement scene-centered shots, in which the gimbal moves independently. In this case, the parameters pan_s, pan_e, $tilt_s$ and $tilt_e$ indicate the pan/tilt starting and ending angles, respectively. Note that in this shot type, the UAV yaw will be such that it coincides with the camera pan angle, which is specified in the global reference frame. Tilt angles are specified in the UAV reference frame for convenience.

Fly-through: The UAV flies through the scene following a predefined path with no specific target to track. As in the previous shot, the only possible RT mode is virtual-traj, as there is no actual target. The flight altitude over the RT path is indicated by the parameter z_0. The ST type is always "none" and there are extra parameters to describe gimbal movement along the shot duration; pan/tilt starting and ending angles (pan_s, pan_e, $tilt_s$, and $tilt_e$). In this case, the UAV yaw follows the general rule to point in the direction of the movement, and both pan and tilt angles are specified in the UAV reference frame.

Elevator: The UAV moves vertically straight up or down tracking an actual target or a static position. The UAV starts the shot above a given position (defined as the initial RT location) at altitude z_s and ends at z_e. Therefore, the RT mode is virtual-traj, but the ST type could be real or virtual. The UAV yaw points to the ST.

Chase/lead: The UAV chases a target from behind with constant or decreasing distance, or moves ahead of it with decreasing or constant distance. All RT modes are

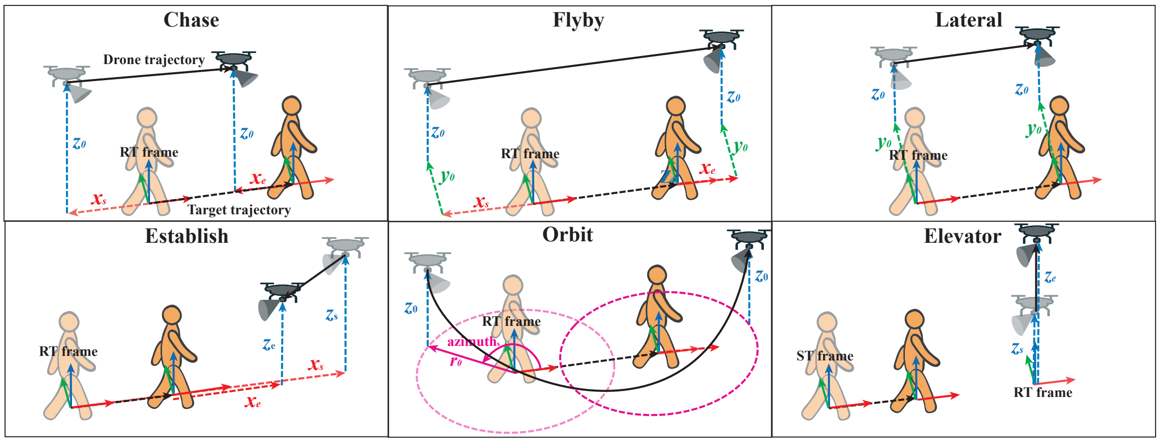


Figure 3.3: Scheme describing the shot geometry for the different shot types. Shot parameters refer to the RT frame. Lighter figures represent the start of the shot, and solid figures the shot end.

possible, depending on whether a virtual or an actual target is followed; for ST, only the "real" type makes sense. Regarding parameters, z_0 determines the UAV height over the RT, and x_s and x_e the starting and ending distances on the X axis (pointing forwards) with respect to the RT.

Flyby: The UAV flies past a target, normally overtaking it as the camera tracks it. The RT could be virtual or real, so all RT modes are possible; for ST only the "real" type makes sense. For parameters, it needs distances with respect to the RT; z_0 for the altitude, x_s and x_e for the starting and ending distances on the X axis, and the constant lateral distance y_0.

Lateral: The UAV flies beside a target with constant distance as the camera tracks it. The RT could be virtual or real, so all RT modes are possible; for ST, only the "real" type makes sense. For parameters, it needs z_0, the altitude with respect to the RT, and the constant lateral distance y_0.

Establish: The UAV moves closer to a target from the front, typically with decreasing altitude. The RT could be virtual or real, so all RT modes are possible. The ST type could be real or virtual (the latter could be for example to descend to a monument or static scene). Both altitude and displacement on the X axis with respect to the RT change during this shot, so it needs as parameters z_s, z_e, x_s and x_e.

Orbit: The UAV moves around a target describing a full or partial orbit. The RT could be virtual or real, so all RT modes are possible. The ST type could be real or virtual (the latter, for example, to orbit around a monument or static scene). The parameters in this case include the altitude above the RT (z_0), the radius of the circle (r_0), the starting azimuth angle for the orbit ($azimuth_s$), and the angular speed ($angular\_speed$).

Finally, it is important to notice the following issue. As the shooting parameters describing the shot geometry are expressed in the RT frame, in the event of this being an actual target, there may be situations where the target is turning at a high rate, causing the UAV to make abrupt maneuvers to relocate itself accordingly. For such a situation, as will be explained in the following sections, we estimate the target position and velocity by means of a stochastic filter implemented in the Target Tracker module, which smooths noisy target direction changes. The controller for shot execution follows a trailer-like approach with respect to the target to produce smooth reference trajectories. For safety reasons, we limited the maximum linear and angular speeds of the UAVs, forbidding them from performing these risky maneuvers in any situation.

3.4 Distributed mission execution

In this section, we describe our autonomous components for cinematography mission execution. More specifically, this is the part of our system architecture that runs on board each of the UAVs.

3.4.1 Scheduler

The execution of UAV shots is carried out by means of a distributed scheduling procedure. Each UAV runs a Scheduler component onboard that receives the plan for that UAV and coordinates the execution of the assigned shots, taking into account the other UAVs involved and the actual evolution of the scene. In particular, the Scheduler receives a list of sequential Navigation and Shooting Actions. Navigation

Actions only imply UAV movement through the scenario, without filming. This is mainly to get to the starting position of an upcoming shot or to go for the landing, so we only consider three types: take-off, land, and go to a waypoint. In the last case, either a single waypoint or a list of waypoints to navigate through can be provided. Shooting Actions are those involving some filming of the scene. They require a special controller to simultaneously take care of UAV and gimbal motion while the particular shot is being executed. Thus, the set of available Shooting Actions coincides with the shots described in Section 3.3.3.

The Scheduler controls the start and end of each action, handling the Shot Executor accordingly. For each Shooting Action, the UAV is sent to its corresponding starting position through a sequence of Navigation Actions that were computed by the Planner. Then it hovers at that starting position waiting for the Event associated with the Shooting Action. Once the Event arrives from the Mission Controller, the Scheduler activates the Shot Executor to start the Shooting Action. These Events represent actual action points of the scene being filmed, such as the start of a race, the runners reaching a particular point of interest or the finish line. Typically, the director may want to assign pre-designed sequences of shots for these moments. Moreover, if the Shooting Action has a specified duration, the Scheduler is in charge of waiting for that time before calling off the shot and continuing with the next action. In the case of Navigation Actions, the Scheduler just waits for notification of completion, and then it goes on with the next action in the sequence.

This event-based mechanism allows us to account for inaccuracies in the planning phase and for required adjustments during the actual filming of the scene. We assume that the Planner can estimate the occurrence time for the Events and an approximate target trajectory, which permits computation of a plan. However, the system does not rely on estimated times for mission execution, but on the actual occurrence of the Events. Thus, we plan so that UAVs arrive earlier than expected at their starting positions, and then wait for Events, to allow for possible delays in the actual scene being filmed. These Events could be detected online by the system automatically. For instance, in rowing races, the launch signal could be communicated to the Mission Controller, and the race reaching specific points of the route could be detected by

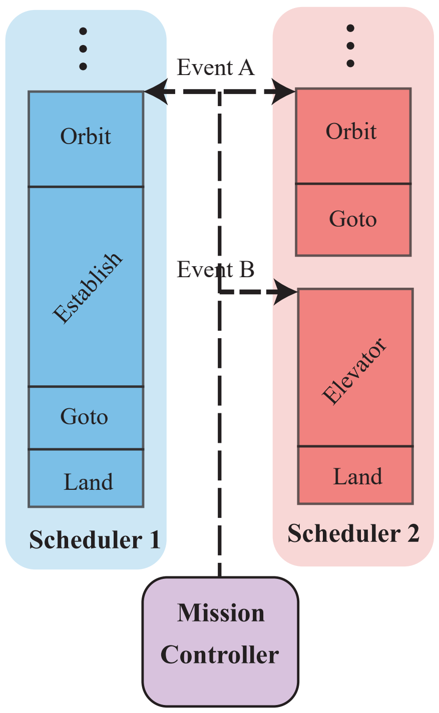


Figure 3.4: Example of the event-based procedure for a distributed execution of a mission with two UAVs.

monitoring GPS trackers on board some of the boats. We also allow the director to send out Events manually to decide on shot triggering. For multi-camera shots, the Events also act as multi-UAV synchronizing signals. All the involved UAVs will be waiting at their starting positions and the Event will ensure that they all start at the right time simultaneously. Figure 3.4 shows an example of how the distributed scheduling works. In the example, two UAVs take an orbit shot in a synchronous manner, being triggered by a certain Event A in the scene. Then, immediately after the orbit, UAV 1 takes an establishing shot and goes back to its station to land, while UAV 2 goes to a new starting location to wait for Event B, which triggers an elevator shot that ends its mission.

Additionally, the Scheduler component integrates functionality for emergency management, which is crucial for safety. Each Scheduler monitors the UAV status for hardware issues. In particular, we implemented low battery alerts, loss of GPS signal and loss of communication with the Ground Station, but any other kind of contingency could be monitored. In the event of failure, the Scheduler reports that event to the Mission Controller on the Ground Station. Then the Mission Controller may decide to launch a re-planning procedure without the affected UAV, reassigning its pending tasks to others. Simultaneously, the Scheduler carries out an emergency maneuver: it cancels the action being executed and commands the UAV to navigate to the closest base station for landing. For this, the Scheduler can plan safe paths that avoid no-fly zones. We implemented an off-the-shelf A* heuristic planner (Hart et al., 1968) on a KML-based map that includes information about the positions of the base stations and the no-fly zones (areas with known obstacles or people gathered as an audience). In the particular case of losing communication with the Ground Station, the UAV continues with the mission as long as possible, and it returns home when some critical information, for example about the target, is required.

3.4.2 Shot executor

This component is in charge of executing Navigation and Shooting Actions. In order to execute a Shooting Action, we need to generate the desired trajectory, which is derived using the shot type, the shooting parameters and the target position. Given the target, which may be real or virtual depending on the RT mode, we make the UAV behave like a trailer attached to that target (Pereira et al., 2017). This method provides smooth reference trajectories to be tracked by the UAV. This is particularly important when the target trajectory is very noisy (e.g., when following a real target) or defined by waypoints (e.g., as a virtual RT path). At the same time, by generating a trailer trajectory, a reference frame tangent to the path is obtained, which can be directly used to define the relative displacements encoded in the shooting parameters of each shot type, as well as the desired heading for the UAV. For example, a chase shot would have the following parameters relative to this trailer reference frame; constant altitude z_0, and starting and ending distances to the target on the X axis x_s and x_e.

Having the desired trajectory and an estimation of the current UAV state, the errors between the current and the desired position and yaw angle are used to generate velocity commands, applying a simple saturated proportional controller with

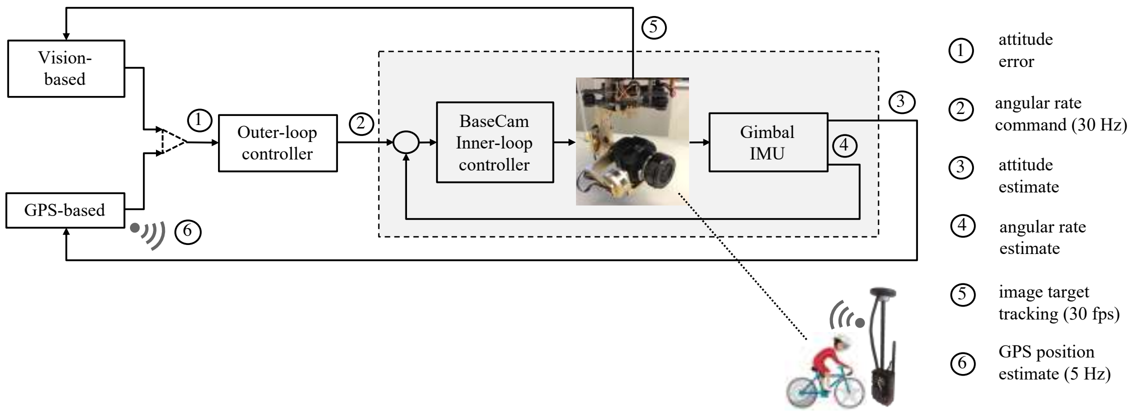

Vision-based
GPS-based
1
Outer-loop controller
2
BaseCam Inner-loop controller
Gimbal IMU
3
4
5
6
1 attitude error
2 angular rate command (30 Hz)
3 attitude estimate
4 angular rate estimate
5 image target tracking (30 fps)
6 GPS position estimate (5 Hz)

Figure 3.5: Flow diagram for gimbal control. Vision-based or GPS-based measurements can be used to feed a attitude controller.

a feedforward velocity term. These velocity commands are sent to the UAV autopilot by means of our software library UAL (UAV Abstraction Layer) (Real et al., 2020). This is a middleware abstraction layer that we developed to abstract UAV navigation algorithms from the specific hardware details and interfaces of each autopilot. Thus, UAL provides a common interface to receive UAV state, including positioning and battery level information. It also allows us to send velocity and position commands, as well as take-off and landing maneuvers. The Shot Executor performs Navigation Actions using our UAL interface directly, as no smooth trajectories are required for those actions.

Apart from UAV control, the Shot Executor is also in charge of controlling the gimbal to point the camera to the specified ST. If the ST type is "none", the controller executes a predefined pan and tilt movement; if the ST type is real or virtual, the controller tracks that target. Figure 3.5 depicts the flow of the gimbal control system. Overall, the gimbal control system is based on an inner–outer loop architecture. A low-level inner-loop controller from BaseCam Electronics is configured to receive angular rate reference commands at a frequency of 30 Hz. This low-level controller relies on an IMU directly attached to the gimbal to obtain estimates of the gimbal orientation, not with respect to the UAV but with respect to an inertial frame. The outer-loop controller, which may be either vision-based or GPS-based, interacts with the inner-loop controller at a frequency of 30 Hz. For the vision-based controller, no information about the 3D target position is required, and the commands are computed directly from 2D target positions on the image and the angular deviation from the horizontal plane, provided by the orientation estimate. The video bandwidth and range to and from the Ground Station did not turn out to be critical issues, because the 2D detection and tracking of the target on the image plane is performed onboard, achieving a rate of 30 fps. For the GPS-based controller, an RTK GPS was used, which provides position updates with a frequency of 5 Hz. As expected, some degradation in tracking performance was observed, due not only to the reduced frequency and reduced accuracy of the position measurements but also to the presence of bias in the orientation estimates. To obtain position estimates at a higher rate and provide some degree of anticipation, the GPS position estimates were fed into a Kalman filter, under the assumption that the UAV motion approximates a constant velocity trajectory in the most immediate time horizon. More specific details about this mathematical formulation can be seen in Cunha et al. (2019). In addition to UAV and gimbal control, we also implemented some camera commands in the Shot Executor. In particular, the component is able to start and stop recording, autofocus, and modify some camera parameters, such as zoom, ISO and white balance.

The Shot Executor requires an estimation of the target position that is provided by the Target Tracker component, which also runs on board the UAV. This target positioning is needed whenever the UAV is tracking an actual target, whether RT or ST. We implemented the two aforementioned options for the Target Tracker in our system: (i) we used a GPS receiver on board the target together with a Kalman filter to estimate 3D target positions that were then sent to the UAVs; and (ii) we used a vision-based algorithm for target tracking that provided 2D positions on the image. The methods we applied for visual target tracking are based on light convolutional neural networks that can run on embedded computers in real time. This image processing part is out of the scope of this thesis and further details can be seen in Nousi et al. (2019); Nousi et al. (2020). In summary, the authors present performance results in well-known benchmarks to demonstrate that the proposed visual tracker is particularly suitable for long-term tracking scenarios, as its success lies behind the ability to efficiently handle severe occlusions, viewpoint changes and scale variations.

Additionally, the Shot Executor needs to take care of collision avoidance for safety reasons. For this, we used a reactive algorithm for collision avoidance in multi-UAV teams (Ferrera et al., 2018). The algorithm resolves UAV conflicts (i.e., possible collisions) with other teammates or external obstacles in a decentralized manner, applying roundabout maneuvers so that they will avoid each other. We integrated this algorithm with our Shot Executor by running it as a reactive layer in parallel. This reactive layer sends warnings to the Shot Executor whenever a conflict is detected, together with velocity commands to resolve the conflict. Thus, the Shot Executor always prioritizes commands coming from the reactive layer over shot execution, in order to avoid collisions. Once the conflict warning disappears, shot execution resumes normally.

Finally, it is important to highlight that our architecture is generic and allows cinematographers to integrate alternative control components, as long as they address UAV and gimbal control and implement the RT and ST concepts that we defined. Along this line, we also tested our system architecture, integrating an algorithm within the Shot Executor that we developed for optimal trajectory planning with multiple UAVs performing cinematography shots. This algorithm, which is described in Chapter 4, plans optimal trajectories to execute multi-UAV shots, dealing with UAV dynamics and collision avoidance constraints, as well as cinematography aspects such as trajectory smoothness and avoidance of mutual visibility between UAVs. The method was tested with this architecture, and was able to compute optimal trajectories on board the UAVs in real time. All the details of the algorithm and its related experiments are shown in Chapter 4.

3.5 Field experiments

We conducted extensive field tests to assess the performance of our complete system filming different outdoor activities. Since the system was developed for the MultiDrone project, our focus was on the sports use cases selected in the project; i.e., cycling/rowing races and parkour runners. The whole consortium devoted much effort to integrating all software components into the team of aerial platforms developed in the project. In particular, we dedicated 9 weeks to physical integration throughout the project's lastyear, as well as 4 weeks for field tests with more than 40 hours of flight, split into two different campaigns in Germany and Spain. We set up several mock-up scenarios to recreate the aforementioned activities with amateur sportsmen, and we even filmed a real regatta event. In Germany, we used a field facility around a farm and next to a lake. The place is located in a village called Bothkamp in the north of Germany, and it has permits to fly UAVs for amateur purposes. In Spain, we used another outdoor site on a farm 30 km from Seville.

3.5.1 System integration

We used a simulation environment for early integration, and also so that the media director could double-check all missions before the actual shooting. Our simulation tool was based on Gazebo (Koenig and Howard, 2004) and the PX4 SITL (Software In The Loop) functionality [2] for UAV autopilots. We added a camera on a gimbal to the UAVs and interfaced them with our open-source [3] UAL library (Real et al., 2020), which abstracts users from the protocol details of each autopilot.

The same software architecture ran in simulation and on our real UAV platforms. We developed all software components as open-source in C++ [4] using ROS Kinetic. We also mounted and integrated several UAVs like the one shown in Figure 3.6 for the experiments. They had the X6 frame from Tarot and were equipped with a PixHawk 2 autopilot running PX4 for flight control, a RTK-GPS for precise localization, a 3-axis gimbal controlled by a BaseCam (AlexMos) controller receiving angle rate commands, a Blackmagic Micro Cinema camera, an Intel NUC i7 computer to run our software for UAV execution, an NVIDIA TX2 computer dedicated to video streaming and image processing for target tracking, and a Thales LTE module to communicate with the

[2]`https://github.com/PX4/sitl_gazebo`
[3]`https://github.com/grvcTeam/grvc-ual`
[4]`https://github.com/grvcTeam/multidrone_planning`

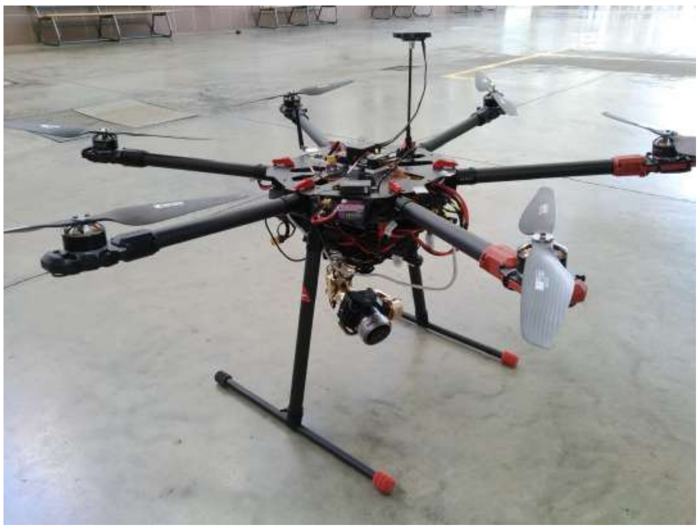


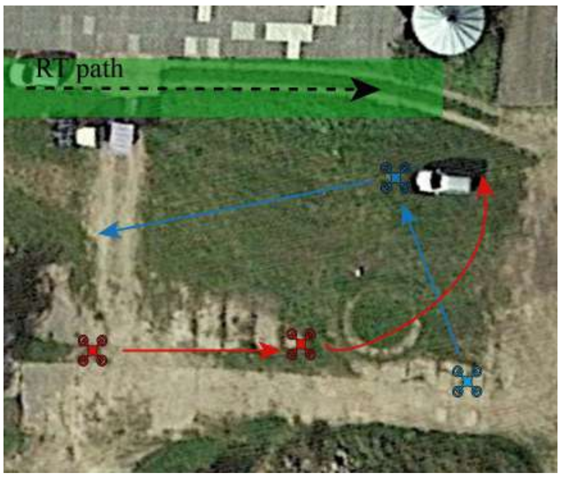
Figure 3.6: One of the UAVs used during our field experiments, with the cinematographic camera mounted on the gimbal.

Ground Station. We selected LTE to achieve better security and performance than WiFi in long-range distances.

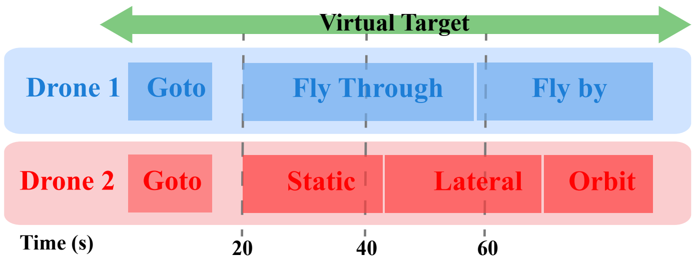
We devised a GPS target to be carried by selected human actors in some of the experiments. The device weighed around 400 grams and consisted of an RTK-GPS receiver with a Pixhawk controller, a radio link and a small battery. This target transmitted target 3D measurements to the Target Tracker on board the UAVs in real time (with a delay below 100 ms). The final 3D target estimation, after being filtered by the Target Tracker, was able to achieve centimeter level. These errors were compensated by our gimbal controller for tracking shooting targets on the video.

3.5.2 Results

In this section, we show example cinematography missions that followed the whole procedure through our architecture for autonomous filming. They were designed by a media expert with the Dashboard facility, then planned and executed autonomously by the UAVs. Our main objectives are to demonstrate: (i) the integration of all the components working together; (ii) the feasibility of our system for autonomous



(a) Top view of the scene with the virtual target and the UAV trajectories for each Shooting Action. In green, the parkour area.

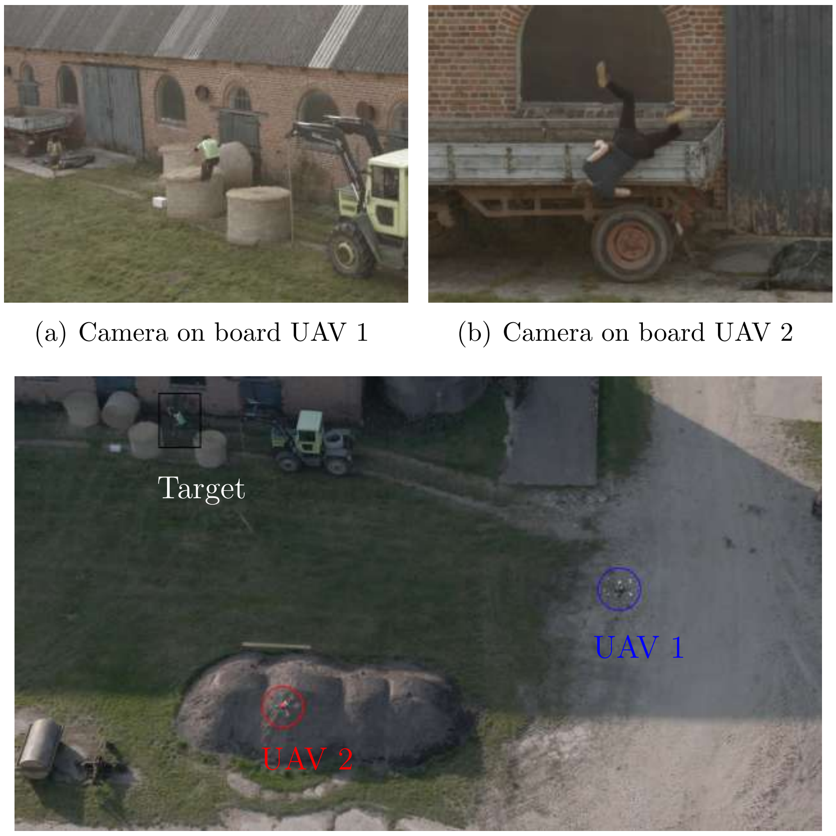


(b) UAV actions over a timeline. Both sequences of consecutive shots are executed in parallel triggered by the `START_RACE` Event at time 20 s. A virtual target is tracked.

Figure 3.7: Parkour mission with two UAVs and five different shots. Blue indicates UAV 1 and red UAV 2.

cinematography with multiple UAVs outdoors; and (iii) the use of different shot types and RT/ST modes.

First, we illustrate parkour filming. Parkour is a sport activity where runners move freely over and through any terrain using only the abilities of their bodies, principally through running, jumping and climbing. In our mock-up, we set up a specific longitudinal course with a variety of obstacles and convened a group of amateur parkourists to perform free-style maneuvers there. Figure 3.7(a) depicts the scheme of a mission designed by our media director. Runners moved in the *parkour zone*



(a) Camera on board UAV 1 (b) Camera on board UAV 2

(c) Top view of the experiment. Both UAVs and one of the runners can be seen.

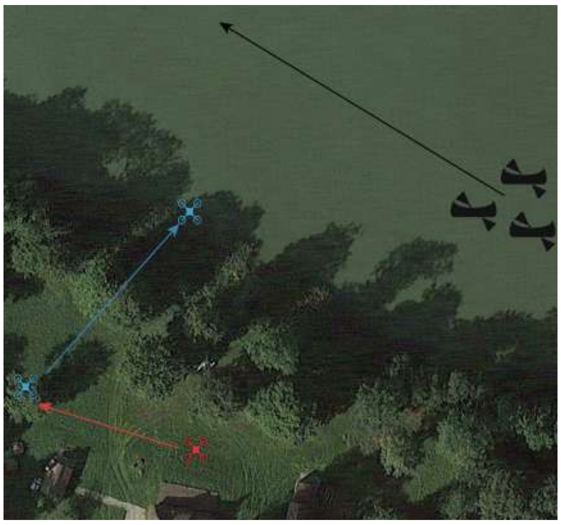
Figure 3.8: Images from a mission with two UAVs filming a parkour activity.

from left to right and the director designed a mission with 5 different shots. First, a sequence of a fly-through shot followed by a flyby, triggered by the `START_RACE` Event. Second, a sequence of a static shot, a lateral and an orbit, also triggered by the same `START_RACE` Event. Since the runners were moving freely in the scene, instead of tracking a particular one, the virtual-traj RT mode was used to specify a virtual trajectory for the RT in the parkour area. This RT path was used by the lateral, the flyby and the orbital shot, while the others had their own RT paths independent of the runners. The ST was “none” for the static and fly-through shots, and configured as virtual for the rest.

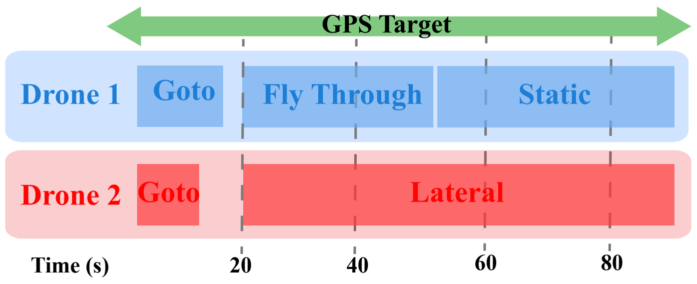
This mission was run with two UAVs and the Planner assigned one of the sequences to each UAV. Figure 3.7(b) depicts a timeline of the schedules for the two UAVs. The UAVs navigate to the starting positions of their first Shooting Actions and wait for the `START_RACE` Event, which triggers both shooting sequences in parallel (only the first Shooting Action of each UAV has a starting Event associated with it; the rest are consecutive). UAV 1 approaches the action scene with a fly-through shot, then performs a flyby in the opposite direction to the runners' movement. UAV 2 starts with a static shot taking an overview of the parkour area. Then it does a lateral shot along the scene, which coincides with the runners coming across a complex obstacle. It finishes with a quarter of an orbit around the final part of the scene. Some images from the experiment can be seen in Figure 3.8; a complete video is accessible at: `https://youtu.be/P_n_PfuEC2A`.

We also demonstrate our system with a mission filming a rowing race. We prepared a mock-up in a lake with four amateur rowing boats recreating a race. Figure 3.9(a) shows a scheme of the mission designed by the director. It consists of three shots to film the rowers from the lake shore as they pass by. There is a sequence with a fly-through shot followed by a static shot, and a lateral shot running in parallel; both tracks triggered by the `START_RACE` Event. One of the boats carried a GPS target, which was used as both RT and ST for the lateral shot. The fly-through and static shots had a ST of type "none". The area with trees on the lakeshore was manually set as a predefined no-fly zone, so that the Planner would not send the UAVs into the trees. Only a narrow corridor without trees was left out of the no-fly zone, so that the UAVs could fly through it to reach the lake.

This mission was run with two UAVs, and the Planner assigned the lateral shot to one UAV and the fly-through and the static shots to the other. Figure 3.9(b) shows a timeline of the schedules for the two UAVs. The UAVs navigate to the starting positions of their first Shooting Actions and wait for the `START_RACE` Event, which triggers both shooting sequences in parallel (only the first Shooting Action of UAV 1 has an associated starting Event; the following one happens consecutively). UAV 1 approaches the rowers, taking a fly-through shot from the lakeshore out over the water. Then it takes a static shot rising up 10 meters and panning to the left



(a) Top view of the scene with the RT and the UAV trajectories for each Shooting Action.



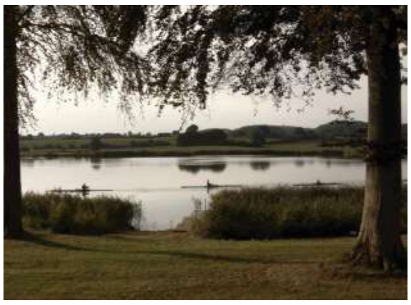
(b) UAV actions over a timeline. The two sequences of consecutive shots are executed in parallel triggered by the `START_RACE` Event at time 20 s. A GPS target is tracked.

Figure 3.9: Rowing race mission with two UAVs and three different shots. Blue indicates UAV 1 and red UAV 2.

to target the boats. UAV 2 performs a lateral shot over a green area beside the lakeshore, tracking the boats at a 50-meter distance and a 3-meter height. Some images from the experiment can be seen in Figure 3.10; a complete video is accessible at: `https://youtu.be/COay0hZsMzk`.

We also ran multiple missions in simulation to test further some functionalities of our system; for instance, re-planning capabilities. As an illustration, we showcase in Figure 3.11 a simulated mission where we simulated a battery alarm in one of the UAVs. The mission starts with an initial plan for 3 UAVs, but UAV 1 runs out

(a) Camera on board UAV 1 during the static shot.


(b) Camera on board UAV 2 during the lateral shot.

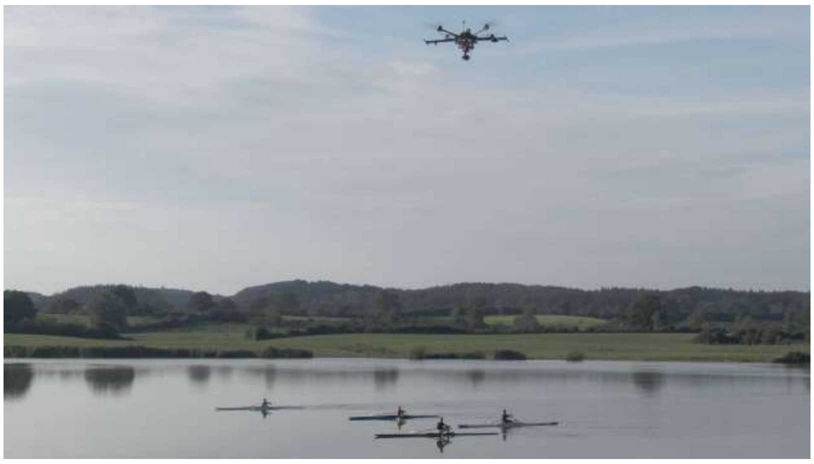

(c) View of UAV 1 taking the static shot as the rowers pass by.

Figure 3.10: Images from a mission with two UAVs filming a rowing race.

of battery during the mission execution. UAV 1 is then commanded to land and a re-planning procedure is triggered for the remaining UAVs. In this case, UAV 2 partially covers the lateral shot missed by UAV 1.

Finally, we demonstrated the system in a real regatta event in Wannsee in Berlin (Germany). We deployed the system in a strategic spot prior to actual race, earlier in the day. Then two of our UAVs waited for a manually triggered Event to run a short mission designed beforehand, consisting of static and fly-through shots in parallel, followed by a flyby. The two first shots used ST of type "none", but we employed a visual ST for the last shot to track the boats as they passed by. Visual tracking worked at a frame rate of 30 fps, which enabled real-time tracking achieving

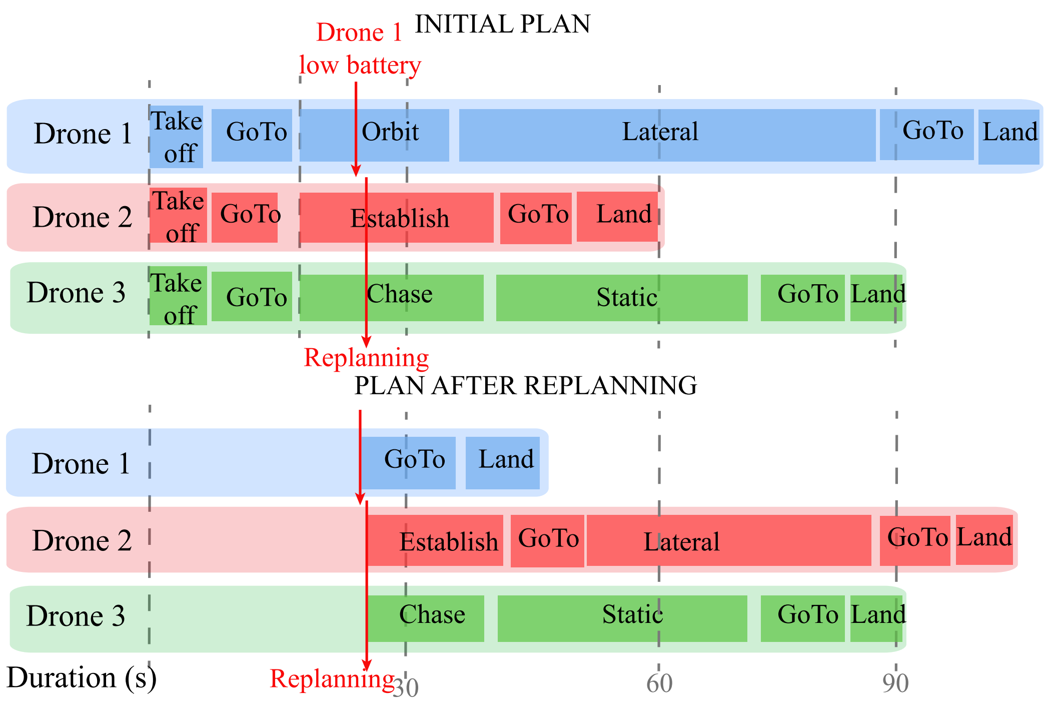


Figure 3.11: Timeline of a simulated mission where re-planning is carried out. The UAVs were executing their initial plans when the battery alarm of UAV 1 went off. A new plan was computed assigning the missing shot from UAV 1 to others. UAV 2 covered part of the lateral shot initially assigned to UAV 1.

a 96% true positives. The main objective was just to showcase to media end-users the possibilities of our system and its fast deployment for covering a real sporting event. A feature video with the main results of our field campaigns can be seen at `https://www.youtube.com/watch?v=iLs6Xo87j78`.

3.5.3 System usability evaluation

Within the framework of the MultiDrone project, we carried out a study based on questionnaires to assess the usability of our system. The evaluation method consisted of one-to-one interviews with various members of the MultiDrone media production crew. During the interviews, the users were asked to rate their agreement or disagreement with a set of statements describing the performance of different functionalities of the system, on a scale of 1 (fully agree) to 4 (totally disagree). The full questionnaire and evaluation study can be found in MultiDrone-Consortium (2019). For the sake of brevity, we do not include all details here but give a summary of the most important conclusions:

- The Planner component was clearly regarded as useful for planning cinematography missions in pre-production.
- The performance of GPS-based target tracking was mostly evaluated positively. However, the users were less convinced that the smoothness of the camera movements and the video quality were sufficient for media production.
- Those interviewed agreed that the UAV flights were stable enough and followed the patterns defined by the director without much deviation.
- Emergency handling was also assessed positively, the system being regarded as safe.
- Most of the interviewees confirmed that the system and its main functionalities were easy to understand and to use. Even though some users pointed out that current methods and current tools (normal camera equipment or consumer UAVs) are still at least as efficient, they also highlighted that a finished and fully functional multi-UAV system would make video production much easier and interesting.

The evaluation, although altogether positive, showed a certain discrepancy regarding the robotics-related and media production-related requirements. On the one hand, the functionalities of the UAV itself such as tracking, autonomous flight and emergency management were mostly highly rated. On the other hand, functionalities that are crucial for high quality media production performed slightly less satisfactorily. We believe that these issues are related to the fact that we tested a system at prototype level to showcase its potential, but more professional or consumer equipment should improve the overall performance in terms of media production quality.

3.5.4 Lessons learned

In this section, we discuss the main lessons that we learnt during our field and integration tests about our system and autonomous cinematography with UAVs in general.

One UAV to rule them all: Due to the ambition of the application, the hardware design of the aerial platform was complex. On the one hand, the UAVs were conceived to fulfill safety and usability concerns. This entails the integration of heavy payload, including a high-performance cinematographic camera, several processing units, an LTE module, and enough batteries to cover a reasonable flight time (20 minutes). On the other hand, more "commercial" products are usually aimed at smaller platforms, mainly due to logistic and cost constraints. From feedback by media end-users, we learned that our UAVs were appropriate to test and demonstrate system functionalities, but a final product should trade off capabilities with payload and size, in order to be more secure and practical for media production.

Camera and gimbal integration: The selection of the camera was quite important for the system. A high-performance camera was a requirement from the media end-users, and our choice fulfilled all the media specifications. However, we discovered throughout our experimentation that this kind of camera is not designed to be integrated in autonomous platforms. First, gimbal calibration was difficult, as off-the-shelf gimbals are designed for lighter cameras. A custom product with an integrated camera would havebeen more appropriate to obtain steadier images. Second, we experienced many issues with drivers for video streaming on the NVIDIA TX2, as the selected carrier board (AUVIDEA) did not have official drivers for HDMI input with TX2. Lastly, we found problems in focusing the camera remotely, which was another requirement from the end-users. The camera offered an expansion port to send commands to be configured, but this port did not send back any feedback from the camera about its properties (e.g., focus, ISO, white balance, etc). Therefore, it was difficult to implement specific controllers, so we opted for interfacing with the built-in autofocus of the camera, which was not perfect when flying far from the target. In general, all these details of integrating commercial cameras and gimbals for high-performance media production on UAVs are not negligible, and they should be considered carefully when designing a final system.

Middleware and communication: We found our choice for ROS and UAL as system middleware quite helpful, as they offered us a good solution to enable abstraction from low-level UAV control and communication, speeding up software development. UAL also allowed us to design the system transparently, regardless of the final selection of the UAV autopilot. In terms of communication, the LTE module provided high-quality video transmission and multi-UAV communication, which was critical for the application. However, the ROS configuration (we used the *multimaster-fkie* package) to operate with multiple UAVs in a distributed fashion was troublesome. We believe that the establishment of ROS 2 will be key for multi-robot applications, as its communication is decentralized and professional middleware can be easily integrated.

Simulation is key in cinematographic applications: We used SITL simulations in Gazebo to integrate and test our system, which was tremendously useful for speeding up the development process. Nonetheless, simulation turned out to be a helpful tool for media production as well. The media director always found it interesting to see a 3D recreation of the mission before the real scene happened. For security, we also used the simulator to graphically show the safety pilots the behavior of the UAVs before every field test. Even though Gazebo was enough for our purposes, the use of simulators with more realistic graphics engines such as AirSim [5] would be more appealing for media users, enhancing their experience.

Media end-users' need for alternative types of targets: We followed media users' recommendations to implement shots based on both actual and virtual targets. Our way of describing shots by means of a Reference and a Shooting Target, and our different RT modes were a success, as they provided the director with the required level of flexibility. Being able to define virtual rails for camera motion independent of the actual target is highly desirable for media directors. We also learnt that, although they appreciate autonomous functionalities, they also feel the need to have the possibility of operating the gimbal and the focus manually, in order to adapt the shots to their

[5]`https://github.com/microsoft/AirSim`

artistic wishes at any moment. Regarding shooting targets, we discovered that a combination of GPS and image processing was the best solution. GPS targets can be detected at a longer range but visual tracking performs better for closer frames, as it runs on board the UAVs with local images, alleviating latency issues. GPS was noisy in locations with adverse conditions, such as nearby high trees or buildings; and a poor radio link connection for sending measurements could cause latency and low-rate measurements. However, note that this radio link connection could be replaced by a higher-performance LTE connection. Thus, a wise trade-off was to use GPS to initially locate the target on the image, and then visual processing to track it more precisely.

Onboard collision avoidance is a must: Even if the system is to be operated in open and well-structured environments, autonomous collision avoidance is quite important, as it provides reassurance to end-users. Thus, the sooner conflicts are detected, the better. This means that planning components minimizing hypothetical conflicts for the UAVs are desirable. Nonetheless, onboard mechanisms for reactive collision avoidance during the mission execution are also necessary to cover dynamic scenes, as there are always unexpected obstacles and inaccuracies in the plan execution, mainly in outdoor events. Moreover, relying only on a safety pilot is sometimes insufficient as their perspective with respect to the UAV is not always ideal.

3.6 Conclusions

This chapter presented a system for autonomous execution of cinematography missions with multiple UAVs. We introduced the complete architecture, including components for mission design, planning and execution. We then focused on the system for mission execution. In particular, we described our parametric way of defining shots, which includes different types of camera motion and target actors in the scene. In addition, we implemented a series of canonical shots and proposed a distributed scheduling procedure to execute cinematography missions, which can include sequential and concurrent shots, as well as single- and multi-camera shots. An event-based mechanism is used to synchronize shot execution and to increase the system robustness against potential inaccuracies during the planning phase.

The system was developed within the framework of the EU-funded MultiDrone project and it has been released as open-source for the community. Our field experiments filming sport activities showcased the feasibility of the system to address outdoor cinematography missions involving multiple UAVs and a variety of shot types. In general, feedback from the media experts in the project was positive, as they found the combination of virtual and actual targets to guide camera motion helpful, as well as the flexibility that our concepts of Reference and Shooting Targets provided.

In future work, more specific subjective user studies could be carried out to better evaluate the artistic possibilities of the system combining multi-camera shots. Moreover, although we integrated solutions for conflict resolution in the planning components and also for reactive collision avoidance between UAVs during the mission execution, we would like to explore mechanisms more oriented to obstacle avoidance in unstructured environments, using onboard sensors for online mapping.

Chapter 4

Optimal trajectory planning for cinematography with multi-UAV coordination

This chapter proposes a method for online trajectory planning with a team of UAVs taking cinematography shots. While Chapter 3 focused on a general architecture for multi-UAV cinematography missions by means of a distributed scheduler that activates different shot controllers depending on the shot type, this chapter presents an algorithm that takes care of the control of the UAV and gimbal motion, implementing an optimization-based trajectory planner that runs on the UAVs in a distributed fashion. Our method is intended to provide smooth trajectories for visually pleasing video output, integrating cinematographic constraints imposed by the shot types, physical limits of the gimbal, mutual visibility between cameras, and avoidance of collisions. This multi-UAV trajectory planner has been integrated within the general framework presented in Chapter 3, and it has been demonstrated in simulated and field experiments.

4.1 Introduction

The use of UAVs for aerial photography and cinematography is increasing substantially. From the application point of view, UAVs present a remarkable potential to produce unique aerial shots at reduced costs, in contrast with other alternatives such as dollies or static cameras. However, UAV trajectory planning for aerial cinematography is still challenging, as multiple aspects need to be considered: flying smooth trajectories to achieve aesthetic videos, tracking actors to be filmed, avoiding collisions with potential obstacles, and keeping other cameras out of the field of view, among others.

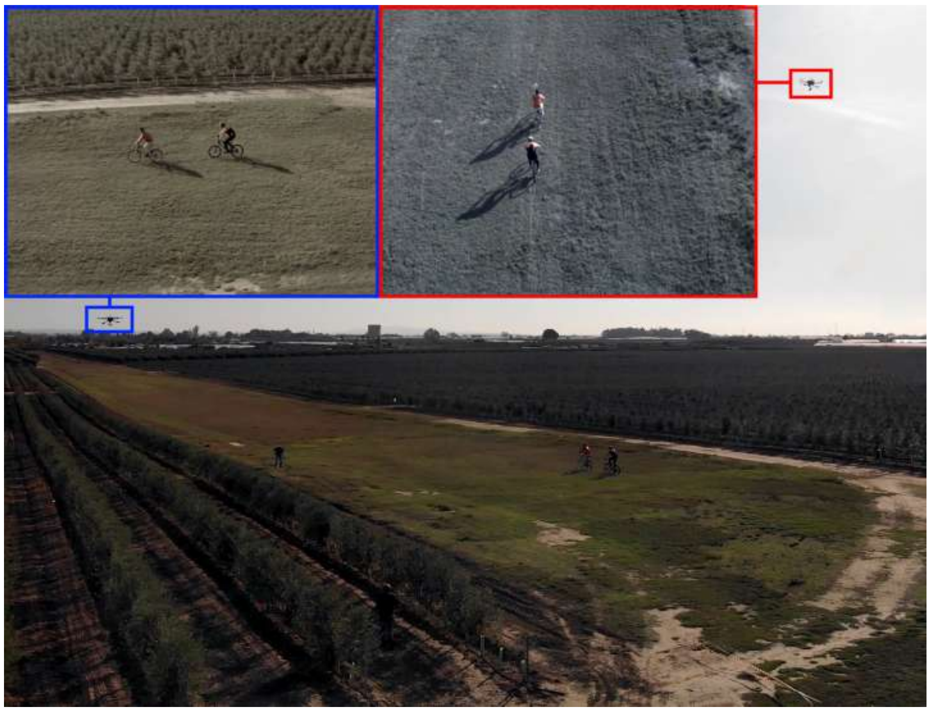


Figure 4.1: Cinematography application with two UAVs filming a cycling event. Bottom, aerial view of the experiment with two moving cyclists. Top, images taken from the cameras on board each UAV.

We propose a novel method to plan optimal online trajectories for a set of UAVs executing cinematography shots. The optimization is performed in a distributed manner, and it seeks to produce smooth trajectories that comply with dynamic and cinematographic constraints. Moreover, we can handle multiple UAVs, integrating new constraints for inter-UAV collisions and mutual visibility. We present results to evaluate the method with different types of shots and we demonstrate the system in field experiments with multiple UAVs filming dynamic scenes. The main novelty of our method is the multi-UAV coordination to combine the execution of several types of shots simultaneously in outdoor scenarios (see Figure 4.1), with the specific challenges that those environments involve. More particularly, our main contributions are the following:

- We propose a novel formulation of the trajectory planning problem for UAV cinematography. We model both UAV and gimbal motion (Section 4.3) but decouple their control actions.
- We propose a non-linear, optimization-based method for trajectory planning (Section 4.4). Using a receding horizon scheme, trajectories are planned and executed in a distributed manner by a team of UAVs providing multiple views of the same scene. The method takes account of dynamic UAV constraints, and imposes them to avoid predefined no-fly zones and collisions with others. Cinematographic aspects imposed by shot definition, camera mutual visibility and gimbal physical bounds are also addressed. Trajectories smoothing UAV and gimbal motion are generated to achieve aesthetic video footage.
- We describe the complete system architecture on board each UAV and the different types of shots that are possible (Section 4.5). The architecture integrates target tracking with trajectory planning and allows different UAVs to execute different types of shots simultaneously.
- We present extensive experimental results (Section 4.6) to evaluate the performance of our method for different types of shots. We prove that our method is able to compute smooth trajectories, reducing jerky movements in real time and complying with cinematographic restrictions. We then demonstrate our system in field experiments with three UAVs planning trajectories online to film a moving actor (Section 4.7).

4.2 Related work

There are several studies related to trajectory planning for virtual camera motion in the computer animation community (Christie et al., 2008). They typically use offline optimization to generate smooth trajectories that are visually pleasing and comply with certain cinematographic aspects, such as the *rule of thirds*. However, many of them do not ensure physical feasibility to comply with dynamic UAV constraints and they assume a full knowledge of the environment map. In terms of optimization functions, several studies consider similar terms to achieve smoothness. For instance, the authors in Joubert et al. (2015) 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 Joubert et al. (2016). Timed reference trajectories are generated from 3D positions specified by the user, and the final timing of the shots is addressed by designing easing curves that drive the UAV along the planned trajectory (i.e., curves that modify the UAV velocity profile). In Gebhardt et al. (2016), aesthetically pleasing 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 (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, such as planning in a *toric space* (Lino and Christie, 2015) or interpolating polynomial curves (Galvane et al., 2016; Joubert et al., 2016). In general, these papers 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 scenesaltitude, without moving actors.

In the robotics literature, there are studies focusing more on filming dynamic scenes and complying with physical UAV constraints. The authors in Huang et al.

(2018), for instance, propose to detect limb movement of a human for outdoor filming. Trajectory planning is carried out online with polynomial curves that minimize the snap. In Bonatti et al. (2019); Bonatti et al. (2020), 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 by minimizing the trajectory jerk, and shot quality by defining objective curves fulfilling cinematographic constraints associated with relative angles with respect to the actor and shot scale. Optimal cinematography trajectories have also been computed in real time through receding horizons with non-linear constraints (Nägeli et al., 2017a). 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 in terms of online numerical optimization but it is only tested in indoor settings.

Some of the aforementioned robotics authors have also looked at UAV cinematography, applying machine learning techniques. In particular, learning from demonstrations to imitate professional camera operators' behaviors (Huang et al., 2019) or reinforcement learning to achieve visually pleasing shots (Gschwindt et al., 2019). In general, most of these cited works on robotics present quite interesting results in terms of outdoor operation or online trajectory planning, but they are always restricted to a single UAV.

Regarding methods for multiple UAVs, there is some related work which is worth mentioning. In Nägeli et al. (2017b), 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 toric UAV space is proposed in Galvane et al. (2018) to ensure that cinematographic properties and dynamic constraints are maintained 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

References	**Online**	**Scene**	**UAV Dynamics**	**Collision Avoidance**	**Mutual Visibility**	**Outdoors**	**Multiple UAVs**
Lino and Christie (2015)	No	Static	No	No	No	No	No
Joubert et al. (2015)	No	Static	Yes	No	No	Yes	No
Gebhardt et al. (2016)	No	Static	Yes	Yes	No	No	No
Joubert et al. (2016)	No	Static	Yes	Actor	No	Yes	No
Galvane et al. (2016)	No	Dynamic	No	No	No	No	No
Nägeli et al. (2017a)	Yes	Dynamic	Yes	Yes	No	No	No
Nägeli et al. (2017b)	Yes	Dynamic	Yes	Actor	Yes	No	Yes
Huang et al. (2019)	Yes	Dynamic	Yes	Actor	No	Yes	No
Galvane et al. (2018)	Yes	Dynamic	Yes	Yes	Yes	No	Yes
Bonatti et al. (2019)	Yes	Dynamic	Yes	Yes	No	Yes	No
Bonatti et al. (2020)	Yes	Dynamic	Yes	Yes	No	Yes	No
Bucker et al. (2021)	Yes	Dynamic	Yes	Yes	Yes	Yes	Yes
Ours	Yes	Dynamic	Yes	Yes	Yes	Yes	Yes

Table 4.1: Related works on trajectory planning for UAV cinematography. We indicate whether computation is online or not, the type of scene and constraints they consider, and their capacity to handle outdoor applications and multiple UAVs.

dynamic targets is coordinated by means of a centralized master–slave approach to solve conflicts. These studies present quite valuable contributions for cinematography with multiple UAVs, but they are evaluated in indoor settings where a *Vicon* motion capture system provides accurate positioning for all targets and UAVs. The specifics of the outdoor scenarios considered in our work are different in several aspects, as the environment is less controlled: among other factors, 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. Bucker et al. (2021) presented a work closer to ours, dealing with outdoor aerial cinematography with multiple UAVs, focusing more on the execution of shots in unstructured cluttered environments.

Table 4.1 summarizes the main related works on trajectory planning for UAV cinematography and their corresponding properties. We indicate whether computation is online or offline, whether the scene contains dynamic targets to be filmed and whether UAV dynamics are included as constraints. We also analyze the type of collision avoidance: none ("No"), with the actor being filmed ("Actor"), or with external obstacles and other UAVs ("Yes"). Studies that address mutual visibility constraints between multiple cameras are mentioned specifically. Finally, we indicate

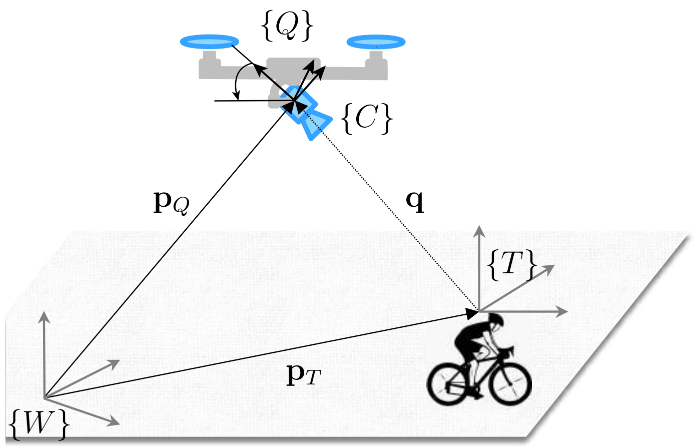


Figure 4.2: Definition of the reference frames used. The origins of the camera and multirotor frames coincide. The camera points to the target.

whether each method includes evaluation in outdoor settings and whether it can handle multiple UAVs.

4.3 Dynamic models

This section presents our dynamic models for UAV cinematographers. We model the UAV as a multirotor with a camera mounted on a gimbal with two degrees of freedom.

4.3.1 UAV model

Let $\{W\}$ denote the world reference frame with origin fixed in the environment and East-North-Up (ENU) orientation. There are three additional reference frames (see Figure 4.2): the multirotor reference frame $\{Q\}$ attached to the UAV with origin at the center of mass, the camera reference frame $\{C\}$ with z-axis aligned with the optical axis but with opposite sign, and the target reference frame $\{T\}$ attached to the moving target that is being filmed. For simplicity, we assumed that the origins of $\{Q\}$ and $\{C\}$ coincide.

The configuration of $\{Q\}$ with respect to $\{W\}$ is denoted by $(\mathbf{p}_Q, \mathbf{R}_Q) \in \mathbb{SE}(3)$, where $\mathbf{p}_Q \in \mathbb{R}^3$ is the position of the origin of $\{Q\}$ expressed in $\{W\}$ and $\mathbf{R}_Q \in \mathbb{SO}(3)$

is the rotation matrix from $\{Q\}$ to $\{W\}$. Similarly, the configurations of $\{T\}$ and $\{C\}$ with respect to $\{W\}$ are denoted by $(\mathbf{p}_T, \mathbf{R}_T) \in \mathbb{SE}(3)$ and $(\mathbf{p}_C, \mathbf{R}_C) \in \mathbb{SE}(3)$, respectively.

We model the multirotor dynamics as a linear double integrator model:

$$\begin{aligned} \dot{\mathbf{p}}_Q &= \mathbf{v}_Q \\ \dot{\mathbf{v}}_Q &= \mathbf{a}_Q, \end{aligned} \tag{4.1}$$

where $\mathbf{v}_Q = [v_x \ v_y \ v_z]^T \in \mathbb{R}^3$ is the linear velocity and $\mathbf{a}_Q = [a_x \ a_y \ a_z]^T \in \mathbb{R}^3$ is the linear acceleration. We assume that the linear acceleration $\mathbf{a}_Q$ takes the form

$$\mathbf{a}_Q = -g\mathbf{e}_3 + \mathbf{R}_Q \frac{T}{m} \mathbf{e}_3, \tag{4.2}$$

where m is the multirotor mass, g the gravitational acceleration, $T \in \mathbb{R}$ the scalar thrust, and $\mathbf{e}_3 = [0 \ 0 \ 1]^T$.

For the sake of simplicity, we use the 3D acceleration $\mathbf{a}_Q$ as control input, although the thrust T and rotation matrix $\mathbf{R}_Q$ could also be recovered from 3D velocities and accelerations. For this, we first parameterize $\mathbf{R}_Q$ by the Z-Y-X Euler angles $\boldsymbol{\lambda}_Q = [\phi_Q, \theta_Q, \psi_Q]^T$, such that

$$\mathbf{R}_Q = \mathbf{R}_z(\psi_Q)\mathbf{R}_y(\theta_Q)\mathbf{R}_x(\phi_Q). \tag{4.3}$$

If we restrict the yaw angle ψ_Q to keep the multirotor's front pointing forward in the direction of motion such that

$$\psi_Q = \operatorname{atan2}(v_y, v_x), \tag{4.4}$$

then the thrust T and the Z-Y-X Euler angles $\boldsymbol{\lambda}_Q = [\phi_Q, \theta_Q, \psi_Q]^T$ can be obtained from $\mathbf{v}_Q$ and $\mathbf{a}_Q$ according to:

$$\begin{cases} T = m\|\mathbf{a}_Q + g\mathbf{e}_3\| \\ \psi_Q = \text{atan2}(v_y, v_x) \\ \phi_Q = -\arcsin((a_y \cos(\psi_Q) - a_x \sin(\psi_Q))/\|\mathbf{a}_Q + g\mathbf{e}_3\|) \\ \theta_Q = \text{atan2}(a_x \cos(\psi_Q) + a_y \sin(\psi_Q), a_z + g) \end{cases} \tag{4.5}$$

4.3.2 Gimbal angles

Let $\boldsymbol{\lambda}_C = [\phi_C, \theta_C, \psi_C]^T$ denote the Z-Y-X Euler angles that parametrize the rotation matrix $\mathbf{R}_C$, such that

$$\mathbf{R}_C = \mathbf{R}_z(\psi_C)\mathbf{R}_y(\theta_C)\mathbf{R}_x(\phi_C). \tag{4.6}$$

In our system, we decouple gimbal motion with an independent gimbal attitude controller that ensures that the camera is always pointing towards the target during the shot, as in Bonatti et al. (2020). This reduces the complexity of the planning problem and allows us to control the camera based on local perception feedback if available, accumulating less errors. We also consider that the time-scale separation between the "faster" gimbal dynamics and the "slower" multirotor dynamics is sufficiently large to neglect the gimbal dynamics and assume an exact match between the desired and actual orientations of the gimbal. In order to define $\mathbf{R}_C$, let us introduce the relative position:

$$\mathbf{q} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^T = \mathbf{p}_C - \mathbf{p}_T, \tag{4.7}$$

and assume that the UAV is always higher than the target; i.e., $q_z > 0$, but not directly above the target; i.e., $[q_x \; q_y] \neq 0$. Then the gimbal orientation $\mathbf{R}_C$ that guarantees that the camera is aligned with the horizontal plane and pointing towards the target is given by:

$$\mathbf{R}_C = \left[-\frac{\mathbf{q} \times \mathbf{q} \times \mathbf{e}_3}{\|\mathbf{q} \times \mathbf{q} \times \mathbf{e}_3\|} \quad \frac{\mathbf{q} \times \mathbf{e}_3}{\|\mathbf{q} \times \mathbf{e}_3\|} \quad \frac{\mathbf{q}}{\|\mathbf{q}\|} \right]$$
$$= \begin{bmatrix} * & \frac{q_y}{\sqrt{q_x^2+q_y^2}} & * \\ * & \frac{-q_x}{\sqrt{q_x^2+q_y^2}} & * \\ \frac{\sqrt{q_x^2+q_y^2}}{\sqrt{q_x^2+q_y^2+q_z^2}} & 0 & \frac{q_z}{\sqrt{q_x^2+q_y^2+q_z^2}} \end{bmatrix}. \tag{4.8}$$

To recover the Euler angles from the expression for $\mathbf{R}_C$ in (4.8), note that if the camera is aligned with the horizontal plane, then there is no roll angle; i.e., $\phi_C = 0$, and $\mathbf{R}_C$ takes the form:

$$\mathbf{R}_C = \begin{bmatrix} \cos(\psi_C)\cos(\theta_C) & -\sin(\psi_C) & \cos(\psi_C)\sin(\theta_C) \\ \cos(\theta_C)\sin(\psi_C) & \cos(\psi_C) & \sin(\psi_C)\sin(\theta_C) \\ -\sin(\theta_C) & 0 & \cos(\theta_C) \end{bmatrix}, \tag{4.9}$$

so we obtain:

$$\begin{cases} \phi_C = 0 \\ \theta_C = \operatorname{atan2}(-\sqrt{q_x^2 + q_y^2}, q_z) \\ \psi_C = \operatorname{atan2}(-q_y, -q_x) \end{cases} \tag{4.10}$$

Our cinematography system is designed to perform smooth trajectories as the UAVs are taking their shots, and then to use more aggressive maneuvers only to fly between shots while not filming. If the UAVs fly smoothly, we can assume that their accelerations a_x and a_y are small, and hence, by direct application of (4.5), that their roll and pitch angles are small and $\mathbf{R}_x(\phi_Q) \approx \mathbf{R}_y(\theta_Q) \approx I_3$. This assumption is important to alleviate the non-linearity of the model and achieve real-time numerical optimization. Moreover, it is reasonable during shot execution, as our trajectory planner will explicitly minimize UAV accelerations, and will limit both UAV velocities and accelerations.

Under this assumption, the orientation matrix of the gimbal with respect to the multirotor ${}^{Q}\mathbf{R}_C$ can be approximated by:

$$\begin{aligned} {}^{Q}\mathbf{R}_C &= (\mathbf{R}_Q)^T\mathbf{R}_C \\ &\approx \mathbf{R}_z(\psi_C - \psi_Q)\mathbf{R}_y(\theta_C)\mathbf{R}_x(\phi_C), \end{aligned} \tag{4.11}$$

and the relative Euler angles ${}^{Q}\boldsymbol{\lambda}_C$ (roll, pitch and yaw) of the gimbal with respect to the multirotor are obtained as:

$$\begin{cases} {}^{Q}\phi_C = \phi_C = 0 \\ {}^{Q}\theta_C = \theta_C = \text{atan2}(-\sqrt{q_x^2 + q_y^2}, q_z) \\ {}^{Q}\psi_C = \psi_C - \psi_Q = \text{atan2}(-q_y, -q_x) - \text{atan2}(v_y, v_x) \end{cases} \tag{4.12}$$

According to(4.5), (4.10) and (4.12), $\boldsymbol{\lambda}_Q$, $\boldsymbol{\lambda}_C$ and ${}^{Q}\boldsymbol{\lambda}_C$ are completely defined by the trajectories of the multirotor and the target, as explicit functions of $\mathbf{q}$, $\mathbf{v}_Q$, and $\mathbf{a}_Q$.

4.4 Optimal trajectory planning

In this section, we describe our method for optimal trajectory planning. We explain how the trajectories are computed online in a receding horizon scheme, under dynamic and cinematographic constraints, and then how coordination between multiple UAVs is addressed. We then detail how to execute the trajectories and control the gimbal. Lastly, we include a thorough discussion of some critical aspects of the method.

4.4.1 Trajectory planning

We plan optimal trajectories for a team of n UAVs as they film a moving actor or target whose position can be measured and predicted. The main objective is to come up with trajectories that satisfy physical UAV and gimbal restrictions, avoid collisions, and respect cinematographic concepts. This means that each UAV needs to perform the kind of motion imposed by its shot type (e.g., stay beside/behind the target in a lateral/chase shot) and generate smooth trajectories to minimize jerky movements of the camera and yield pleasing video footage. Each UAV will have a shot type and a desired 3D position ($\mathbf{p}_D$) and velocity ($\mathbf{v}_D$) to be reached. This desired state is determined by the type of shot and may move along with the receding horizon. For instance, in a lateral shot, the desired position ($\mathbf{p}_D$) moves along with the target, to place the UAV beside it; in a flyby shot, this position is such that the UAV reaches a position above the target by the end of the shot. More details about the different types of shot and how to compute the desired position will be given in Section 4.5.

We plan trajectories for each UAV in a distributed manner, assuming that the plans from other neighboring UAVs are communicated (we denote this set of neighboring UAVs as $\mathcal{N}$). To do so, we solve a constrained optimization problem for each UAV where the optimization variables are its discrete state with 3D position and velocity $\mathbf{x}_k = [\mathbf{p}_{Q,k}\ \mathbf{v}_{Q,k}]^T$, and its 3D acceleration as control input ($\mathbf{u}_k = \mathbf{a}_{Q,k}$). A non-linear cost function is minimized for a horizon of N timesteps, using as input at each solving iteration the current observation of the system state $\mathbf{x}'$. In particular, the following non-convex optimization problem is formulated for each UAV:

$$\underset{\substack{\mathbf{x}_0,\ldots,\mathbf{x}_N \\ \mathbf{u}_0,\ldots,\mathbf{u}_N}}{\text{minimize}} \sum_{k=0}^{N} (w_1||\mathbf{u}_k||^2 + w_2 J_\theta + w_3 J_\psi) + w_4 J_N \tag{4.13}$$

$$\begin{aligned}
\text{subject to } & \mathbf{x}_0 = \mathbf{x}' && \text{(4.13.a)}\\
& \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad k = 0, \ldots, N-1 && \text{(4.13.b)}\\
& \mathbf{v}_{\min} \leq \mathbf{v}_{Q,k} \leq \mathbf{v}_{\max} && \text{(4.13.c)}\\
& \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} && \text{(4.13.d)}\\
& \mathbf{p}_{Q,k} \in \mathcal{F} && \text{(4.13.e)}\\
& ||\mathbf{p}_{Q,k} - \mathbf{p}_{O,k}||^2 \geq r_{\text{col}}^2, \quad \forall O && \text{(4.13.f)}\\
& \theta_{\min} \leq^Q \theta_{C,k} \leq \theta_{\max} && \text{(4.13.g)}\\
& \psi_{\min} \leq^Q \psi_{C,k} \leq \psi_{\max} && \text{(4.13.h)}\\
& \cos(\beta_k^j) \leq cos(\alpha), \quad \forall j \in \mathcal{N} && \text{(4.13.i)}
\end{aligned}$$

As constraints, we impose the initial UAV state (4.13.a) and the UAV dynamics (4.13.b), which are obtained by numerically integrating the continuous model in Section 4.3 with the Runge-Kutta method. We also include bounds on the UAV velocity (4.13.c) and acceleration (4.13.d), to ensure trajectory feasibility. The UAV position is restricted in two ways. On the one hand, it must stay within the volume $\mathcal{F} \in \mathbb{R}^3$ (4.13.e), which is a not necessarily convex space that excludes predefined no-fly zones. These are static zones provided by the director before the mission to keep the UAVs away from known hazards such as buildings, high trees, crowds, etc. On the other hand, the UAV must stay at a minimum distance r_{col} from any additional obstacle O detected during flight (4.13.f), in order to avoid collisions. $\mathbf{p}_{O,k}$ represents the obstacle position at timestep k. One of these constraints is added for each other UAV in the team to model them as dynamic obstacles, using their communicated trajectories to extract their positions along the planning horizon. Other dynamic obstacles; e.g., the actor to be filmed, could also be considered. For this, a model to predict the future position of the obstacle within the time horizon would be required. Mechanical limitations of the gimbal to rotate around each axis are enforced by means of bounds on the pitch (4.13.g) and yaw angles (4.13.h) of the camera with respect to the UAV. Lastly, there are mutual visibility constraints (4.13.i) for each other UAV in the team, to ensure that they do not get into the field of view of the active camera. More details about how to compute this constraint are given in Section 4.4.2.

The cost function consists of four weighted terms to be minimized. The terminal cost $J_N = ||\mathbf{x}_N - [\mathbf{p}_D \; \mathbf{v}_D]^T||^2$ is added to guide the UAV to the desired state imposed by the shot type. The other three terms are related to the smoothness of the trajectory, penalizing UAV accelerations and jerky movements of the camera. Specifically, the terms $J_\theta = |{}^Q\dot{\theta}_{C,k}|^2$ and $J_\psi = |{}^Q\dot{\psi}_{C,k}|^2$ minimize the angular velocities to penalize quick changes in the gimbal angles. Deriving (4.12) analytically, J_θ and J_ψ can be expressed in terms of the optimization variables and the target trajectory. We assume that the target position at the initial timestep is measurable and we apply a kinematic model to predict its trajectory for the time horizon N. An appropriate tuning of the different weights of the terms in the cost function is key to enforcing shot definition while still maintaining smooth camera motion.

4.4.2 Multi-UAV coordination

Our method plans trajectories for multiple UAVs as they do cinematography shots. The cooperation of several UAVs can be used to execute different types of shot simultaneously or to provide alternative views of the same subject. This is particularly appealing for outdoor filming; e.g., for sport events, where the director may want to orchestrate the views from multiple cameras in order to show the surroundings during the line of action. In this section, we provide further insight into how we coordinate the motion of several UAVs while filming.

The first point to highlight is that we solve our optimization problem (4.13) on board each UAV in a distributed manner, but being aware of constraints imposed by neighboring teammates. This is reflected in (4.13.f) and (4.13.i), where we force UAV trajectories to keep a safe distance from others and to stay out of each others' field of view for aesthetic purposes. For this, we assume that UAVs are operating close together to film the same scene, which allows them to communicate their computed trajectories after each planning iteration. However, there are different ways to synchronize the distributed optimization process so that UAVs act in a coordinated fashion. Let us discuss other approaches from key related studies and then our proposal.

In the literature there are multiple proposals for multi-UAV optimal trajectory planning, but as we showed in Section 4.2, only a few of these specifically addressed cinematography aspects. A master–slave approach is applied in Galvane et al. (2018) to solve conflicts between multiple UAVs. Only one of the UAVs (the master) is supposed to be shooting the scene at a time, while the others act as relay slaves that provide complementary viewpoints when selected. The slave UAVs fly in formation with the master, avoiding visibility issues by staying out of its field of view. Fully distributed planning is used in Nägeli et al. (2017b) by means of a sequential consensus approach. Each UAV receives the current planned trajectories from all the others, and computes a new collision-free trajectory taking into account the whole set of future positions from the teammates and the rest of the restrictions. In addition, it is ensured that the trajectories for each UAV are planned sequentially and communicated after each planning iteration. In the first iteration, this is equivalent to establishing a priority plan, but not in subsequent iterations, yielding more cooperative trajectories.

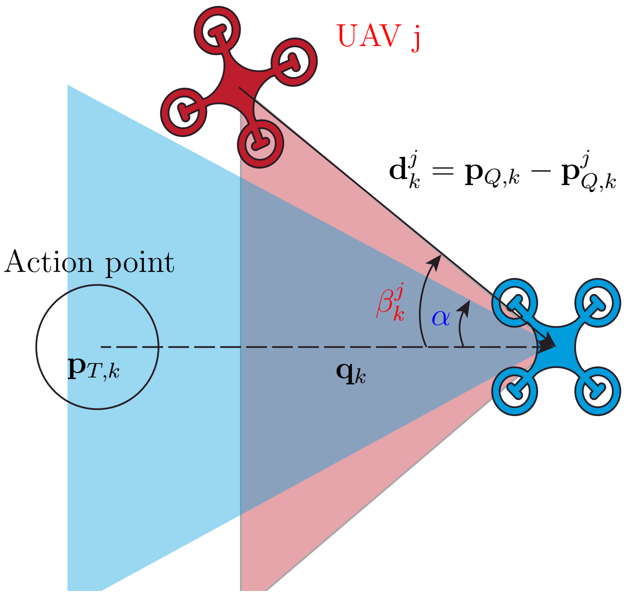


Figure 4.3: Mutual visibility constraint for two UAVs. The UAV on the right (blue) is filming an action point at the same time that it keeps the UAV on top (red) out of its angle of view α.

We follow a hierarchical approach in between these two approaches. In contrast to Galvane et al. (2018), all UAVs can film the scene simultaneously with no preferences, but there is a scheme of priorities to solve multi-UAV conflicts, as in Nägeli et al. (2017b). Thus, the UAV with top priority plans its trajectory ignoring the others; the second UAV generates an optimal trajectory applying collision avoidance and mutual visibility constraints given the planned trajectory from the first UAV, the third UAV avoids the two previous ones, and so on. This scheme helps coordinate UAVs without deadlocks and reduces computational cost as UAV priority increases. Moreover, we do not recompute and communicate the trajectories after each control timestep as in Nägeli et al. (2017b), but instead, replanning is done at a lower frequency, and meanwhile, the UAVs execute their previous trajectories as we will describe in the next section.

In terms of multi-UAV coordination, constraint (4.13.f) copes with collisions between teammates and (4.13.i) with mutual visibility. We consider all neighboring UAVs as dynamic obstacles whose trajectories are known (plans are communicated), and we enforce a safe inter-UAV distance r_{col} along the entire planning horizon N. The procedure to formulate the mutual visibility constraint is illustrated in Figure 4.3. The objective is to ensure that each UAV's camera does not have any other UAVs within its field of view (the angle of view is denoted as α). We approximate the actual field of view of the camera with a circular shape, and α is the semi-cone angle of the cone surrounding the real field of view. We think this is a good approximation for long-range shots and it simplifies the formulation of the mutual visibility constraints, which alleviates the problem of non-linearity and helps in computing a solution. Geometrically, we model UAVs as points that need to stay out of the field of view, but select α large enough to account for the dimensions of the UAVs. If we consider the UAV that is planning its trajectory at position $\mathbf{p}_{Q,k}$ and another neighboring UAV j at position $\mathbf{p}^j_{Q,k}$, then β^j_k refers to the angle between vectors $\mathbf{q}_k = \mathbf{p}_{Q,k} - \mathbf{p}_{T,k}$ and $\mathbf{d}^j_k = \mathbf{p}_{Q,k} - \mathbf{p}^j_{Q,k}$:

$$\cos(\beta^j_k) = \frac{\mathbf{q}_k \cdot \mathbf{d}^j_k}{||\mathbf{q}_k|| \cdot ||\mathbf{d}^j_k||}, \tag{4.14}$$

where $\cos(\beta^j_k) \leq \cos(\alpha)$ is the condition to keep UAV j out of the field of view.

Finally, it is important to note that there may be certain situations where our priority scheme to apply mutual visibility constraints could fail. If we plan a trajectory for the UAV with priority 1, and then another one for the UAV with lower priority 2, then ensuring that UAV 1 is not within the field of view of UAV 2 does not imply the reverse; i.e., UAV 2 could still appear on UAV 1's video. However, these situations are rare in our cinematography application, as there are not many cameras pointing in random directions, but only a few and all of them filming a target typically on the ground. Moreover, since we favor smooth trajectories, we experienced in our tests that our solver tends to avoid crossings between different UAVs' trajectories, as that would result in more curves. Therefore, establishing UAV priorities in a smart way, based on their height or distance to the target, was enough to prevent these issues related to mutual visibility.

4.4.3 Trajectory execution

Our trajectory planners produce optimal trajectories containing UAV positions and velocities sampled at the control timestep, denoted as Δt. As we do not recompute trajectories at each control timestep for computational reasons, we use another independent module for trajectory following, whose task is flying the UAV along its current planned trajectory. This module is executed at a rate of $1/\Delta t$ Hz and keeps track of the last computed trajectory, which is replaced after each planning iteration. Each trajectory follower computes 3D velocity references for the velocity controller on board the UAV. For this purpose, we take the closest point in the trajectory to the current UAV position, and then select another point in the trajectory at least L meters ahead. The 3D velocity reference is a vector pointing to that *look-ahead* waypoint and with the required speed to reach the point within the specified time in the planned trajectory.

At the same time that the UAVs are following their trajectories, a gimbal controller is executed at a rate of $1/\Delta t_G$ Hz to point the camera toward the target being filmed. We assume that each gimbal has an IMU and a low-level controller receiving angular rate commands defined with respect to the world reference frame $\{W\}$. Using feedback about the target position, we generate references for the gimbal angles to track the target, and compensate for the UAV motion and possible errors in trajectory planning. These references are sent to an attitude controller that computes angular velocity commands based on the error between the current and desired orientation in the form of a rotation matrix $\mathbf{R}_e = (\mathbf{R}_C)^T\mathbf{R}_C^*$, where the desired rotation matrix $\mathbf{R}_C^*$ is given by (4.9). Recall that we assumed that $\mathbf{R}_C$ instantaneously takes the value of $\mathbf{R}_C^*$. To design the angular velocity controller, we use a standard first-order controller for stabilization on the special orthogonal group $\mathbb{SO}(3)$, which is given by $\boldsymbol{\omega} = k_\omega(\mathbf{R}_e - \mathbf{R}_e^T)^\vee$, where the *vee* operator $\vee$ transforms 3×3 skew-symmetric matrices into vectors in $\mathbb{R}^3$ (Lee, 2013). More specific details about the mathematical formulation of the gimbal controller can be seen in Cunha et al. (2019).

4.4.4 Discussion

In this section, we discuss some critical aspects of our method for trajectory planning. In particular, its optimality and convergence time, as well as how it deals with issues such as delays in computing solutions, external disturbances due to bad weather, and obstacle representation.

Optimality: We apply numerical methods to solve the optimization problem described in Section 4.4.1, thus converging to an optimal solution for a single UAV. Even though there are no theoretical guarantees of achieving the global optimum when solving a non-linear and non-convex optimization problem, we experienced good results with the numerical solver that we used, both in terms of local optimality and computation time. A proper solver initialization is essential for fast convergence, so we use the last computed trajectory to initialize the solution search. In any case, as we are considering a formulation with multiple UAVs acting simultaneously, our method does not achieve the optimal solution for the complete team. This is because we impose a priority scheme and solve each UAV trajectory assuming the other trajectories fixed for the given time horizon. Even though it would be more optimal to recompute and exchange solutions after each execution time step for all UAVs (Nägeli et al., 2017b), the quality of our solutions was sufficient for the purpose of the application. Moreover, the UAV priorities were fixed in our experiments, but the method could easily be adapted to consider priorities that vary during the mission depending on certain circumstances, to be more efficient. We leave as future work a further analysis to establish bounds on the quality degradation of our solution compared with the complete multi-UAV optimum.

Convergence time: Our trajectory planning problem is a non-linear and non-convex optimization that is complex to solve; even if the team of UAVs does not encounter external obstacles, they need to consider inter-UAV collision avoidance and mutual visibility. Therefore, the time to converge to a solution is not negligible. We tackle this by limiting the time horizon for trajectory planning (which reduces computation time) and using different rates for trajectory planning and execution. Trajectory planning is performed at lower rates to reduce computation (between 0.5 and 2 Hz in our experiments). In addition, we limit the computation time for the solver and keep tracking the last computed trajectory until it converges to a new solution. If the maximum computation time is reached without convergence, there are no guarantees regarding the quality of the computed solution, so we recalculate it initializing the search with the current UAV state, which is usually enough to converge to a new solution. In the unlikely case of reaching the end of the previous computed trajectory without a new solution, the UAV would keep hovering and recomputing trajectories with different initial solutions until convergence.

In addition, we do not assume that solutions are generated instantly and we deal with delays when planning trajectories. The generated trajectories have timestamps associated with each waypoint. The trajectory follower component described in Section 4.4.3 receives these trajectories with a certain delay (due to the solution computation time) and synchronizes them by discarding the initial waypoints corresponding to time instants already gone by.

Performance under external perturbations: Keeping flight stability and smooth trajectories even under external disturbances, such as bad weather conditions, is critical in our method. In the presence of bad weather, the trajectory planning components (Section 4.4.1) would still generate smooth trajectories; however, windy conditions could result in inaccurate following of the trajectory due to external perturbations. Therefore, the key to improving stability under bad weather conditions would be implementing more robust UAV controllers. In our case, we implemented a trajectory follower based on a *pure pursuit* algorithm with a *look-ahead* parameter and a velocity controller. Nonetheless, alternative control techniques (Kamel et al., 2017; Kostadinov and Scaramuzza, 2020) that take external perturbations and uncertainties into account, or integrate non-linear models for the UAV, could be applied to increase flight stability in case of wind gusts. In terms of trajectory planning, we could also adapt the weights of the cost function in the event of bad weather, penalizing these costs more based on UAV accelerations and gimbal angular velocities, and relaxing the cost associated with the desired final state. Thus the generated trajectories would be more conservative from the smoothness point of view, which would help following trajectories in these adverse conditions.

Obstacle representation: In our problem formulation, we include predefined no-fly zones and additional dynamic obstacles. The former are used to indicate static hazards with known positions, such as buildings or treed areas. Dynamic obstacles are other UAVs in the team or external obstacles; e.g., the target being filmed or other actors in the scene, among others. As explained in Section 4.4.1, we represent these dynamic obstacles by means of spherical objects of radius r_{col}, since the included constraint is to keep that safe distance between the 3D obstacle position and the corresponding UAV. We also explained that we need a prediction model to estimate object trajectories within the planning horizon time. We use a constant velocity model to compute these future predictions, although more complex models could be used, too. Moreover, alternative geometrical representations could be used for the obstacles if more information about their shape were known. For instance, 3D ellipsoids with three different axis lengths are used in Nägeli et al. (2017b). In our context, we do not foresee UAVs getting so close to targets that their geometrical shape really matters, and hence, we preferred to use spherical shapes, which ease mathematical formulation.

Obstacle detection is out of the scope of this thesis, so we assume that there is a perception module providing estimates of 3D positions and velocities (for motion prediction) of obstacles. In practice, in our experiments we used dynamic obstacles whose positions could be measured with a GPS and communicated; i.e., other UAV teammates and the filmed target. Nonetheless, this information could be obtained by algorithms processing measurements from pointcloud-based sensors on board the UAVs, such as 3D LIDARs or RGB-D cameras. In this case, alternative obstacle representations based on distance to the points (e.g., to the centroid or to the closest point) within the corresponding pointclouds could be used, as in Chapter 5.

4.5 System architecture

In this section, we present our system architecture, describing the different software components required for trajectory planning, and their interconnection. In particular, the trajectory planning method in this chapter has been integrated into the overall architecture for multi-UAV cinematography presented in Chapter 3.

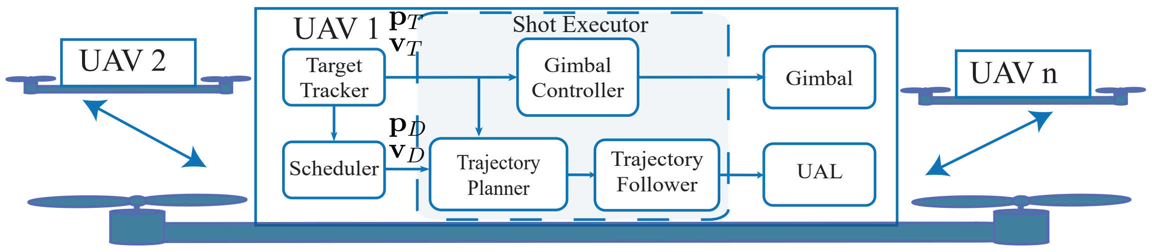


Figure 4.4: System architecture on board each UAV. A Scheduler initiates the shot and continuously updates the desired state for trajectory planning. The Shot Executor plans optimal trajectories to carry out the shot. UAVs exchange their plans for coordination.

Figure 4.4 shows the components running on board each UAV, which are those devoted to shot execution. Recall that each UAV has a Scheduler module that receives shot assignments from the Ground Station and indicates when a new shot should be started. Thee Shot Executor is then in charge of planning and executing optimal trajectories to do each shot, implementing the method described in Section 4.4. As input, the Shot Executor receives the future desired 3D position $\mathbf{p}_D$ and velocity $\mathbf{v}_D$ for the UAV, which is updated continuously by the Scheduler depending on the shot parameters and the target position. For instance, in a lateral shot, the dynamic model of the target is used to predict its position by the end of the horizon time, and to then place the desired position of the UAV at the lateral distance indicated by the shot parameters. Additionally, the target positioning provided by the Target Tracker is required by the Shot Executor to point the gimbal and place the UAV properly.

The Shot Executor, as explained in Section 4.4, consists of three submodules: the *Trajectory Planner*, the *Trajectory Follower*, and the *Gimbal Controller*. The Trajectory Planner computes optimal trajectories for the UAV solving the problem in (4.13) in a receding fashion, trying to reach the desired state indicated by the Scheduler. The Trajectory Follower calculates 3D velocity commands at a higher rate so that the UAV follows the optimal reference trajectory, which is updated any time the Planner generates a new solution. The Gimbal Controller generates commands for the gimbal motors in the form of angular rates in order to keep the camera pointing towards the target. UAL (Real et al., 2020) is used to interface with the position and velocity controllers of the UAV autopilot. Finally, recall that each UAV has a communication link with its teammates in order to share their current computed trajectories, which are used for multi-UAV coordination by the Trajectory Planner.

4.5.1 Cinematography shots

In Chapter 3, we described in detail all the types of shots implemented by our multi-UAV cinematography framework. Each shot has a type, a time duration, and a set of geometric parameters that are used by the system to compute the desired camera position with respect to the target. From all the shots, we take a set of the most representative ones for evaluating trajectory planning in this chapter. For the sake of clarity, let us recall the main features of those selected:

- *Chase/lead:* The UAV chases a target from behind or leads from in front at a certain distance and with a constant altitude.
- *Lateral:* The UAV flies beside a target with constant distance and altitude as the camera tracks it.
- *Flyby:* The UAV overtakes a target with a constant altitude as the camera tracks it. The initial distance behind the target and final distance in front of it are also shot parameters.
- *Orbit:* The UAV flies with a constant altitude orbiting around the target from a certain distance, as the camera tracks it.

Even though our complete system implements additional shots, such as static, elevator, among others., the other shots follow similar behaviors or are not relevant for the evaluation of trajectory planning. In particular, we distinguish between two groups of shots for assessing the performance of the trajectory planner: (i) shots where the relative distance between the UAV and the target is constant (e.g., chase, lead, lateral), denoted as Type I shots; and (ii) shots where this relative distance varies throughout the shot (e.g., flyby, orbit), denoted as Type II shots. Note that an orbit shot can be built with two consecutive flyby shots. In Type I shots, the relative motion of the gimbal with respect to the UAV is quite limited, and the desired camera position does not vary with the shot phase; i.e., it is always at the same distance from the target. In Type II shots though, there is a significant relative motion of the gimbal with respect to the UAV, and the desired camera position depends on the shot phase; e.g., during a flyby shot, it transitions from behind to in front. These two kinds of patterns will result in different behaviors of our trajectory planner, so for a proper evaluation, we test it with shots from both groups.

4.6 Performance evaluation

In this section, we present the results of experiments used to assess the performance of our method for trajectory planning in cinematography. We evaluate the behavior of the resulting trajectories for the two categories of shots, demonstrating that our method achieves smooth, less jerky camera movements. We also show the effect of considering physical limits for gimbal motion, as well as multi-UAV constraints due to collision avoidance and mutual visibility.

We implemented our trajectory planner described in Section 4.4 by means of Forces Pro (Zanelli et al., 2017), which is software that creates domain-specific solvers in C language for non-linear optimization. Forces Pro uses direct multiple shooting (Bock and Plitt, 1984) for problem discretization, approximating the state trajectories to achieve a finite-dimensional optimization problem. Then an algorithm based on the interior-point method is used to solve this non-linear optimization. Table 4.2 depicts common values for some parameters of our method used in all the experiments, where physical constraints correspond to our actual UAV prototypes. All the experiments in this section were performed with a MATLAB-based simulation environment integrating the C libraries from Forces Pro, on a computer with an Intel Core i7 CPU @ 3.20 GHz and 8 GB RAM.

Parameter	**Value**
$\mathbf{u}_{min}, \mathbf{u}_{max}$	± 5 m/s^2
$\mathbf{v}_{min}, \mathbf{v}_{max}$	± 10 m/s
$\theta_{min}, \theta_{max}$	$-\pi/2, -\pi/4$ rad
ψ_{min}, ψ_{max}	$-3\pi/4, 3\pi/4$ rad
α	$\pi/6$ rad
Δt, Δt_G	0.1 s
L	1 m

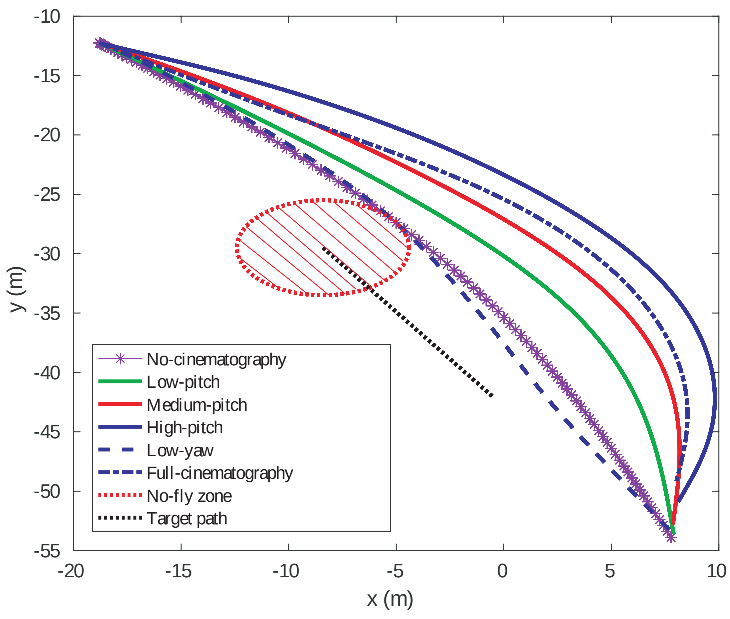
Table 4.2: Values of the method parameters for the experiments.

4.6.1 Cinematographic aspects

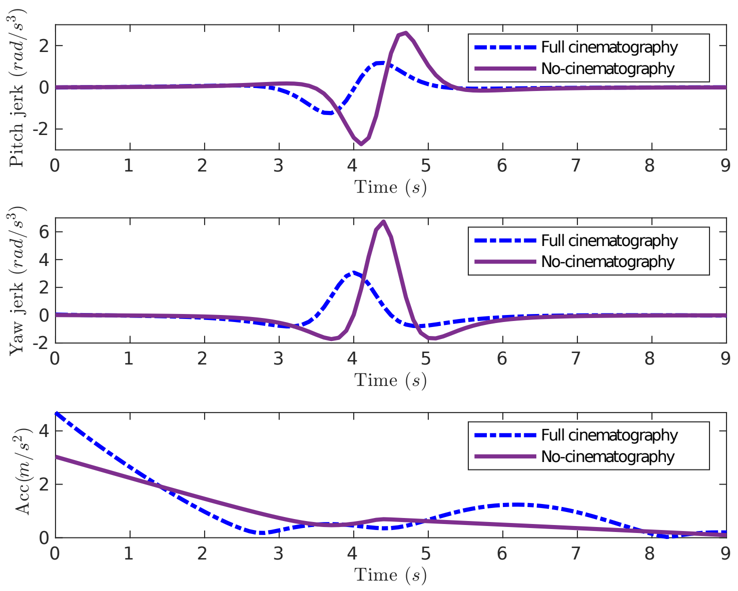
First, we evaluate the effect of imposing cinematography constraints in UAV trajectories. To do so, we selected a shot of Type II, since their relative motion between the target and the camera makes them richer for analyzing cinematographic effects. We did a flyby shot with a single UAV, filming a target moving on the ground with a constant velocity (1.5 m/s) along a straight line (this constant motion model is used to predict the target movement). The UAV had to take a 10-second shot at a constant altitude of 3 m, starting 20 m behind the target and overtaking it to end up 15 m ahead of it. We placed a circular no-fly zone at the starting position of the target, simulating the existence of a tree.

We evaluated the quality of the trajectories computed by our method, setting the planning horizon to $N = 100$ (10 s), in order to calculate the complete trajectory for the whole duration of the shot in a single step, instead of using a receding horizon [1]. We tested different configurations for comparison: *no-cinematography*, uses $w_2 = w_3 = 0$; *low-pitch*, *medium-pitch*, and *high-pitch*, use $w_3 = 0$ and $w_2 = 100$, $w_2 = 1\,000$, and $w_2 = 10\,000$, respectively; *low-yaw* and *high-yaw*, use $w_2 = 0$ and $w_3 = 0.5$, and $w_3 = 1$, respectively; and *full-cinematography*, uses $w_2 = 10\,000$ and $w_3 = 0.5$. For all configurations, we set $w_1 = w_4 = 1$. These values were selected empirically to analyze the planner behavior under a wide spectrum of weighting options in the cost function. Figure 4.5 (top) shows the trajectory followed by the target and the UAV trajectories generated with the different options. Even though trajectory planning was done in

[1]The average time to compute each trajectory was ~ 100 ms, which allows for online computation.



(a)



(b)

Figure 4.5: At the top, top view of the resulting trajectories for different solver configurations of the cost weights. *High-yaw* is not shown as it was too similar to *low-yaw*. The target follows a straight path and the UAV has to execute a flyby shot (10 s) starting 20 m behind and ending up 15 m ahead of the target. The predefined no-fly zone simulates the existence of a tree. At the bottom, temporal evolution of the jerk of the camera angles and the norm of its 3D acceleration. We compare the *full-cinematography* configuration against *no-cinematography*.

	Dist (m)	**Acc (m/s^2)**	**Yaw jerk (rad/s^3)**	**Pitch jerk (rad/s^3)**
No-cinematography	0.00	0.86	0.61	0.31
Low-pitch	1.81	1.13	0.35	0.15
Medium-pitch	6.38	1.25	0.19	0.07
High-pitch	6.13	1.42	0.10	0.03
Low-yaw	0.00	0.81	0.52	0.25
High-yaw	0.00	1.00	0.50	0.23
Full-cinematography	4.44	1.27	0.10	0.03
Full-cinematography (receding)	4.19	1.45	0.08	0.03

Table 4.3: Resulting metrics for a flyby shot. *Dist* is the minimum distance to the no-fly zone. *Acc*, *Yaw jerk* and *Pitch jerk* are the average norms along the trajectory of the 3D acceleration and the jerk of the camera yaw and pitch, respectively.

3D, the altitude did not vary much, as the objective was to perform a shot with a constant altitude. Therefore, a top view is depicted to better evaluate the effect of the weights.

Table 4.3 shows a quantitative comparison of the different configurations. For this comparison, we define the following metrics. First, we measure the minimum distance to any obstacle or no-fly zone in order to check the collision avoidance constraints. We then measure the average norm of the 3D acceleration along the trajectory, and of the jerk (third derivative) of the camera angles θ_C and ψ_C. These three metrics provide an idea of whether the trajectory is smooth and whether it implies jerky movement for the camera. Note that jerky motion has been identified in the literature on aerial cinematography (Bonatti et al., 2020; Gebhardt et al., 2016) as an important cause of low video quality. Figure 4.5 (bottom) depicts the temporal evolution of jerk of the camera angles and the norm of its 3D acceleration.

Our experiment allows us to derive several conclusions. The *no-cinematography* configuration produces a trajectory that gets as close as possible to the no-fly zone and minimizes 3D accelerations (curved trajectory). However, when increasing the weight on the pitch rate, the trajectories get further from the target and their accelerations increase slightly (as longer distances need to be covered in the same shot duration), but the jerk in the camera angles is reduced.In contrast, activating the weight on the yaw rate brings the trajectories closer to the target again. With the *full-cinematography* configuration, we achieve the lowest values in angle jerks and a medium value in 3D acceleration, which seems to be a fairly reasonable trade-off. It can also be seen in Figure 4.5 (bottom) how this configuration reduces camera acceleration and angle jerks compared to *no-cinematography*, obtaining smoother trajectories.

Finally, we also tested the *full-cinematography* configuration in a receding horizon manner. In that case, the solver was run at 1 Hz with a time horizon of 5 s ($N = 50$). The resulting metrics are included in Table 4.3. Using a receding horizon with a horizon shorter than the shot's duration is suboptimal, and the average acceleration increases slightly. However, we achieve similar values for the angle jerk, plus a reduction in the computation time [2]. Moreover, this option of recomputing trajectories online would allow us to correct possible deviations on predictions for the target motion in the event of more random movements (in these simulations, the target moved with a constant velocity).

4.6.2 Time horizon

We also performed an experiment to evaluate the performance of shots of Type I. We selected a lateral shot to show our results, but the behavior of other shots such as a chase or lead shot was similar, as they all do the same movement but with a different relative position with respect to the target. We executed a lateral shot with a duration of 20 s to film a target from a lateral distance of 8 m and a constant altitude of 3 m. As in the previous experiment, the target moved on the ground with a constant velocity (1.5 m/s) along a straight line, and we used that motion model to predict its movement. In normal circumstances, the type of trajectories followed to film the target are not so interesting, as the planner only needs to track it laterally at a constant distance. Therefore, we used this experiment to analyze the effects of modifying the time horizon, which is a critical parameter in terms of both computational load and capacity for anticipation. We used our solver in receding horizon recomputing trajectories at 2 Hz, and we placed a static no-fly zone in the

[2]The average time to compute each trajectory was ~ 7 ms.

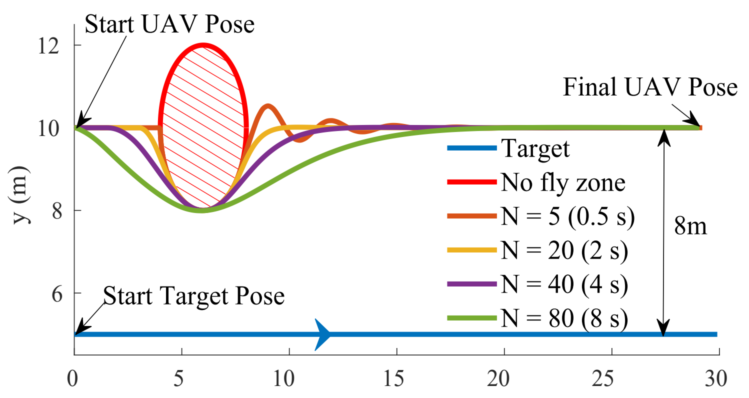


Figure 4.6: Receding horizon comparison for a lateral shot. Top view of the trajectories resulting from different time horizons. The target follows a straight path and the UAV has to execute a lateral shot (10 s) at a distance of 8 m.

Time horizon (s)	**Acc. (m/s^2)**	**Yaw jerk (rad/s^3)**	**Pitch jerk (rad/s^3)**	**Solution time (s)**
0.5	0.15	10.68	1.35	0.004
2	0.08	5.12	0.32	0.016
4	0.05	5.06	0.32	0.029
8	0.04	4.8	0.29	0.101

Table 4.4: Metrics for the lateral shot. *Solution time* is the average time to compute each trajectory.

middle of the UAV trajectory to check its avoidance during the lateral shot under several values of the time horizon N. The top view of the resulting trajectories (the altitude did not vary significantly) can be seen in Figure 4.6. Table 4.4 shows also the performance metrics for the different trajectories.

We can conclude that the trajectories with a longer time horizon were able to predict the potential collision farther in advance and react more smoothly, while shorter horizons resulted in more reactive behaviors. In general, for the kind of outdoor shots that we did in all our experiments, we realized that time horizons on the order of several seconds (consistent with Bonatti et al. (2020)) were enough, as the dynamics of the scenes were not extremely high. We also tested that the computation time for our solver was short enough to calculate these trajectories online.

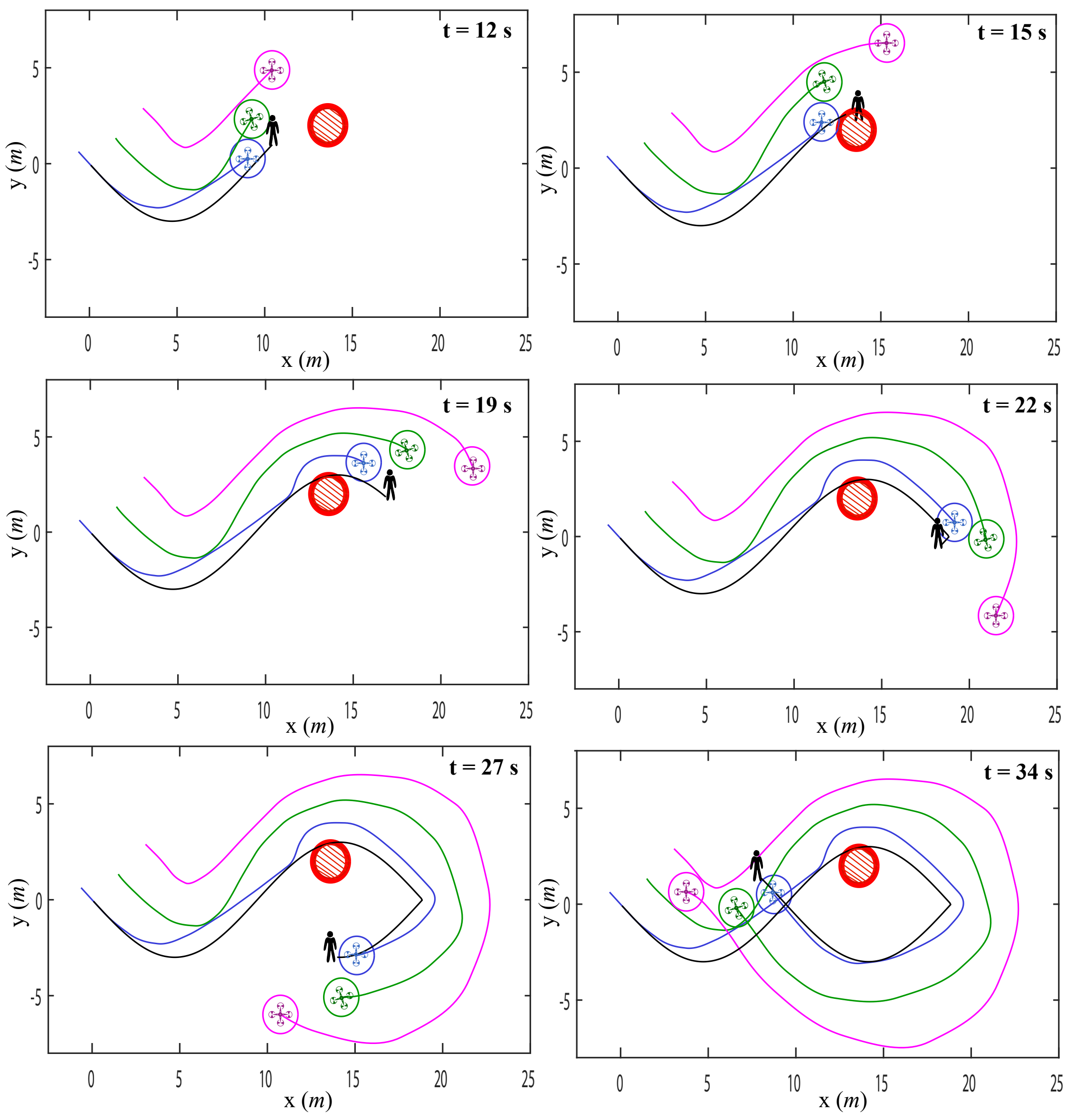


Figure 4.7: Top view of different time instants of an experiment with three UAVs filming a target in a coordinated manner. The target (black) follows an 8-shaped path. UAV 1 (blue) performs a chase shot 2 m behind the target, UAV 2 (green) a lateral shot 2 m aside the target, and UAV 3 (magenta) an orbit shot with a 4 m radius. A no-fly zone (red) is placed in the middle of the target trajectory for multi-UAV avoidance. Each UAV is represented with a 2 m circle around to show the collision avoidance constraint.

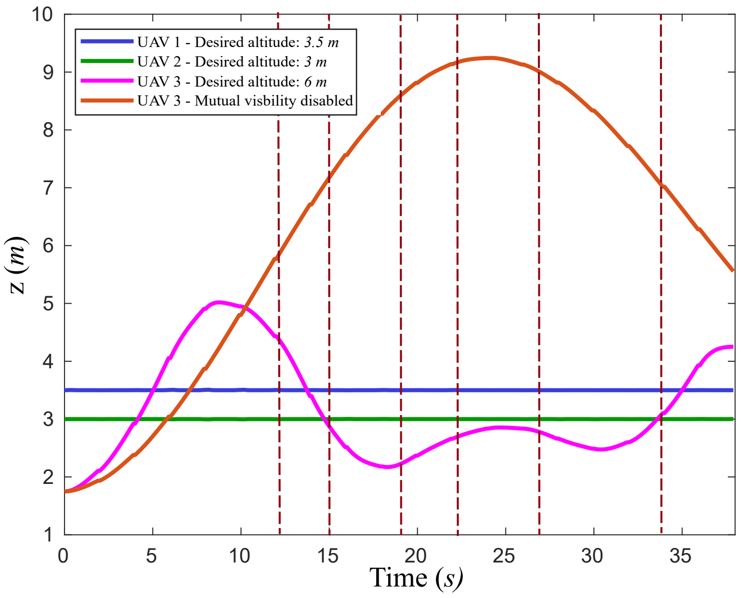


Figure 4.8: Temporal evolution of the UAV altitudes during the multi-UAV coordination experiment. The vertical lines mark the time instants of the snapshots depicted in Figure 4.7.

4.6.3 Multi-UAV coordination

In order to test multi-UAV coordination, we performed another experiment with three UAVs filming a target that followed a figure 8–shaped path at a speed of 1 m/s. We combine heterogeneous shots of Types I and II, with a duration of 40 seconds each: UAV 1 performs a chase shot at a 3.5 m altitude and 2 m behind the target; UAV 2 a lateral shot at a 3 m altitude and a 2 m lateral distance from the target; and UAV 3 is commanded to make an orbit of radius 4 m at an altitude of 6 m. Each UAV ran our method with a receding horizon of $N = 40$ (4 s), recomputing trajectories at 0.5 Hz. We set $r_{\text{col}} = 2$ m as the distance for collision avoidance and the low-pitch configuration for the cost function weights, as we saw this was working better for this experiment. The purpose of this experiment is twofold. First, we show how the method works with a non-rectilinear target motion. We assume the course of the *road* followed by the target is known, and so we use a model that constrains the target motion to that path. Second, we show the main features related to multi-UAV coordination. A no-fly zone is used to test obstacle avoidance in a coordinated manner, including inter-UAV collision avoidance and mutual visibility avoidance.

Figure 4.7 shows several snapshots of the experiment [3]. We set UAV 1 as the one with top priority in the trajectory planner, then UAV 2, and least priority for UAV 3. Between $t = 15$ s and $t = 19$ s, UAV 1 comes across the no-fly zone in its trajectory and deviates to avoid it. Consequently, UAV 2 also deviates in a coordinated manner so as not to collide with UAV 1. Although UAV 1 and 2 are close together throughout the whole experiment, they keep a safe distance of at least 2 m apart. UAV 3 is assigned an orbit of 4 m, but between $t = 19$ s and $t = 27$ s, it moves incrementally further from the target as it begins the orbit. This makes sense in order to minimize the variation in the camera angles, since UAV 3 increases its altitude slightly during this period. Figure 4.8 shows the temporal evolution of the UAV altitudes during the experiment. UAV 3 starts the experiment at an altitude 1.5 m and its assigned altitude for the orbit is 6 m. We provided that desired altitude for the shot to enforce coordination, as we detected that at that altitude the other two UAVs were appearing within the field of view of UAV 3. UAVs 1 and 2 start at their assigned altitudes and maintain them throughout the entire experiment, as they have no issues of mutual visibility. However, UAV 3 starts ascending to reach its assigned altitude, which is never reached, to comply with the mutual visibility constraint. As UAV 3 has less priority, it is the one changing its altitude during the experiment to avoid getting UAVs 1 and 2 within its field of view. We also included in Figure 4.8 the altitude trajectory of UAV 3 when the mutual visibility constraint is disabled in the planner. In that case, it can be seen that the UAV ascends above 6 m, which causes the other two UAVs to appear within its field of view.

4.7 Field experiments

In this section, we report on field experiments where we test our method with 3 UAVs filming a human actor outdoors. This allows us to verify the feasibility of the

[3]For the sake of clarity, a video with the temporal evolution of the simulation is available at `https://youtu.be/u5Vi4leni7U`.

method with actual equipment for UAV cinematography and assess its performance in a scenario with uncertainties in target motion and detection.

The cinematography UAVs used in our experiments were like those in Chapter 3 (see Figure 4.9). Recall that they were custom-designed $1.80 \times 1.80 \times 0.70$ m hexacopters made of carbon fiber and the following onboard 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 their plans with each other and communicate with our Ground Station. 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 weighed 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, which 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.

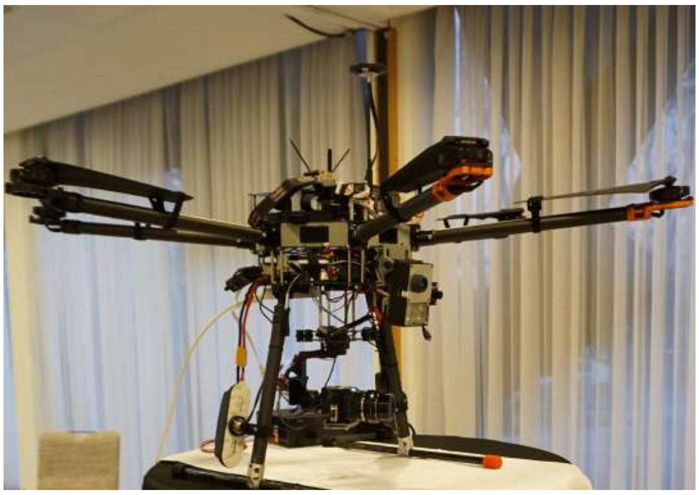


Figure 4.9: One of the UAVs used during the field experiments.

We integrated the architecture described in Section 4.5 into our UAVs, using the ROS framework. We developed our method for trajectory planning (Section 4.4) in

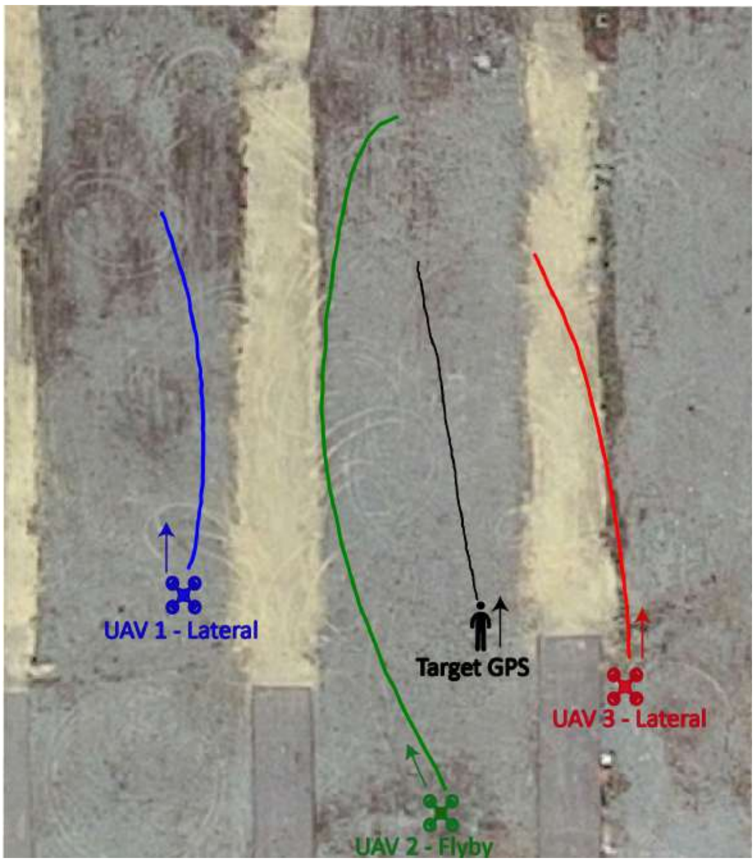


Figure 4.10: Trajectories followed by the UAVs and the human target during the field experiment. UAV 1 (blue) does a lateral shot, UAV 2 (green) a flyby, and UAV 3 (red) another lateral.

C++ [4], using Forces Pro (Zanelli et al., 2017) to generate a compiled library for the non-linear optimization of our specific problem. The parameters used in the experiments were also those in Table 4.2. For collision avoidance, we used $r_{\text{col}} = 5$ m, a value slightly larger than in our simulations for safety reasons. We also limited the maximum velocity of the UAVs to 1 m/s for safety reasons. All trajectories were computed on board the UAVs online at 0.5 Hz, with a receding horizon of $N = 100$ (10 s). Then the Trajectory Follower modules generated 3D velocity commands at 10 Hz to be sent to the autopilot controllers through the UAL component. We assumed a constant speed model for the target motion. This model was inaccurate, as the actual

[4] `https://github.com/alfalcmar/optimal_navigation`

target speed was unknown, but these uncertainties were addressed by recomputing the trajectories with a receding horizon.

We designed a field experiment with 3 UAVs simultaneously taking different shots of a human target walking on the ground. UAV 1 takes a lateral shot following the target sideways with a lateral distance of 20 m; UAV 2 does a flyby shot starting 15 m behind the target and finishing 15 m ahead of it in the target motion line; and UAV 3 does another lateral shot, but from the other side and with a lateral distance of 15 m. For safety reasons, we established different altitudes for the UAVs, 3 m, 10 m, and 7 m, respectively. In our decentralized trajectory planning scheme, UAV 1 had top priority, followed by UAV 2 and then UAV 3. In order to design the shots of the mission safely and with good aesthetic outputs, we created a realistic simulation in Gazebo with all our components integrated and a SITL approach for the UAVs (i.e., the actual PX4 software of the autopilots was run in the simulator).

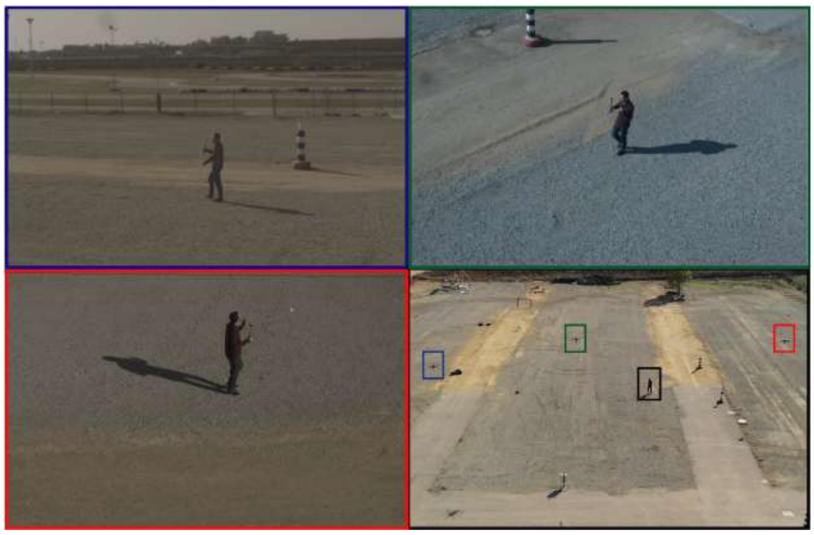


Figure 4.11: Examples of images from the cameras on board the UAVs during the experiment: top left, UAV 1 (blue); top right, UAV 2 (green); and bottom left, UAV 3 (red). Bottom right, an overall view of the experiment showing the three UAVs and the target.

The full video of the field experiment can be found at `https://youtu.be/M71gYva-Z6M`, and the actual trajectories followed by the UAVs are depicted in Figure 4.10. Figure 4.11 shows some example images captured by the onboard cameras

UAV	**Distance traveled (m)**	**Acc. (m/s^2)**	**Dist. (m)**
1	95	0.125	9.543
2	141.4	0.108	9.543
3	98.6	0.100	14.711

Table 4.5: Metrics of the trajectories followed by the UAVs during the field experiment. We measure the total distance traveled for each UAV, the average norm of the 3D accelerations, and the minimum distance (horizontally) to other UAVs.

during the experiment. The experiment demonstrates that our method is able to generate online trajectories for the UAVs coping with cinematographic constraints (i.e., no jerky motion, gimbal mechanical limitations, and mutual visibility) and safety constraints (i.e., inter-UAV collision avoidance); and keeping the target in the cameras' field of view, even under conditions of noisy target detection and uncertainties in its motion. We measured some metrics of the resulting trajectories (see Table 4.5) in order to evaluate the performance of our method. It can be seen that UAV accelerations were smooth, in line with those produced in our simulations, and the minimum distances between UAVs were always higher than the collision avoidance constraint limit (5 m).

4.8 Conclusions

In this chapter, we presented a method for planning optimal trajectories with a team of UAVs in a cinematography application. We proposed a novel formulation for non-linear trajectory optimization, executed in a decentralized and online fashion. Our method integrates UAV dynamics and collision avoidance, as well as cinematographic aspects such as gimbal limits and mutual camera visibility. Our experimental results demonstrate that our method can produce coordinated multi-UAV trajectories that are smooth and reduce jerky movements. We also show that our method can be applied to different types of shots and compute trajectories online for time horizons with lengths of up to 10 seconds, which seems enough for the outdoor cinematographic scenes that we considered. Our field experiments proved the applicability of the method with an actual team of UAV cinematographers filming outdoors.

As future work, we plan to study alternative schemes for decentralized multi-UAV coordination instead of our priority-based computation. Our objective is to compute in multi-UAV approximate solutions a distributed manner that are closer to the optimum, but without significantly increasing the computation time. We believe that a comparison with methods based on reinforcement learning could also be of high interest.

Chapter 5

Aerial filming with distributed lighting by a team of UAVs

In Chapter 3, we presented a framework for cinematography with multiple UAVs, and in Chapter 4 we presented a specific algorithm for optimal trajectory planning to provide aesthetic views. Despite the advantages of using UAVs for outdoor cinematography, they face limitations in scenarios with insufficient natural light and without artificial lighting. This chapter describes a method for autonomous aerial filming with distributed lighting by a team of UAVs. We formulate a novel optimization problem for multi-UAV trajectory planning and propose a leader–follower formation to film a target with one UAV (leader), while the other UAVs (followers) illuminate from several directions, which is one of the fundamental techniques of traditional filming. In order to tackle a problem with non-convex optimization due to the obstacle avoidance constraints, the multi-UAV trajectory planning problem is decomposed into two parts: non-linear cinematographic aspects are formulated without obstacle avoidance to generate reference trajectories, then these are used to generate collision-free regions that are convex and transform the final problem into a QP optimization task. This decomposition allows us to integrate collision avoidance into trajectory planning in a more scalable way, yielding a method that can work in dynamic, cluttered environments. The chapter presents experimental results for aerial filming to monitor inspection activities, both in simulated and real scenarios.

5.1 Introduction

The use of UAVs as flying cameras presents a remarkable potential not only for recreational cinematography, but also for monitoring inspection operations in outdoor infrastructures with complex access. For instance, the EU-funded AERIAL-CORE project [1] proposes the use of UAVs to monitor the safety of human workers at great heights during maintenance operations on electrical power lines (see Figure 5.1). Although the footage is not used for entertainment purpose in this case, a high-quality video would speed up the assessment by human operators at the ground station. Teams with multiple UAVs expand upon these possibilities, as they may be able to provide alternative points of view or even supplementary illumination. Similarly, in the DRONUMENT project of the NAKI II program, efficient variable illumination plays a key role in documenting the interiors of historical buildings.

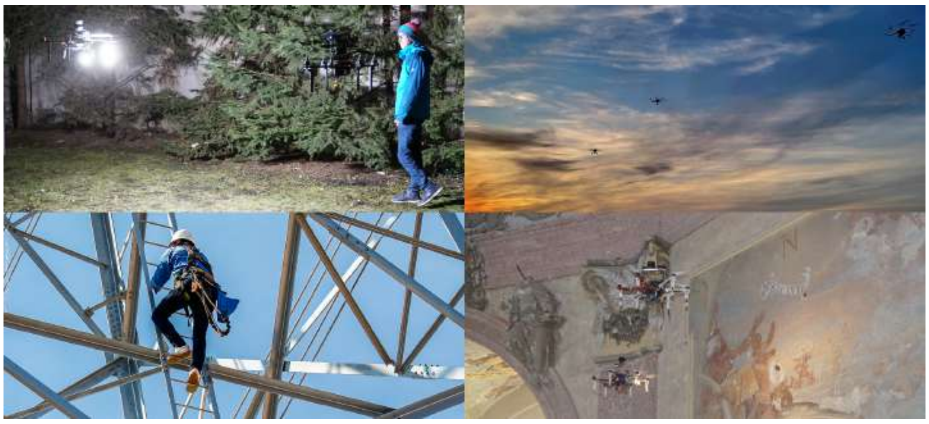


Figure 5.1: UAV filming applications to provide external lighting, to capture smooth shots outdoors, and to monitor dangerous maintenance operations for power lines. Pictures were obtained within AERIAL-CORE and DRONUMENT projects, for which the proposed technology is being developed.

Proper lighting techniques are fundamental for bringing out details in an image and in creating natural-looking film scenes that more closely represent real life. Thus, cinematography sets are packed with different lighting sources, since digital sensors are not as reactive to light as the human eye. This can be especially important in

[1] `https://aerial-core.eu`

monitoring maintenance operations scheduled at times of the day with poor illumination. Although aerial cinematography has been attractive to the scientific community as of late, lighting techniques have yet to be applied to improve the performance of filming.

In this context, navigating a team of UAVs for filming tasks with distributed lighting is complex. Save, smooth trajectories are required to achieve pleasing shots that do not compromise safety in dynamic scenarios. In this chapter, we propose a method for online trajectory planning and execution with multiple UAVs. Our team of UAVs obeys a leader–follower scheme where the formation leader carries an onboard camera to film a moving target and the followers generate trajectories that enable distributed lighting of the target while maintaining desired lighting angles. We formulate a non-linear, optimization-based method that plans trajectories for the filming UAV that will produce visually pleasing footage and distributes the surrounding UAVs in a specified formation. At the same time, we focus on safety by including a systematic framework for obstacle avoidance. Safe flight corridors for the UAVs are generated by forming sets of convex polyhedrons that model the free space. Optimal and safe trajectories are thereafter computed within these convex sets. Our main contributions are summarized as the following:

- We formulate a novel optimization problem for aerial filming with distributed lighting. Using a leader–follower scheme, we plan and execute trajectories in a distributed manner. Optimization is run in a receding horizon setting to compute smooth trajectories with pleasing footage for the UAV that is filming (the leader), which takes shots of a dynamic target indicated by an external user. The followers compute their trajectories to maintain a formation with specified lighting angles on the target.

- We propose a new method to tackle non-convex trajectory optimization with obstacle avoidance in real time. We decompose the problem into two parts. Non-linear cinematographic aspects are formulated as a problem without obstacle avoidance to generate reference trajectories. These are used to generate collision-free regions which are convex and to transform the problem into a final QP optimization task.

- We present experimental results to evaluate the performance of our method for different types of cinematographic shots. We prove that our method is capable of computing smooth trajectories that reduce jerky movements in real time and we show that the distributed formation improves the illumination of the footage. The system is evaluated in various outdoor scenarios in a realistic simulator, including the filming of a moving target in a cluttered environment.

5.2 Related work

The modification of lighting angles to improve images is a fundamental topic in cinematography (Hall, 2015). An onboard light used together with a camera on a single UAV can compensate for insufficient lighting, but positioning lights at different angles with respect to the camera axis would require the use of several UAVs. Despite the unquestionable importance of lighting for shot quality, its usage for aerial cinematography has not been well-studied. Utilizing UAVs to provide supplementary illumination has been proposed for building documentation tasks (Petracek et al., 2020) and for tunnel inspection (Petrlík et al., 2020). A formation of UAVs with one carrying a camera and others carrying lights was deployed to document the overshadowed parts of historical buildings (Saska et al., 2017). A similar system has been used to carry out specialized documentation techniques (Krátký et al., 2020). However, these studies have proposed lighting for tasks in static scenes, whereas the present thesis deals with filming moving targets in dynamic and potentially cluttered environments; e.g., to monitor inspection operations in large outdoor infrastructure.

In order to guarantee safe trajectory planning in multi-UAV cinematography, most studies (Nägeli et al., 2017b; Galvane et al., 2018; Alcántara et al., 2021) only consider collision avoidance with actors, other UAVs, or static objects that can be modeled with previously known no-fly zones. The work in Bonatti et al. (2020) integrates

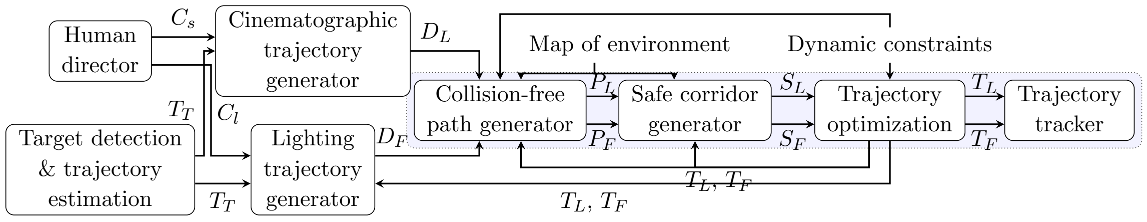


Figure 5.2: The architecture of the proposed system. C_s and C_l represent the desired cinematographic shot type and lighting configuration specified by a human director; T_T is the target estimated trajectory; D_L, D_F are reference trajectories for the leader UAV and the follower UAVs, respectively; P_L, P_F are collision-free paths generated along the desired trajectories; S_L, S_F are safe corridors along the collision-free paths; and T_L, T_F are optimized trajectories for the camera and lighting UAVs, respectively. The modules enclosed in the blue rectangle run on both types of UAVs.

local mapping with onboard sensors to penalize proximity to obstacles in the cost function and solves an unconstrained optimization problem. An alternative approach to obstacle avoidance applied in standard UAV trajectory planning is to create a convex representation of the free space by means of a set of linear inequality constraints (Yu et al., 2016; Mohta et al., 2018; Liu et al., 2017; Tordesillas et al., 2021) in order to obtain a QP formulation for real-time motion planning. We have been inspired by these single-UAV studies to develop a fundamental framework for the representation of obstacles in our non-linear optimization problem for multi-UAV cinematography.

5.3 System overview

Figure 5.2 shows the architecture of the entire system. The leader UAV carries a camera for filming while the others carry light sources to provide proper illumination. A human director specifies the cinematographic parameters for the scene. These parameters include the shot type (i.e., the camera motion relative to the target), the camera shooting angle for the leader, and the desired lighting angles for the followers. This information, together with an estimation of the target trajectory, is used to generate reference trajectories for the UAVs (Section 5.4.2). These initial trajectories do not consider obstacle avoidance but only cinematographic aspects. The leader attempts to execute the commanded shot smoothly, while the followers maintain a surrounding formation with the desired lighting angles.

Safety is ensured by integrating information from a local map for collision avoidance (Section 5.4.3). First, a collision-free path is generated for each UAV using the map and the initial cinematographic trajectories as guidelines. Then a safe corridor along each of these paths is computed, consisting of a set of obstacle-free polyhedrons generated by the convex decomposition of the free space (see Figure 5.5). Finally, the UAV trajectories are obtained as a result of a trajectory optimization process that computes dynamically feasible trajectories inside each safe corridor (Section 5.4.4). Inter-UAV collision avoidance is achieved by including the team-mates' planned trajectories as obstacles in the map.

The entire pipeline shown in Figure 5.2 (except for the *Human director* component) runs on board each UAV in a receding horizon manner. This enables the online planning module to react properly to changes in the behavior of the target being filmed, as well as to malfunctioning team-members or previously unseen obstacles. Note that either the *Cinematographic trajectory generator* or the *Lighting trajectory generator* is activated on each UAV, depending on whether it carries a camera or a light source. The component for trajectory tracking on each UAV corresponds to the low-level control pipeline described in Báča et al. (2021).

5.4 Autonomous aerial cinematography

In this section, we begin by detailing the dynamic UAV model (Section 5.4.1). We then describe our procedure to generate optimal and safe trajectories for each UAV (Sections 5.4.2, 5.4.3, and 5.4.4). Lastly, we explain how the orientation of each UAV is controlled (Section 5.4.5).

5.4.1 Multirotor aerial vehicle dynamic model

An independent trajectory tracker (Báča et al., 2021) for UAV attitude control is used, which allows for planning with a simplified positional dynamic UAV model.

In addition, the orientation of the camera or light source on board (depending on the UAV) needs to be modeled. We assume the existence of a gimbal mechanism to compensate angle deviations due to changes in the UAV's attitude. Therefore, it is assumed that camera roll is negligible and we only control pitch and heading. Since the heading of a multirotor vehicle can be controlled independently of its position, we fix the relative position between the camera/light and the UAV to always point forward and control its heading through the UAV heading. We use the same reference frames and dynamic model as in Chapter 4. The positional part for the vehicle (we assume that the camera/light and the vehicle position coincide) of the dynamic model is defined as a linear double integrator:

$$\begin{aligned} \dot{\mathbf{p}}_Q &= \mathbf{v}_Q, \\ \dot{\mathbf{v}}_Q &= \mathbf{a}_Q, \end{aligned} \tag{5.1}$$

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

$$\begin{aligned} \dot{\mathbf{o}}_C &= \boldsymbol{\omega}_C, \\ \dot{\boldsymbol{\omega}}_C &= \boldsymbol{\theta}_C, \end{aligned} \tag{5.2}$$

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

5.4.2 Generation of reference trajectories

The first step of our method for trajectory planning is to generate a reference trajectory D_j for each UAV j. The problem complexity is alleviated by removing collision avoidance constraints and focusing only on cinematographic aspects. For the filming UAV, the objective is to reach a position relative to the target as provided by the shot type C_s, while minimizing changes in the camera angle to produce pleasing images. A specific camera shooting angle ψ_d over the target needs to be maintained. The following non-linear optimization problem is formulated [2] for the filming UAV:

$$\underset{\substack{\mathbf{x}_0,\ldots,\mathbf{x}_{N-1}\\ \mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}}{\text{minimize}} \sum_{k=1}^{N} (||\mathbf{u}_{k-1}||^2 + \alpha_1 J_{\psi,k}) + \alpha_2 J_N, \tag{5.3}$$

$$\begin{aligned}
\text{subject to } \mathbf{x}_0 &= \mathbf{x}', && && (5.3.a)\\
\mathbf{x}_{k+1} &= \mathrm{f}_p(\mathbf{x}_k, \mathbf{u}_k) && \forall k \in \{0,\ldots,N-1\}, && (5.3.b)\\
\mathbf{v}_{\min} &\le \mathbf{v}_{Q,k} \le \mathbf{v}_{\max} && \forall k \in \{1,\ldots,N\}, && (5.3.c)\\
\mathbf{u}_{\min} &\le \mathbf{u}_k \le \mathbf{u}_{\max} && \forall k \in \{0,\ldots,N-1\}, && (5.3.d)\\
q_{\mathrm{z},\min} &\le q_{z,k} && \forall k \in \{1,\ldots,N\} && (5.3.e)
\end{aligned}$$

where $\mathrm{f}_p(\cdot)$ represents the positional part of the dynamic model defined in Section 5.4.1; $\mathbf{v}_{\min}$, $\mathbf{v}_{\max}$ are velocity limitations; and $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ control input limitations.

The first two terms in the cost function aim to produce smooth trajectories by penalizing UAV accelerations and reducing gimbal movements. The director specifies an aesthetic objective through the desired camera shooting angle ψ_d to film the target (see Figure 5.3). Emphasis is placed on positioning the UAV to keep this angle constant without moving the gimbal. In doing so, the angular changes in the gimbal are reduced in order to favor less jerky camera motion and therefore produce pleasing footage. In order to define J_ψ, the relative position between the leader UAV camera ($\mathbf{p}_L$) and the target ($\mathbf{p}_T$) is introduced as:

$$\mathbf{q} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^T = \mathbf{p}_L - \mathbf{p}_T. \tag{5.4}$$

[2]For simplicity of description, $\mathbf{x} := \mathbf{x}_p$ and $\mathbf{u} := \mathbf{u}_p$. We use the Runge-Kutta method for numerical integration.

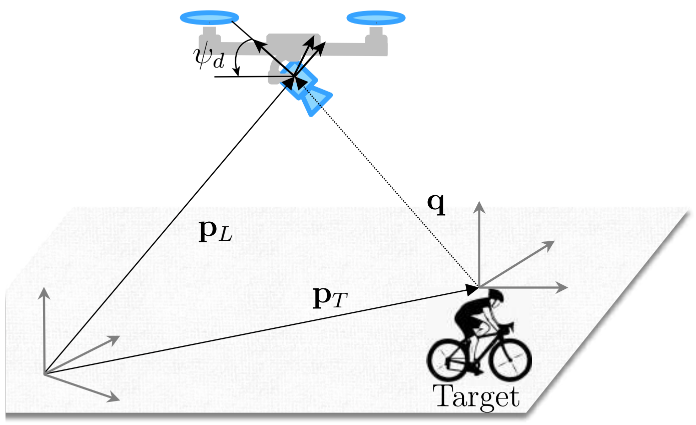


Figure 5.3: The camera shooting angle, the leader UAV position, the target position, and their relative position.

Then we define J_ψ as:

$$J_{\psi,k} = \left(\tan(\psi_d) - \frac{q_{z,k}}{\sqrt{q_{x,k}^2 + q_{y,k}^2}} \right)^2 . \tag{5.5}$$

By minimizing this cost, we implicitly minimize variations in the camera pitch angle as the relative pitch with respect to the target is kept constant. The camera heading corresponds to the UAV heading, whose variations are also smoothed, as explained in Section 5.4.4. The idea is to generate UAV trajectories where the gimbal only needs to move slightly to compensate for small disturbances.

The terminal cost $J_N = ||\mathbf{x}_{xy,d} - \mathbf{x}_{xy,N}||^2$ guides the UAV to a desired state (on the horizontal XY plane) imposed by the shot type; e.g., at a certain distance beside the target's final position in a lateral shot. Note that a final UAV height is not imposed, as we want the planner to compute the optimal p_z to maintain the camera shooting angle specified by the director. Lastly, the constraint (5.3.e) establishes a minimum distance above the target for safety purposes.

The reference trajectories for the lighting UAVs are computed to achieve a desired leader–follower formation around the target. The desired position of the followers is influenced by the corresponding leader position $\mathbf{p}_L$ and camera orientation $\mathbf{o}_L$, the target position $\mathbf{p}_T$, the desired lighting angles of the j-th light χ_j and ϱ_j, and the desired distance of the light to the target d_j. The desired position of the j-th follower $\mathbf{p}_j$ is then given by the equation:

$$\mathbf{p}_j = \mathbf{p}_T + d_j \begin{bmatrix} -\cos(\varphi_j)\cos(\xi_j) \\ -\sin(\varphi_j)\cos(\xi_j) \\ \sin(\xi_j) \end{bmatrix}, \tag{5.6}$$

where $\varphi_j = \varphi_L + \chi_j$ and $\xi_j = \xi_L + \varrho_j$ are desired lighting angles relative to the camera's optical axis (see Figure 5.4). To avoid quick changes in the desired follower UAV positions that could be caused by switching to a new target, the concept of a virtual target placed a certain distance in front of the camera is applied. The position of this virtual target is given by:

$$\mathbf{p}_v = \mathbf{p}_L + d_v \begin{bmatrix} \cos(\varphi_L)\cos(\xi_L) \\ \sin(\varphi_L)\cos(\xi_L) \\ \sin(\xi_L) \end{bmatrix}, \tag{5.7}$$

where d_v is the desired distance between the virtual target and the center of the camera, and $\mathbf{p}_v$ denotes the virtual target position. Substituting position $\mathbf{p}_v$ for $\mathbf{p}_T$ in Equation (5.6), a new leader–follower scheme is acquired, resulting in a fixed shape formation for constant lighting angles χ_j, ϱ_j and a more compact formation when these angles vary.

5.4.3 Generation of safe corridors

The initial reference trajectories are computed without considering obstacles. They are therefore used as seeds to generate a safe corridor S_j for each UAV j where collision-free trajectories can then be computed. First, we convert each trajectory D_j into a collision-free path P_j. We iterate over each waypoint in D_j and add it directly to P_j if it is collision-free. Otherwise, we label the previous collision-free waypoint as A and keep moving along D_j until we find the next collision-free waypoint B. Then we try to find an alternative collision-free path from A to B, to be appended to P_j

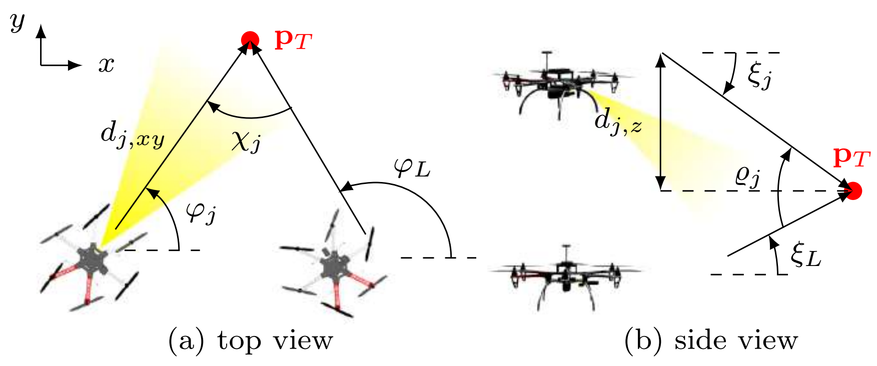


Figure 5.4: The leader–follower scheme defined by Equation (5.6).

and continue iterating. For that alternative path, we use the Jump Point Search (JPS) algorithm introduced in Harabor and Grastien (2011, 2014) and extended to 3D in Liu et al. (2017). A real-time performance is ensured by introducing a timeout for the JPS path search.

If the JPS algorithm fails to find a path within the given timeout from A to B, we run it again to connect A directly to the last waypoint in D_j (let this waypoint be C). If this is not found either, we append to P_j the path to the node closest to C from all those expanded during the JPS search. Once completed, P_j is post-processed so that the waypoints are sampled evenly with a maximum sampling distance, as in the initial D_j. Since these collision-free paths are used as a guide for trajectory optimization in subsequent steps, this limitation helps to avoid dynamic infeasibility of the final trajectories. The process to create a collision-free path P_j and its corresponding safe corridor S_j is illustrated in Figure 5.5.

Safe corridors are generated around these collision-free paths using a map of the environment represented by a point cloud O_{pcl} and by using the convex decomposition method proposed in Liu et al. (2017). This method is based on an iterative procedure for generating polyhedrons. It begins by inflating an ellipsoid aligned with each path segment. In the next step, tangent planes are constructed at the contact points between the ellipsoid and any obstacles. Next, all points lying behind this plane are removed from O_{pcl}. The next iteration starts by again inflating the ellipsoid up to the nearest point in O_{pcl}. This procedure is terminated if there are no remaining points in

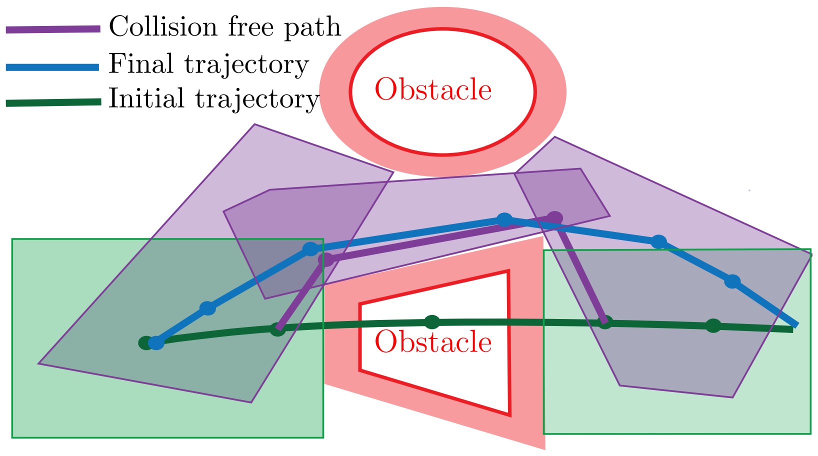


Figure 5.5: The safe corridor–generating process. The initial reference trajectory (dark green) is converted into a collision-free path (purple), and the obstacle-free polyhedrons are generated along this path. The final optimized trajectory within the safe corridor is also shown (blue). We inflate the obstacles for safety purposes (light red).

O_{pcl}. The tangent planes generated define an obstacle-free polyhedron $\mathcal{P}$ enclosing the corresponding path segment, and the set of all the polyhedrons along the path constitutes the safe corridor.

5.4.4 Trajectory optimization

Given a collision-free path P and its corresponding safe corridor S, a final optimal trajectory is computed through a QP problem in receding horizon. The particular optimization task [3] attempts to track a desired trajectory $\mathbf{p}_d$ corresponding to the

[3]For simplicity of description, $\mathbf{x} := \mathbf{x}_p$, and $\mathbf{u} := \mathbf{u}_p$.

reference trajectory D_j:

$$\underset{\substack{\mathbf{x}_0,\dots,\mathbf{x}_N\\ \mathbf{u}_0,\dots,\mathbf{u}_{N-1}}}{\text{minimize}} \sum_{k=1}^{N}(||\mathbf{p}_{d,k} - \mathbf{p}_{Q,k}||^2 + \beta||\mathbf{u}_{k-1}||^2), \tag{5.8}$$

$$\text{subject to } \mathbf{x}_0 = \mathbf{x}', \tag{5.8.a}$$

$$\mathbf{x}_{k+1} = \mathrm{f}_p(\mathbf{x}_k, \mathbf{u}_k) \quad \forall k \in \{0, \dots, N-1\}, \tag{5.8.b}$$

$$\mathbf{v}_{\min} \leq \mathbf{v}_{Q,k} \leq \mathbf{v}_{\max} \;\; \forall k \in \{1, \dots, N\}, \tag{5.8.c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} \;\; \forall k \in \{0, \dots, N-1\}, \tag{5.8.d}$$

$$\mathbf{p}_{Q,k} \in \mathcal{P}_k \qquad\qquad \forall k \in \{1, \dots, N\}, \tag{5.8.e}$$

where $\mathrm{f}_p(\cdot)$ represents the positional part of a dynamic model defined in Section 5.4.1; $\mathbf{v}_{\min}$, $\mathbf{v}_{\max}$ are velocity limitations; $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ control inputs limitations; and $\mathcal{P}_k$ is a convex polyhedron representing the free space associated with the k-th transition point. The last constraint ensures a safe resulting trajectory without collisions. Given that the constraint (5.8.e) can be decoupled in a set of linear constraints, the problem becomes a quadratic convex program.

The optimization formulation is the same for both the leader and follower UAVs. However, there are a couple of important differences. First, the desired reference trajectories are computed in a different way, following either filming or lighting criteria (see Section 5.4.2). Second, the followers encode mutual-collision avoidance through constraint (5.8.e). To prevent negative effects on the cinematographic quality of the performed shot, mutual collision avoidance is left entirely to the followers. A fixed priority scheme is defined for the UAVs where the occupied space O_{pcl} of each follower is updated with the current planned trajectories from the leader and other followers of higher priority. At each waypoint from the trajectories that are to be avoided, a spherical object is placed with the desired collision avoidance radius. This mechanism for mutual collision avoidance simplifies the optimization problem and does not significantly increase computational demands. This is due to the dynamic nature of the targeted environment where local maps already need to be updated at each planning iteration.

Another important issue for applications of multi-UAV cinematography is how to prevent other UAVs from appearing in the field of view (FoV) of the filming UAV. However, including this in the optimization task as either a constraint or a cost term can increase the complexity of the problem considerably. We considered including the FoV of the leader camera as an obstacle in the local maps of the followers, so that they could avoid it. Even so, relatively small changes in camera orientation could result in significant changes in the map representation and hence lead to unstable planned trajectories. Therefore, the camera's FoV is avoided by the lighting UAVs only through penalizing deviations from the desired trajectories $\mathbf{p}_d$. Thus, FoV avoidance is mostly determined by the choice of the lighting parameters that describe the desired formation.

5.4.5 Orientation control

In this application, both the camera and the light sources need to be always pointing at the filmed target. Hence, their desired orientation is given by

$$\mathbf{o}_d = \begin{bmatrix}\varphi_d \; \xi_d\end{bmatrix}^T = \begin{bmatrix}\arctan(q_y, q_x) \;\; \sin\left(\frac{q_z}{||q||}\right)\end{bmatrix}^T. \tag{5.9}$$

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:

$$\underset{\substack{\mathbf{x}_0,\dots,\mathbf{x}_N \\ \mathbf{u_0},\dots,\mathbf{u_{N-1}}}}{\text{minimize}} \sum_{k=1}^{N} (||\mathbf{o}_{d,k} - \mathbf{o}_{C,k}||^2 + \gamma||\mathbf{u}_{k-1}||^2), \tag{5.10}$$

$$\begin{aligned}
\text{subject to } \mathbf{x}_0 &= \mathbf{x}', && (5.10.\text{a})\\
\mathbf{x}_{k+1} &= \mathrm{f}_o(\mathbf{x}_k, \mathbf{u}_k) \quad \forall k \in \{0,\dots,N-1\}, && (5.10.\text{b})\\
\boldsymbol{\omega}_{\min} \leq \boldsymbol{\omega}_{C,k} &\leq \boldsymbol{\omega}_{\max} \;\; \forall k \in \{1,\dots,N\}, && (5.10.\text{c})\\
\xi_{\min} \leq \xi_k &\leq \xi_{\max} \quad \forall k \in \{1,\dots,N\}, && (5.10.\text{d})\\
\mathbf{u}_{\min} \leq \mathbf{u}_k &\leq \mathbf{u}_{\max} \;\; \forall k \in \{0,\dots,N-1\}, && (5.10.\text{e})
\end{aligned}$$

where $\mathrm{f}_o(\cdot)$ represents the orientation aspect of the dynamic model defined in Section 5.4.1; $\omega_{\min}$, $\omega_{\max}$ are limitations on the angular velocities; $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ control input limitations; and $\xi_{\min}$, $\xi_{\max}$ represent gimbal hardware limitations on adjusting pitch angles. The heading and pitch angles of the camera or light can be controlled independently. Thus, (5.10) was decoupled into two simpler problems. The optimal solution for each problem can be found analytically with a standard framework for linear MPC).

5.5 Experimental evaluation

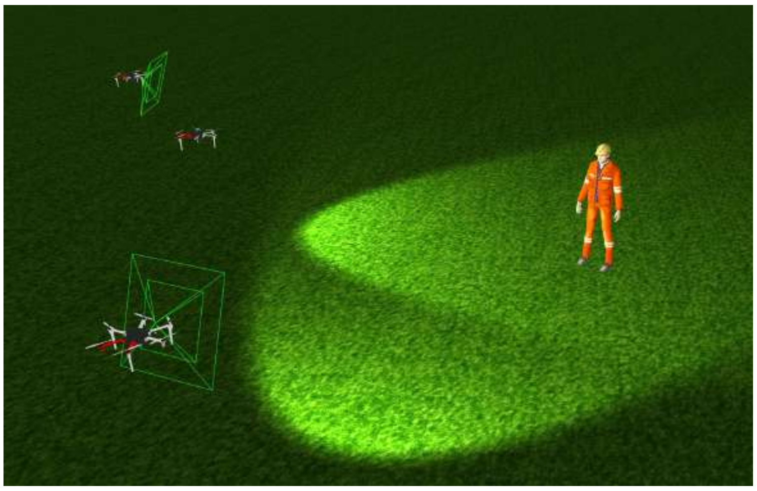


Figure 5.6: Illustration of the proposed system filming a worker under the required illumination.

In this section, experimental results are presented to demonstrate the performance of our method for multi-UAV trajectory planning. Figure 5.6 depicts an illustrative scene of our simulation with one UAV filming a human worker and two others providing proper illumination. We have assessed that the proposed method is capable of computing smooth cinematographic trajectories in real time. Additionally, we have evaluated that the trajectories of the follower UAVs which provide lighting for the target are capable of complying with formation constraints to improve the quality of the shot. The safety of our method has also been proved through experiments in the presence of multiple obstacles.

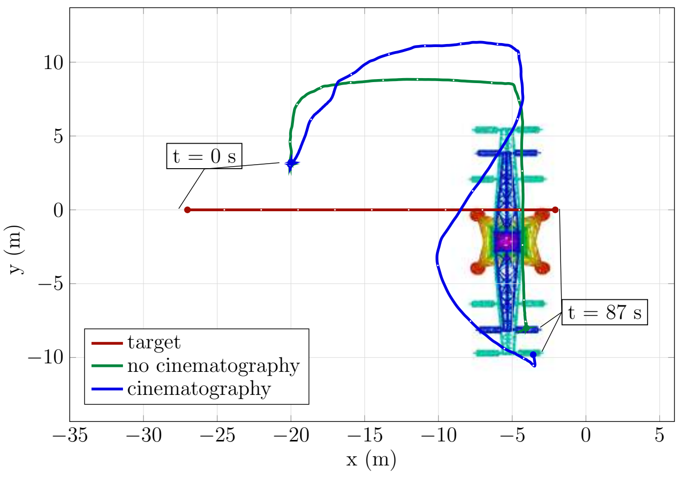
5.5.1 Experimental setup

We implemented our architecture described in Section 5.3 in C++ using the ROS framework. The ACADO Toolkit (Houska et al., 2011) was used to solve the optimization problems. We conducted SITL simulations using Gazebo to simulate the physics and to equip the UAVs with a camera and lights. To solve the optimization problems, a horizon length of 8 s and a time step of 0.2 s were chosen. The cinematographic parameters were set to $\psi_d = 6°$ and $q_{\mathrm{z,min}} = 0.5\,\mathrm{m}$.

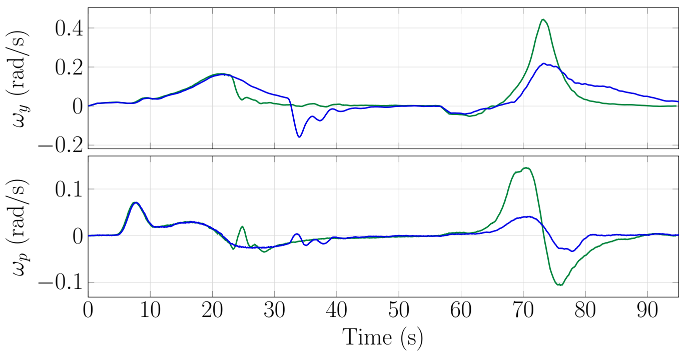
5.5.2 Simulation experiment – cinematography trajectories

The objective of this experiment was twofold: to demonstrate how the method computes smoother camera trajectories for the leader UAV while complying with cinematographic aspects, and how the trajectories of the followers maintain the formation constraints to light the target properly. We simulated a human worker performing a maintenance operation on a transmission tower while being monitored by a team of three UAVs (one filming and two lighting the worker). While the worker approached and climbed the tower, the system was commanded to perform a lateral shot followed by a sequence of flyby shots.

The flyby shots were selected to film the operation as they impose relative motion between the camera and the target. This feature is regarded as richer from a cinematographic point of view. We further demonstrate how our method is able to execute these relative movements more aesthetically than with a baseline approach where the specific term to smooth variations in the camera angles has been removed (i.e., $\alpha_1 = 0$ in 5.3). Figure 5.7 compares the trajectories for the camera-carrying UAV generated with both our method and the baseline approach. It can be seen that the baseline approach generates straight trajectories, whereas our method results in orbital trajectories, which are known to produce more pleasing video footage. Additionally, our method reduces the camera angular rates (Figure 5.7) and jerk. Note that in



(a)



(b)

Figure 5.7: Trajectories for the camera-carrying UAV while monitoring a worker on a transmission tower. For simplicity, only the lateral shot and the first flyby shot are shown. We compare the trajectories generated by our method (blue) with those from a baseline approach without cinematographic costs (green). The upper image displays a top view of the UAV's and target's trajectories. The small white dots on the trajectories depict transition points sampled every 5 s to give a notion of the speed. The bottom image depicts the evolution of the camera angular rates.

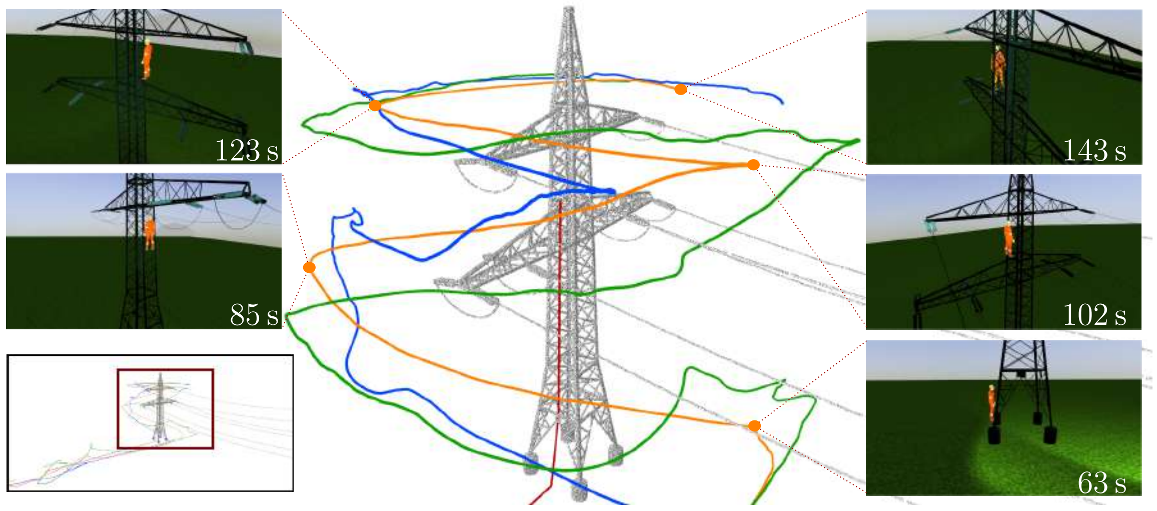


Figure 5.8: Illustration of the experiment where an operator is filmed working on a transmission tower. The trajectories of the camera-carrying leader (orange), both followers carrying lights (blue and green), and the human worker (red) are shown. The obstacle map is represented by a point cloud, including the power lines and tower. The worker is tracked with a lateral shot as he walks to the tower and then with a sequence of flyby shots while he climbs up. Several onboard images taken during the experiment are also shown.

aerial cinematography literature, camera motion jerk (third derivative of the angles) has been identified as a key aspect for shot quality (Bonatti et al., 2020; Gebhardt et al., 2016). We measured the root mean square of the jerk of φ and ξ along the full trajectories and obtained $0.0161\,\mathrm{rad\,s^{-3}}$ and $0.0057\,\mathrm{rad\,s^{-3}}$, respectively, for our method; and $0.0184\,\mathrm{rad\,s^{-3}}$ and $0.0104\,\mathrm{rad\,s^{-3}}$, respectively, for the baseline without the cinematographic cost term.

Figure 5.8 shows the trajectories followed by the whole UAV formation throughout the experiment to film the maintenance operation. It can be seen that the formation is properly maintained to avoid collisions between the UAVs and the tower, and to provide the required lighting of the filmed object. Moreover, none of the UAVs appear in the camera's field of view. A video of the complete simulation can be found at `https://www.youtube.com/watch?v=v4f7kb_fxjA`.

5.5.3 Simulation experiment – cluttered environment

The aim of this experiment was to demonstrate the performance of our method for trajectory planning in a cluttered environment while assessing its scalability with numerous obstacles. We simulated a forest-like scenario with multiple trees as obstacles. As a human target walks through the forest, the filming UAV executes a chase shot from behind while the lighting UAVs follow the leader side by side. Figure 5.9 depicts the distribution of the obstacles around the forest and the trajectories generated for the UAVs. In this figure, it can be seen that the UAVs were able to follow the human in formation and to simultaneously avoid obstacles.

Lastly, we analyze the scalability of our method in terms of computational demand. Simulations were run with a 4-core Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz. Table 5.1 shows the results of our method corresponding to the total planning time for each iteration that was run on the leader UAV. As expected, most time was spent in the non-convex optimization step described in Section 5.4.2. The results for the followers are not included because they skip this non-convex optimization and thus consume less time. The results are similar for the two experiments, although the second scenario was significantly more cluttered.

Since the map of the environment is transformed into safe corridors composed of convex polyhedrons, cluttered environments do not represent a significant increase in the computational demands of the trajectory optimization method. Therefore, we are able to plan the leader's trajectories at a rate of 1 Hz with horizon lengths of 8 s. This rate is adequate for real-time performance in the dynamic scenarios that we target. The lower computational complexity required to generate the initial trajectories of the followers allows us to plan follower trajectories at a higher rate of 2 Hz, enabling faster reactions to changes in the leader's behaviour and thus a more efficient mutual collision avoidance.

	Time (s)			
	Total $(Avg \pm std)$	ITG (%)	SCG (%)	FTO (%)
Tower	0.70923 ± 0.10557	70.9982	11.81564	17.18615
Forest	0.71274 ± 0.05792	72.41338	8.77989	18.80673

Table 5.1: The planning times of our method per iteration. The total average values are shown for the two experiments, followed by percentage of time consumed at each step. ITG stands for the procedure indicated in Section 5.4.2, SCG for the procedure described in Section 5.4.3, and FTO for the trajectory optimization described in Section 5.4.4.

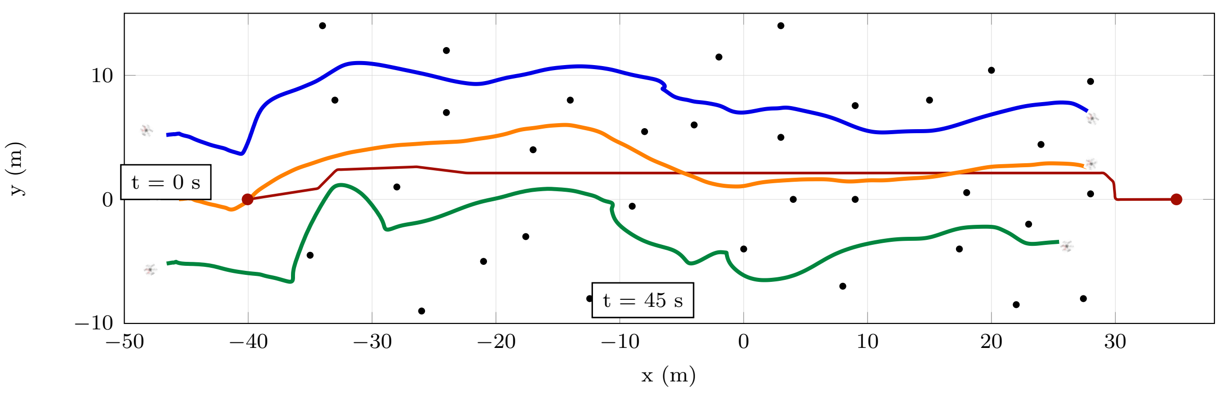


Figure 5.9: A top view of the trajectories generated in the cluttered forest scenario. The trajectories of the target (red), the leader (orange), and the two light-carrying UAVs (blue and green) are shown. The trees are represented by black dots.

5.6 Conclusion

This chapter has presented a method for autonomous aerial cinematography with distributed lighting by a team of UAVs. We have proposed a novel methodology for multi-UAV trajectory planning, addressing non-linear cinematographic aspects and obstacle avoidance in separate optimization steps. We have demonstrated that the method is capable of generating smooth trajectories complying with aesthetic objectives for the filming UAV, and trajectories for the follower UAVs that allow them to maintain a formation while lighting the target properly and staying out of the camera FoV. Our results indicate that we can plan trajectories in a distributed and online manner, and that the method is suitable for obstacle avoidance even in cluttered environments. In future work, we plan to address occlusions caused by obstacles within the camera FoV. Our idea is to compute the regions where these occlusions would take place and include them in the representation of the occupied space.

Chapter 6

Conclusions and future work

This chapter summarizes the main contributions and results of this thesis, drawing overall conclusions about the work. It then identifies interesting research directions for future work.

6.1 Conclusions

The use of teams with multiple UAVs is an important promising development for aerial filming, as a way of covering large outdoor scenarios offering different or supplementary views concurrently (or sequentially). However, the evolution of autonomous aerial cinematography is impossible without the development of methods to generate UAV trajectories that ensure compliance with system dynamics, smoothness requirements, and safety constraints. Recent advances in optimization-based techniques using numerical optimization algorithms enable the resolution of complex non-linear problems in milliseconds. This is the case for multi-UAV filming applications, where UAVs need to cope in real time with numerous constraints (including non-convex ones) such as UAV and camera dynamics, obstacle avoidance, inter-UAV collisions, and mutual UAV visibility.

These reasons motivated the work presented in this thesis. In particular, we have developed a set of algorithms for optimal trajectory planning in multi-UAV filming applications, and a complete framework for executing cinematography missions with multiple UAVs. The published contributions have pushed the frontiers of the current state of the art in multi-UAV filming applications by introducing the following:

- A parametric way to define aerial shots, accounting for different types of camera motion and target actors in the scene. We implemented a set of canonical shots in a multi-UAV system for cinematography mission execution, and proposed a distributed scheduling procedure to execute those missions, coping robustly with UAV failures.

- A novel method for optimal trajectory planning with a team of UAVs in cinematography applications. We formulated a new problem incorporating cinematography and safety constraints and then used numerical methods to solve the resulting non-linear optimization in a decentralized and online fashion.

- A new methodology for autonomous filming with distributing lighting by a team of UAVs. We proposed a method that addresses non-linear cinematographic aspects and obstacle avoidance in separate optimization steps. The method is scalable in terms of obstacle avoidance, and hence trajectory planning can be carried out in real time even in cluttered environments.

We would like to emphasize the experimental part of the thesis, especially complex in outdoor multi-UAV applications. All the multi-UAV methods included have been tested in outdoor field experiments, dealing with GPS positioning inaccuracies, communications delays, and disturbances such as moderate wind. Overall, this thesis has been a long journey, along which a large variety of aspects that are inherent to multi-UAV trajectory planning have been tackled. In particular, we would like to centre the final discussion of our conclusions around the following important issues: (i) field experimentation, (ii) non-convex numerical optimization methods, and (iii) obstacle avoidance. We discuss each of these aspects in more detail.

Our diverse **field experiments** filming multiple activities in dynamic scenes showcased the feasibility of the developed framework to address outdoor cinematography with multiple UAVs. As a general conclusion, the feedback from the media experts involved in the projects was positive, as they found the number of shots and autonomous features implemented by the system. rich enough In particular, they found the possibility of combining virtual and actual targets to guide camera motion quite helpful. However, the media experts were also critical about the quality of the final footage produced. Autonomous UAVs are an interesting tool for aerial filming and cinematography, but the planning of smooth trajectories is not enough on its own to produce professional videostreams. The aesthetically pleasing views yielded by our methods needs to be complemented by hardware adequate for filming purposes. Thus, professional high-resolution cameras and stable gimbals are essential to obtain high-quality videos. These types of cameras used in media production are heavy equipment that is not easy to integrate with standard gimbals and UAVs. Onboard communication devices providing stable long-range links with a broad bandwidth also turned out to be key for the type of outdoor scenarios addressed. Therefore, we believe that the methods for trajectory planning in filming applications developed in this thesis could strongly benefit from their integration with custom-made UAVs for professional media applications.

Regarding the use of numerical methods for **trajectory optimization in non-convex formulations**, our results demonstrate that current software solvers are efficient enough to plan trajectories on board in real time for multi-UAV settings. Moreover, we can conclude that our particular formulation was able to produce trajectories that are smooth and reduce jerky camera motion. However, despite the computation capabilities of these software tools for non-linear optimization, they are not always robust enough. The underlying algorithms implemented to tackle non-convex optimization are quite sensitive to the initial conditions, and small variations in the contour conditions may lead to different solutions, or to failures and non-convergence. Therefore, it would be interesting to devise new methods that could establish certain guarantees for the computed solutions, avoiding significant jumps between consecutive trajectories obtained in receding horizon and ensuring convergence under certain conditions.

In terms of **obstacle avoidance**, the idea of building safe corridors that can be transformed into convex constraints for trajectory optimization turns out to be promising. Our results demonstrated that the technique implemented behaved in a scalable manner and was able to cope with cluttered environments. However, our tests were always performed with preloaded maps and not in highly dynamic scenarios. Therefore, we believe that additional effort and testing is necessary to evaluate this methodology integrated within a pipeline that includes online mapping and obstacle detection.

6.2 Future work

The outcome of this thesis suggested future research directions, which could further advance the state of the art in multi-UAV filming applications. Some of them are straightforward extensions of the work done in the thesis, while others focus on the multi-UAV trajectory planning problem from a more general perspective.

Additional evaluation. Our media experts, having evaluated the shots, asked for the videos to be produced by the UAVs to be more stable. The heavy equipment used for filming was not easy to integrate with standard gimbals. Thus, additional field experiments with custom-made UAVs for professional media applications could be carried out, providing high-quality videos to the media experts and facilitating the evaluation of shots. Moreover, we encountered difficulties in evaluating the artistic component of the shots, given the subjectivity evaluating aesthetics. Throughout the thesis, we applied metrics used in the literature to evaluate the smoothness of the trajectories and the pleasing qualitites of the shots. However, the artistic principle of filming has a subjective component that depends on the viewer. Additional subjective user studies could be carried out to better evaluate the artistic possibilities of the system combining multi-camera shots.

Actor occlusion. In future work, actor occlusions could be addressed by computing regions where these occlusions would take place and adding them as no-fly zones. This would make sense in scenarios where few obstacles might occlude the target, since in scenarios with many obstacles, either occlusion would be unavoidable or its avoidance would severely affect the camera trajectories from a cinematographic point of view.

Uncertainties. We dealt with model and environmental uncertainties by generating trajectories at a high frequency in a receding horizon scheme. Although we employed GPS in our experiments, it would be interesting to further investigate how the methods proposed in this thesis could be adapted to account for more complex sensors and their uncertainties, such as depth cameras or laser-based sensors. A possible line of research could be the formulation of a trajectory planning problem that minimizes a type of cost related to positioning uncertainty; e.g., minimizing the distance to certain landmarks so that the positioning sensors can perform better. Moreover, in multi-UAV teams, the uncertainties associated with inter-vehicle communication could be modeled and integrated into the optimization problem, to pursue UAV formation shapes that optimize the communication throughput.

Collision avoidance. Our methods for obstacle avoidance can be used in dynamic scenarios, as our trajectories adapt online thanks to the use of receding horizon planning. However, our computation of safe corridors may be conservative and problematic in highly dynamic scenarios, since the convex decomposition of the free space becomes harder to compute due to the extra time dimension. An alternative method has been presented in Tordesillas et al. (2021). They deal with dynamic obstacles by calculating outer polyhedral representations of every interval of the trajectories, and then including the plane that separates each pair of polyhedrons as a decision variable in the optimization problem. This method looks promising for integrations with trajectory planning in filming applications.

New decentralized planning schemes. Alternative schemes for decentralized multi-UAV coordination could be studied instead of our priority-based planning. The objective would be to compute in a distributed manner multi-UAV approximate solutions that are close enough to the optimum, but without significantly increasing the computation time. Fully decentralized methods based on sequential consensus Nägeli et al. (2017b) may be worth exploring. We also believe that a comparison with methods based on reinforcement learning could be of high interest.

Appendix A

Open-source code repositories

All the methods developed throughout this thesis have been released as open-source code by means of online *GitHub* repositories. In this Appendix, we compile a list of all these repositories with their main content and their relation with the different thesis chapters. The complete list of repositories with code generated as a result of the thesis is the following:

- Multidrone Planning.
  `https://github.com/grvcTeam/multidrone_planning`
- Cinematography Trajectory Planner.
  `https://github.com/alfalcmar/optimal_navigation`
- Shot Executor.
  `https://github.com/grvcTeam/shot_executor`
- Safe Corridor Generator.
  `https://github.com/alfalcmar/safe_corridor_generator`
- Trajectory Follower.
  `https://github.com/alfalcmar/trajectory_follower`

In addition, some of the above repositories have dependencies with UAL `https://github.com/grvcTeam/grvc-ual`, a library created by some members of the Group

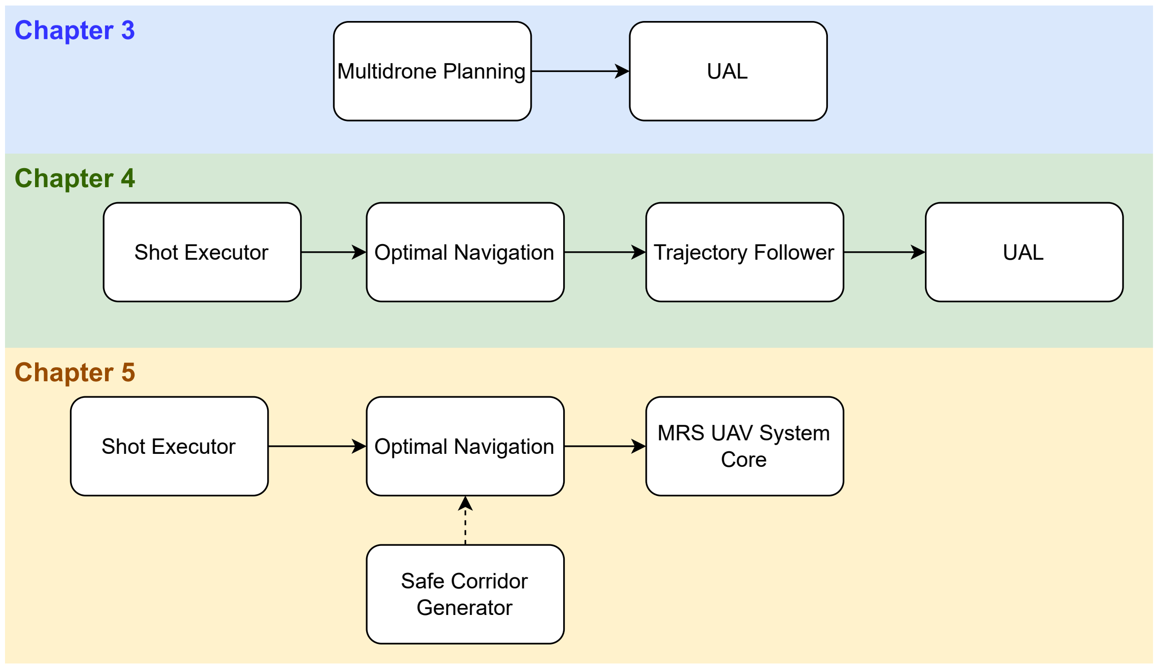


Figure A.1: Interconnections between the different repositories. Each module represents a code repository. Solid lines imply connections based on ROS topics and services, whereas dashed lines refer to connections made by library headers. In Chapter 3 and Chapter 4, UAL is used as UAV navigation framework, while the MRS UAV System is used in Chapter 5.

of Robotics, Vision, and Control (GRVC) of the University of Seville. It consists of an interface to abstract the user from the hardware specifics of each autopilot, as a way of easing UAV navigation. Instead, to integrate the work in Chapter 5, the MRS UAV System `https://github.com/ctu-mrs/uav_core` is used for trajectory tracking and UAV navigation. This repository was developed by the Multi-robot Systems Group at the Czech Technical University in Prague. It should be noted that all the code in the thesis is ROS compatible, using interfaces based on ROS topics and services for inter-process communication.

Figure A.1 shows a diagram of the interconnection between the listed repositories and their relationship to the chapters of this thesis. In the following, we provide more details about the content of each specific repository.

A.1 Multidrone Planning

This repository contains the software implementation of the architecture for multi-UAV cinematography presented in Chapter 3 and published in Alcántara et al. (2020). As this work was developed within the framework of the MultiDrone project, the repository also contains some additional components related to the project, such as those related with mission planning (Caraballo et al., 2020). The following software packages are included in this repository:

- **Dashboard Interface**. A dummy implementation of the Director's Dashboard (Montes-Romero et al., 2020) (the graphical tool to design missions in the MultiDrone project, which was not released as open-source code). It contains XML files with mission examples to be sent to the Mission Controller.
- **Multidrone Planning**. Contains the Mission Controller and the High-Level Planner. The Mission Controller receives cinematography missions from the Director's Dashboard in XML format, and computes mission plans using the High-Level Planner, sending the corresponding tasks to each UAV while monitoring the overall mission execution.
- **Onboard Scheduler**. Runs on board each UAV and receives the respective UAV's list of actions from the Mission Controller. It is then in charge of executing them sequentially, synchronizing their start and finish, and calling the Action Executor for the actual execution of the different navigation or shooting actions.
- **Action Executor**. Runs on board each UAV and is responsible for executing the navigation and shooting actions as they are received from the Onboard Scheduler. It computes reference trajectories for the UAV and takes care of the camera gimbal motion, taking into account the type of action and the target position estimation.
- **Gimbal Camera Interface**. Implements the communication bridge between the Action Executor and the gimbal hardware.

- **Global Tracker**. Provides position estimates for the targets in the scene, fusing visual and GPS measurements.
- **Multidrone Simulator**. Contains a set of GAZEBO models to simulate missions in the context of the MultiDrone project.
- **Multidrone Visualizer**. Uses a set of RViz plugins to implement a detailed visualizer for simulating and testing multi-UAV cinematography missions.

All the modules listed above are implemented as ROS nodes. The use of nodes in ROS provides several benefits to the overall system. There is additional tolerance to failures, as crashes are isolated to individual nodes. Also, code complexity is reduced and implementation details are well hidden. A detailed description of the modules involved, the interfaces, and ROS use can be found in the wiki of the repository `https://github.com/grvcTeam/multidrone_planning/wiki`.

A.2 Cinematography Trajectory Planner

This repository implements the UAV trajectory planning methods developed in Chapter 4 and Chapter 5, to calculate optimal cinematography trajectories. The repository includes two different formulations of the optimization problem. One is the formulation presented in Alcántara et al. (2021), which was tackled using the Forces Pro solver. The other is the formulation presented in Krátký et al. (2021), which used the ACADO solver. The repository also contains the interfaces to connect with the Trajectory Follower repository and the MRS UAV System Core (used in Chapter 4 and Chapter 5 for trajectory tracking, respectively).

A.3 Shot Executor

This repository contains the implementation of the Shot Executer, which is a specific version of the Action Executer module included in the Multidrone Planning repository. This version was created to integrate the methods presented in Chapter 4 and Chapter 5 within the Action Executer. The Shot Executer is used in Alcántara et al. (2021); Krátký et al. (2021) to compute the desired position references for the UAV, so that it carries out the specified shot. These desired positions are computed using the current position of the UAV, the shot type, and the target position estimate. They are then fed to the optimal cinematography trajectory planner. This module is implemented as a ROS node that offers an interface to be connected to the trajectory planning modules by means of ROS topics.

A.4 Safe Corridor Generator

This repository contains the implementation of the generator of safe corridors used for collision avoidance in the method presented in Chapter 5 and published in Krátký et al. (2021). An example of the use of this safe corridor generator library can be found in the Cinematography Trajectory Planner repository.

In particular, the repository implements a ROS library that generates a safe corridor for the input trajectory. We iterate over each waypoint of the input trajectory and add it directly if it is collision-free. Otherwise, we try to find an alternative collision-free path to be appended, and continue iterating. For that alternative path, we use the Jump Point Search (JPS) algorithm presented in Liu et al. (2017). The JPS code is also available in the third party folder included in the repository.

A.5 Trajectory Follower

This repository contains a package for UAV trajectory following, which is used in Chapter 4 for the field experiments presented in Alcántara et al. (2021). This trajectory follower is a simplified version of a 'Carrot Chasing' algorihm (Perez-Leon et al., 2020). The original software package, including additional functionalities such as trajectory interpolation and path following, can also be found online `https://github.com/hecperleo/upat_follower`.

The Trajectory Follower consists of an ROS node that receives trajectories in the `trajectory_to_follow` topic. Then it computes and sends the proper velocity commands to the UAL library, in order to track the instructed trajectory.

Bibliography

3DR (2015). Solo Drone. `https://www.3dr.com/company/about-3dr/solo/`.

Alcántara, A., Capitán, J., Cunha, R., and Ollero, A. (2021). Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems*, 140:103778.

Alcántara, A., Capitán, J., Torres-González, A., Cunha, R., and Ollero, A. (2020). Autonomous execution of cinematographic shots with multiple drones. *IEEE Access*, 8:201300–201316.

Alonso-Mora, J., Baker, S., and Rus, D. (2015a). Multi-robot navigation in formation via sequential convex programming. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4634–4641.

Alonso-Mora, J., Beardsley, P., and Siegwart, R. (2018). Cooperative Collision Avoidance for Nonholonomic Robots. *IEEE Transactions on Robotics*, 34(2):404–420.

Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., and Siegwart, R. (2013). *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*, pages 203–216. Springer Berlin Heidelberg, Berlin, Heidelberg.

Alonso-Mora, J., Montijano, E., Schwager, M., and Rus, D. (2016). Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5356–5363.

Alonso-Mora, J., Naegeli, T., Siegwart, R., and Beardsley, P. (2015b). Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39(1):101–121.

Alotaibi, E. T., Alqefari, S. S., and Koubaa, A. (2019). LSAR: Multi-UAV collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832.

Ascher, U. M., Ruuth, S. J., and Spiteri, R. J. (1997). Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151–167.

Augugliaro, F., Schoellig, A. P., and D'Andrea, R. (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1917–1922.

Baca, T., Hert, D., Loianno, G., Saska, M., and Kumar, V. (2018). Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6753–6760.

Báča, T., Petrlik, M., Vrba, M., Spurny, V., Penicka, R., Hert, D., and Saska, M. (2021). The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 26.

Bazaraa, M. S. (2013). *Nonlinear Programming: Theory and Algorithms.* Wiley Publishing, 3rd edition.

Benjumea, D., Alcántara, A., Ramos, A., Torres-Gonzalez, A., Sánchez-Cuevas, P., Capitan, J., Heredia, G., and Ollero, A. (2021). Localization system for lightweight unmanned aerial vehicles in inspection tasks. *Sensors*, 21(17).

Betts, J. T. (2009). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming.* Cambridge University Press, USA, 2nd edition.

Bianco, N., Bertolazzi, E., Biral, F., and Massaro, M. (2018). Comparison of direct and indirect methods for minimum lap time optimal control problems. *Vehicle System Dynamics*, 57:1–32.

Bock, H. and Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems*. *IFAC Proceedings Volumes*, 17(2):1603–1608.

Boggs, P. and Tolle, J. (1995). Sequential quadratic programming. *Acta Numerica*, 4:1–51.

Bonatti, R., Ho, C., Wang, W., Choudhury, S., and Scherer, S. (2019). Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 229–236.

Bonatti, R., Wang, W., Ho, C., Ahuja, A., Gschwindt, M., Camci, E., Kayacan, E., Choudhury, S., and Scherer, S. (2020). Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 37(4):606–641.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundatinos and Trends in Maching Learning*, 8(3–4):231–357.

Bucker, A., Bonatti, R., and Scherer, S. (2021). Do You See What I See? Coordinating Multiple Aerial Cameras for Robot Cinematography. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7972–7979.

Caraballo, L.-E., Montes-Romero, Á., Díaz-Báñez, J.-M., Capitán, J., Torres-González, A., and Ollero, A. (2020). Autonomous planning for multiple aerial cinematographers. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1509–1515.

Carlos, B. B., Sartor, T., Zanelli, A., Frison, G., Burgard, W., Diehl, M., and Oriolo, G. (2020). An Efficient Real-Time NMPC for Quadrotor Position Control under Communication Time-Delay. In *IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 982–989.

Castillo-Lopez, M., Sajadi-Alamdari, S. A., Sanchez-Lopez, J. L., Olivares-Mendez, M. A., and Voos, H. (2018). Model Predictive Control for Aerial Collision Avoidance in Dynamic Environments. In *Mediterranean Conference on Control and Automation (MED)*, pages 198–203.

Cesare, K., Skeele, R., Yoo, S.-H., Zhang, Y., and Hollinger, G. (2015). Multi-UAV exploration with limited communication and battery. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2230–2235.

Cheng, H., Zhu, Q., Liu, Z., Xu, T., and Lin, L. (2017). Decentralized navigation of multiple agents based on ORCA and model predictive control. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3446–3451.

Christie, M., Olivier, P., and Normand, J. M. (2008). Camera control in computer graphics. *Computer Graphics Forum*, 27(8):2197–2218.

Cunha, R., Malaca, M., Sampaio, V., Guerreiro, B., Nousi, P., Mademlis, I., Tefas, A., and Pitas, I. (2019). Gimbal Control for Vision-based Target Tracking. In *EUSIPCO. Satellite Workshop on Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, La Coruña, Spain.

Deits, R. and Tedrake, R. (2015). Efficient mixed-integer planning for UAVs in cluttered environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49.

Diehl, M., Bock, H., Diedam, H., and Wieber, P.-B. (2006). *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*, pages 65–93. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ding, W., Gao, W., Wang, K., and Shen, S. (2019). An Efficient B-Spline-Based Kinodynamic Replanning Framework for Quadrotors. *IEEE Transactions on Robotics*, 35(6):1287–1306.

DJI (2018). Mavic Pro 2. `https://www.dji.com/es/mavic`.

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

Erunsal, I. K., Ventura, R., and Martinoli, A. (2019). Nonlinear Model Predictive Control for 3D Formation of Multirotor Micro Aerial Vehicles with Relative Sensing in Local Coordinates. In *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*.

Faria, M., Marín, R., Popović, M., Maza, I., and Viguria, A. (2019). Efficient Lazy Theta* Path Planning over a Sparse Grid to Explore Large 3D Volumes with a Multirotor UAV. *Sensors*, 19(1).

Ferrera, E., Alcántara, A., Capitán, J., Castaño, A. R., Marrón, P. J., and Ollero, A. (2018). Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors*, 18(12).

FreeSkies (2019). FreeSkies CoPilot. `http://freeskies.co/`.

Galvane, Q., Fleureau, J., Tariolle, F.-L., and Guillotel, P. (2016). Automated Cinematography with Unmanned Aerial Vehicles. In *Eurographics Workshop on Intelligent Cinematography and Editing*.

Galvane, Q., Lino, C., Christie, M., Fleureau, J., Servant, F., Tariolle, F.-l., and Guillotel, P. (2018). Directing cinematographic drones. *ACM Transactions on Graphics*, 37(3).

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074.

Garage, A. (2019). Skywand. `https://skywand.com/`.

Gebhardt, C., Hepp, B., Nägeli, T., Stevšić, S., and Hilliges, O. (2016). Airways: Optimization-Based Planning of Quadrotor Trajectories according to High-Level User Goals. In *Conference on Human Factors in Computing Systems (CHI)*, pages 2508–2519.

Gill, P. E., Murray, W., and Saunders, M. A. (2005). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131.

Gopalakrishnan, B., Singh, A. K., Kaushik, M., Krishna, K. M., and Manocha, D. (2017). PRVO: Probabilistic Reciprocal Velocity Obstacle for multi robot navigation under uncertainty. In *IEEE International Conference on Intelligent Robots and Systems(IROS)*, pages 1089–1096.

Gschwindt, M., Camci, E., Bonatti, R., Wang, W., Kayacan, E., and Scherer, S. (2019). Can a robot become a movie director? learning artistic principles for aerial cinematography. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1107–1114.

Hairer, E., Norsett, S., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. Springer.

Hall, B. (2015). *Understanding cinematography.* Crowood.

Harabor, D. and Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. In *AAAI Conference on Artificial Intelligence*, page 1114–1119.

Harabor, D. and Grastien, A. (2014). Improving jump point search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 24.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Ho, C., Jong, A., Freeman, H., Rao, R., Bonatti, R., and Scherer, S. (2021). 3D Human Reconstruction in the Wild with Collaborative Aerial Cameras. In *IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 5263–5269.

Houska, B., Ferreau, H. J., and Diehl, M. (2011). ACADO toolkit - An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312.

Huang, C., Gao, F., Pan, J., Yang, Z., Qiu, W., Chen, P., Yang, X., Shen, S., and Cheng, K. T. (2018). ACT: An Autonomous Drone Cinematography System for Action Scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7039–7046.

Huang, C., Yang, Z., Kong, Y., Chen, P., Yang, X., and Cheng, K.-T. (2019). Learning to capture a film-look video with a camera drone. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1871–1877.

Janson, L., Schmerling, E., Clark, A., and Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921.

Joubert, N., E, J. L., Goldman, D. B., Berthouzoz, F., Roberts, M., Landay, J. A., and Hanrahan, P. (2016). Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *ArXiv e-prints*.

Joubert, N., Roberts, M., Truong, A., Berthouzoz, F., and Hanrahan, P. (2015). An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics*, 34(6):1–11.

Kamel, M., Alonso-Mora, J., Siegwart, R., and Nieto, J. (2017). Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

Karloff, H. (1991). *The Simplex Algorithm*, pages 23–47. Birkhäuser Boston, Boston, MA.

Koenig, N. and Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2149–2154.

Kostadinov, D. and Scaramuzza, D. (2020). Online Weight-adaptive Nonlinear Model Predictive Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6.

Krátký, V., Alcántara, A., Capitán, J., Štěpán, P., Saska, M., and Ollero, A. (2021). Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 6(4):7580–7587.

Krátký, V., Petráček, P., Spurný, V., and Saska, M. (2020). Autonomous reflectance transformation imaging by a team of unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 5(2):2302–2309.

Krishnan, S., Rajagopalan, G. A., Kandhasamy, S., and Shanmugavel, M. (2020). Continuous-time trajectory optimization for decentralized multi-robot navigation. *IFAC-PapersOnLine*, 53(1):494–499.

Landry, B., Deits, R., Florence, P. R., and Tedrake, R. (2016). Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1469–1475.

LaValle, S. and Kuffner, J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Workshop on Algorithmic Foundations on Robotics*.

Lee, T. (2013). Robust adaptive attitude tracking on SO(3) with an application to a quadrotor UAV. *IEEE Transactions on Control Systems Technology*, 21(5):1924–1930.

Li, J., Ran, M., and Xie, L. (2021). Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization. *IEEE Robotics and Automation Letters*, 6(2):405–412.

Lindqvist, B., Mansouri, S. S., Agha-Mohammadi, A. A., and Nikolakopoulos, G. (2020). Nonlinear MPC for Collision Avoidance and Control of UAVs with Dynamic Obstacles. *IEEE Robotics and Automation Letters*, 5(4):6001–6008.

Lino, C. and Christie, M. (2015). Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics*, 34(4):82:1–82:12.

Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C. J., and Kumar, V. (2017). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695.

Liu, Y. and Bucknall, R. (2018). A survey of formation control and motion planning of multiple unmanned vehicles. *Robotica*, 36:1019 – 1047.

Loianno, G., Brunner, C., McGrath, G., and Kumar, V. (2017). Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411.

Luis, C. E. and Schoellig, A. P. (2019). Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control. *IEEE Robotics and Automation Letters*, 4(2):375–382.

Mademlis, I., Mygdalis, V., Nikolaidis, N., Montagnuolo, M., Negro, F., Messina, A., and Pitas, I. (2019). High-level multiple-uav cinematography tools for covering outdoor events. *IEEE Transactions on Broadcasting*, 65(3):627–635.

Mademlis, I., Nikolaidis, N., Tefas, A., Pitas, I., Wagner, T., and Messina, A. (2019a). Autonomous uav cinematography: A tutorial and a formalized shot-type taxonomy. *ACM Computing Surveys*, 52(5).

Mademlis, I., Nousi, P., Barz, C. L., Goncalves, T., and Pitas, I. (2019b). Communications for Autonomous Unmanned Aerial Vehicle Fleets in Outdoor Cinematography Applications. In *EUSIPCO. Satellite Workshop on Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, La Coruña, Spain.

Mattingley, J. and Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27.

Maza, I., Capitan, J., Merino, L., and Ollero, A. (2015). *Multi-UAV Cooperation*, pages 1–10. John Wiley & Sons.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525.

Mohta, K., Watterson, M., Mulgaonkar, Y., Liu, S., Qu, C., Makineni, A., Saulnier, K., Sun, K., Zhu, A., Delmerico, J., Karydis, K., Atanasov, N., Loianno, G., Scaramuzza, D., Daniilidis, K., Taylor, C. J., and Kumar, V. (2018). Fast, autonomous flight in GPS-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120.

Mondal, A., Bhowmick, C., Behera, L., and Jamshidi, M. (2018). Trajectory tracking by multiple agents in formation with collision avoidance and connectivity assurance. *IEEE Systems Journal*, 12(3):2449–2460.

Montes-Romero, Á., Torres-González, A., Capitán, J., Montagnuolo, M., Metta, S., Negro, F., Messina, A., and Ollero, A. (2020). Director tools for autonomous media production with a team of drones. *Applied Sciences*, 10(4).

Moulard, T., Chretien, B., and Yoshida, E. (2014). Software Tools for Nonlinear Optimization. *Journal of the Robotics Society of Japan*, 32(6):536–541.

MultiDrone-Consortium (2019). Deliverable D6.3: Experimental Demonstration in Media Production. `https://multidrone.eu/wp-content/uploads/2020/01/D6.3_Experimental-Demonstration-in-Media-Production.pdf`.

Nash, A., Daniel, K., Koenig, S., and Feiner, A. (2007). Theta*: Any-angle path planning on grids. In *National Conference on Artificial Intelligence*, pages 1177–1183.

Nash, A., Koenig, S., and Tovey, C. (2010). Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D. In *Symposium on Combinatorial Search (SoCS)*.

Natalizio, E., Zema, N., Yanmaz, E., Di Puglia Pugliese, L., and Guerriero, F. (2019). Take the Field from your Smartphone: Leveraging UAVs for Event Filming. *IEEE Transactions on Mobile Computing*, pages 1–1.

Nousi, P., Triantafyllidou, D., Tefas, A., and Pitas, I. (2019). Joint lightweight object tracking and detection for unmanned vehicles. In *IEEE International Conference on Image Processing (ICIP)*, pages 160–164.

Nousi, P., Triantafyllidou, D., Tefas, A., and Pitas, I. (2020). Re-identification framework for long term visual object tracking based on object detection and classification. *Signal Processing: Image Communication*, 88:1–11.

Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D., and Hilliges, O. (2017a). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, 2(3):1696–1703.

Nägeli, T., Meier, L., Domahidi, A., Alonso-Mora, J., and Hilliges, O. (2017b). Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics*, 36(4):1–10.

Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., and Galceran, E. (2016). Continuous-time trajectory optimization for online uav replanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339.

Oleynikova, H., Lanegger, C., Taylor, Z., Pantic, M., Millane, A., Siegwart, R., and Nieto, J. (2020). An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. *Journal of Field Robotics*, 37(4):642–666.

Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J. (2018). Safe Local Exploration for Replanning in Cluttered Unknown Environments for Microaerial Vehicles. *IEEE Robotics and Automation Letters*, 3(3):1474–1481.

Park, J. and Kim, H. J. (2021). Online Trajectory Planning for Multiple Quadrotors in Dynamic Environments Using Relative Safe Flight Corridor. *IEEE Robotics and Automation Letters*, 6(2):659–666.

Pereira, P. O., Cunha, R., Cabecinhas, D., Silvestre, C., and Oliveira, P. (2017). Leader following trajectory planning: A trailer-like approach. *Automatica*, 75:77–87.

Perez-Grau, F. J., Ragel, R., Caballero, F., Viguria, A., and Ollero, A. (2018). An architecture for robust UAV navigation in GPS-denied areas. *Journal of Field Robotics*, 35(1):121–145.

Perez-Leon, H., Acevedo, J. J., Maza, I., and Ollero, A. (2020). A 4D trajectory follower based on the 'Carrot chasing' algorithm for UAS within the U-space context. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1860–1867.

Petracek, P., Kratky, V., and Saska, M. (2020). Dronument: System for reliable deployment of micro aerial vehicles in dark areas of large historical monuments. *IEEE Robotics and Automation Letters*, 5(2):2078–2085.

Petrlík, M., Báča, T., Heřt, D., Vrba, M., Krajník, T., and Saska, M. (2020). A robust UAV system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176.

Potra, F. A. and Wright, S. J. (2000). Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302.

Price, E., Lawless, G., Ludwig, R., Martinovic, I., Black, M., and Ahmad, A. (2018). Deep Neural Network-based Cooperative Visual Tracking through Multiple Micro Aerial Vehicles. *IEEE Robotics and Automation Letters*, 3(4):3193–3200.

Real, F., Castaño, A. R., Torres-González, A., Capitán, J., Sánchez-Cuevas, P. J., Fernández, M. J., Romero, H., and Ollero, A. (2021a). Autonomous fire-fighting with heterogeneous team of unmanned aerial vehicles. *Field Robotics*, 1:158–185.

Real, F., Castaño, A. R., Torres-González, A., Capitán, J., Sánchez-Cuevas, P. J., Fernández, M. J., Villar, M., and Ollero, A. (2021b). Experimental evaluation of a team of multiple unmanned aerial vehicles for cooperative construction. *IEEE Access*, 9:6817–6835.

Real, F., Torres-González, A., Soria, P. R., Capitán, J., and Ollero, A. (2020). Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, 17(4):1–13.

Richter, C., Bry, A., and Roy, N. (2016). Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In *International Symposium on Robotics Research (ISRR)*, pages 649–666.

Robinson, D. R., Mar, R. T., Estabridis, K., and Hewer, G. (2018). An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments. *IEEE Robotics and Automation Letters*, 3(2):1215–1222.

Sabetghadam, B., Alcántara, A., Capitán, J., Cunha, R., Ollero, A., and Pascoal, A. (2019). Optimal trajectory planning for autonomous drone cinematography. In *European Conference on Mobile Robots (ECMR)*, pages 1–7.

Saska, M., Krátký, V., Spurný, V., and Báča, T. (2017). Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8.

Scherer, J. and Rinner, B. (2020). Multi-UAV Surveillance With Minimum Information Idleness and Latency Constraints. *IEEE Robotics and Automation Letters*, 5(3):4812–4819.

Schittkowski, K. and Zillober, C. (1995). *Sequential Convex Programming Methods*, pages 123–141. Springer Berlin Heidelberg, Berlin, Heidelberg.

Skydio (2019). Skydio 2. `https://www.skydio.com/`.

Smith, C. (2016). *The Photographer's Guide to Drones.* Rocky Nook, Inc.

Spurný, V., Báča, T., Saska, M., Pěnička, R., Krajník, T., Thomas, J., Thakur, D., Loianno, G., and Kumar, V. (2019). Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles. *Journal of Field Robotics*, 36(1):125–148.

Tallamraju, R., Rajappa, S., Black, M. J., Karlapalem, K., and Ahmad, A. (2018). Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–8.

Tanner, H. G. and Kumar, A. (2005). Formation stabilization of multiple agents using decentralized navigation functions. In *Robotics: Science and Systems*, pages 49–56.

Topputo, F. and Zhang, C. (2014). Survey of direct transcription for low-thrust space trajectory optimization with applications. *Abstract and Applied Analysis*, 2014.

Tordesillas, J., Lopez, B. T., Everett, M., and How, J. P. (2021). FASTER: Fast and safe trajectory planner for navigation in unknown environments. *IEEE Transactions on Robotics*, pages 1–17.

Torrente, G., Kaufmann, E., Fohn, P., and Scaramuzza, D. (2021). Data-Driven MPC for Quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776.

Torres-Gonzalez, A., Alcantara, A., Sampaio, V., Capitan, J., Guerreiro, B., Cunha, R., and Ollero, A. (2019). Distributed Mission Execution for Aerial Cinematography with Multiple Drones. In *EUSIPCO. Satellite Workshop on Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, La Coruña, Spain.

Torres-González, A., Capitán, J., Cunha, R., Ollero, A., and Mademlis, I. (2017). A Multidrone Approach for Autonomous Cinematography Planning. In *Third Iberian Robotics Conference*, volume 693 of *Advances in Intelligent Systems and Computing*, pages 337–349.

van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935.

Vanderbei, R. J. and Shanno, D. F. (1999). An Interior-Point Algorithm for Nonconvex Nonlinear Programming. *Computational Optimization and Applications*, 13(1-3):231–252.

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., and Diehl, M. (2022). Acados: a Modular Open-Source Framework for Fast Embedded Optimal Control. *Mathematical Programming Computation*, 14:147–183.

Wu, C., Huang, X., Luo, Y., Leng, S., and Wu, F. (2020). An Improved Sparse Hierarchical Lazy Theta* Algorithm for UAV Real-Time Path Planning in Unknown Three-Dimensional Environment. In *IEEE International Conference on Communication Technology (ICCT)*, pages 673–677.

Xie, C., van den Berg, J., Patil, S., and Abbeel, P. (2015). Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4187–4194.

Yu, C., Wang, J., Shan, J., and Xin, M. (2016). Multi-UAV UWA video surveillance system. In *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6.

Yuneec International (2018). Typhoon H Plus. `https://us.yuneec.com/typhoon-h-overview`.

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2017). FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, pages 13–29.

Zhou, B., Pan, J., Gao, F., and Shen, S. (2021). RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 37(6):1992–2009.

Zhou, Z. J. and Yan, N. N. (2014). A survey of numerical methods for convection-diffusion optimal control problems. *Journal of Numerical Mathematics*, 22(1):61–85.

Zhu, H. and Alonso-Mora, J. (2019). Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments. *IEEE Robotics and Automation Letters*, 4(2):776–783.

Čáp, M., Novák, P., Kleiner, A., and Selecký, M. (2015). Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 12(3):835–849.