POLITECNICO DI MILANO
DEPARTMENT DEIB
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

PATH PLANNING FOR MULTICOPTER NETWORKS AND APPLICATIONS TO THE CONDITION MONITORING OF THE BUILT ENVIRONMENT

Doctoral Dissertation of:
Michele Bolognini

Supervisors:
Prof. Lorenzo Mario Fagiano
Prof. Maria Pina Limongelli

Tutor:
Prof. Simone Garatti

The Chair of the Doctoral Program:
Prof. Luigi Piroddi

Year 2022 – Cycle XXXV

Abstract

Unmanned Aerial Vehicles have been among the hottest technology trends of the last two decades. Their versatility is testified by their widespread adoption in several different fields. Within a rich research field, this thesis presents novel theoretical results and experimental applications of such technology in the inspection of the built environment. In the first part, the problem of path planning and coordination for Unmanned Aerial Vehicle networks is tackled both for tethered and non-tethered vehicles. In the latter case, several formulations and variations of the Traveling Salesman Problem exist, but none of them tackles the issue of robustness with respect to a single point of failure. We therefore offer a multi-agent version of the Traveling Salesman Problem, also known as Constrained Vehicle Routing Problem, where the ability of completing the planned mission in case of failure of one drone is guaranteed theoretically by the addition of suitable constraints. In the former case, the relatively newer introduction and narrower field of application of systems of tethered drones explains the lack of suitable path planners in literature. We present a real-time planning algorithm for such a system in an unknown environment, assuming vehicles are only equipped with 2D LiDARs, and then refine such planner to embed a Model Predictive Control formulation, endowing safety guarantees. We furthermore illustrate another planner for a human-collaborative tethered drone, based on a Riemanniann Motion Policy framework and test it in a real world spray painting scenario.

In the second part, two case studies on the application of drones to inspections are presented. While modal analysis is a known and thoroughly

tested set of techniques, it is mainly performed through accelerometer measurements or with cameras fixed with respect to the ground. Mounting the camera on a drone enables it to be positioned wherever it is most convenient, at the cost of introducing noise in the measurement. We present a methodology for performing vision-based modal analysis by synchronizing multiple footage sources, from cameras mounted on different drones. This enables the simultaneous measurement of entire sections of a structure. We also verify that the proposed measurement technique is accurate with and without markers present on the structure, provided that an adequate trade-off between camera resolution, oscillation amplitude and distance from the structure is kept. Finally, we present a study of the feasibility of automating the pipeline for energy efficiency assessment of a building, from the data collection planning to the data analysis phase. A drone with a dual (color and infrared) sensor is employed to survey a real-world building by taking pictures. These are then combined to obtain a 3D geometrical model of the appearance of the target building and to map onto it the energy-related information gathered through the infrared images, making the comparison between the building model and the measurements easy and facilitating the tracking of the evolution of its health.

Sommario

I velivoli senza pilota sono tra le tendenze più popolari nell'ultimo ventennio. La loro versatilità è testimoniata dal loro impiego in diversi ambiti. In un ambito di ricerca affermato e avanzato, questa tesi presenta innovativi risultati teorici e applicazioni sperimentali di questa tecnologia, applicata all'ispezione dell'ambiente costruito. Nella prima parte, si affronta il problema della pianificazione di percorso e di missione per una rete di velivoli, sia connessi da cavi sia non. Nel secondo caso, esistono già varie formulazioni del Problema del Commesso Viaggiatore, ma nessuna di esse tratta di robustezza, intesa come resilienza alla possibilità che uno dei droni vada improvvisamente fuori uso durante la missione. Pertanto, si offre una formulazione multi-agente del problema, anche noto come Constrained Vehicle Routing Problem, in cui la capacità di portare a termine la missione a fronte di un guasto è garantita teoricamente attraverso opportuni vincoli al problema di ottimizzazione. Nel primo caso, l'introduzione recente e l'applicazione ridotta dei sistemi di droni connessi via cavo è una concausa della mancanza in letteratura di metodi per la pianificazione di percorso per tali sistemi. Si presenta quindi un algoritmo per la pianificazione in tempo reale in ambiente ignoto, supponendo che i velivoli siano equipaggiati con sensori LiDAR bi-dimensionali, per poi raffinare l'approccio formulandolo come un problema di Model Predictive Control, che aggiunge garanzie di sicurezza. Si illustra inoltre un altro approccio alla pianificazione per un singolo drone cablato nel contesto di una collaborazione con un operatore umano. L'approccio è poi testato in un caso realistico.

Nella seconda parte, si presentano due casi studio dell'applicazione di

tali velivoli all'ispezione di edifici. L'analisi modale è una tecnica nota e approfondita, che però viene normalmente applicata sulla base di misure con accelerometri, oppure con videocamere fisse al terreno. Installare una videocamera su un drone consente di posizionarla idealmente, senza vincoli spaziali, seppure al costo di introdurre del rumore nella misura. Si presenta un metodo per applicare l'analisi modale basata su video sincronizzando diverse fonti, da camere montate su droni. Ciò facilita la misurazione simultanea di sezioni intere della struttura. Si verifica che la tecnica proposta produce misure accurate sia con marker sia senza. Infine, si illustra uno studio di fattibilità dell'automazione della procedura di verifica dell'efficienza energetica di un edificio, dalla fase di raccolta dei dati a quella di analisi. Si impiega un drone con doppio sensore (camera a colori e camera a infrarossi), combinando le fotografie ottenute per ottenere una rappresentazione 3D dell'edificio e mappare su di esso i difetti identificati attraverso le fotografie nello spettro infrarosso. Ciò facilita il confronto tra le misurazioni effettuate ed un modello esistente dell'edificio, aiutando a tracciare il suo stato di salute nel tempo.

Contents

CHAPTER 1

Introduction

1.1 Unmanned Aerial Vehicles in the Inspection Industry

Ever since the origin of Unmanned Aerial Vehicles (UAVs), commonly identified in an 1849 Austrian military project [1], humans have studied and developed them to achieve diverse objectives. Because it is a known and apparent fact that destroying is easier than building, it is easier and faster to develop technologies that help us do the former than the latter. Many technologies we use today were born for military applications: our personal computers descend from the Turing machine, which the famous mathematician Alan Turing developed for the British Government Code and Cypher School in a successful effort to crack the German Enigma cipher code [2]. UAVs are no exception: they were also initially mainly designed for wartime use, consisting of dropping explosives and supporting reconnaissance efforts [1], from their inception in the late eighteenhundreds to their deployment on today's battlegrounds: many people still associate the word "drone" to unmanned and remotely controlled military planes famously adopted, for example, by the US army in the Middle East.

Nevertheless, these technologies have been successfully employed in the civil sector. Development of ever smaller and cheaper UAV prototypes in-

tensified around the beginning of the last century, pushed by companies like Parrot and DJI [3, 4]. This opened the door to the applications of drones, defined here as small UAVs for civil use, in numerous sectors, including parcel delivery [5], precision agriculture [6, 7], forestry [8], firefighting [9], entertainment [10], architecture and civil engineering [11–15] and, above all, visual inspection of the built environment [1, 16–18]. This flourishing of applications was made possible by technological advancements that rendered rotary-wing drones stable enough to mount almost any lightweight sensor on them (infrared, LiDAR, RGB and thermal cameras, contact sensors, sonar, ...), and their batteries capable of supporting them for several minutes in flight. In the world of structural inspection, UAVs are often employed as flying cameras, enabling a single human operator to quickly assess, for example, the integrity of a roof or solar panel field, or the progress of a construction site. While human-piloted drones can make these kinds of inspections significantly faster and easier to perform, many research groups worldwide are pushing the limits of the technology to allow these aircraft to perform the same tasks autonomously. Many believe UAVs will become fundamental tools for monitoring and inspection tasks because they can revolutionize how such tasks are carried out, making surveys safer, cheaper, more repeatable, and more cost-effective. These improvements are generally achieved by reducing the reliance on trained human personnel (and therefore relaxing some of the necessary safety constraints) and automatically performing both data gathering and data analysis, minimizing human arbitrariness and bias. From a research perspective, this is today a very active field where some significant results have been achieved, like surface reconstruction and analysis [12] and the 3D modeling of both geometrical and energy aspects [13, 19]. In contrast, others are still sought after, such as a drone-enabled method for vision-based Structural Health Monitoring (SHM) of a real-world scale building [14, 20]. Presenting new contributions to the field is one of the aims of the present thesis. Furthermore, it hopes to illustrate the potential advantages of employing tethered, autonomous multi-copter networks, especially for recurrent inspections. In fact, while autonomous drones already constitute a significant step forward by reducing the cost of inspections, their tethered counterparts present even more features. Not only can they fly for virtually unlimited time, completely removing one of the main limits of drones, but the physical connection to a ground station also makes them inherently safer, by constraining the area where they could potentially fly or fall in the event of an accident. For example, a fixed tethered drone system could be installed on a bridge after the construction is over, and enable autonomous, repeatable and remote

sensing, while at the same time providing a real-time coverage of any unexpected behavior of or on the structure. A similar result can be obtained with portable systems: as long as there is electrical power available the system can operate, and transport is a small issue due to such systems weighing in the tens of kilograms. The inherent safety advantage is already recognized in the legal framework in many regions, which classify drones connected through any kind of physical harness as kites, a category of flying objects with significantly lower safety requirements.

It should finally be noted that the development and deployment of fully autonomous drones in general, while still technically challenging, is also slowed down by the legal framework in which research takes place. For ethical and legal reasons, in most countries, while a drone flies autonomously, one human pilot per vehicle must be present, in the visual line of sight of the aircraft and ready to intervene whenever necessary. This is equivalent to a human pilot being present in a self-driving car. While these laws introduce a high cost, especially in the operation of multi-agent autonomous systems, they are necessary to assign responsibility in case the vehicles cause some damage. Therefore, for the sake of safety, experimental activities mainly take place in private settings.

1.2 Contributions

This thesis is the result of an interdisciplinary PhD project, studying both the high level control of drone formations and their application to the condition monitoring of the built environment. As such, it contains contributions spanning the two fields of research. Throughout this work, the following contributions will be presented in detail:

- **A graph-based mission planner for a set of non-tethered UAVs**. The planner aims to assign inspection tasks to the available drones minimizing either mission time or travelled distance. It is formulated as a mixed integer optimization problem, and it is endowed with a set of constraints that theoretically ensure the mission can be carried out to completion, even if one of the drones fails, by distributing his workload across others.

- **A LiDAR-based motion planner for a network of tethered drones**, capable of guiding the leader drone of the formation to an arbitrarily specified position in 3D space within a previously unknown environment. This is achieved by only relying on real-time sensor readings. The problem is formulated as a convex optimization one.

- **An extension of the previous planner, formulated through Model Predictive Control (MPC)**. This new formulation increases the safety of the planned trajectories by under-approximating free space with a convex polytope. An optimal configuration planner is also presented, exploiting known information about the environment to obtain ideal configurations, which the system then tracks.

- **A reactive motion planning approach for a tethered drone based on RMP (Riemanniann Motion Policy)**. While this planner was developed and tested specifically for collaborating with a human in a spray-painting task, it is applicable to tethered drone networks in general.

- **A study of a pipeline for autonomous energy efficiency assessment**, combining existing techniques for automating the whole process, from data gathering to the comparison of the measurements to an existing model of the building to highlight differences potentially indicating defects or damage.

- **A drone-enabled, vision-based modal analysis technique** based on footage collected by multiple drones and synchronized *a posteriori*. The ability to stitch together measurements from different sources enables the recording of a whole structure or some significant sections of it at once. This, in turn, alleviates the need for constantly exciting the target structure for a prolonged amount of time.

- **An algorithm for autonomous mapping for a single non-tethered drone**, through which the vehicle decides on its own when enough data has been collected in the current area and where to move next to minimize exploration time.

1.3 Thesis Structure

The thesis is divided into two parts. The first is centered around high level control and planning for drone networks, while the second focuses on their applications for inspections. In particular, Chapter 2 contains a description of the graph-based planner for non-tethered drones, while all planners for their tethered counterparts are discussed in Chapter 3. Within part II, the vision-based modal analysis approach is presented in Chapter 4 and Chapter 5 contains the evaluation of the pipeline for autonomous energy assessment.

Note that some of the contributions presented here have already been published:

- Preliminary results regarding the multi-agent planner in Chapter 2 have been published in [21] and [22].
- The first LiDAR-based planner presented in Chapter 3 has been published in [23], while its MPC extension has been accepted for publication at the 2022 Conference and Decision and Control.
- The vision-based modal analysis pipeline described in Chapter 4 has been published in [20]

Part I

Path Planning and Coordination for UAV Networks

CHAPTER 2

Non-Tethered Multicopter Networks

UAVs are precious tools for infrastructure inspection due to their versatility and agility. One of their main limitations, especially for rotary wing drones, is the maximum flight time allowed by the onboard battery. While it depends on various factors such as vehicle weight and the mounted sensors' power consumption, the maximum flight time is usually thirty to forty minutes with current technology. This can make inspection times on a large structure very long. One way to tackle this is to employ multiple units to decrease the mission duration and avoid extra travel to change batteries. The deployment of multiple vehicles makes mission planning more complex; moreover, a fault-tolerant planner resilient at least to a single failure is desirable. A new algorithm is proposed here to solve these problems, which, through hierarchical decomposition and numerical optimization, deals with:

1. the automated generation of points of interest given a digital model of the building/structure;

2. the generation of trajectories for each drone;

3. the guarantee of mission robustness against a single fault.

Simulation in two separate realistic scenarios shows that the approach delivers close-to-optimal solutions with short computational time, thus being suitable for real-world operation.

2.1 A fault-tolerant automatic mission planner for a fleet of aerial vehicles

2.1.1 Motivation

As inspection tools, drones empower the latest industry trends of higher automation in construction and civil engineering via big data collection and analysis. Automating the data-gathering pipeline allows stakeholders to exploit them in every phase of a building's life cycle, storing correct and updated information in its digital twin, a vital tool of the Building Information Modeling (BIM) framework. Simultaneously deploying several drones speeds up operations significantly at the cost of more complex mission planning and control requirements. A mission planning algorithm is introduced to divide a set of tasks and assign them to available drones to carry out a data-gathering mission on a given building. While we focus on the particular case of UAVs in infrastructural inspection, the graph-based approach can be generalized to a broad class of mission planning problems.

2.1.2 Related work

Within the maintenance and inspection field, drones are mainly used to gather pictures of buildings and construction sites, especially from otherwise hard-to-reach places, as they extend the range of a human operator [24, 25]. Such images are then manually or automatically analyzed to track construction progress or detect surface defects, including cracks in concrete and road pavement, spalling, exposed rebars, loosened bolts, and surface moisture [26–28]. Regardless of their aim, most approaches rely either fully or partially on human intervention for both the data-gathering and the data-analysis stages. Reaching full autonomy would be a significant step forward, enabling quicker, more repeatable, and cheaper inspections. This can be achieved by automating two main steps.

The first one is the automatic derivation, from a building model to be inspected and for a specific type of task, of the Points of Interest (POI) the aircraft must visit. Although an approach to multi-agent coordination is to plan entire trajectories and constrain those to avoid collisions [29, 30], for inspections in particular, it is more common to develop rules and then automatic procedures that generate suitably distributed POI [18] in space. For

example, sub-modular path planning techniques have been proposed [31] to address the 3D coverage problem. The authors of [32] and [33] developed iterative approaches to solve it. They aim to maximize uniformity of coverage of the structure and minimize the computational cost to generate a trajectory for the UAV while also aiming at ideal camera orientation. Another approach [17] is to exploit Signal Temporal Logic [34], a formal language designed to concisely specify a mission task by expressing fundamental behaviors and time constraints. This also allows one to describe more complex tasks than a sequence of POI. If the aim is to employ the gathered images in a Structure from Motion (SfM) algorithm, such as in [35], points are distributed in space so as to introduce some overlap between consecutive pictures.

After distributing the POI, the second step is to compute an optimal route across them according to a chosen optimality criterion, such as the total mission time. In the case of multiple drones, a path for each one shall be generated, avoiding obstacles and collisions and optimally splitting the overall tasks among the units. This problem is usually formulated and solved via graph search methods, exploiting the Traveling Salesman Problem (TSP) or one of its variations [36]. The graph nodes are the POI, and its edges represent all feasible (i.e., obstacle-free) routes between them, together with their lengths or costs. The solution returned is the wanted list of nodes, i.e., the traversal order. The abstract nature of graphs and their considerable descriptive capabilities make them very powerful and versatile tools [37]. For example, approaches that account for communication constraints [38] and exclusion zones [39] have been proposed. Such abstraction expands the applicability of the proposed path planning method to multi-agent systems in general, including ground and naval vehicles. However, in graph-based formulations, the complexity of path planning increases exponentially with both the number of POI and the number of drones due to the exponential growth of both the search space and the number of Sub-tour Elimination Constraints (SEC) [36]. Heuristics are thus often introduced, even for relatively small instances, to obtain a feasible solution in a reasonable amount of time, at the cost of sub-optimality. In some cases, lower bounds on the quality of a solution with respect to the optimal one can be found [40]. Generalizations of the TSP include its multi-agent version (mTSP) and the Constrained Vehicle Routing Problem (CVRP) [41], where multiple vehicles are considered, each with its maximum capacity. Each node of the graph represents a customer with different capacity demand. In this framework, developed in the '60s within the petrol delivery industry, capacity refers to the quantity of a particular good the customers

require, which each vehicle can transport in a limited amount. The aim is to maximize the customer demand that is met compatibly with capacity limits related to the vehicles. Costs are simultaneously minimized, related to the number of vehicles used in the solution, the traveled distance (due to fuel and vehicle amortization), and travel time (driver payrolls).

As the probability of at least one drone failing, for example, due to battery aging and consequent reduced stored energy, increases with the number of deployed UAVs, it is wise to address this issue at the mission planning level. The concepts of robustness and robust solutions of the CVRP are present in literature, but they typically concern uncertainty on various parameters, such as travel time [42] customer demand [43], and service times [44], rather than the fault of one unit. Indeed, the requirement of robustness to a single point of failure has never been addressed in the formulation of a CVRP.

Addressed challenges and contributions

As summarized, maximizing drone autonomy in monitoring tasks comes with many challenges. In this work, the following are tackled:

- the automation of the POI generation procedure for a specific inspection type, establishing where exactly the UAVs should go to perform a certain task;
- the coordinated path planning of multiple drones with a shared objective;
- the minimization of metrics describing a mission, namely total inspection time and total traveled distance;
- the guarantee of robustness against a single point of failure, thus securing mission accomplishment even if one drone is not able to complete its task, with the possibility of re-planning it in real-time;
- the generation of trajectories that prevent collisions with the building under inspection.

We present an integrated, systematic approach to deal with these aspects and provide a ready-to-use tool. In doing so, we build on our preliminary work [22] and add as original contributions the theoretical proof of the robustness of a solution concerning the failure of one drone, a thorough analysis of computational times with various algorithm settings, and the application of the approach to the models of two real-world buildings.

Our method uses a geometric approach to generate the POI and then a hierarchical decomposition of the vehicle routing problem [45], obtained via clustering. For each cluster of POI, we solve a TSP, while a CVRP is solved to plan the paths among clusters. We present simulated results of the mission planning approach on the models of one of the buildings at the University of California, Los Angeles campus, and of a radio and meteorological research station in Spino d'Adda, Cremona, Italy (see figure 2.1). The obtained performance indicates that the approach can plan the mission in a short time, also for rather large buildings and several drones. A MATLAB library containing all the described algorithms was developed, freely available at [46]. We assume that each drone is equipped with low-level attitude and altitude controllers, and it accepts and tracks position references. Several solutions exist in the literature that ensure accurate position tracking, and collision avoidance [47]. Our experiments show that models of controlled drones are reliable enough to be adopted in complex simulations (see Fig. 2.2).

In Section 2.1.3 the main elements of the considered problem are formalized, and in Section 2.1.4 a solution approach and its guaranteed fault-tolerance property are presented. Section 2.1.5 deals with implementation aspects, while results are described and commented in Section 2.1.6. Section 2.1.7 contains some concluding remarks.

2.1.3 Problem Formulation

An instance of the problem to be solved is defined by a 3D mesh model of the considered building(s) or infrastructure, and a set of criteria that the POI must meet to accomplish the task at hand, also considering the employed data-gathering devices. Furthermore, a number $M \in \mathbb{N}$ of drones available for the mission is also given. The goal is to generate the set of POI:

$$\mathcal{P} = \{p_1, \ldots, p_n\} \subset \mathbb{R}^3, \tag{2.1}$$

where p_i is a single POI in the three-dimensional space, and to compute ordered sub-sets $\mathcal{S}_m$, $m = 1, \ldots M$ of $\mathcal{P}$ and assign them to the vehicles, such that each $\mathcal{S}_m$ represents a sequence of POI:

$$\begin{aligned} &\mathcal{S}_m = \{p_{i^m(1)}, \ldots, p_{i^m(n_m)}\} \\ &\bigcup_{m=1}^{t} \mathcal{S}_m = \mathcal{P} \\ &\mathcal{S}_\ell \cap \mathcal{S}_m = \emptyset,\ \forall \ell \in \{1, M\},\ \forall m \in \{1, M\} : \ell \neq m. \end{aligned} \tag{2.2}$$

In (2.2), $p_{i^m(j)}$ is the j^{th} POI, with index $i = i^m(j)$, of path $\mathcal{S}_m$, whose total length is n_m. In this generic form, a solution could describe various

Figure 2.1: *Buildings used in our tests. From the top: University of California, Los Angeles building, and the meteorological station in Spino d'Adda, Cremona, Italy.*

cooperative path planning or vehicle routing problems, not limited to UAVs. These paths should be optimal according to a suitable criterion, such as minimization of the mission time, under the constraints given by:

- the limited capacity of each drone in terms of flight time/distance;
- the need to compute obstacle-free three-dimensional paths;
- guaranteed robustness of the solution, defined as follows: a solution is said to be robust if, given a failure of a drone at any point during

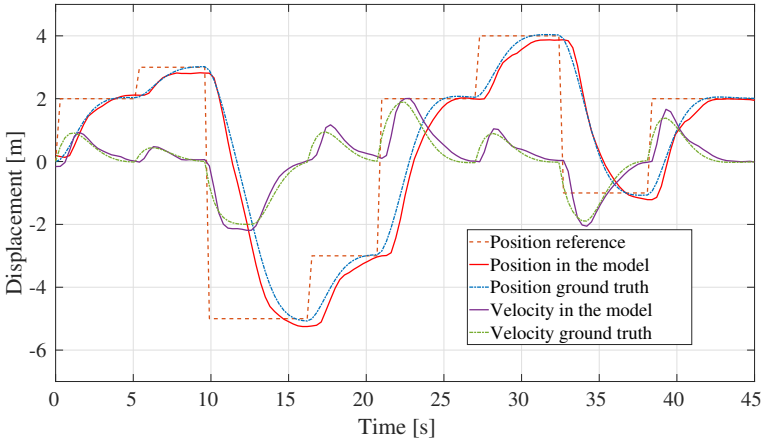

Figure 2.2: *Data gathered in a model identification test on one of our drones, highlighting the reliability of the numerical model*

the execution of the mission, it is possible to still visit all POI by only relying on the remaining UAVs.

It is assumed that the drones execute the mission at roughly constant velocity and stop at each POI for a roughly fixed amount of time. In this way, we can use the path length as a proxy for mission time for each drone and solve the problem as a static path planning one, neglecting the UAVs' dynamics. A different speed for each drone can be considered to account for different UAV models. Thus, the length of the longest path among the employed drones (out of M available ones) should be minimized to minimize mission time. Furthermore, without loss of generality, a single base station is assumed to be positioned at $p_0 \in \mathbb{R}^3$, from which the drones take off and to which they return after the mission. Different take-off and return points for each drone can be specified straightforwardly in our approach. We finally introduce a requirement that the paths be computed in a few minutes at most to make the approach feasible in practical applications.

2.1.4 Proposed methodology

For realistic scenarios, with problem instances featuring thousands of POI, it is challenging to meet the requirements outlined in Section 2.1.3, because generating the paths becomes exponentially more time-consuming as the number of POI and vehicles increases.

We thus propose to break the problem complexity by adopting a hierarchical decomposition of the path planning task. Conceptually, our approach features the following steps, also summarized in Figure 2.3:

A. Automatic generation of the set $\mathcal{P}$ of POI from a mesh of the building/infrastructure under study, according to application-specific criteria;

B. Clustering of $\mathcal{P}$ in k partitions;

C. Generation of in-cluster navigation graphs and estimation of in-cluster traversal cost;

D. Definition of a high-level navigation graph where each node represents a cluster, and each edge the obstacle-free shortest paths between two connected clusters;

E. Assignment of sequences of clusters among the drones by solving a CVRP over the graph defined at step D, accounting for capacity and robustness constraints;

F. Generation and concatenation of in-cluster paths for each drone to build the final ordered list of POI and the obstacle-free routes connecting them.

We describe next each step of the approach in detail.

Generation of points of interest

While our approach is, in principle, applicable to any collaborative path planning problem that a graph can describe, we focus our presentation on the task of visual inspection of a building's surface. We assume to start from a mesh of the building's shell, typically defined by a list of vertices and a list of surface elements. Each element contains the indexes of its vertices and its normal pointing out of the surface. We assume the mesh is geometrically closed, so it is always possible to verify if a given point is inside, outside, or on the surface. Such a mesh model can either be obtained from a CAD file of the structure or defined manually through CAD software. In practice, more than one mesh may be used, with different levels of detail. In this work, we adopt a finer detailed mesh to generate the POI, and a coarser one for trajectory collision checking, to speed up computation. Without loss of generality, we consider triangular meshes in the remainder. See for example figure 2.4. The location of POI around the mesh model depends on the kind of data-gathering task to be carried out. For example, suppose the aim is to obtain a 3D rendering of a building. In that case, the camera Field of View (FOV), its resolution, and the desired overlap between consecutive pictures should all be factored in. Supposing we want to carry out such a mission, we can calculate the distance d from the structure from which the

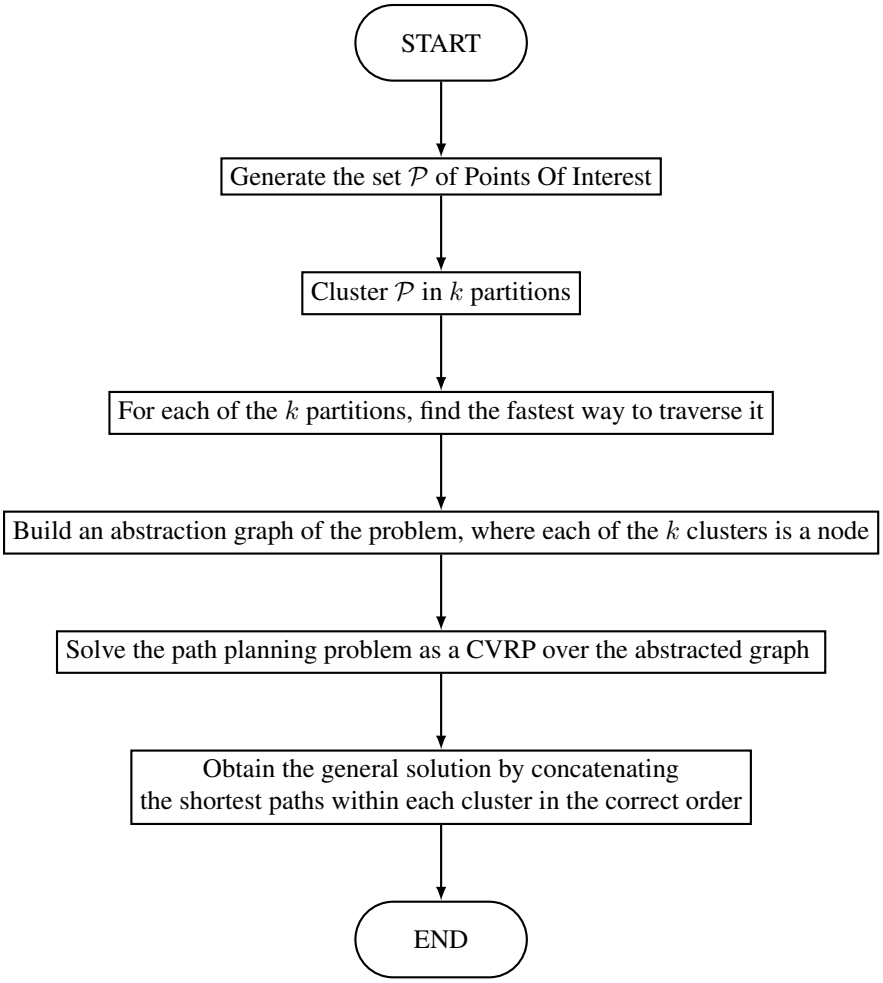

Figure 2.3: *Proposed procedure schema*

pictures shall be taken. Then, we generate the set $\mathcal{P}$ by placing a point along the normal to each surface element of the mesh at a distance d from the surface itself and outside the building. In this way, all POI are equally far away from the structure's surface, and they are more densely distributed in those areas where the building shows a more complex shape, conveyed by a higher number of surface elements, and more pictures would be necessary to obtain a precise reconstruction. We can also assign a direction to each POI, representing the ideal orientation of the camera: in our example, it would be perpendicular to the surface to photograph.

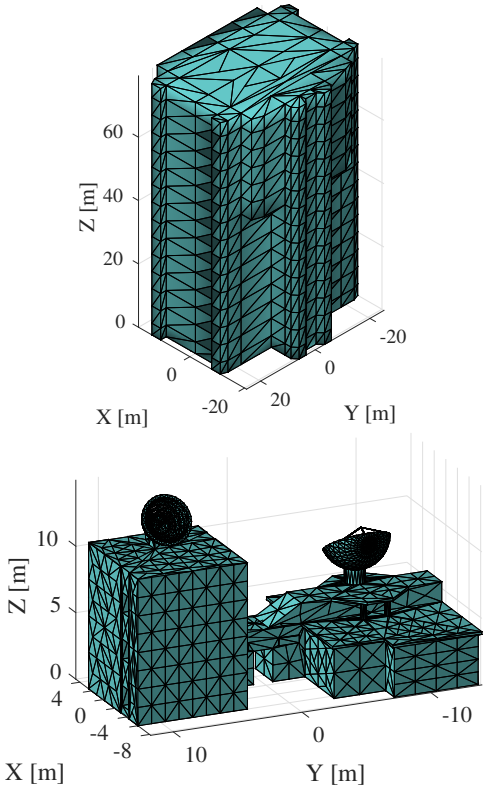

Figure 2.4: *Tri-mesh model of the external surface of the University of California - Los Angeles (UCLA) (top) and Spino (bottom) buildings.*

Clustering of POI

As the number n of POI generated for a real building can be in the order of thousands, finding a global solution to the path-planning in one shot is impractical, even for a single unit. We thus adopt clustering to reduce the complexity, from a single large-scale optimization problem to a hierarchy of smaller-scale ones. Since we aim to cluster the POI based on spatial proximity, we adopt the so-called $k-$means clustering [48]. This approach ensures that each resulting cluster is composed of all the points closer to its centroid than those of the other clusters. Note that the provided partition into clusters depends on the randomly selected initial condition of the algorithm. This can be a source of non-deterministic behavior across repeated runs of the planner, even if starting from the same environment and with the same POI. The choice of $k-$means based on POI distance is natural since we can expect that in the globally optimal solution, the resulting path

would be a sequence of points close to each other rather than far away. We denote the obtained clusters with $\mathcal{C}_i$, $i = 0, \ldots, k$, where $\mathcal{C}_0$ is a particular cluster containing only the home point p_0, which will be necessary to calculate paths originating from and ending at such location:

$$\bigcup_{j=1}^{k} \mathcal{C}_j = \mathcal{P} \tag{2.3a}$$

$$\mathcal{C}_i \cap \mathcal{C}_j = \emptyset,\ \forall i \in \{1, k\},\ \forall j \in \{1, k\} : j \neq i \tag{2.3b}$$

$$\mathcal{C}_0 = \{p_0\} \tag{2.3c}$$

To obtain a sensible approach, the number of clusters shall be larger than the number of available drones, i.e. $k > M$. In Section 2.1.6 we present results on the sensitivity of the obtained solution with different values of k and M.

The procedure up to this point (POI generation and clustering) is summarised in algorithm 1, while figure 2.5 presents an example on the UCLA building.

Algorithm 1 POI generation and clustering

$\mathcal{P} \leftarrow \emptyset$
$d \leftarrow \bar{d}$
$k \leftarrow \bar{k}$
for all surface elements t in mesh **do**
  $p \leftarrow t.centroid + d * t.normal$
  $\mathcal{P}.append(p)$
end for
$\mathcal{C}_1, \ldots, \mathcal{C}_k \leftarrow kMeans(\mathcal{P}, k)$
$\mathcal{C}_0 \leftarrow p_0$

In-cluster navigation graphs and traversal cost estimates

For each cluster $i = 1, \ldots, k$ we generate a graph $\mathcal{G}_{ci}$:

$$\mathcal{G}_{ci} = (\mathcal{C}_i, \mathcal{E}_i, w_c) \tag{2.4}$$

where the set of nodes is $\mathcal{C}_i$, i.e., the UAV belonging to that cluster, and the set of edges $\mathcal{E}_i$ is generated as follows:

$$\mathcal{E}_i = \{(p_r, p_\ell) \in \mathcal{C}_i : \|p_r - p_\ell\|_2 < r,\ r \neq \ell\} \subset \mathcal{C}_i \times \mathcal{C}_i, \tag{2.5}$$

i.e, the set of all pairs of indexes whose corresponding UAV are closer than a distance r, which is a user-defined parameter. Moreover, we compute and

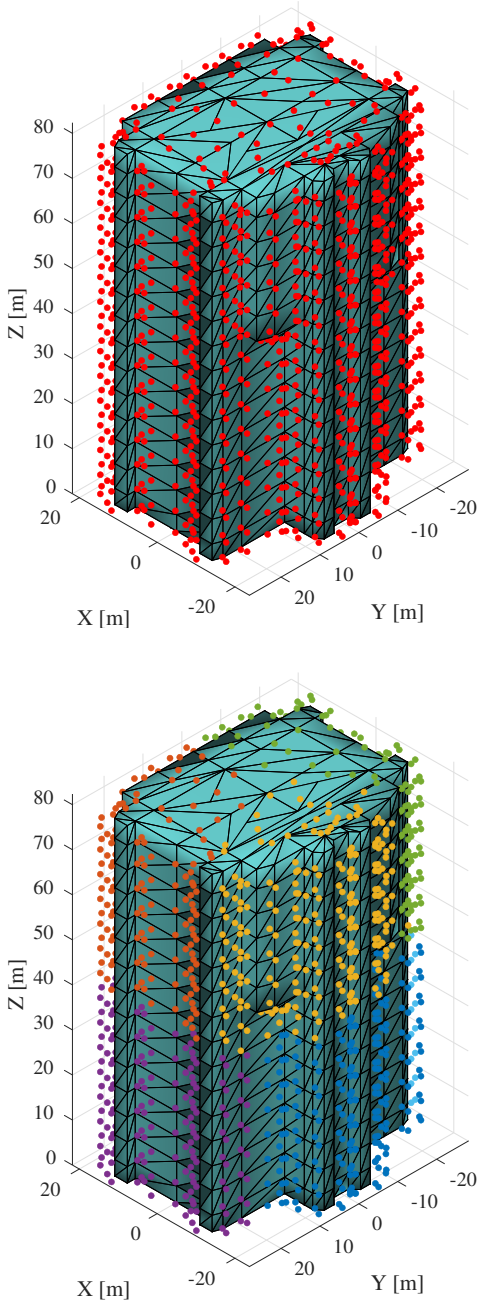

Figure 2.5: *UAV generated from the mesh (top) and clustered into groups based on spatial proximity through $k-$means (bottom).*

store, for each edge, an obstacle-free path $s^{(r,\ell)}$ between the corresponding UAV:

$$s^{(r,\ell)} = \left\{ p_r, q^{(r,\ell),1}, \ldots, q_i^{(r,\ell),\bar{j}^{(r,\ell)}}, p_\ell \right\} \tag{2.6}$$

where $q^{(r,\ell),j}$, $j = 1, \ldots, \bar{j}^{(r,\ell)}$ are intermediate way-points used to travel around possible obstacles that may be be present on the segment connecting p_r to p_ℓ. In this work, we define such intermediate navigation points by calculating the normal to each vertex of the mesh (i.e., the average of the vectors that are normal to the surface elements containing such vertex) and taking the point at distance $\bar{d}$ along its direction. Another possible approach is to randomly sample the free space around the mesh surface. The path $s^{(r,\ell)}$ (2.6) is then computed as the shortest, obstacle-free one between the two UAV (p_r, p_ℓ), possibly passing trough one or more intermediate waypoints. We further associate to the in-cluster graphs a cost function w_c, defined as:

$$\begin{aligned} w_c &: \mathcal{E}_i \longmapsto \mathbb{R}^+, \, i = 1, \ldots, k \\ w_c(p_r, p_\ell) &= \sum_{j=2}^{|s^{(r,\ell)}|} \| s^{(r,\ell),j} - s^{(r,\ell),j-1} \|_2 \end{aligned} \tag{2.7}$$

where $|s^{(r,\ell)}|$ is the cardinality of the sequence $s^{(r,\ell)}$, and $s^{(r,\ell),j}$ is the j^{th} point in the sequence. The weight function (2.7) thus returns the length of the obstacle-free path between two connected UAV.

The choice of connecting only UAV that are closer than a maximum distance r is justified by the same rationale behind the clustering approach: we expect optimal solutions to the local trajectory generation problems to result in series of points that are close to each other, rather than far apart.

To evaluate whether the segment connecting two points is obstacle-free, we verify that its intersection with any of the mesh surface elements is empty. To limit the computational cost of this check, we employ a coarser mesh, as anticipated in Section 2.1.4.

Finally, for each $\mathcal{C}_i$ we want to compute the cost of traversing the whole cluster by visiting all of its UAV. These cost values are needed at the higher hierarchical level to optimally assign a sequence of cluster to each of the available drones (see Section 2.1.4) and, correspondingly, the entry and exit UAV of each cluster. However, the latter would be needed in turn to compute the cluster's traversal cost. To sort out this chicken-and-egg problem, we compute an estimate, $\hat{w}_i$, of the cluster traversal cost, by solving a TSP problem inside it. For each UAV $p_r \in \mathcal{C}_i$, the set of points connected to it inside the same cluster is:

$$\mathcal{N}(p_r) = \{ p_\ell : (p_r, p_\ell) \in \mathcal{E}_i \}, \tag{2.8}$$

and for each edge $(p_r, p_\ell) \in \mathcal{E}_i$ the Boolean variable $\lambda_{r,\ell}$ takes the value 1 if that edge is part of the cluster-traversing path and 0 otherwise. These Boolean variables are collected in the set:

$$\Lambda_i = \{\lambda_{r,\ell} \in \{0,1\},\ \forall (p_r, p_\ell) \in \mathcal{E}_i\}\,. \tag{2.9}$$

The total number of Boolean variables thus introduced for cluster $\mathcal{C}_i$ is:

$$|\Lambda_i| = \sum_{r=1}^{|\mathcal{C}_i|} |\mathcal{N}(p_r)|$$

Then, we compute the value of $\hat{w}_i$ by solving the TSP:

$$\hat{w}_i = \min_{\Lambda_i} \sum_{r\,=\,1}^{|\mathcal{C}_i|} \sum_{\ell\,=\,1}^{|\mathcal{N}(p_r)|} w_c(p_r, p_\ell)\lambda_{r,\ell} \tag{2.10a}$$

subject to

$$\sum_{\ell=1}^{|\mathcal{N}(p_r)|} \lambda_{r,\ell} = 1,\ r = 1, \ldots, |\mathcal{C}_i| \tag{2.10b}$$

$$\sum_{r=1}^{|\mathcal{N}(p_\ell)|} \lambda_{r,\ell} = 1,\ \ell = 1, \ldots, |\mathcal{C}_i| \tag{2.10c}$$

The cost (2.10a) is the total length of the path and constraints (2.10b) and (2.10c) require that only one edge is picked respectively to and from each point. Provided suitable subtour elimination constraints are added, problem (2.10) results in a single loop that visits all UAV in cluster $\mathcal{C}_i$. We will show in Section 2.1.6 that the error between $\hat{w}_i$ and the final cost of traversing the cluster, after its entry and exit point have been defined, is very small in realistic scenarios. As a final remark, note that problem (2.10) may be unfeasible, for example if the graph $\mathcal{G}_{ci}$ is not connected or if it can not be traversed by a single loop. In these cases, one can simply increase the value of r until (2.10) becomes feasible, which is guaranteed to happen when

$$r \geq \max_{p_r, p_\ell \in \mathcal{C}_i} \|p_r - p_l\|_2.$$

High-level navigation graph

After obtaining the clusters, we generate another navigation graph at higher level, named "global graph" and denoted as $\bar{\mathcal{G}}$:

$$\bar{\mathcal{G}} = \left(\bar{\mathcal{V}}, \bar{\mathcal{E}}, w_g\right) \tag{2.11a}$$

$$\bar{\mathcal{V}} = \{0, 1, \ldots, k\} \tag{2.11b}$$

$$\bar{\mathcal{E}} \subset \bar{\mathcal{V}} \times \bar{\mathcal{V}}, \;\; |\bar{\mathcal{E}}| = (k+1)k \tag{2.11c}$$

$$w_g : \bar{\mathcal{E}} \;\longmapsto\; \mathbb{R}^+. \tag{2.11d}$$

Each node i in $\bar{\mathcal{V}}$ represents cluster $\mathcal{C}_i$, and each edge is a pair of different clusters:

$$\bar{\mathcal{E}} = \{(i,j) \in \bar{\mathcal{V}} \times \bar{\mathcal{V}} : i \neq j\}. \tag{2.12}$$

For simplicity, we consider a fully connected global graph: we discuss options to reduce the number of edges, thus the computational burden, in Section 2.1.5. In a way similar to the in-cluster navigation graphs, we associate and store to each edge $(i,j) \in \bar{\mathcal{E}}$ an obstacle-free path that allows a drone to travel from $\mathcal{C}_i$ to $\mathcal{C}_j$. To do so, we first find the UAV $\bar{p}_i^{(j)}$ and $\bar{p}_j^{(i)}$ as:

$$\bar{p}_i^{(j)}, \bar{p}_j^{(i)} = \arg \min_{\substack{p \in \mathcal{C}_i \\ q \in \mathcal{C}_j}} \|p - q\|_2, \tag{2.13}$$

i.e. the points belonging to the two different clusters whose distance is minimal (corresponding to the distance between the two clusters). Then, the inter-cluster obstacle-free path $\bar{s}^{(i,j)}$ is computed as in (2.6) with $\bar{p}_i^{(j)}$ and $\bar{p}_j^{(i)}$ as starting and ending points, respectively, and the possible addition of intermediate navigation points whenever the segment between $\bar{p}_i^{(j)}$ and $\bar{p}_j^{(i)}$ is not obstacle-free, as discussed in Section 2.1.4. We denote with $l_{i,j} = w_c(\bar{p}_i^{(j)}, \bar{p}_j^{(i)})$ the weight (length) of path $\bar{s}^{(i,j)}$ (compare (2.7)) and store this quantity as well, and we compute the maximum inter-cluster travel distance as:

$$l_{max} = \max_{(i,j) \in \bar{\mathcal{E}}} l_{i,j} \tag{2.14}$$

Finally, the weight function w_g computes the cost of traversing a given edge (i,j), as the cost $l_{i,j}$ of traveling from cluster $\mathcal{C}_i$ to cluster $\mathcal{C}_j$, plus the cost $\hat{w}_j$ of traversing the arrival cluster $\mathcal{C}_j$:

$$w_g(i,j) = l_{i,j} + \hat{w}_j. \tag{2.15}$$

Cluster assignment and guaranteed fault tolerance

Exploiting the global graph (2.11), we can now formulate a tractable problem to assign to each drone a sequence of clusters to visit. We start from a CVRP, which aims at finding the optimal set of routes for a fleet of vehicles to visit a given set of nodes, and include additional constraints to account for capacity limits and fault tolerance, while the obstacle-free requirement of the resulting paths is implicitly satisfied by how the edges among and within clusters have been computed.

The Boolean optimization variables $x_{(i,j),m} \in \{0,1\}$, $(i,j) \in \bar{\mathcal{E}}$, $m = 1,\dots,M$ take value 1 if the m^{th} drone travels from cluster $\mathcal{C}_i$ to cluster $\mathcal{C}_j$ and traverses inside the latter, and 0 otherwise. We collect these variables in vector $X \in \{0,1\}^{M[(k+1)k]}$. Moreover, we denote the capacity of drone m as $\bar{Q}_m$ (maximum flight time or, as considered here, available travel distance), and the maximum capacity among drones as:

$$\bar{Q} = \max_{m=1,\dots,M} \bar{Q}_m.$$

For a given value of X, we can then compute the residual capacity of the generic drone m as:

$$Q^*_m = \bar{Q}_m \; - \sum_{(i,j)\in\bar{\mathcal{E}}} w_g(i,j)\, x_{(i,j),m}, \tag{2.16}$$

i.e, the difference between the capacity of a drone and the length of the path assigned to it.

We can now formulate the optimization problem:

$$\min_X \sum_{(i,j)\in\bar{\mathcal{E}}} \sum_{m\,=\,1}^{M} w_g(i,j)\, x_{(i,j),m} \tag{2.17a}$$

subject to

$$\sum_{\substack{i=0\\ i\neq j}}^{k} \sum_{m=1}^{M} x_{(i,j),m} = 1,\ j = 1,\dots,k, \tag{2.17b}$$

$$\textstyle\sum_{\substack{i=0\\ i\neq p}}^{k} x_{(i,p),m} = \sum_{\substack{j=0\\ j\neq p}}^{k} x_{(p,j),m},\ p = 1,\dots,k,\ m\ =\ 1,\ \dots,\ M \tag{2.17c}$$

$$Q^*_m \geq 2l_{max} \qquad m = 1,\dots,M \tag{2.17d}$$

$$\sum_{m=1}^{M} Q^*_m \geq 2M\, l_{max} + \bar{Q} \tag{2.17e}$$

The cost function (2.17a) is the sum of trajectory lengths over all vehicles, also known as total makespan. Other cost functions can be adopted, such as the length of the longest individual trajectory, which corresponds to minimising mission time. We address this case in Section 2.1.5. Constraint (2.17b) imposes that only one drone can reach or leave each cluster, with the exception of the home cluster $\mathcal{C}_0$ where they all start, whereas (2.17c) ensures that each node is reached and left by the same vehicle, imposing continuity in each path (i.e., teleportation is not allowed). Constraint (2.17d) imposes that each drone has enough residual capacity to cover twice the maximum travel distance among all cluster pairs, l_{max}. At the same time it ensures no drone is assigned a longer trajectory than its capacity allows for. Finally, constraint (2.17e) requires that, in addition to the amount set by (2.17d), the residual capacities have enough margin to collectively exceed $\bar{Q}$, i.e. the maximum capacity among all drones. Infeasibility of (2.17) can occur only if the drones do not have have enough capacity to tackle the data-gathering problem at hand, thus providing an indication that the available units are not sufficient to carry out the mission without recharging the batteries once or more.

Constraints (2.17d) and (2.17e) allow the approach to obtain solutions that are guaranteed to be tolerant to any single point of failure. Specifically, we consider the following definition of robustness:

Definition 1 (Robustness)**.** *A solution to* (2.17) *is robust if and only if, given the failure of any single vehicle at any point during the mission, it is possible to carry out the remainder of the mission with the remaining vehicles without violating the constraints on maximum capacity.*

Lemma 1. *All feasible solutions to* (2.17) *are robust.*

Proof. Assume, without loss of generality, that the drone fails immediately, i.e. all the nodes that were assigned to it have not been visited. The necessary capacity to visit those nodes Q_f is by definition

$$Q_f = \bar{Q}_m - Q^*_{\underline{m}}, \tag{2.18}$$

where $\underline{m}$ denotes the failed drone. This is the worst case scenario, as if the drone does not fail immediately, then $Q_f < \bar{Q}_m - Q^*_{\underline{m}}$. Thanks to (2.17d), it is true that

$$Q^*_{\underline{m}} \geq 2l_{max} \implies Q^*_{\underline{m}} = 2l_{max} + \alpha, \tag{2.19}$$

α being the non-negative term that describes exactly the residual capacity in excess of $2l_{max}$. The total residual capacity left in the remaining drones

is equal to the total residual capacity minus that of the failed drone, and thanks to (2.17e):

$$\sum_{m=1,m\neq\underline{m}}^{M} Q_m^* = \sum_{m=1}^{M} Q_m^* - Q_{\underline{m}}^* \geq 2Ml_{max} + \bar{Q} - Q_{\underline{m}}^*. \tag{2.20}$$

The available residual capacity is greater than the necessary one Q_f, in fact:

$$2Ml_{max} + \bar{Q} - Q_{\underline{m}}^* \geq Q_f = \bar{Q}_m - Q_{\underline{m}}^* \tag{2.21}$$

$$2Ml_{max} + \bar{Q} - 2l_{max} - \alpha \geq \bar{Q}_m - 2l_{max} - \alpha \tag{2.22}$$

$$2Ml_{max} + \bar{Q} \geq \bar{Q}_m, \tag{2.23}$$

which is always true as M and l_{max} are positive and $\bar{Q} \geq \bar{Q}_m$ by definition. □

Remark 1. *In practice, constraints* (2.17d)-(2.17e) *ensure that, if a drone fails at any point, the remaining ones have enough capacity left to complete the mission without the need recharge. Note that the additional capacity* $2Ml_{max}$ *in* (2.17e) *is required as well, to ensure that all drones can eventually return to the starting position.*
If a drone fails, an immediate automatic re-planning is computed by solving again problem (2.17) *with* $M = M - 1$*, after substituting the robustness constraints* (2.17d)-(2.17e) *with a constraint on maximum capacity*

$$Q_m^* \geq 0,\ m\ =\ 1,\ \ldots,\ M \tag{2.24}$$

and updating the global graph (2.11) *by eliminating the nodes (i.e., clusters) that have already been visited when the failure occurs.*

Generation of the overall paths

X^* is the optimal vector obtained by solving problem (2.17), and $x_{(i,j),m}^*$ its entries. Our goal is now to assemble the overall path for each drone. For simplicity, in this section we omit the notation $m = 1, \ldots, M$ and consider a generic drone with index m, the procedure being identical for all drones. The path is generated through the following steps.

Procedure 1. Path generation

1. Extract the optimized sequence $i^m(0), \ldots, i^m(N_m)$ of indexes that identify the clusters to be visited. The sequence cardinality is $N_m + 1$,

where N_m depends on the obtained solution for drone m. Such a sequence is computed incrementally as follows:

$$\begin{aligned} i^m(0) &= 0 \\ i^m(1) &= \ell : x^*_{(i^m(1),\ell),m} = 1 \\ &\vdots \\ i^m(j) &= \ell : x^*_{(i^m(j-1),\ell),m} = 1 \end{aligned} \tag{2.25}$$

i.e., by concatenating the edges that shall be travelled by drone m. In virtue of constraints (2.17b)-(2.17c), all the indexes in (2.25) are unique and $i^m(N_m) = 0$, that is, the drone returns to $\mathcal{C}_0$ at the end of its mission.

2. On the basis of sequence (2.25), identify the entry and exit POI of each cluster, denoted respectively by the indexes $in(i^m(j))$ and $out(i^m(j)), j = 1, \ldots, N_m - 1$, as follows (recall (2.13)):

$$\begin{aligned} p_{in(i^m(j))} &= \bar{p}^{(i^m(j-1))}_{i^m(j)} \\ p_{out(i^m(j))} &= \bar{p}^{(i^m(j+1))}_{i^m(j)}. \end{aligned} \tag{2.26}$$

3. Use the obtained entry and exit points to compute the final in-cluster path for each $\mathcal{C}_{i^m(j)}$, $j = 1, \ldots, N_m - 1$, by solving again the TSP (2.10) this time adding, and forcing its usage in the solution, a virtual edge with zero cost between the POI $p_{in(i^m(j))}$ and $p_{out(i^m(j))}$. This corresponds to computing the shortest Hamiltonian path between these POI over the graph $\mathcal{G}_{ci^m(j)}$. $s^*(\mathcal{C}_{i^m(j)})$ denotes such a path, comprising only POI, and with $t^*(\mathcal{C}_{i^m(j)})$ the corresponding obstacle-free trajectory, comprising the same POI plus the intermediate navigation points, when needed.

4. Build the desired sequence $\mathcal{S}_m$ (2.2) as:

$$\mathcal{S}_m = \left\{ s(\mathcal{C}_{i^m(1)}), \ \ldots, s(\mathcal{C}_{i^m(N_m-1)}) \right\} \tag{2.27}$$

and the corresponding overall obstacle-free trajectory, denoted with $\mathcal{T}_m$, as:

$$\begin{aligned} \mathcal{T}_m = \{ & s^{(0,in(i^m(1)))}, \ t^*(\mathcal{C}_{i^m(1)}), \\ & s^{(out(i^m(1),in(i^m(2)))}, t^*(\mathcal{C}_{i^m(2)}), \\ & s^{(out(i^m(j-1),in(i^m(j)))}, t^*(\mathcal{C}_{i^m(j)}), \\ & \ldots, t^*(\mathcal{C}_{i^m(N_m-1)}), s^{(out(i^m(N_m-1)),0)} \} \end{aligned} \tag{2.28}$$

Figure 2.6 presents an example of obstacle-free trajectories obtained after executing Procedure 1.

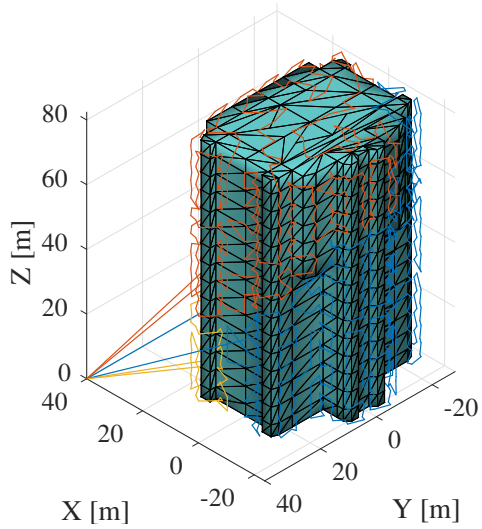

Figure 2.6: *Example of solution obtained with 3 drones, UCLA environment.*

2.1.5 Implementation aspects and overall approach

We now discuss an alternative formulation and relevant implementation aspects, and summarize the proposed approach.

Minimizing mission time

A valid alternative to cost function (2.17a) is to replace it with the following:

$$\max_{m} \sum_{i=1}^{K} \sum_{j=1}^{K} w_g(i,j) x_{(i,j),m}, \tag{2.29}$$

i.e., the length of the longest trajectory. Solutions obtained through this approach are characterised by evenly distributed workloads across the drones, thus also shorter mission duration, with generally larger total makespan (i.e., higher cumulative flight time and thus total energy cost). This min-max formulation entails the inclusion of a real optimization variable, turning problem (2.17) from a Binary Integer Linear Problem (BILP) to a Mixed-Integer Linear Problem (MILP), with significantly longer solution time (see figure 2.12). To decrease the problem complexity and solution time, one can however limit the number of arcs in the global graph by connecting only the clusters that are closer than a certain threshold or are not completely separated by obstacles, as discussed next.

Reducing Solution Time

When solving problem (2.17), we expect the computational time t_{CVRP} to be an exponential function of the number of optimization variables:

$$t_{CVRP} \propto e^{\alpha(Mk(k+1))+\beta}, \tag{2.30}$$

where α, β are positive scalars. In practical applications, a trade-off between computational time and optimality must be achieved: for example, since we aim at using this formulation not only for planning, but also for online re-planning in case of failure, we want to ensure that we can elaborate a new mission plan as soon as a failure occurs and without the need to stop the other aircrafts. This limits the total computational time for re-planning to some seconds at most. To decrease t_{CVRP} while preserving as much as possible the quality of solutions, we propose a heuristic that reduces the available degrees of freedom in a sensible way. In particular, in (2.17) there are $Mk(k+1)$ binary optimization variables due to the graph $\bar{\mathcal{G}}$ being completely connected: M binary variables for each one of the $k(k+1)$ edges of

the graph. To reduce this number, we set some of these to zero, by exploiting once again the assumption that an optimal solution to (2.17) is made of series $\mathcal{S}_m$ of consecutive clusters that are close to each other, in terms of the distance $l_{i,j}$ introduced in Section 2.1.4. The procedure we adopt, illustrated in Algorithm 2, removes edges one by one in decreasing weight order while ensuring that each node ends up with degree (i.e., the number of connections) at least a user-defined value $\underline{\mathrm{m}}$ and the starting node remains connected to all the other nodes, to maintain a direct route to/from the base station. This procedure is applied right before solving the CVRP, after the global graph has been completed. The aim is to reduce the solution time t_{CVRP} by solving the problem on a graph where the connections are significantly less than $k(k+1)$ and their number scales linearly with the number of clusters k. In fact, by applying procedure 2 the number of optimization variables tends to $k\underline{\mathrm{m}}$.

Algorithm 2 Edge removal

for $i = 1, \ldots, k$ **do**
  $numEdges \leftarrow |\{j : (i,j) \in \bar{\mathcal{E}}\}|$
  while $numEdges > \underline{\mathrm{m}}$ **do**
    $(i, j^*) = arg \max_{j:(i,j)\in\bar{\mathcal{E}}} l_{i,j}$
    if $j^* \neq 0$ **and** $|\{k : (j^*, k) \in \bar{\mathcal{E}}\}| > \underline{\mathrm{m}}$ **then**
      $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{(i,j)\}$
    end if
    $numEdges \leftarrow numEdges - 1$
  end while
end for

Subtour Elimination Constraints (SEC)

In routing problems such as (2.10) and (2.17), the issue of sub-tours must be addressed. Sub-tours are defined as loops that do not include the starting point, which are still feasible in both (2.10) and (2.17). There are two possible solution strategies to avoid sub-tours. One is to introduce special Sub-tour Elimination Constraints, in one of the forms that exist in literature, however resulting in a number of additional constraints that is exponentially increasing with the number of nodes in the graph. We thus adopt the second strategy, which is to solve the problem as stated, check for sub-tours in the solution and, if found, include additional constraints to specifically exclude those sub-tours from the solution and solve again the routing problem, iterating the procedure until no sub-tours remain. Specifically, we identify the sets $\mathcal{V}_{sub,n} \subset \bar{\mathcal{V}} \setminus \{0\}$ of nodes belonging to the generic n-th sub-tour,

$n = \ldots, N$, and impose that no more than $|\mathcal{V}_{sub,n}| - 1$ edges are taken in the next solution within that subset:

$$\begin{aligned} \sum_{(i,j)\in\mathcal{V}_{sub,n}} x_{(i,j),m} \leq & |\mathcal{V}_{sub,n}| - 1 \\ m = 1, \ldots, M \qquad & n = 1, \ldots, N \end{aligned} \tag{2.31}$$

These constraints avoid that the same sub-tour is present in the next solution.

Overall approach

The following procedure summarizes the proposed approach, also including the considered implementation aspects. At steps 2) and 5), the described sub-tour elimination strategy shall be adopted as well.

Procedure 2. Overall approach
Given the building mesh and the drones' capacities $\bar{Q}_1, \ldots, \bar{Q}_M$:

1. Run Algorithm 1;
2. For each cluster $i = 1, \ldots, k$, define the graph (2.4) and solve (2.10);
3. Define the graph (2.11);
4. Run Algorithm 2;
5. Solve problem (2.17);
6. Apply Procedure 1.

Notes on implementation in the real world

While this approach is applicable to real world scenarios with existing technology, some remarks on implementation are due. First of all, the majority of UAVs available to the general public rely on a combination of the onboard Inertial Measurement Unit (IMU) and GPS for self-localization in space. As long as the drones are endowed (and most of them are) with a mechanism for real-time obstacle avoidance, the low position accuracy of a regular GPS system poses no safety threat to the vehicle. Nevertheless, issues could arise in the precise tracking of the POI, as consumer grade GPS typically show inaccuracies in the order of meters. To pursue a precise

localization of the aircraft within the environment and centimeter-level accuracy in tracking the POI, it is advisable to rely on a Real Time Kinematics (RTK) capable GPS.

Secondly, because the formulation of the problem is centralized, it is necessary that all employed drones be able to communicate at all times with a ground station through a wireless connection to receive their setpoints and any possible update. The proposed problem formulation cannot be translated into a decentralized version, and even if it was, the on-board computers of commercially available drones have very limited computational resources, which are mostly already taken up by low-level control and reference tracking tasks. We therefore conclude that solving the problem on a centralized ground station without stringent hardware limits and communicating the references only to each drone is the best design. Furthermore, one should make sure that the chosen UAVs allow the real-time communication of waypoints from a device different than the remote they are shipped with.

Finally, the laws and regulations of the country in which the mission is performed must be taken into account. While at this time rules are evolving quickly, in most of the world flying autonomous drones is still prohibited, at least in public spaces, meaning that there must always be one pilot per aircraft holding the remote and taking full responsibility for damages the UAV may cause. What is and is not allowed in private settings varies by country.

2.1.6 Results

Simulation Setup

We performed several simulations based on the models of both buildings introduced in Section 2.1.2, with around 1200 generated POI each. All simulations have been performed in MATLAB [49], installed on Ubuntu 20, on a machine with an AMD Ryzen™9 5950X 3.4 GHz processor. CVRP instances were solved through the CPLEX® optimizer, while TSP problems were solved with LKH 3.0 [50]. The most relevant metrics are solution time, which determines if the approach is suitable for real-world applications, and the length of planned trajectories, taken as a measure of solution quality. The sum of all trajectory lengths is a proxy for energy consumption, whereas their maximum across vehicles characterizes mission duration. We solved the problem with a varying number of drones (3 to 5) and of clusters (6 to 20). The MATLAB code we developed for the simulations is available both through a GitHub repository [51] and as a MATLAB toolbox [46].

Numerical Results

With reference to the first mesh shown in figure 2.4, 1279 POI were generated and we studied the behaviour of total length of the trajectories, which is minimised through the cost function, with respect to the number of clusters. Figure 2.7 shows that the solution's quality is roughly independent of the number of clusters (given that it is larger than the number of drones). From this we conclude that there is no point in increasing the number of

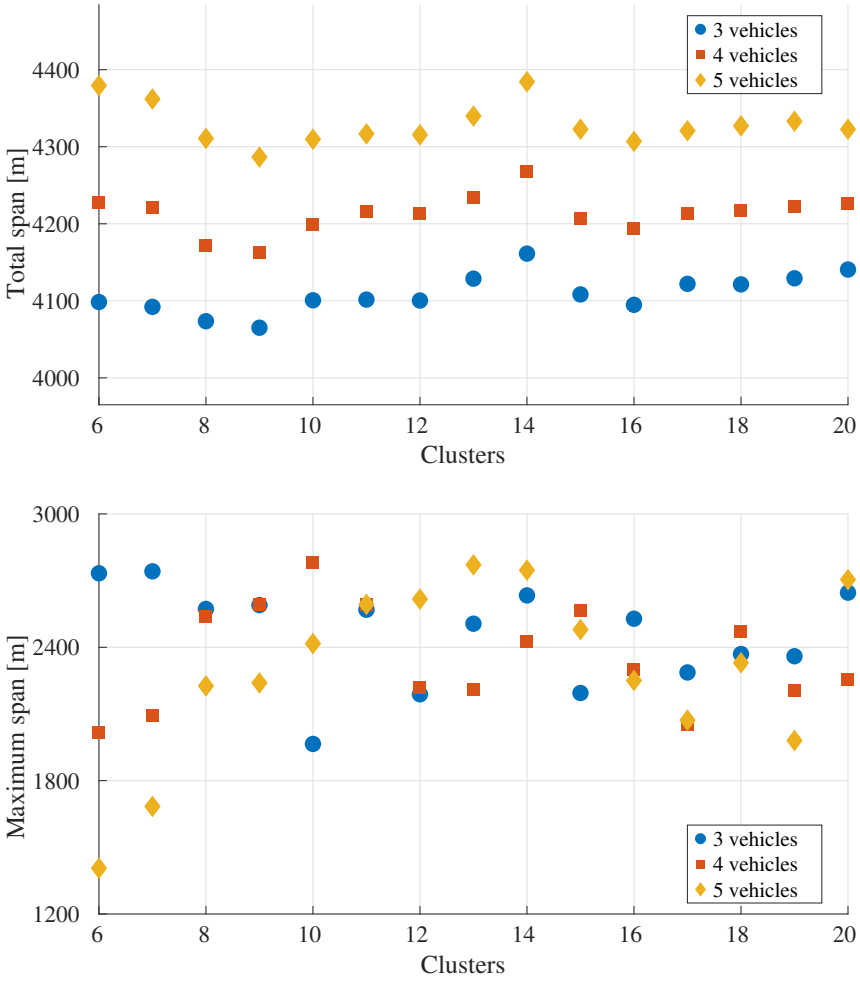

Figure 2.7: *Sum of the lengths (top) and maximum (bottom) across all trajectories as a function of both the number of clusters and the number of drones, UCLA environment. The sum of all lengths reliably increases with the number of vehicles because the solver was forced to use them all in the solution, therefore they all have to leave and go back to the starting point.*

clusters beyond a certain number, at least with respect to the total trajectories' length. This also suggests that clustering still offers a sufficiently detailed representation of the environment to obtain near-optimal solutions. Indeed, it would be faster to solve the intra-cluster trajectory planning problem first and the inter-cluster drone assignment second, because in this case the TSP problem would only be solved once [21]. However, with such an approach the entry and exit points of each cluster (recall 2.26) would not be chosen optimally, leading systematically to worse solutions, whose quality decreases further as the number of clusters increases. Furthermore, according to our simulations, the difference between the optimal intra-cluster path length as calculated in the third step of procedure 1 and the estimate $\hat{w}_i$ obtained as solution of (2.10) is always lower than 1%.

We measure solution time directly in seconds instead of number of iterations of the optimization algorithm. On one hand, this makes the measurement dependent on the machine used to solve the problems, but on the other it makes CVRP and TSP solution times homogeneous and comparable quantities, even though the two problems are structurally different and they are tackled with separate solvers. Still, we expect our considerations to be applicable in general, meaning that the shape of the curves we identify in the graphs is tied to the number of iterations and it is machine-independent, only their particular numerical values are not.

While a low number of clusters k keeps the complexity of the CVRP within reasonable bounds, the opposite is true for the TSPs. In fact if the overall number of POI n is fixed, then the number of points per cluster, on average $\frac{n}{k}$, increases, thus originating TSP instances with more points. The computational time required to solve the CVRP and the TSP increases exponentially as the number of nodes in their respective graphs grows. Figure 2.8 illustrates this behaviour.

A large number of smaller clusters is beneficial for their distribution among drones, because it allows the path planner to better exploit the available capacity of each drone and to obtain a robust solution more easily. We expect the solution to tend to the globally theoretically optimal one as k increases. In fact at the extreme, where $k = n$, the CVRP problem coincides with the original problem, without clustering. Note how, in figure 2.8, even though the number of optimisation variables $|\bar{\mathcal{E}}|M$ is proportional to the number of drones M, the CVRP solution times do not reliably increase with M as they do with $|\bar{\mathcal{E}}|$ (and a similar behavior, but inversely proportional to k, is observed for the TSP solution times). This is because, even if the number of optimisation variables increases, the newly added ones are linked to the previous ones by mutual exclusion constraints, ensuring that

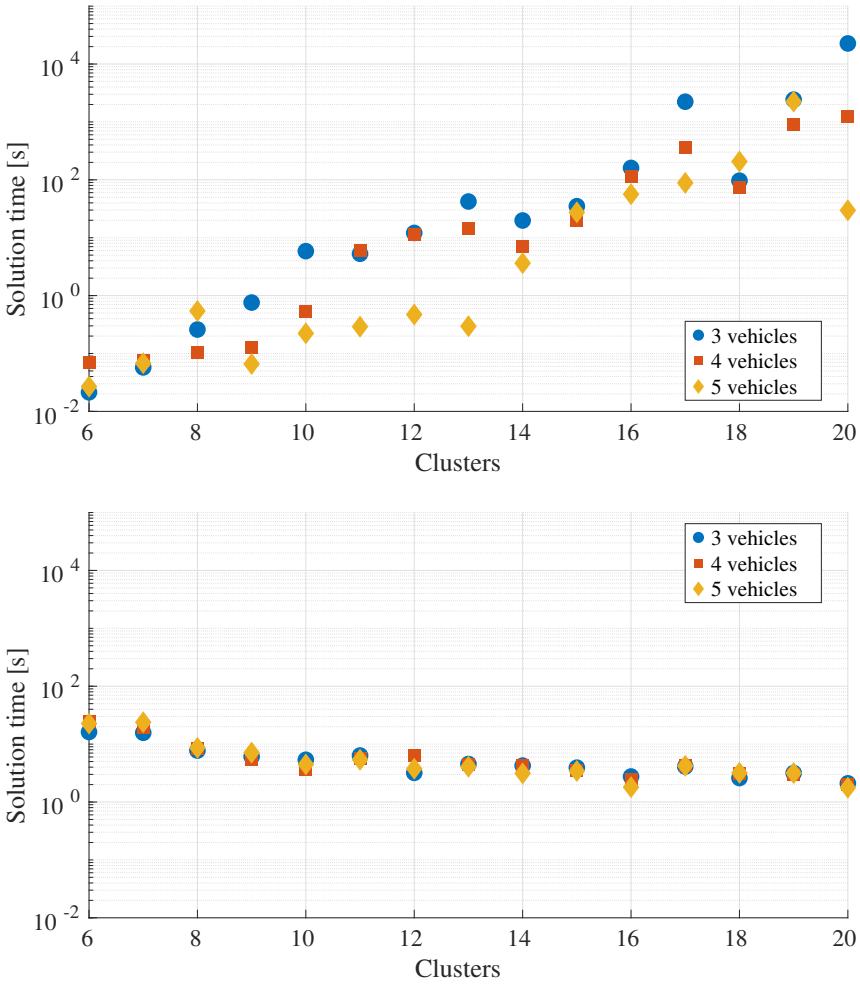

Figure 2.8: *Total solution time of CVRP problem (top) and TSP problems (bottom), as a function of both number of clusters and number of drones. In the latter, each data point represents the sum total of all the times required to solve all the TSPs for that instance of the problem. The y-scale is logarithmic.*

an edge is not assigned to two drones at the same time. As a consequence, the feasible space does not increase in size as much as the solution space, and solution time is not severely affected. The total makespan does increase with M instead, because the solver typically exploits all the drones it has available, and the path from the starting point to the structure has a non-negligible length.

By applying the connectivity minimisation procedure described in 2.1.5, we can reduce the solution time, in particular the part pertaining to the

CVRP. The quality of obtained solutions, as measured by the total length of generated trajectories (figure 2.9), is comparable to that obtained without reducing the degree of the nodes (figure 2.7), which we set to $\underline{m} = 4$. On the other hand, the solution time is significantly lower (figure 2.10), thanks to the lower number of optimization variables. This effect is particularly noticeable in instances with many clusters, where solution time decreases by three orders of magnitude. This is especially useful in the re-planning phase, which otherwise could take even more time than the first planning, as figure 2.11 suggests. This behavior is due to the total available capacity of the remaining vehicles being close to the capacity needed to finish the mission. More generally, we argue that when such a situation occurs, capacity constraints become active and significantly reduce the size of the feasible space, thus also increasing solution time. This is also apparent from figure 2.13.

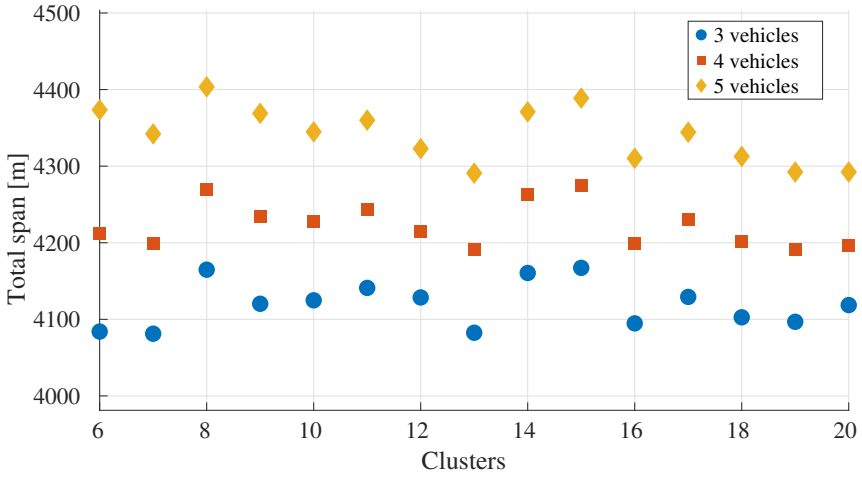

Figure 2.9: *Total trajectory length with graph connectivity minimisation procedure, UCLA environment. This is comparable to figure 2.7, hence we conclude solution quality is not affected significantly.*

With regards to the min-max formulation, figure 2.14 shows that it does divide the workload across vehicles more evenly than the previous one, but minimising cost function (2.29), i.e. minimising mission time, leads to a significantly increased complexity and solution time (see figure 2.12), due to the BILP becoming a MILP. Adopting the aforementioned complexity reduction approach, though, is helpful in this case too: solution times are decreased to values that make min-maxing valuable in a real-time framework, especially in the re-planning phase, which is apparent from figure 2.15. The significant difference between the times taken to plan and re-

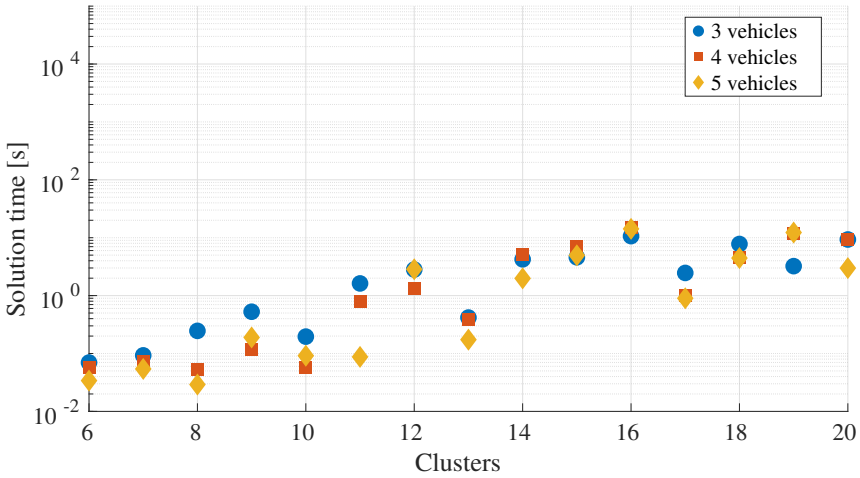

Figure 2.10: *CVRP solution times with graph connectivity minimisation procedure, UCLA environment. Especially with a high number of clusters, this is orders of magnitude faster than the fully-connected case (figure 2.8). Note how the growth trend is less than linear, as a fixed number of edges per node means the number of optimisation variables grows linearly, not exponentially, with the number of clusters.*

plan the mission can be explained by the removal of robustness constraints (2.17d) and (2.17e) on capacity in the formulation of the latter problem.

On the ideal number of drones

All the scenarios were simulated with a number of drones $M = 3$, 4 and 5, to explore the effect of this variable on the total makespan, maximum trajectory length and solution time of the CVRP. Furthermore, in each case the solver was forced to assign at least one cluster to each drone, so as to employ them all. As apparent from Figures 2.7 (top) and 2.9, the first direct consequence is that the total makespan increases with M regardless of the number of clusters k and of whether the graph connectivity minimization procedure is applied. This is because each drone must leave the starting point, go to the structure and come back. The effect on the maximum length of a trajectory assigned to a drone, instead, is unclear, but we note that M seems not to influence it significantly (see Figure 2.7 (bottom)). Similarly, the effect of M on the CVRP solution time seems negligible (see Figures 2.8 (top), 2.10, 2.11). The number of optimization variables in an instance of the CVRP problem is $|\bar{\mathcal{E}}|M$, with $|\bar{\mathcal{E}}|$ being the number of edges and M the number of vehicles. Increasing M thus also increases the number of optimization variables. Nevertheless, the solution time of the CVRP problem is not significantly increased, because when adding a new vehicle, all

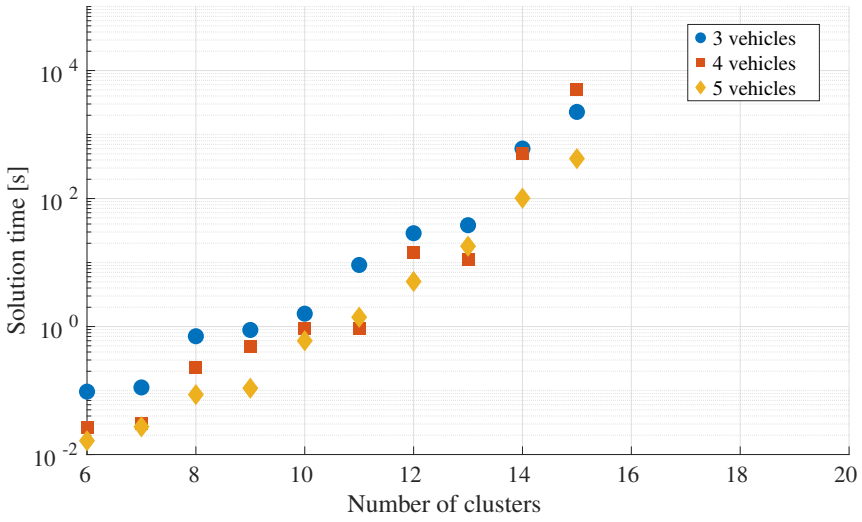

Figure 2.11: *CVRP time for solving the re-planning problem, in the same conditions as figure 2.8, i.e., with fully connected global graph. These simulations were carried out by always imposing that the drone with the longest assigned trajectory fails after visiting the first cluster assigned to it. This means every other drone will finish visiting the first cluster assigned to it and the rest of its trajectory will be re-planned. Due to the long required time, simulations with many clusters were not carried out.*

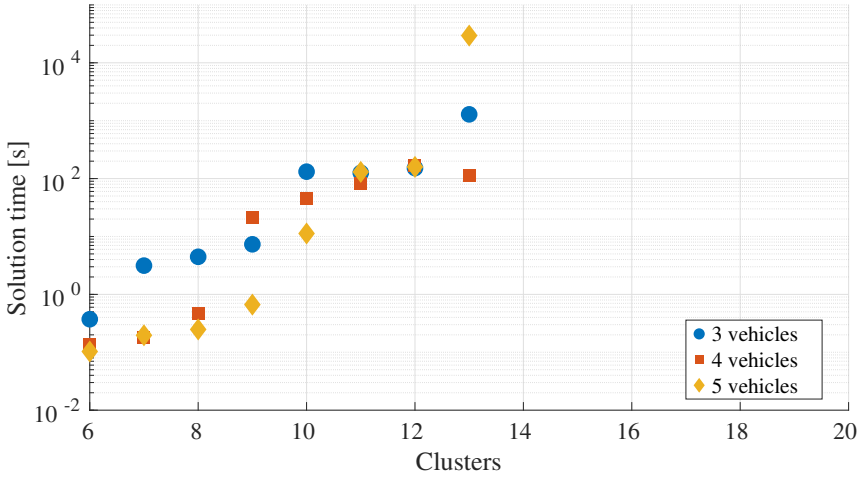

Figure 2.12: *CVRP solution times in min-max formulation, UCLA environment. Note how they rise much quicker with respect to the number of clusters. Simulations with many clusters were not carried out.*

of newly introduce the decision variables are tied to the ones of the other vehicles by mutual exclusion constraints. In other words, a cluster can only

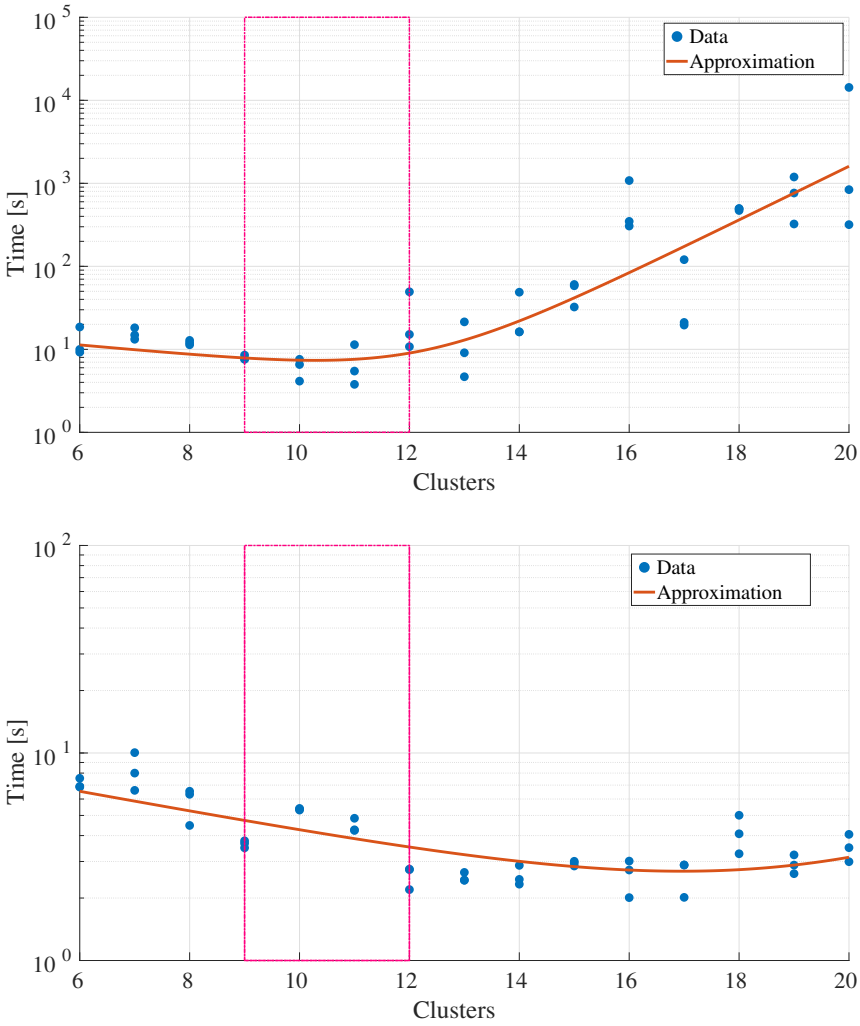

Figure 2.13: $t_{tot} = t_{CVRP} + t_{TSPs}$, *as measured in simulations with the first environment, the UCLA building (top) and with the second environment, the Spino d'Adda station (bottom). The ideal value range of* k *is highlighted by dashed line.*

be assigned to one vehicle, regardless of the total number of vehicles. As a consequence, the space of feasible solutions does not increase significantly, and the search performed by the solver in such space does not take significantly more time.

In the application of this method to a real mission planning task, the number of drones M is an arbitrary variable chosen by the designer, likely by finding the ideal trade-off between two criteria: on one hand the push to use the minimum amount of UAVs that can get the particular mission

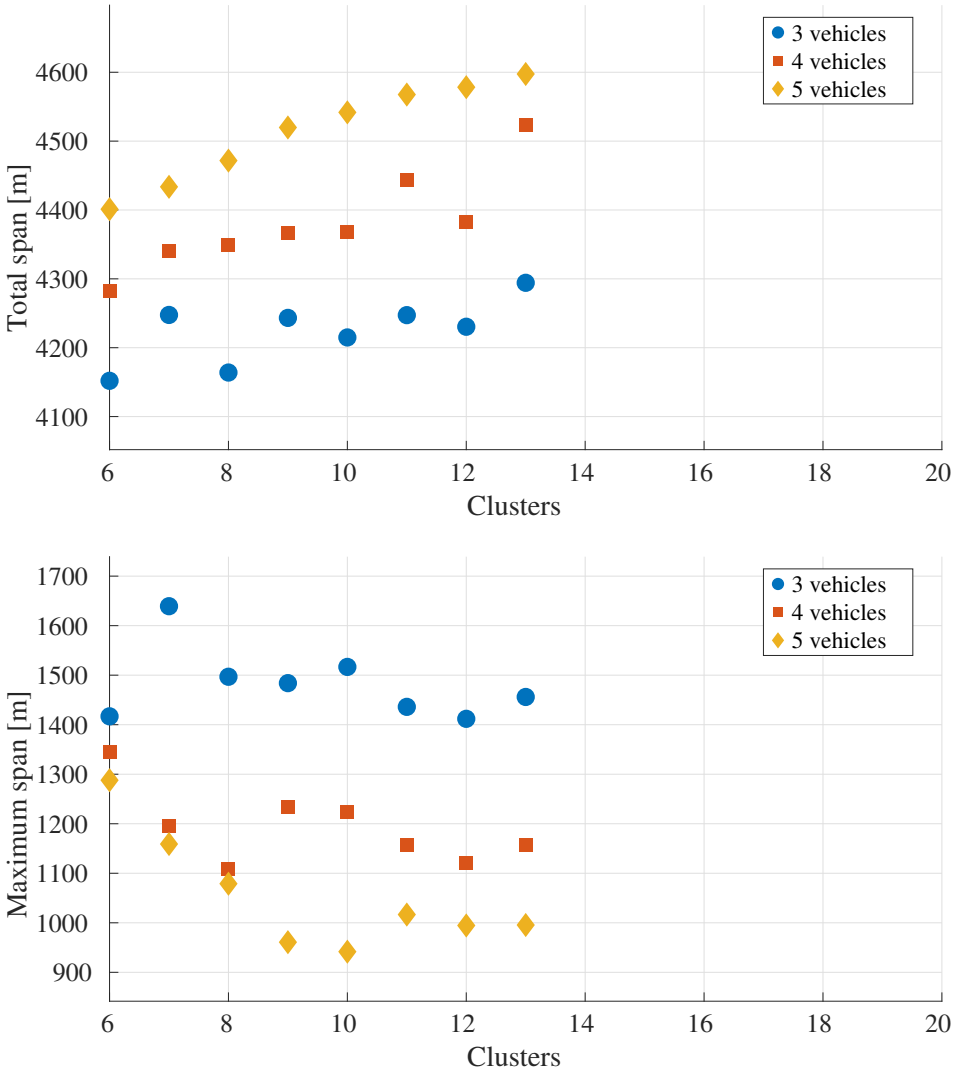

Figure 2.14: *Total (top) and maximum (bottom) lengths of trajectories assigned to drones with min-max formulation, UCLA environment. Workload is distributed as evenly as possible across vehicles. Simulations with high number of clusters were not carried out.*

at hand done, on the other hand the necessity of using more to guarantee that the mission will be completed regardless of the possible failure of one aircraft. The higher cost of buying, maintaining, charging and flying additional drones, in terms of both money and time, is the price of a robustness guarantee. The ideal value of M, therefore, is the minimum number for which a feasible solution to (2.17) exists.

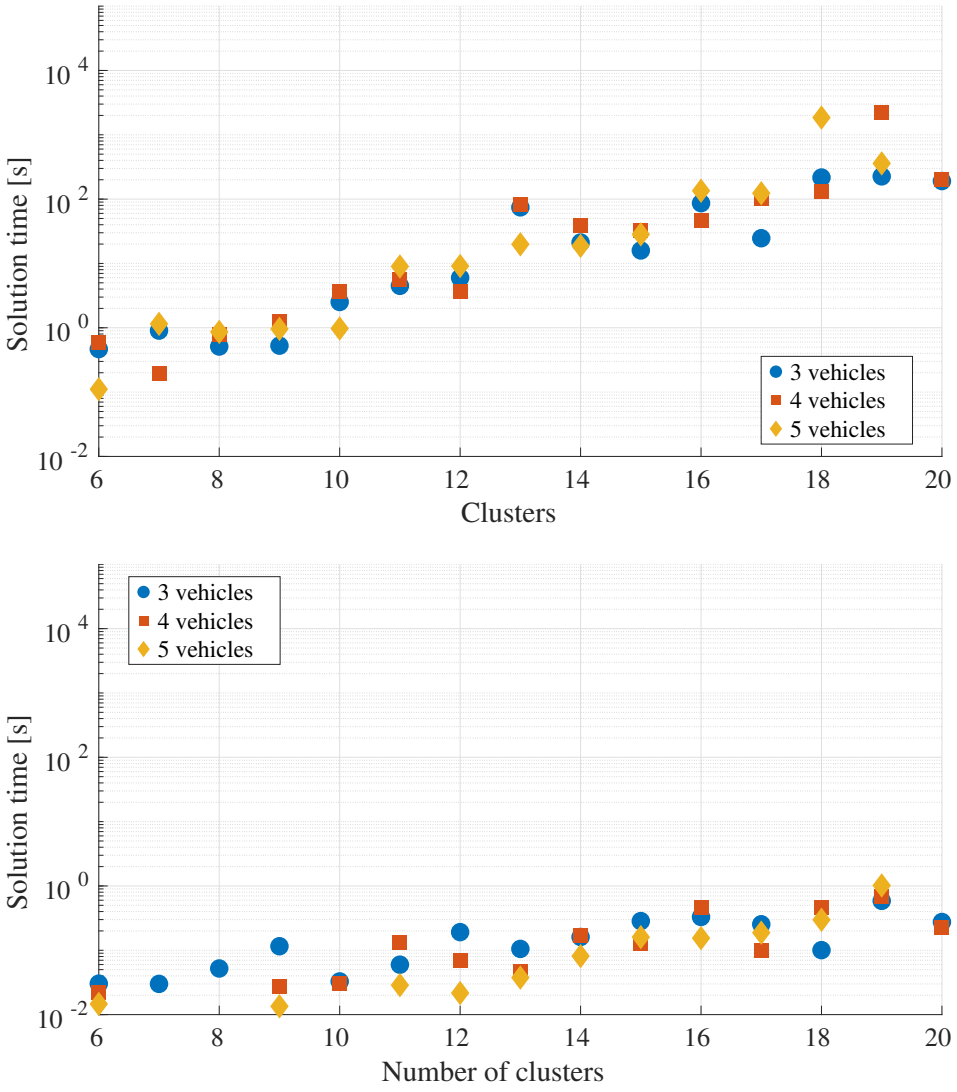

Figure 2.15: *CVRP solution times in min-max formulation with graph connectivity minimisation, UCLA environment. First planning (top) and re-planning after failure (bottom). Note that the distinction between the number of involved drones has been dropped. With this procedure, min-maxing becomes a viable strategy, especially for re-planning.*

On the ideal number of clusters

As we have shown, the arbitrarily chosen number of clusters significantly impacts the solution times of both the CVRP and the TSP problems. The former tends to increase exponentially, whereas the latter decreases with a

similar behaviour (figure 2.8). We can thus approximate the total time it takes to solve the two problems as the sum of two exponential functions:

$$t_{tot}(k) = e^{\alpha k+\beta} + ke^{\gamma\frac{n}{k}+\delta}, \tag{2.32}$$

where n represents the number of UAV, k is the number of clusters and $\alpha, \beta, \gamma, \delta$ are unknown parameters. If each of the two contributions is non-negligible with respect to the other, at least for some values of k, then it is sensible to look for an ideal number $\bar{k}$ of clusters which minimizes the overall time spent solving the optimization problems. In fact under this assumption function (2.32) must have a minimum with respect to the value of k.

$$\bar{k} = \arg\min_{k\in\mathbb{N}^+} t_{tot}(k). \tag{2.33}$$

Such a minimum does not have an explicit form, yet it can be computed numerically by imposing null derivative:

$$\frac{\partial t_{tot}}{\partial k} = \alpha e^{\beta}e^{\alpha k} + e^{\delta}e^{\gamma\frac{n}{k}}\left(1-\gamma\frac{n}{k}\right) = 0. \tag{2.34}$$

The solution to this problem is unique, since the cost function (2.32) after relaxing $k \in \mathbb{R}^+$ is convex in the considered domain (as it can be inferred by its curvature, which is positive everywhere for positive k). Moreover, it is useful to look for rules of thumb that guide the choice of its value *a priori*. First, it must be $k \geq M$, for the problem to be well-defined. Furthermore, if $k = M$ there is at least a trivial solution: assign one cluster to each drone, checking that capacity constraints are not violated. In order to better exploit the capacity of the drones, it makes sense to choose $k \geq 2M$, so that the planner can profit from a more fine-grained representation of the environment, the residual capacity (2.16) can be distributed more evenly and more robust solutions become available. On the other hand, in the limit case $k = n$ (i.e. each UAV is itself a cluster) the cluster assignment problem is reduced to the non-hierarchical case, and the local TSP instances lose meaning. Anyway we expect that even for some $k = k_{lim} \ll n$ the exponentially increasing complexity of the CVRP overcomes the exponentially decreasing one of the TSPs and the overall solution time becomes impractical. Given the results of our simulations, we observe that $k \in [3M, 4M]$ is a good choice in terms of both solution time and solution quality, but this might depend on the characteristics of the problem, such as the overall number of POI n: with the same number of clusters, increasing n leads to longer times for solving the TSP instances.

A comparison between the two presented environments (figure 2.13), though, highlights that the CVRP solution time can vary significantly across

instances of the overall problem. The solver we adopted, in fact, reliably takes more more to find a solution when capacity constraints are demanding with respect to the necessary span, which is due at least in part to such constraints substantially reducing the size of the set of feasible solutions for the CVRP. The two environments differ in scale: given the same amount of UAV, we expect optimal trajectories to be shorter if the points are more densaly placed in a region. In fact, though the number of UAV is the same, the sum of the lengths of the generated trajectories in the UCLA case exceeds 4000 m, whreas in the Spino case it is lower than 1000 m. Provided the maximum capacities $\bar{Q}_m$ are the same, capacity constraints (2.17d) and (2.24) are not limiting in the second case, hence the feasible space is larger. Furthermore, the former environment is such that the solver tends do find more sub-tours before identifying a feasible solution as optimal, which implies the problem must be solved several times due to the sub-tour elimination strategy we adopted (see Section 2.1.5). In conclusion, the ideal number $\bar{k}$ of clusters depends heavily on the instance of the problem, and in particular on the capacity of the drones with respect to the overall distance that needs to be travelled. It is reasonable to assume that in the worst case scenario the capacity is critically limited, therefore the related constraints are active and the CVRP solution time grows significantly. The opposite would mean that more vehicles are available than actually needed, even for a robust solution. Finally, the ideal number of clusters $\bar{k}$ also depends on the number of UAV n, as t_{TSP_s} depends on the average number of UAV per cluster $\frac{n}{\bar{k}}$. By inspection of equation 2.32, as n increases we expect $\bar{k}$ to grow too. In fact, since t_{TSP} depends on $\frac{n}{\bar{k}}$, it is possible to predict how the TSP solution time will change if n changes.

2.1.7 Conclusion and Future Work

We presented an automated approach to plan data-gathering missions with multiple drones in reasonable time, guaranteeing robustness to single faults. This is achieved by introducing a hierarchical separation between high level planning and locally optimal path generation, which makes the problem tractable in a reasonable amount of time without sacrificing solution quality. We proposed an alternative cost function that minimizes mission time overall, under the assumption that all vehicles work in parallel. We addressed the issue of robustness with respect to a single point of failure, providing a proof that missions can be re-planned and carried out to completion should one of the drones fail. We also discussed implementation aspects and provided an overall procedure with related available code. Finally, we analysed the performance of our algorithm in two different environments modeling real buildings, and discussed the possibility of reducing solution time through the choice of an *a priori* optimal number of clusters.

Possible directions for future research are concerned with the development of ad-hoc graph generation methods that are guaranteed to yield nodes with fixed degree, the use of specific algorithms [52, 53] that are known to have polynomial complexity, and the development of a tailored sub-tour elimination strategy for instances of the CVRP problem with a high number of clusters.

CHAPTER 3

Tethered Multicopter Networks

3.1 LiDAR-based Autonomous Flight for a System of TEthered Multicopters[2]

In the previous chapter, a strategy for tackling the limited flight time problem was detailed. A second approach is possible: powering drones through tethers. While introducing a physical connection limits the vehicle's freedom of movement, it enables it to hover for a potentially unlimited amount of time. To gain back some of the agility, multiple UAVs can be connected in series, creating a System of TEthered Multicopters (STEM). The agility and long flight time come at the cost of a more complex multi-agent control problem. The tether also anchors the system to the ground, making it inherently safer with respect to fly-away risks and in terms of land area potentially harmed as a consequence of a fault and crash event.

3.1.1 State of the Art

In some applications where uninterrupted operation of the vehicle is more important than its agility, systems of drones powered through electrical ca-

[2]This work was previously published by the author in a different form, see [23].

Figure 3.1: *A STEM prototype with drones during a field test.*

bles have already been explored [23, 54, 55] and commercial products already exist [56]. The autonomous operation of a STEM poses novel control challenges regarding movement planning and execution, regarding collision avoidance in particular. In fact, both the drones and the tethers must be kept away from obstacles, and only the former are directly controllable. Furthermore, tethers constrain the movement of the system significantly. Addressing the resulting challenges in a systematic way is the aim of this work. Figure 3.1 shows a real-world example of such a system in action in a test site close to Politecnico di Milano, Italy. The topic of trajectory generation for individual vehicles has been extensively researched, both in an on-line framework (MPC [57, 58], Reinforcement Learning [59]) and in an off-line one (A^*, Potential Fields [60]). MPC can handle multiple inputs and constraints, introduced to guarantee safety. Regarding multi-aircraft tethered systems, a few designs and control procedures have been proposed, dealing with control of the UAVs [55, 61, 62] and of tether tension force [63]. In some works, the tethers are assumed to have fixed

length [61, 63], whereas in others they are extendable [55], but the overall control challenges are the same. A popular idea throughout the literature is that such a system can be conceptualized as a robotic manipulator, where drones act as three-dimensional spherical joints and tethers as mono-axial prismatic ones. Exploiting this similarity, a method has been proposed for path planning in a known and static environment [64], borrowing known techniques from classical robotics. This research builds on a novel tethered multi-drone system first described in proposed in [65]. It consists of two or more drones, tethered to each other and to a ground station in a series topology.

The problem at hand can be classified as a collision avoidance one, which has been tackled in literature with *planning* techniques, such as graph search ([66–68]), Rapidly-exploring Random Tree (RRT) ([69]), and potential fields ([70]), and *reactive* techniques, e.g., [71], [72]. The existing approaches, though, do not solve all the issues at hand:

1. the presence of the tethers, that mechanically couple the drones and must be taken into account as well in the obstacle avoidance functionality;
2. the unknown nature of the environment;
3. the reliance on real-time LiDAR readings only.

In the following sections, two slightly different approaches to solve these issues will be presented, based on a real-time algorithm that exploits the information from the sensors and transforms it into polytopic constraints for an optimization problem. Constraints provide a local convex approximation of free space at each time step, marking a safe area for the vehicles and their tethers to traverse. The non-linear shape of the tether in the vertical direction is also taken into account through constraints. Finally, an optimization problem is solved calculate a reference position, then fed to each drone. A model of the tether as a nonlinear multi-body dynamical system is also introduced to perform simulations.

3.1.2 System model and problem formulation

The model of the system at hand is represented in Figure 3.2. It consists of a set of three of drones connected in a series topology. The aim of the high-level controller, described in the following sections, is to guide the leader drone, *i.e.* the farthest one from the ground station, to an arbitrarily specified position in 3D space. The centralized controller operates in discrete time with a sampling period T_s. At each time instant, it elaborates the

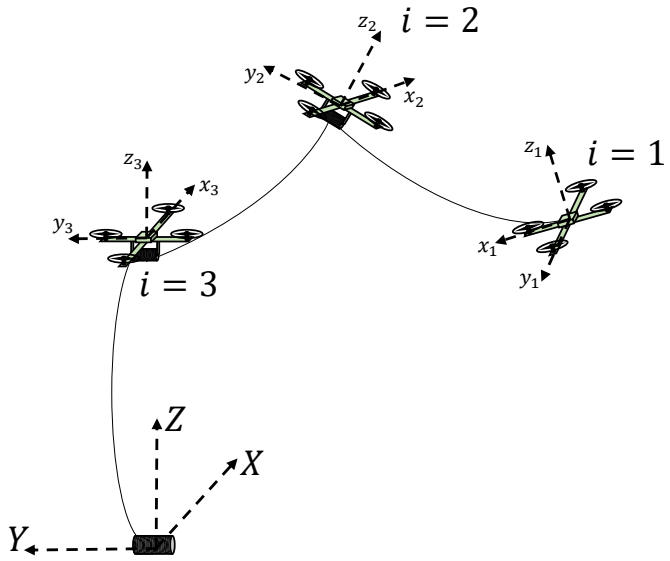

Figure 3.2: *Model of a considered STEM system with three drones.*

latest LiDAR readings and calculates a suitable position reference for each drone.

Multicopter drones

Each UAV is endowed with a local position controller, which is fed a position reference by the centralized computer. Six Degrees of Freedom (DOF), nonlinear, and continuous-time models of the drones, are employed for simulation and described in [65]. The procedure can support an arbitrary number $N \in \mathbb{N}$ of drones connected in series, identified by indexes $i = 1, 2, ..., N$. As apparent from Figure 3.2, the leader drone is denoted with the index 1. $\boldsymbol{p}_i$ and $\dot{\boldsymbol{p}}_i \in \mathbb{R}^3$ describe the position and velocity vectors of the i-th drone in a world-fixed, right-handed inertial reference frame (X, Y, Z), with Z pointing up, where $\boldsymbol{p}_i = [p_{X,i}, p_{Y,i}, p_{Z,i}]^T$ and $\cdot^T$ is the matrix transpose operation. Without loss of generality, the origin of the reference frame is the position of the ground station. Winches are present on the ground station and on all drones except the leader. Each winch control the length of the tether coiled around it. L_i is the reeled-out length of the tether from drone i to drone $i+1$. L_i is thus known to the position controller of the drone $i+1$. Aircraft and winches are locally controlled so that UAVs track a reference position vector, $\boldsymbol{p}_{ref,i}$, and tether tension forces are always within suitable bounds. A controller for the winches is proposed in [65].

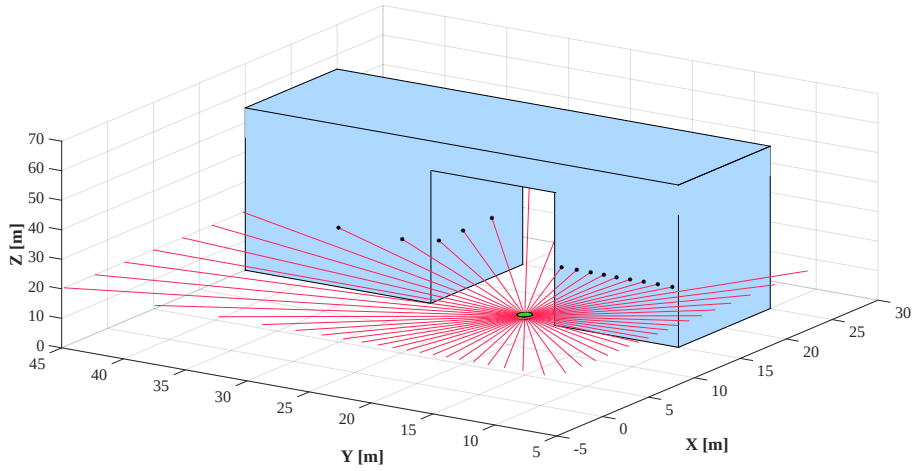

Figure 3.3: *A LiDAR scan along the horizontal plane in simulation. The UAV is represented as a green circle, obstacles in light blue. Red lines represent laser beams.*

LiDAR sensors

Planar LiDAR sensors with a field of view $\Theta = 360°$, are considered, so that each sensor scans an entire plane without blind areas. The sensors have angular resolution θ_s, *i.e.* the angle between two consecutive laser beams. A sensor scan is thus collected in a vectorr $\boldsymbol{s} \in \mathbb{R}^M$ with $M = \lfloor \frac{\Theta}{\theta_s} \rfloor$ entries $s(j)$, with $j = 0, \ldots, M-1$, each one measuring the distance of the closest obstacle along direction $j\theta_s$. For instance the measured obstacles positions can be described in Cartesian coordinates in the horizontal plane XY as:

$$\begin{aligned} X_j^{obs} &= s_i(j)\cos(j\theta_s) \\ Y_j^{obs} &= s_i(j)\sin(j\theta_s); \quad j = 0, \ldots, M-1, \end{aligned} \tag{3.1}$$

as represented in Figure 3.3.

To simulate the sensors, the vector $\boldsymbol{s}$ is obtained via a collision detection routine between the M segments originating from the drones and directed along the aforementioned directions and the obstacles, defined as sets of polytopes. The sensors have maximum detection range s_{max} (30 m for the LiDARs considered here), therefore no obstacle is detected within this range if its distance is greater that the threshold, and as a result $s(j) = \infty$. Each drone i mounts two planar LiDAR sensors with $\Theta = 360°$, one in the horizontal plane, one in the vertical. The sensors' attitude is assumed stable with respect to the inertial frame notwithstanding the drone's motion. This

is practically achievable by mounting them on stabilizing gimbals. With reference to Figure 3.2, the horizontal plane is always parallel to (X, Y) and has height equal to the drone's $Z-$coordinate, while the vertical one, is always perpendicular to (X, Y) and can be rotated through the gimbal's yaw angle. Note that this only allows the aircraft to gather information on obstacles within either of the two planes, but not in the remaining space. The choice of LiDARs for navigation is advantageous because they are fast, hence it is possible to use them in real-time scenarios, they are active sensors, thus they work in conditions where light and contrast are scarce or excessive, and they retain accuracy at a distance. This makes them ideal candidates for a real-time, reactive-only control approach, which does not rely on prior knowledge or on the construction of a representation of the surrounding space, but only on the interpretation of data available at each sampling period.

Tethers

The tethers not only transfer electrical power from the ground source to the vehicles, but also establish a power line communication network, enabling a centralized control of the system. Their length is controlled through onboard winches. Accounting for the sag in the vertical direction is necessary to develop and test a functional navigation algorithm in a cluttered environment. Tethers are modeled as multi-body entities, whereby the $i-$th tether is a chain of N_t inner nodes with mass $m_{t,i}$ (see Figure 3.4), computed as:

$$m_{t,i} = \frac{L_i \rho_t}{N_t}, \tag{3.2}$$

where ρ_t is the tether linear density. Each node has a position and a velocity, $\boldsymbol{p}_{t,i,l}, \dot{\boldsymbol{p}}_{t,i,l} \in \mathbb{R}^3, l = 1, \ldots, N_t$. The extreme points are fixed to the drones or to the ground station, therefore there are $N_t + 1$ segments. Inner nodes are subject to their own weight and to the forces applied by the two adjacent tether segments. Elastic and internal friction forces are considered, neglecting the aerodynamic ones by assuming that the relative speed between the tethers and air, due to wind and motion, is negligible. The tether segments are modeled as parallel pairs of linear dampers and nonlinear springs, only applying forces when the tether segment is taut. To model the inner tether forces, the nominal length and spring constant of each segment, $\ell_{t,i}$ and $k_{t,i}$,

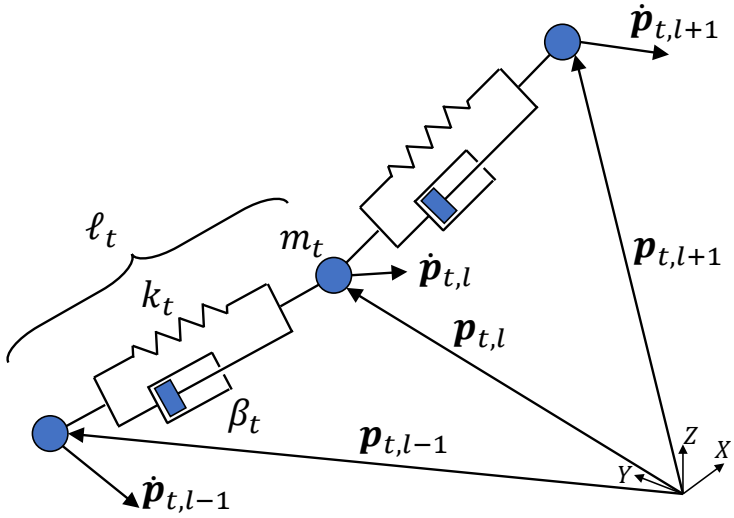

Figure 3.4: *Multi-body tether model employed for the numerical simulations and to compute the map employed for constraint tightening in the navigation algorithm.*

are computed as:

$$\begin{aligned} \ell_{t,i} &= \frac{L_i}{N_t + 1} \\ k_{t,i} &= \frac{\overline{F}_t}{\overline{\varepsilon}_t \ell_t} \end{aligned} \tag{3.3}$$

where $\overline{F}_t$ is the breaking load of the tether and $\overline{\varepsilon}_t$ the corresponding maximum elongation. Finally, β_t is the constant friction coefficient of each tether segment. These parameters are needed to compute the forces acting on the nodes. Considering node l, the applied nonlinear elastic forces read:

$$\begin{aligned} \boldsymbol{F}_{e,i,l+1} &= \max\left(0, k_t\left(\|\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}\| - \ell_t\right)\right) \frac{\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}}{\|\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}\|} \\ \boldsymbol{F}_{e,il-1} &= \max\left(0, k_t\left(\|\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}\| - \ell_t\right)\right) \frac{\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}}{\|\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}\|}, \end{aligned} \tag{3.4}$$

where $\|\cdot\|$ denotes the 2-norm, and the saturation to zero implies that no elastic force is present when the distance between two nodes is smaller than the nominal segment length. The friction forces are supposed to act only in

axial direction and are computed as:

$$\begin{aligned}\boldsymbol{F}_{f,i,l+1} &= \beta_t \frac{\left(\dot{\boldsymbol{p}}_{t,i,l+1} - \dot{\boldsymbol{p}}_{t,i,l}\right)^T \left(\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}\right)}{\|\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}\|^2} \left(\boldsymbol{p}_{t,i,l+1} - \boldsymbol{p}_{t,i,l}\right) \\ \boldsymbol{F}_{f,i,l-1} &= \beta_t \frac{\left(\dot{\boldsymbol{p}}_{t,i,l-1} - \dot{\boldsymbol{p}}_{t,i,l}\right)^T \left(\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}\right)}{\|\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}\|^2} \left(\boldsymbol{p}_{t,i,l-1} - \boldsymbol{p}_{t,i,l}\right)\end{aligned} \tag{3.5}$$

For the l-th node, the equations of motion thus read:

$$\ddot{\boldsymbol{p}}_{t,i,l} = \frac{\left(\boldsymbol{F}_{e,i,l+1} + \boldsymbol{F}_{e,i,l-1} + \boldsymbol{F}_{f,i,l+1} + \boldsymbol{F}_{f,i,l-1}\right)}{m_t} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.6}$$

where g is the gravity acceleration, $\boldsymbol{F}_f$ represents friction forces and $\boldsymbol{F}_e$ elastic ones The position and velocity vectors of the first and last nodes, $l = 1$ and $l = N_t$, are known as those of the tether ends, thus equal to the position and velocity vectors of the devices attached there. This time-varying, nonlinear dynamical tether model is employed both in simulations (see Section 3.1.4) and in the pre-computation of a map which estimates the height differential $\underline{Z}_i$ between the lowest point of the tether's catenary, assumed at steady-state, and the drone to which it is attached (see Figure 3.5).

$$\underline{Z}_i = f(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}, L_i) \tag{3.7}$$

Such a map, which is rather accurate when the drones are moving at low speed, is useful to tighten the constraints used by the navigation algorithm, in order to avoid an impact between the tether (also accounting for its catenary) and the obstacles.

Problem Formulation

The goal of the system is to reach an arbitrarily defined point of interest with the first drone (leader) of the chain. The other drones in the formation will adapt their trajectories to allow the leader maximum freedom of movement, while ensuring obstacle avoidance for themselves and the tethers. Since obstacles are unknown *a priori*, the centralized navigation algorithm can rely only on the partial, real-time information gathered by each UAV. The supevisory controller's role is highlighted in Figure **??**. Its inputs are the positions of the drones $\boldsymbol{p}_i$, $i = 1, \ldots, N$, the tether states L_i, $i = 1, \ldots, N$, and the LiDAR readings, $\boldsymbol{s}_i$, $i = 1, \ldots, N$ gathered by the drones, while it outputs reference position vectors $\boldsymbol{p}_{ref,i}$, $i = 1, \ldots, N$.

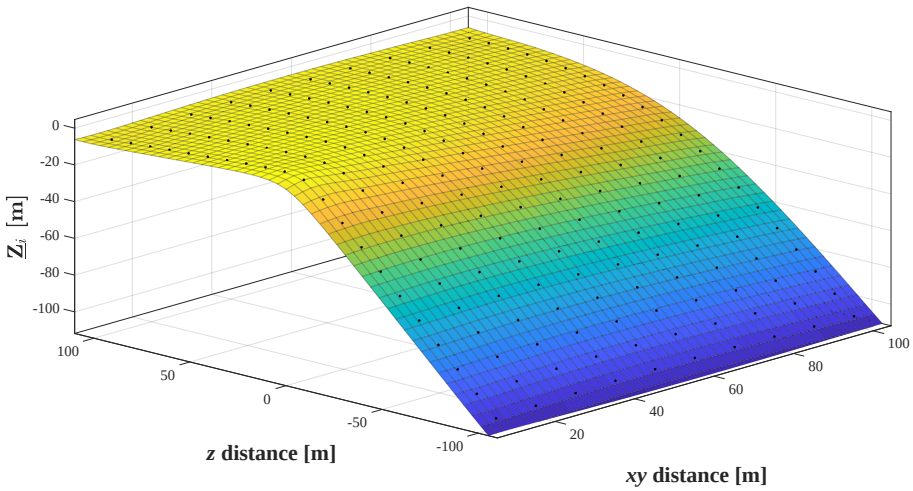

Figure 3.5: *Visualization of look-up function used to take into account the tether catenary in the proposed navigation algorithm. Dots represent the vertical distance between a drone and the lowest point of the cable (Δz) as a function of distance along the (X, Y) plane and Z difference of the two tether ends. Essentially, this is a representation of the function describing how far lower the lowest point of the cable is with respect to a drone, given the positions of the two drones the cable hangs from.*

The problem at hand is the design of such a supervisory controller such that the leader drone reaches its destination $\boldsymbol{p}_{poi}$ in the inertial reference (X, Y, Z), while obstacle avoidance is ensured.

3.1.3 Real-time navigation and obstacle-avoidance algorithm for STEM

The vector of decision variables for the optimization problem is $\boldsymbol{x} = [\boldsymbol{p}_{ref,1}^T, \ldots, \boldsymbol{p}_{ref,N}^T]^T \in \mathbb{R}^{3N}$, and a matrix $\bar{\Lambda} = \text{diag}(\Lambda_1, \ldots, \Lambda_N) \in \mathbb{R}^{3N \times 3N}$ is defined, where $\text{diag}(\cdot)$ is a block-diagonal matrix and, for $i = 1, \ldots, N$, Λ_i are 3×3 diagonal matrices with the tuning parameters $\lambda_i \in (0, 1)$ on the diagonal. The navigation algorithm performs the following steps:

1. collect the data regarding drone positions, tether lengths and sensor readings: $\boldsymbol{p}_i$, L_i, and $\boldsymbol{s}_i$, $i = 1, \ldots, N$;
2. calculate suitable *goals* for each drone, in the form of position vectors $\boldsymbol{p}_{goal,i} \in \mathbb{R}^3$ in the inertial frame. Collect the goals in vector $\boldsymbol{x}_{goal} = [\boldsymbol{p}_{goal,1}^T, \ldots, \boldsymbol{p}_{goal,N}^T]^T$;
3. build a matrix $A \in \mathbb{R}^{p \times 3N}$ and vector $\boldsymbol{b} \in \mathbb{R}^p$ that describe free space as a polytopic region, to be used as constraints on the position references;
4. Obtain $\boldsymbol{x}^*$ as:

$$\begin{aligned} \boldsymbol{x}^* = \quad & \arg\min_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_{goal})^T \bar{\Lambda} (\boldsymbol{x} - \boldsymbol{x}_{goal}) \\ & \text{subject to} \\ & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned} \tag{3.8}$$

5. Feed the optimal position references in $\boldsymbol{x}^*$ to the corresponding drones, repeat from (1) at the next time step.

The optimization problem (3.8) is a strictly convex Quadratic Program (QP) of small size with diagonal Hessian $\bar{\Lambda}$ The problem is solvable in very short times due to its properties [73, 74], therefore it is suitable for real-time application. The parameters γ_i in the cost function weigh the distance of the different drones drone from their goals. A large weight γ_1 prioritizes goal-reaching for the leader over the followers, should the two objectives be in direct contrast. The core of the approach is the computation of the goals $\boldsymbol{p}_{goal,i}, i = 1, \ldots, N$ and of A and $\boldsymbol{b}$. The goal for the leader drone is always its final destination $\boldsymbol{p}_1^{goal} = \boldsymbol{p}_{poi}$. For the followers, goal $\boldsymbol{p}_{goal,i+1}$

of drone $i+1$ is determined from the current position $\boldsymbol{p}_i$ and goal $\boldsymbol{p}_{goal,i}$ of drone i:

$$\boldsymbol{p}_{goal,i+1} = \boldsymbol{p}_{goal,i} - \bar{d}\frac{\boldsymbol{p}_{goal,i} - \boldsymbol{p}_i}{\left\|\boldsymbol{p}_{goal,i} - \boldsymbol{p}_i\right\|}, \tag{3.9}$$

with $\bar{d}$ and arbitrary, tunable distance parameter. Equation (3.9) formalized the intuition conveyed by Figure 3.6: vehicle $i+1$ should position itself behind drone i, on the line from latter to its goal, at a distance $\bar{d}$.

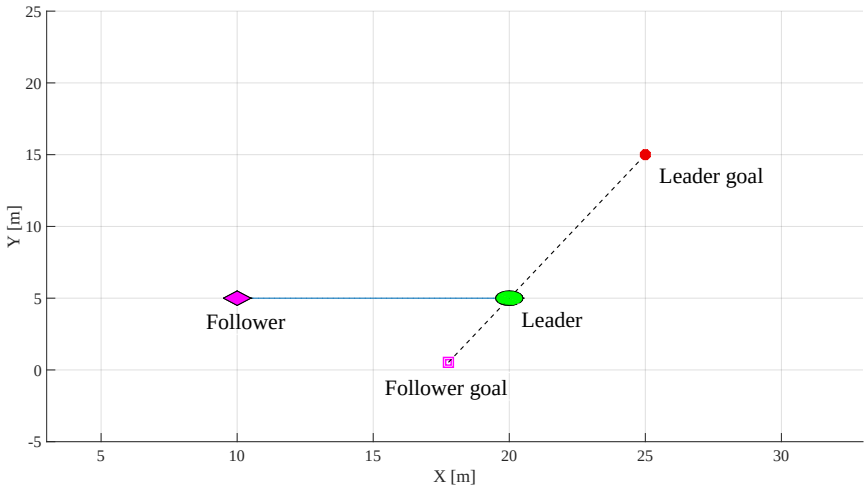

Figure 3.6: *Goal assignment on the (X, Y) plane. The goal of drone $i+1$ is lies at distance $\bar{d}$ (here 5 m) behind drone i on the line connecting $\boldsymbol{p}_{goal,i}$ and $\boldsymbol{p}_i$.*

Computation of constraints in 2D

Horizontal and vertical constraints are treated differently. The former are obtained by only considering the horizontal plane (X, Y). First, obstacle avoidance for the single UAVs is introduced. LiDAR readings $\boldsymbol{s}_i$ are infinite if no obstacle is detected along the corresponding direction within the maximum distance. It is then assumed that if the drone can move along one of the obstacle-free directions in $\boldsymbol{s}_i$ until the next sampling period. Furthermore if θ_s is small (*i.e.* the scans are dense), it is reasonable to assume that all sectors of space containing obstacle-free scans also represent obstacle-free areas (see Figure 3.3). The maximum distance s_{max} form Section 3.1.2 can be set to a very small value to aide navigation in cluttered environments. It is however necessary to make sure that, given the maximum speed of the drone (around 2 $\frac{m}{s}$ in our simulations), the vehicle cannot travel further than

s_{max} within the time allotted for the solution of the problem. The following indices are defined:

$$\begin{array}{rcll} \underline{j}_i & = & \min\limits_{j=1,\dots,M} & j \ : s_i(j) = \infty \\ \bar{j}_i & = & \max\limits_{j=\underline{j}_i+1,\dots,M} & j \ : s_i(k) = \infty, k = \underline{j}_i + 1, \dots, j, \end{array} \tag{3.10}$$

so that the pair $(\underline{j}_i, \bar{j}_i)$ identifies an angular sector of space in which LiDAR readings detect no obstacles. The sector can be described by two lines:

$$\underbrace{\begin{bmatrix} -\tan(\underline{j}_i\theta_s) & 1 \\ -\tan(\bar{j}_i\theta_s) & 1 \end{bmatrix}}_{A_i(\underline{j}_i,\bar{j}_i)} \begin{bmatrix} p_X \\ p_Y \end{bmatrix} = \underbrace{\begin{bmatrix} -\tan(\underline{j}_i\theta_s)p_{X,i} + p_{Y,i} \\ -\tan(\bar{j}_i\theta_s)p_{X,i} + p_{Y,i} \end{bmatrix}}_{\boldsymbol{b}_i(\underline{j}_i,\bar{j}_i)} \tag{3.11}$$

As long as $\theta_s \left(\bar{j}_i - \underline{j}_i\right) \leq \pi$, the obstacle-free sector can be described as a polyhedron :

$$A_i \begin{bmatrix} p_X \\ p_Y \end{bmatrix} \leq \boldsymbol{b}_i \tag{3.12}$$

and is therefore convex.

Remark 2. *Some considerations are necessary:*

1. *If $\underline{j}_i\theta_s$ and/or $\bar{j}_i\theta_s$ are equal to $\dfrac{\pi}{2}$ or $\dfrac{3\pi}{2}$, then the corresponding row of matrix $A_i\left(\underline{j}_i, \bar{j}_i\right)$ and vector $\boldsymbol{b}_i\left(\underline{j}_i, \bar{j}_i\right)$, should respectively be changed to $[1\ 0]$ and $p_{X,i}$;*
2. *If $N_c > 1$ obstacle-free, non-contiguous angular sectors are identified, then a N_c pairs $\left(\underline{j}_{i,l}, \bar{j}_{i,l}\right)$, $l = 1, \dots, N_c$, are extracted from $\boldsymbol{s}_i$ instead of one. One of those is eventually selected as:*

$$A_i = A_i\left(\underline{j}_{i,l^*}, \bar{j}_{i,l^*}\right), \ \ \boldsymbol{b}_i = \boldsymbol{b}_i\left(\underline{j}_{i,l^*}, \bar{j}_{i,l^*}\right) \tag{3.13}$$

where

$$l^* = \arg\max_{l=1,\dots,N_c} c_l \tag{3.14}$$

and

$$c_l = \left\| \begin{bmatrix} \cos\left(\underline{j}_{i,l}\theta_s\right) & \sin\left(\underline{j}_{i,l}\theta_s\right) \\ \cos\left(\bar{j}_{i,l}\theta_s\right) & \sin\left(\bar{j}_{i,l}\theta_s\right) \end{bmatrix} \begin{bmatrix} (p_{X,goal,i} - p_{X,i}) \\ (p_{Y,goal,i} - p_{Y,i}) \end{bmatrix} \right\|_\infty \tag{3.15}$$

Namely, (3.13)-(3.15) *select the angular sector whose borders feature the largest inner product with the vector pointing from the drone to its goal. Moving within this angular sector guarantees that the distance to the goal can be minimized more than it would be in any other one;*

3. *If* $\theta_s\left(\bar{j}_i - \underline{j}_i\right) > \pi$*, then the sector spanning the obstacle-free directions does not correspond to a convex set. In this case the line with smallest absolute inner product with vector* $(\boldsymbol{p}_{goal,i} - \boldsymbol{p}_i)$ *is dropped, thus selecting a half-plane within the non-convex angular sector. An example of this is shown in Figure 3.7.*

4. *If the goal lies between the drone and an obstacle, the sensor readings can be manipulated to enable movement in the angular sector where the goal lies.*

Figure 3.8 shows an example of this procedure being simulated.

After obtaining the constraints A_i, $\boldsymbol{b}_i$, $i = 1, \ldots, N$ for all drones, the overall constraint matrix can be assembled:

$$\bar{A}_i \begin{bmatrix} p_{X,ref,i} \\ p_{Y,ref,i} \end{bmatrix} \leq \bar{\boldsymbol{b}}_i \tag{3.16}$$

where $\bar{A}_N = A_i$, $\bar{\boldsymbol{b}}_N = \boldsymbol{b}_N$ and, for $i < N$:

$$\bar{A}_i = \text{diag}(A_i, A_{i+1}), \quad \bar{\boldsymbol{b}}_i = \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{b}_{i+1} \end{bmatrix}. \tag{3.17}$$

Namely, the position reference for each drone must lie inside the intersection of the obstacle-free regions obtained for the drone itself and of the next one in the series (*i.e.* its follower). This forces the two UAVs in line-of-sight of each other at all times, thus ensuring that the $i-$th tether connecting them does not impact obstacles (see Figure 3.9). Finally, A and $\boldsymbol{b}$ from (3.8) are obtained by considering constraints (3.16) altogether:

$$A = \text{diag}(\bar{A}_1, \ldots, \bar{A}_N), \quad \boldsymbol{b} = \begin{bmatrix} \bar{\boldsymbol{b}}_1 \\ \vdots \\ \bar{\boldsymbol{b}}_N \end{bmatrix}. \tag{3.18}$$

Computation of constraints in 3D

An extension to 3D is not straightforward, because the assumptions imply a pair of 2D LiDARs is used, which is not equivalent to a 3D sensor. This

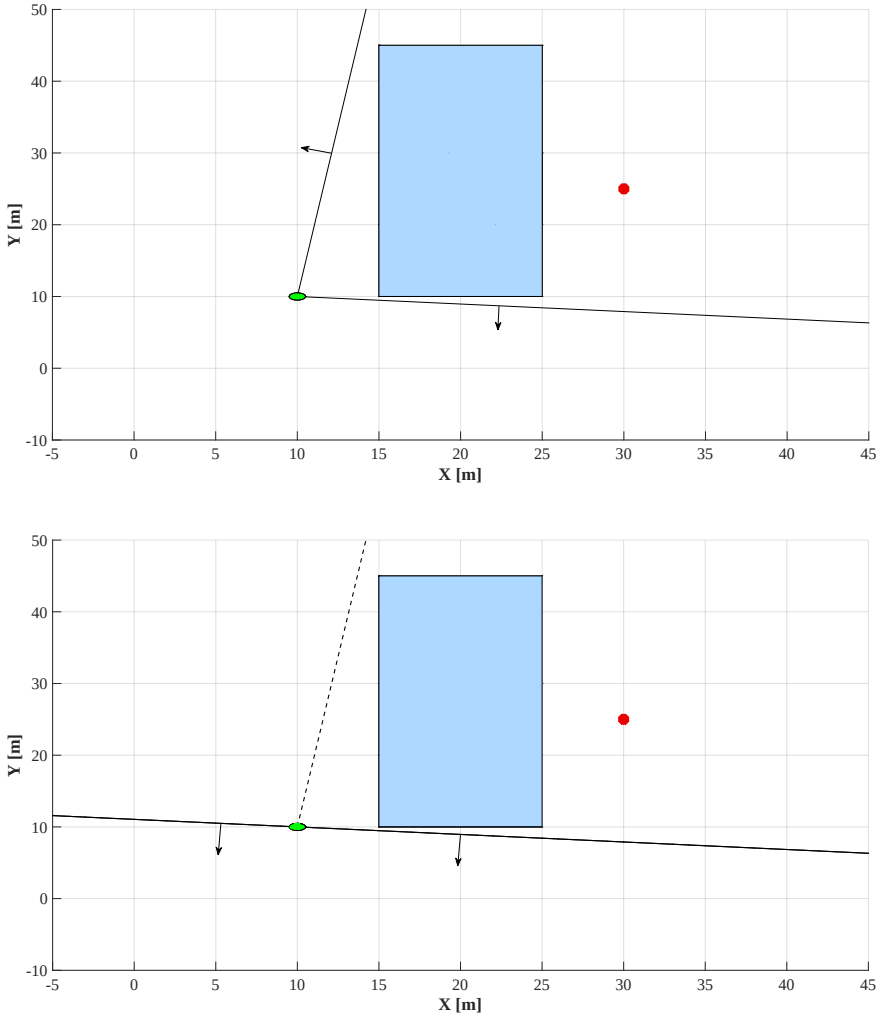

Figure 3.7: *The non-convex area is reduced to a convex one by only considering a half-plane. Green circle: drone i. Red circle: goal $\boldsymbol{p}_{goal,i}$.*

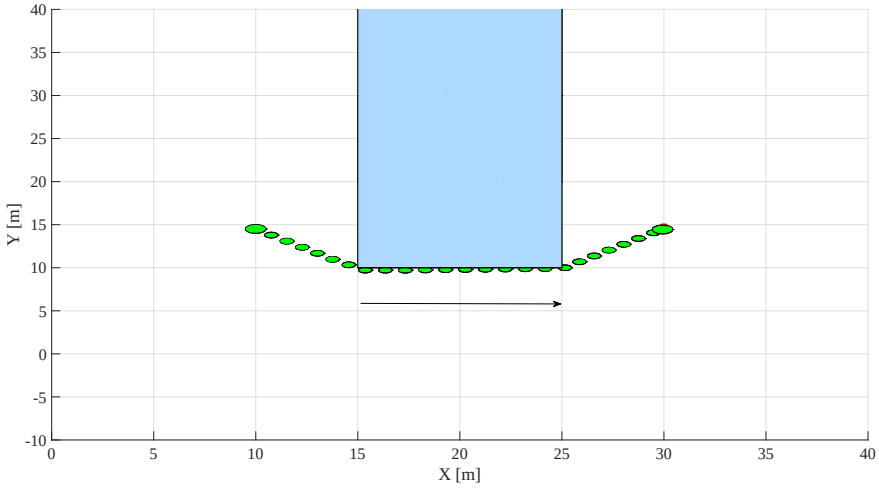

Figure 3.8: *A drone flies around an obstacle (from left to right). Note how it follows a linear path, dictated by a constraint.*

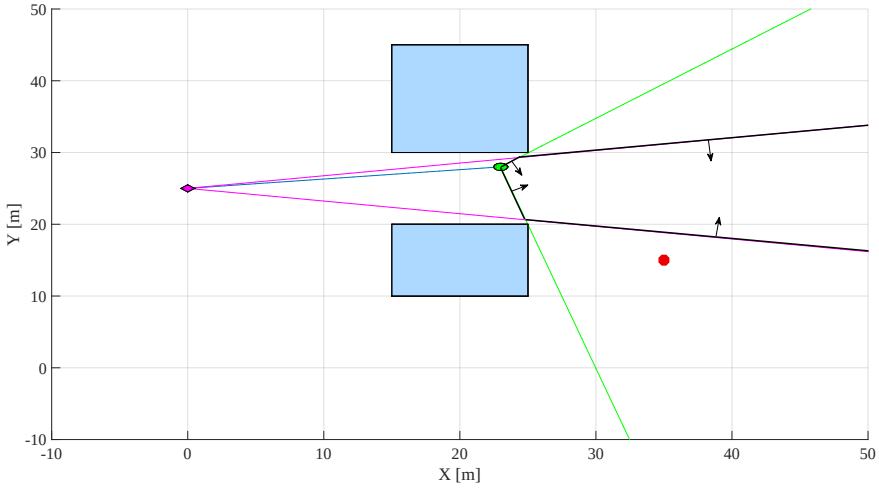

Figure 3.9: *The feasible space for the position reference of drone i (green circle) is limited by the one seen by drone $i+1$ (magenta square). If drone i were to travel within its own constraints only, the tether might impact with the obstacles.*

limits the collected information to two planes out of 3D space. As a consequence, only obstacles lying in the geometrical union of the two planes are detected, and their shape outside such planes in unknown. Therefore, at each time step the algorithm must choose whether to move within the horizontal or the vertical plane for the current sampling period. Through the adoption of gimbals for rotating the LiDAR scanning the vertical plane, it is possible for the leader to carry out the vertical scan in the plane that contains its goal, while the other drones $i{+}1,\ i = 1, \dots, N{-}1$, scan the vertical plane that contains drone i. As long as the presented assumptions hold, the vertical plane where drones i and $i{+}1$ lie also contains the $i{-}th$ tether. Constraints in the vertical plane are calculated with the same approach adopted for the horizontal one (3.10)-(3.18). Denoting with $\bar{A}_{H,i}$, $\bar{\boldsymbol{b}}_{H,i}$ the terms in (3.16) related to the horizontal plane, and with $\bar{A}_{V,i}$, $\bar{\boldsymbol{b}}_{V,i}$ those related to the vertical one, after the constraint sets have been computed the following quantities are derived to determine if a drone should move along the horizontal or vertical plane:

$$c_H = \begin{cases} 0 & \text{if } \bar{A}_{H,i} \begin{bmatrix} p_{X,goal,i} \\ p_{Y,goal,i} \end{bmatrix} \leq \bar{\boldsymbol{b}}_{H,i} \\ d\left(\begin{bmatrix} p_{X,goal,i} \\ p_{Y,goal,i} \end{bmatrix}, \mathcal{C}_H \right) & \text{otherwise} \end{cases}$$

$$c_V = \begin{cases} 0 & \text{if } \bar{A}_{V,i} \begin{bmatrix} p_{XY,goal,i} \\ p_{Z,goal,i} \end{bmatrix} \leq \bar{\boldsymbol{b}}_{V,i} \\ d\left(\begin{bmatrix} p_{XY,goal,i} \\ p_{Z,goal,i} \end{bmatrix}, \mathcal{C}_V \right) & \text{otherwise} \end{cases}$$

where $p_{XY,goal,i} = \left\| \begin{matrix} p_{X,goal,i} - p_{X,i} \\ p_{Y,goal,i} - p_{Y,i} \end{matrix} \right\|$ is the relative position on plane (X, Y) of the goal with respect to the drone (note that the vertical plane always contains both points for the leader), $d(\boldsymbol{v}, \mathcal{A})$ is the distance from point $\boldsymbol{v}$ to set $\mathcal{A}$, finally

$$\mathcal{C}_H = \left\{ \begin{bmatrix} p_X \\ p_Y \end{bmatrix} : \bar{A}_{H,i} \begin{bmatrix} p_X \\ p_Y \end{bmatrix} \leq \bar{\boldsymbol{b}}_{H,i} \right\}; \mathcal{C}_V = \left\{ \begin{bmatrix} p_{XY} \\ p_Z \end{bmatrix} : \bar{A}_{V,i} \begin{bmatrix} p_{XY} \\ p_Z \end{bmatrix} \leq \bar{\boldsymbol{b}}_{V,i} \right\}. \tag{3.19}$$

The drone will move along the horizontal plane if $c_H \leq c_V$, otherwise along the vertical one. This approach privileges the plane in which the distance between the obstacle-free space and the point of interest (possibly projected on the considered plane) is smaller. When an obstacle lies between the drone and its goal, this procedure effectively chooses to travel around it

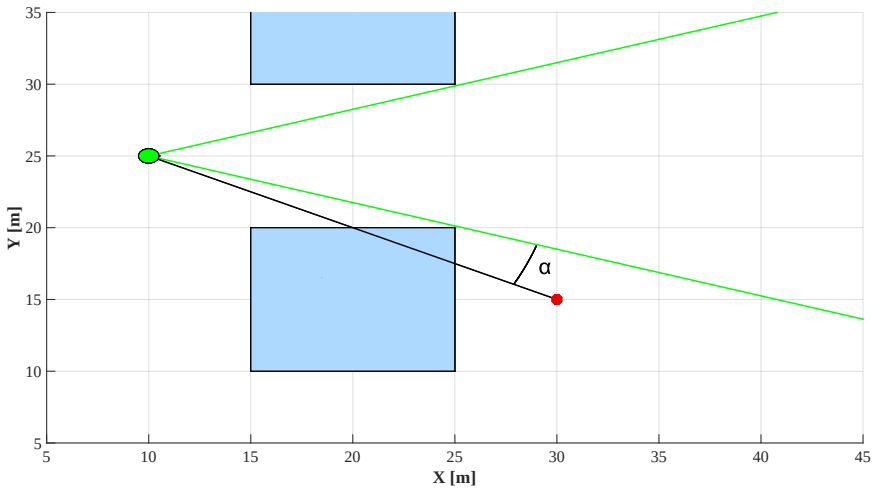

Figure 3.10: *An example of the calculation of the cost. Since the projection of the goal on the xy plane lies outside the feasible area, the cost is equal to the angle α, otherwise it would have been zero. The same calculation is performed with in the zt plane, then the two are compared to choose a plane where the reference must lie.*

in the plane where the required travel length appears smaller. Once the plane has been selected, its constraints are included in the QP (3.8), together with additional, linear constraints forcing the movement to happen exclusively within that plane. The constraints related to the discarded plane are dropped. Figure 3.10 represents this criterion. In the vertical direction, the tether is not a straight line. The following linear constraints are included, exploiting the map (3.7), to prevent collisiong between the sagging tether and the underlying obstacles:

$$\begin{aligned} p_{Z,ref,i} &\geq (\overline{Z}_{obs} + \Delta Z) + (p_{Z,i} - \underline{Z}_i) \\ p_{Z,ref,i+1} &\geq (\overline{Z}_{obs} + \Delta Z) + (p_{Z,i+1} - \underline{Z}_i), \end{aligned} \tag{3.20}$$

where $\overline{Z}_{obs}$ is the height of the tallest obstacle detected by the vertical LiDAR sensor of drone $i+1$ and ΔZ is a user-defined safety margin. Constraints (3.20) are enforced on both drones i and $i+1$. Once again, LiDAR readings are modified if the goal lies between the drone and an obstacle. This is more common in 3D because the ground is an obstacle and if the goal is lower than the leader's position it would otherwise be impossible to descend. Some stricter linear constraints are also introduced regarding the reference position of drone N, $\boldsymbol{p}_{ref,N}$, which prevent it from moving too far away in the (X, Y) plane from the ground station. This is necessary

because the tether segment connecting them is the only one with a fixed end.

The extension to the case of multiple drones follows the same principles: a new goal is calculated in the same way, the distance of the reference from it is penalized in the cost function and constraints are enforced on each drone and the connected ones one to prevent drone and cable collisions.

$$\begin{aligned} \left\| x_N^{ref} - x_{GS} \right\| &\leq r \\ \left\| y_N^{ref} - y_{GS} \right\| &\leq r. \end{aligned} \tag{3.21}$$

3.1.4 Simulation results

Simulations were run with MATLAB and Simulink. The latter ran the continuous-time dynamics of the drones and the tethers and the discrete-time local controllers between the sampling instants of the supervisory controller, which was instead implemented in MATLAB. The QP (3.8) was solved with MATLAB's `quadprog`. Several simulations were carried out with different obstacle shapes and positions, covering realistic application scenarios. It is apparent from simulations that the proposed algorithm can

Parameter	Value
m_t	3 Kg
$\ell_{t,max}$	100 m
m_d	10.92 Kg
R	0.5 m
T_s	1 s
θ_s	1°
Tether mass	3 Kg
Max tether length	100 m
Drone mass	12 Kg
Drone radius	1.5 m
Supervisor sampling period	1 s
LiDAR resolution	1°

Table 3.1: *Parameter values employed in the simulation tests.*

guide a formation of three drones plus the ground station in an unknown environment. Obstacle avoidance for both the drones and the tethers is exemplified in Figure 3.11, where the leader is prevented from heading directly towards its goal by the intersection of his constraints with those of its follower. The last few time steps also show how the algorithm correctly

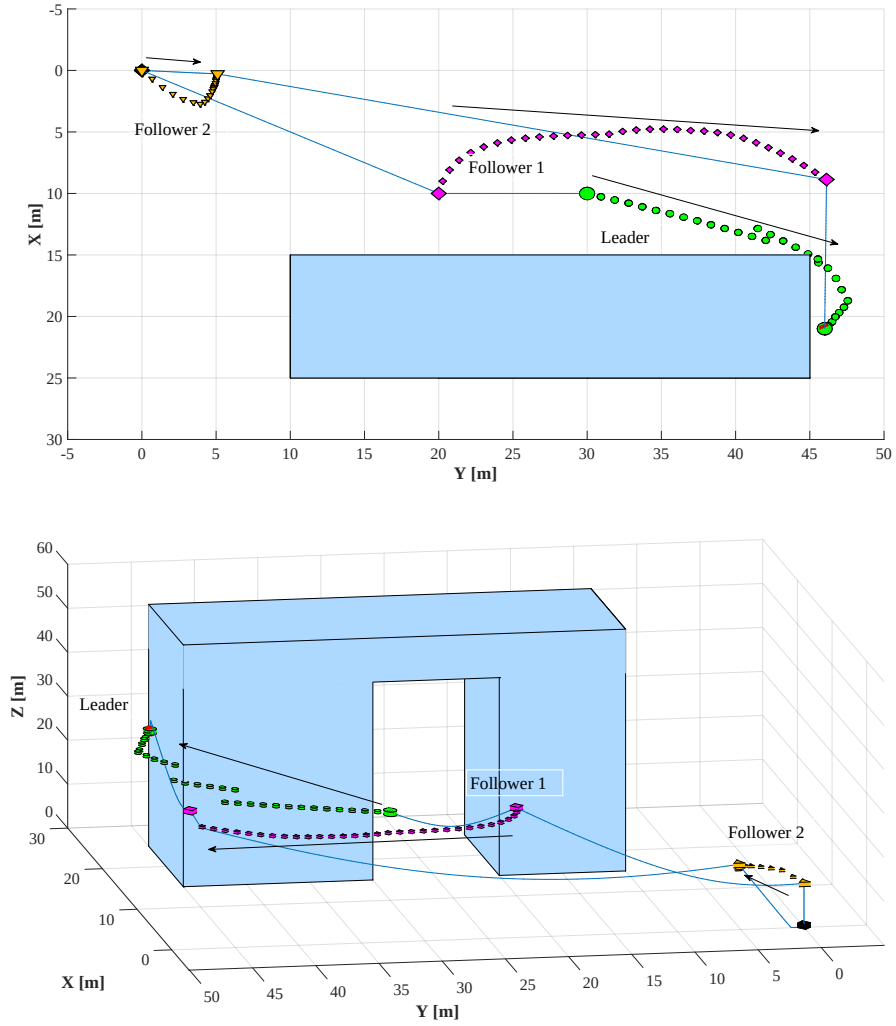

Figure 3.11: *A simulation with two drones and no ground station (the drones move towards increasing y values). Constraints are such that the follower drone limits the feasible space for the leader, temporarily blocking it from reaching the destination, in order to prevent cable collisions. The resulting behavior avoids the obstacle.*

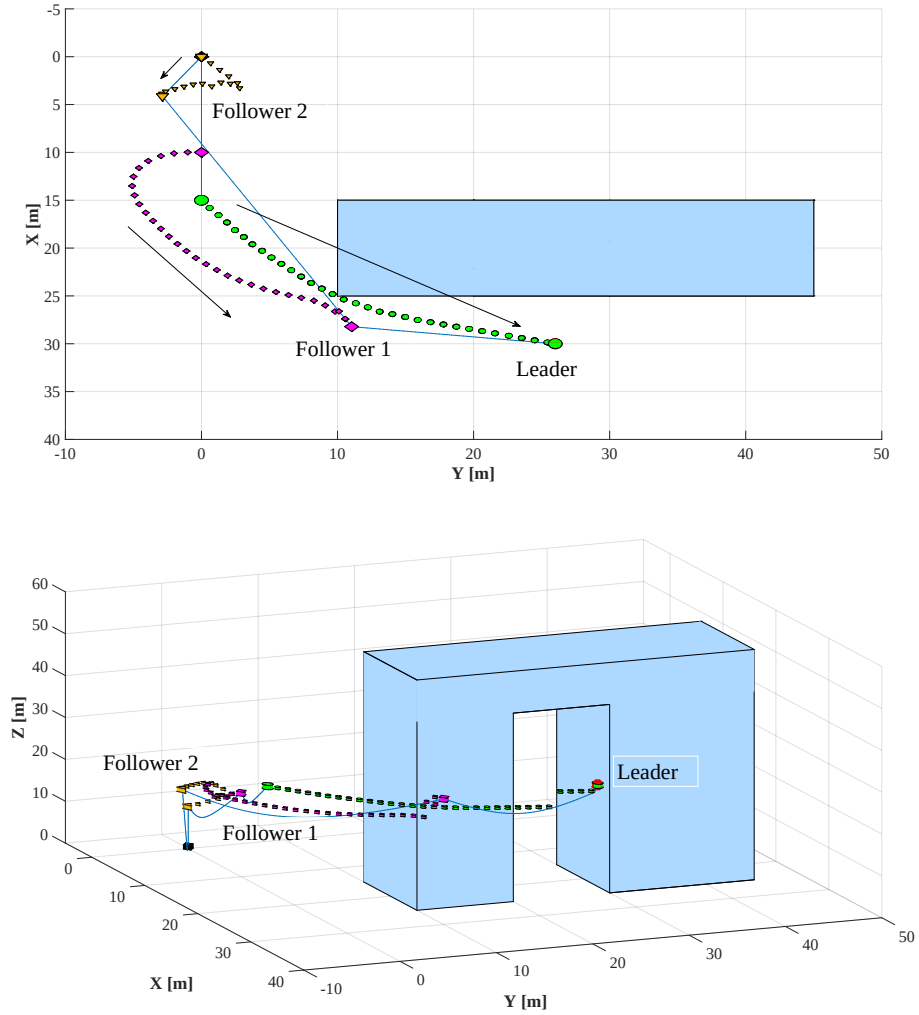

Figure 3.12: *A simulation of the whole system with three drones and a ground station (the drones move towards increasing x and y values). Note how the third drone (yellow triangle) is confined within a restricted area above the ground station.*

detects that the goal lies along a non-free direction but is closer than the obstacle, and lets the drone approach it in order to get to its destination. Figure 3.12 illustrates the behavior of the system when the goal is behind an obstacle, besides highlighting other features:

- The third drone of the formation, which is connected to the ground station, is constrained to hover close to it:
- The $i-$th drone aligns itself to the $i-1$-th, along the direction connectong the latter to its goal;
- Each aircraft is assigned to move either within the horizontal or vertical plane, depending on its goal and the surroundings obstacles;
- The second drone (Figure 3.12), in the last moments of the simulation, hovers in placeto prevent a collision between the obstacle and the second cable (connecting drones 2 and 3), letting its distance from the leader grow instead.

3.1.5 Conclusion

A centralized approach was presented to guide a network of tethered drones in an unknown environment, relying solely on LiDAR measurements. The algorithm is based on a QP and it operates in real-time, without building a map, thus it is capable of navigating some dynamical environments. It drives the formation to incremental exploration of the environment towards the final goal. It privileges safety over performance, because the obstacle-avoidance feature is a key for the automated flight of a STEM.

3.2 The MPC approach

Owing to the relatively recent introduction of STEM, several research gaps still exist, pertaining to path planning and safe on-line trajectory generation. The optimization problem presented in the previous section can be modified and refined to address some of them through a MPC formulation. The following sections address such gaps through the following contributions:

- the formulation of an optimal mission planner in a nominal environment, aiming at obtaining a reference configuration to be tracked by the tethered drones;
- the development of a reactive path planner based on MPC, capable of driving the system to the desired configuration while avoiding collisions with static obstacles, possibly different from the nominal ones, using range-limited sensors;
- the validation of the proposed techniques in simulations, with sophisticated drone and tether models.

The issue of optimal planning in a known environment is addressed here, though it was not tackled in the previous section. This is an expansion of the scope of the work, independent of the real-time MPC-based algorithm for path planning, which does not exploit *a priori* knowledge of the environment.

3.2.1 System description and model

The system is largely the same but not identical to the one presented in section 3.1. For the sake of clarity and correctness, a slightly different and more detailed notation is introduced here. A series ofN_d drones is considered, numbered from the closest to the furthest from the ground station. Therefore, $i = N_d$ denotes the leader drone, and the i-th drone is linked by tether i to drone $i-1$ and by tether $i+1$ to drone $i+1$, with $i = 1$, the first drone being connected by tether 1 to the ground station. Kinematic quantities are expressed with respect to an inertial, right-handed coordinate system (x_f, y_f, z_f), with origin at ground level, the (x_f, y_f) coordinates defining a plane parallel to the ground and z_f positive upwards. The continuous time variable is $t \in \mathbb{R}$, while $k \in \mathbb{N}_{\geq 0}$ the discrete one. Bold symbols indicate vectors in the 3D space, $\cdot^T$ is the matrix transpose operation and $v_{(a:b)}$ denotes the entries from the a-th to b-th position of the generic vector v.

Tether and winch model

Each of the N_d tethers is modeled as a chain of N_t inner nodes with mass $m_{t,i,l}$, $l = 1, \ldots, N_t$, where for simplicity all tethers are modeled with the same number of masses. As each tether can be reeled in and out from a winch, the lumped masses are time varying. The position and velocity of each node are, respectively, $\boldsymbol{s}_{t,i,l}$, $\dot{\boldsymbol{s}}_{t,i,l} \in \mathbb{R}^3$, $i = 1, \ldots, N_d$, $l = 1, \ldots, N_t$. Two extreme points are added, fixed either to a drone or to the ground station, for a total of $N_t + 1$ segments. Each inner node is subject to the gravitational pull and to the forces applied by the two neighboring tether segments. Elastic and internal friction forces are considered, but aerodynamic ones are neglected, assuming again that the speed relative to the air is small. Each tether segment is modeled as the parallel of a nonlinear spring, capable of transferring force only when taut, and a linear damper (see Figure 3.4). Dependence on the continuous time t is omitted. Based on Newton's second law, the equations of motion for the l-th node in the inertial reference read:

$$\ddot{\boldsymbol{s}}_{t,i,l} = \frac{(\boldsymbol{F}_{e,i,l+1} + \boldsymbol{F}_{e,i,l-1} + \boldsymbol{F}_{f,i,l+1} + \boldsymbol{F}_{f,i,l-1})}{m_{t,i,l}} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.22}$$

where g is the gravity acceleration, $\boldsymbol{F}_{e,i,l+1}$ denotes the elastic force pulling the l-th node towards the next one and $\boldsymbol{F}_{f,i,l+1}$ the corresponding friction force. The position and velocity vectors of the first and last nodes are equal to those of the connected elements, either a drone or the ground station, and are adopted as inputs to the tether model. On the other hand, the forces pertaining to the first and last nodes act directly on the connected drones and affect their motion. Thus, for the i-th vehicle the force applied by the tethers connected to it is:

$$\boldsymbol{F}_{t,i} = (\boldsymbol{F}_{e,i,N_t} + \boldsymbol{F}_{f,i,N_t} + \boldsymbol{F}_{e,i+1,0} + \boldsymbol{F}_{f,i+1,0}) \tag{3.23}$$

Each tether is coiled around a winch, fixed either to the ground station or to a drone, so that it can be extended or retracted to maintain the tension force within acceptable bounds. Denoting with $\theta_i(t)$, $\dot{\theta}_i(t)$ the angular position and speed of the i-th winch, its total time-varying mass depends on the mass of the winch without the tether $\underline{\mathrm{m}}_{w,i}$, its external radius radius $r_{e,i}$, the maximum length of the tether $\bar{l}$, and its linear mass density $\rho_{t,i}$:

$$m_{w,i} = \underline{\mathrm{m}}_{w,i} + \left(\bar{l} - r_{e,i}\theta_i(t)\right) \rho_{t,i}, \tag{3.24}$$

assuming $\theta_i(t) = 0$ when the cable is completely coiled. The mass $m_{t,i,l}$ of each point of tether i is then:

$$m_{t,i,l}(t) = \frac{\rho_{t,i} r_{e,i} \theta_i(t)}{N_t}$$

To deal with the cable sag effect, an approximation is introduced. Given that the height $\underline{z}_i$ in coordinate frame (x_l, y_l, z_l) of the lowest hanging segment of the i-th cable depends on the relative positions of the vehicles and the unreeled length of the tether itself, by assuming that the tether is always at an equilibrium configuration, it is possible to build a lookup function to be used online, yielding:

$$\underline{z}_i = f_{tether}(\boldsymbol{P}_i, \boldsymbol{P}_{i+1}, r_{e,i}\theta_i). \tag{3.25}$$

Where $\boldsymbol{P}_i$, $\boldsymbol{P}_{i+1}$ are the positions of drone i and $i+1$ (or the ground station) respectively. Therefore, in order to prevent the tether from colliding with the ground, a linear constraint is added to our planning problems, ensuring that the height reference for a drone is always greater than $\underline{z}_i$ by an arbitrary margin z_{margin} for both of the tethers it is attached to:

$$\begin{aligned} P^{z_f}_{ref,i} &\geq \underline{z}_i + z_{margin} \\ P^{z_f}_{ref,i} &\geq \underline{z}_{i+1} + z_{margin} \end{aligned} \tag{3.26}$$

Multi-copter model

Without loss of generality identical, symmetric quad-copter models are employed, each described through six-degrees-of-freedom nonlinear equations. This is a common choice in literature [75]. The position of drone i in space is $\boldsymbol{P}_i(t) = \left[P_i^{x_f}(t), P_i^{y_f}(t), P_i^{z_f}(t)\right]^T$. A local, right-handed reference frame (x_l, y_l, z_l) fixed to the drone is defined, where x_l, y_l are aligned with the drone's arms, and the z_l axis points up. Considering a single drone (thus dropping the index i for clarity), its control inputs are defined as:

$$\begin{aligned} u_1(t) &= \sum_{j=1}^{4} b\Omega_j^2(t) \\ u_2(t) &= a\, b \left(\Omega_4^2(t) - \Omega_2^2(t)\right) \\ u_3(t) &= a\, b \left(\Omega_3^2(t) - \Omega_1^2(t)\right) \\ u_4(t) &= d \left(\Omega_2^2(t) + \Omega_4^2(t) - \Omega_1^2(t) - \Omega_3^2(t)\right) \end{aligned} \tag{3.27}$$

where u_1 is the total thrust force along axis z_l, and u_2, u_3, u_4 are the yaw moments around the three axes of the local reference frame. b and d are

the lift and drag coefficients, a is the distance of a rotor from the center of the aircraft and Ω_j the rotational speed of the j-th rotor. All these inputs are bounded, since $\Omega_j \in [0, \bar{\Omega}]$, $j = 1, \ldots, 4$, where $\bar{\Omega}$ is the maximum rotational speed of the motors. By applying Newton's law the dynamical model reads:

$$\begin{aligned}
\ddot{\boldsymbol{P}}(t) &= \tfrac{1}{m_d(t)} R(t) \begin{bmatrix} 0 \\ 0 \\ u_1(t) \end{bmatrix} + \boldsymbol{F}_t(t) - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\
\dot{p}(t) &= \tfrac{I_{y_l} - I_{z_l}}{Ix} q(t) r(t) + \tfrac{u_2(t)}{Ix} - \tfrac{J_p}{Ix} q(t) \Omega_r(t) + d_x F_t(t) \\
\dot{q}(t) &= \tfrac{I_{z_l} - I_{x_l}}{Iy} p(t) r(t) + \tfrac{u_3(t)}{Iy} + \tfrac{J_p}{Iy} p(t) \Omega_r(t) - d_y F_t(t) \\
\dot{r}(t) &= \tfrac{I_{x_l} - I_{y_l}}{Iz} p(t) q(t) + \tfrac{u_4(t)}{Iz}
\end{aligned} \tag{3.28}$$

where g is the gravity acceleration, $m_d(t)$ is the time-varying overall drone's mass composed by the mass of the drone, the winch and the wrapped tether, $I_{x_l}, I_{y_l}, I_{z_l}$ are the rotational moments of inertia around the local axes, J_p is the moment of inertia of each motor, $p(t)$, $q(t)$, $r(t)$ are the angular velocities around axes x_l, y_l, z_l, $\Omega_r(t)$ is the rotational speed around the related axis and $R(t)$ is the rotational matrix from local to global coordinates. d_x and d_y express the distance between the center of mass and the point where tether forces act, along the x and y direction respectively.

Each autonomous quad-copter is endowed with a low level controller that stabilizes it and accepts position vectors as references to track in the form $\boldsymbol{P}_{ref} = [P_{ref}^{x_f}, P_{ref}^{y_f}, P_{ref}^{z_f}]^T$. From the standpoint of a high-level motion planner, how these references are tracked is not relevant. The reader is referred to [55] for an example implementation of such low level controller.

Control-oriented model

The whole system's model entails a chain of nonlinear multi-copter models. However, if tether forces are kept low by the winches and/or by proper maneuvering resulting from relative position constraints to be met by the mission planner, the low-level control system on each drone can reject their perturbing effects on the positioning precision. Then, the overall system behavior from the perspective of the high-level navigation strategy is well described by decoupled linear dynamics for each vehicle, as shown by the system response in [55]. Moreover, a linear, time-invariant model allows

the system dynamics to be included as a set of linear constraints in a Finite Horizon Optimal Control Problem (FHOCP), while the coupling introduced by tethers can be considered in a MPC problem, as detailed in Section 3.2.3. The control-oriented model is then a continuous time double integrator with state feedback for each drone:

$$\begin{bmatrix} \dot{\boldsymbol{P}}(t) \\ \dot{\boldsymbol{V}}(t) \end{bmatrix} = \begin{bmatrix} 0^{3\times3} & \mathbb{I}^{3\times3} \\ -K_{vel}K_{pos} & -K_{vel} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}(t) \\ \boldsymbol{V}(t) \end{bmatrix} + \begin{bmatrix} 0^{3\times3} \\ K_{vel}K_{pos} \end{bmatrix} \boldsymbol{P}_{ref}(t) \quad (3.29)$$

where $\boldsymbol{V}(t) = \dot{\boldsymbol{P}}(t) \in \mathbb{R}^3$, $\boldsymbol{P}(t) \in \mathbb{R}^3$ are the drone velocity and position respectively and $0^{3\times3}$, $\mathbb{I}^{3\times3}$ are a matrix of zeros and the identity matrix. Furthermore, the acceleration of the vehicle is denoted as:

$$\boldsymbol{A}(t) = K_{vel}\left(K_{pos}\left(\boldsymbol{P}_{ref}(t) - \boldsymbol{P}(t)\right) - \boldsymbol{V}(t)\right). \quad (3.30)$$

The matrices K_{vel}, $K_{pos} \in \mathbb{R}^{3\times3}$ are suitable gain matrices representing the state feedback that have to be tuned to obtain a closed loop response as close as possible to the one of the real system, see Figure 2.2 for an example with experimental data collected on one of the tethered drones in our lab.

The continuous time model (3.29) is then converted to discrete time with sampling time T_s, obtaining the desired control-oriented model for the i-th drone:

$$x_i(k+1) = A_i x_i(k) + B_i \boldsymbol{u}_i(k) \quad (3.31)$$

where $x_i(k) = [\boldsymbol{P}_i(k),\ \boldsymbol{V}_i(k)]^T \in \mathbb{R}^6$ is the state of the i-th position-controlled vehicle and $\boldsymbol{u}_i(k) = \boldsymbol{P}_{ref_i} \in \mathbb{R}^3$ its input. Thus, the control-oriented model features as input the position reference $\boldsymbol{P}_{ref_i}(k)$ consistently with the nonlinear position-controlled model, and as state the position and velocity of the vehicle, making it possible to include in the FHOCP constraints on these quantities and the acceleration as well, through equation (3.30). Finally, the control-oriented model of the whole system composed of N_d vehicles is defined as:

$$x(k+1) = A_s x(k) + B_s u(k) \quad (3.32)$$

where $x(k) \in \mathbb{R}^{6N_d}$ is a column vector containing the states of the vehicles, $u(k) \in \mathbb{R}^{3N_d}$ is the vector of inputs, while $A_s = diag(A_1, \ldots, A_{N_d})$ and $B_s = diag(B_1, \ldots, B_{N_d})$ are block-diagonal matrices defining the overall system dynamics. Finally, let the selection matrix $\mathbb{P}_{xy} \in \mathbb{R}^{6N_d\times2N_d}$ be defined such that $x(k)\mathbb{P}_{xy} = [x_{1_{(1:2)}}(k)^T, \ldots, x_{N_{d_{(1:2)}}}(k)^T]^T$, i.e. it selects the drones' positions components along the x_f and y_f axes.

Sensors

All drones are endowed with a Global Positioning System (GPS) for absolute localization, and an Inertial Measurement Unit (IMU) with filtering algorithms to estimate reliably the full state. For the purposes of environment perception, each drone is equipped with a 360° wide, planar LiDAR. Such sensor is used to locate obstacles in real time during flight with respect to the drone. Its readings are exploited in the formulation of constraints for the FHOCP, in the form of a convex under-approximation of free space, as detailed in Section 3.2.3. Each LiDAR is assumed to provide a vector $\delta(k)$ of $M = \frac{2\pi}{\alpha_s}$ measurements with angular resolution α_s. Each reading can be expressed as the vector $d_i(k) = \delta_i(k)[cos(\alpha_i),\ sin(\alpha_i)]^T, i = 0, \dots, M-1$ where $\delta_i(k)$ is the measured distance between the drone and the closest obstacle in the direction α_i. If no obstacle is detected in the α_i direction, then the measurement $\delta_i(k)$ is equal to R_L, the maximum detection range. Several LiDAR sensors provide data with frequency in the order of 10 Hz or greater. Furthermore, the readings can be directly employed in formulating constraints with efficient approaches, as presented in Section 3.2.3. These two attributes make them convenient for real-time reactive planning.

Environment

A 3D environment is modeled, where the obstacles have constant cross-section along the vertical direction z_f and infinite height. Although this limits the representable environments, in realistic scenarios obstacles often can be considered or over-approximated in this way. Furthermore, a method for dealing with fully 3D obstacles has been presented in [23] and can be directly implemented within the FHOCP framework presented here. In order to introduce a safe distance margin from obstacles and to ease their treatment in the mathematical formulation of the problem, approximated as elliptical hulls, containing their actual shape, as shown in Figure 3.13. In this framework, each obstacle is described as a compact set with shape matrix H_j:

$$\mathcal{O}_j \doteq \{\chi \in \mathbb{R}^2 :\ (\chi - \chi_{c_j})^T H_j (\chi - \chi_{c_j}) \leq 1\} \tag{3.33}$$

where χ_{c_j} is the center of the j-th obstacle and the set of all obstacles is defined as $\mathcal{O} \doteq \bigcup_{j=1}^{N_o} \mathcal{O}_j$.

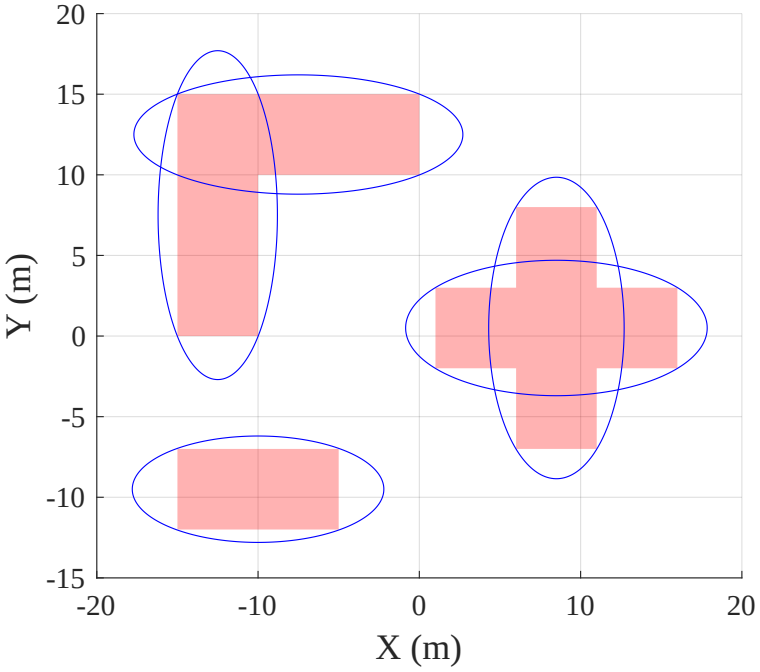

Figure 3.13: *Example of the elliptical approximation of non-convex obstacles.*

3.2.2 Problem formulation

The position of all the drones are gathered in a single vector, named *configuration* of the system $\boldsymbol{C}(t) = \bigcup_{i=1}^{N_d} \boldsymbol{P}_i(t)$. A configuration is admissible with respect to the set of obstacles $\mathcal{O}$ if the positions of all the drones and the tethers connecting them belong to the free space $\boldsymbol{S}_{free} \doteq \mathbb{R}^2 \setminus \mathcal{O}$ and the tether lengths are within a maximum value $\bar{l}$, and a minimum one $\underline{l}$:

$$\boldsymbol{P}_{i_{(1:2)}} + \alpha \left(\boldsymbol{P}_{i+1_{(1:2)}} - \boldsymbol{P}_{i_{(1:2)}} \right) \in \boldsymbol{S}_{free}$$
$$i = 0, \ldots, N_d, \quad \forall \alpha \in [0, 1] \tag{3.34}$$
$$\underline{l} \leq \|\boldsymbol{P}_{i+1} - \boldsymbol{P}_i\|_2 \leq \bar{l} \quad i = 0, \ldots, N_d - 1. \tag{3.35}$$

$\boldsymbol{P}_0$ denotes the position of the ground station, from here assumed equal to the origin of (x_f, y_f, z_f). $\bar{l}$ and $\underline{l}$ are bounds on the Euclidean distance between two drones and as such they do not account for cable sag. Therefore the upper bound $\underline{l}$ in particular shall be assigned a conservative value, lower than the real maximum length of the tether. A twofold problem is defined, with the aim to obtain:

- An off-line procedure that, given a set of obstacles $\mathcal{O}$ and an arbitrary target position in free space $\boldsymbol{P}_{goal}$, yields, if it exists, an admissible configuration $\boldsymbol{C}^*$ for the system such that the position of the last drone of the chain coincides with the goal: $\boldsymbol{P}^*_{N_d} = \boldsymbol{P}_{goal}$.
- An on-line path follower that tracks such configuration $\boldsymbol{C}^*$ so that the system converges to it from any arbitrary feasible starting configuration, while guaranteeing avoidance of collisions both with known obstacles $\mathcal{O}$ and with new and unknown ones the system may encounter during the flight.

To solve this problem, a high-level, centralized controller is designed which can communicate with all drones through the tethered network.

3.2.3 Proposed method

Offline Configuration Planning

To obtain the optimal admissible configuration $\boldsymbol{C}^*$ that allows the leader drone to reach the target while avoiding obstacles, the following optimization problem is solved off-line. The adopted cost function is:

$$J(\boldsymbol{P}_i, \boldsymbol{P}_{goal}) = \|\boldsymbol{P}_{goal} - \boldsymbol{P}_{N_d}\|^2_{W_T} + \sum_{i=0}^{N_d-1} \|\boldsymbol{P}_{N_d-i} - \boldsymbol{P}_{N_d-i-1}\|^2_{W_D} \tag{3.36}$$

where $\|x\|_W^2 \doteq x^T W x$. In (3.36), the first term aims at weighting the distance of the leader from the target, while the second one accounts for the distance between a drone and the subsequent one in the chain through the symmetric positive-definite weighting matrices W_T, $W_D \in \mathbb{R}^{3\times 3}$. To guarantee that the drones and the tether don't collide with an obstacle, the following constraints are introduced, where $\hat{\boldsymbol{P}}_{i,\lambda} \doteq (\lambda \boldsymbol{P}_i + (1-\lambda)\boldsymbol{P}_{i+1})$:

$$\begin{aligned} &(\hat{\boldsymbol{P}}_{i,\lambda} - \chi_{c_j})^T H_j (\hat{\boldsymbol{P}}_{i,\lambda} - \chi_{c_j}) > r(\theta) + \epsilon, \\ &\forall \lambda \in [0,1], \quad j = 1,\dots,N_O, \quad i = 1,\dots,N_d - 1 \qquad (3.37a) \\ &\underline{l} \le \|\boldsymbol{P}_i - \boldsymbol{P}_{i-1}\| \le \bar{l}, \ i = 1,\dots,N_d \qquad (3.37b) \\ &P_i^{z_f} \ge \underline{z}_i + z_{margin}, \quad i = 1,\dots,N_d, \qquad (3.37c) \end{aligned}$$

where $r(\theta)$ is the radius of the ellipsoid and ϵ is a user-selected safety distance from the obstacles. In particular, constraint (3.37a) ensures that all the possible convex combinations of the position of the two drones lay outside the ellipsoidal set by a distance ϵ. The following Non-Linear Program (NLP) is thus obtained:

$$\begin{aligned} &\min_{\boldsymbol{P}_1,\dots,\boldsymbol{P}_{N_d}} J(\boldsymbol{P}_i, \boldsymbol{P}_{goal}) \qquad (3.38) \\ &\text{subject to:} \quad (3.37a), (3.37b), (3.37c). \end{aligned}$$

Convex Approximation of Free Space

To derive a set representing an obstacle-free area for two consecutive drones and the tether, through the definition of convexity, a set containing two consecutive vehicles is derived. Let $\mathcal{L}_i(k) \doteq \{d_0(k),\dots,d_{M-1}(k)\} \in \mathbb{R}^M$ be the set containing the LiDAR measurements of the i-th drone at time k. To obtain the non-convex obstacle-free area defined by the LiDARs readings of two consecutive drones i and $i+1$, the set of overlapping measurements is defined:

$$\begin{aligned} &\mathcal{L}_d(k) = \{d_m(k) \in \mathcal{L}_i(k), \ d_n(k) \in \mathcal{L}_{i+1}(k) : \\ &\|d_m(k) - P_{i+1_{(1:2)}}(k)\|_2 < R_L \wedge \|d_m(k) - P_{i_{(1:2)}}(k)\|_2 = R_L, \\ &\|d_n(k) - P_{i_{(1:2)}}(k)\|_2 < R_L \wedge \|d_n(k) - P_{i+1_{(1:2)}}(k)\|_2 = R_L, \\ &\forall m,n = 0,\dots,M-1\} \qquad (3.39) \end{aligned}$$

Thus, the merged readings are:

$$\mathcal{L}_{i,i+1}(k) = \mathcal{L}_i(k) \cup \mathcal{L}_{i+1}(k) \setminus \mathcal{L}_d(k). \qquad (3.40)$$

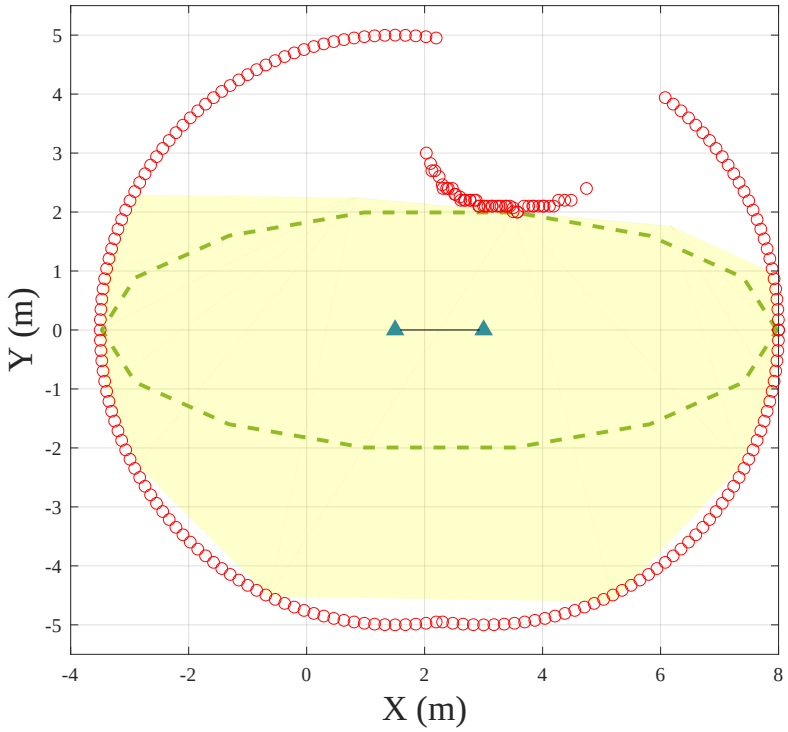

Figure 3.14: *A representation of the described quantities. The triangles represent two connected drones. The red points represent the merged LiDAR readings $\mathcal{L}_{i,i+1}(k)$, the dashed line depicts the discretized ellipsoid and the yellow area corresponds to the convex approximation of free space.*

$L_{i,i+1}(k)$ is the cardinality $|\mathcal{L}_{i,i+1}(k)|$ of the merged readings. Starting from the set $\mathcal{L}_{i,i+1}(k)$, the goal is to derive the largest convex polytopic set containing the two vehicles. To do so, a two step procedure is advanced: (i) solution of a Linear Program (LP) to compute the largest ellipsoidal set in $\mathcal{L}_{i,i+1}(k)$ with major axis directed along the tether (ii) derivation of a polytopic under-approximation of the ellipsoidal set and iterative expansion of its vertices to maximize the area of the polytope. The equation of an ellipsoid in the Cartesian plane reads:

$$k_1\tilde{x}(x,y)^2 + k_2\tilde{y}(x,y)^2 = 1, \tag{3.41}$$

and functions $\tilde{x} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\tilde{y} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are defined, taking into account its offset from the origin and its rotation around it:

$$\begin{aligned} \tilde{x}(x,y) &= (x - x_0)cos(\alpha) + (y - y_0)sin(\alpha) \\ \tilde{y}(x,y) &= (y - y_0)cos(\alpha) + (x - x_0)sin(\alpha) \end{aligned} \tag{3.42}$$

where (x_0, y_0) is the center of the ellipsoid and α is the angle of the rotation of its major axis. To obtain the set with maximum surface area in $\mathcal{L}_{i,i+1}(k) \doteq \{d_0(k), \ldots, d_{L_{i,i+1}(k)-1}(k)\}$ the following LP is solved:

$$\begin{aligned} \min_{k_1,k_2} \quad & k_1 + k_2 \\ & \text{subject to} \\ & k_1\tilde{x}(P_{i_1}(k), P_{i_2}(k)) + k_2\tilde{y}(P_{i_1}(k), P_{i_2}(k)) \leq 1, \\ & k_1\tilde{x}(P_{i+1_1}(k), P_{i+1_2}(k)) + k_2\tilde{y}(P_{i+1_1}(k), P_{i+1_2}(k)) \leq 1, \\ & k_1\tilde{x}(d_m) + k_2\tilde{y}(d_m(k)) \geq 1, \quad \forall m = 0, \ldots, L_{i,i+1}(k) - 1, \end{aligned} \tag{3.43}$$

where the first two constraints impose the membership of the position of the drones to the ellipsoidal set while the last one ensures that LiDAR measurements $\mathcal{L}_{i,i+1}(k)$ lie outside the set. Then, the convex-hull of samples placed on the ellipsoid at a selected constant angular distance one from the other allows its discretized under-approximation as shown in Fig. 3.14. This is exploited as a starting base to build a polytopic under-approximation of the free space $\mathcal{D}_j(k)$, $j = 1, \ldots, N_d$, according to the procedure described in Section III-A of [58], where the obtained vertices are iteratively expanded until they reach a LiDAR reading. To take into account the dimensions of the vehicles, its maximum encumbrance is removed from the LiDAR readings before the computation of the convex under-approximation of the free space. The result of this elaboration is summarized in Fig. 3.14.

Remark 3. *By construction, the proposed algorithm ensures that the current position (x_f, y_f) of two consecutive vehicles $x_{i_{(1:2)}}(k)$, $x_{i+1_{(1:2)}}(k)$ and*

their tether is contained in the convex under-approximation of the free-space $\mathcal{D}_j(k)$, i.e. a safe set.

Finally, the $N_d - 1$ convex polytopes computed at time k are collected for each pair of drones in the following set:

$$\mathcal{S}(k) = \{\mathcal{D}_j(k), j = 1, \dots, N_d\} \tag{3.44}$$

Online Path Following

A real-time path following strategy is now necessary, to have the system state track the configuration $\boldsymbol{C}^*$ obtained in Section 3.2.3. To derive an approach able to react to unexpected obstacles, only real-time information provided by the LiDARs is exploited. To guarantee feasibility in a receding horizon fashion, for each vehicle and at each time instant, a trajectory is sought for, contained by the obstacle-free regions $\mathcal{S}(k)$ and null velocity at the end the prediction horizon $N \in \mathbb{N}$. The leader drone sequentially tracks the terminal positions of all drones $\boldsymbol{C}^*$, one at a time and in order, from i to N_d. When one is reached within a certain distance, the setpoint switches to the following one, up until the last. These are collected in the vector $\hat{C} = [\boldsymbol{C}^{*T}_{1:3}, \boldsymbol{C}^{*T}_{4:6}, \dots, \boldsymbol{C}^{*T}_{3N_d-2,3N_d}]^T \in \mathbb{R}^{3N_d}$. The follower drones (i.e. from the first to the $N_d - 1$-th) are given as reference at time k a point in space belonging to the line connecting the following drone to its goal, at an arbitrary, user-defined distance $\bar{d}_i$ behind it:

$$\begin{aligned}\boldsymbol{P}_{ref_i}(k) = \frac{\boldsymbol{P}_{i+1}(k) - \boldsymbol{P}_{ref_{i+1}}(k)}{\|\boldsymbol{P}_{i+1}(k) - \boldsymbol{P}_{ref_{i+1}}(k)\|} \left(\|\boldsymbol{P}_{i+1}(k) - \boldsymbol{P}_{ref_{i+1}}(k)\| + \bar{d}_i\right) \\ i = 1, \dots, N_d - 1.\end{aligned} \tag{3.45}$$

This approach aligns each drone behind the following one, allowing the latter maximum freedom to move towards its goal. Thus, at each time step k the vector of position references for all the drones is defined as $P_t(k) = [\boldsymbol{P}_{t_1}(k)^T, \dots \boldsymbol{P}_{t_{N_d}}]^T$. $x(j|k)$ is the state of system (3.32) at time $k + j$ predicted at time k. To track the position references $P_t(k)$ the following cost function is considered:

$$J(x(k), P_t(k)) = \sum_{j=1}^{N} \sum_{i=1}^{N_d} \|\boldsymbol{x}_{i_{(1:3)}}(j|k) - \boldsymbol{P}_{t_i}(k)\|^2_Q \tag{3.46}$$

where $Q \in \mathbb{R}^{3\times3}$ is a symmetric positive-definite weighting matrix. Now, denoting the vector of decision variables $U = [u(0|k)^T, \dots, u(N-1|k)^T]^T \in$

$\mathbb{R}^{3N_dN}$, the FHOCP $\mathscr{P}(x(k), \mathcal{S}(k), P_t(k))$ can be stated:

$$\min_{U} J(x(k), P_t(k)) \tag{3.47a}$$

subject to

$$x(j|k) = A_s x(j-1|k) + B_s u(j-1|k), \quad \forall j \in \mathbb{N}_1^N, \tag{3.47b}$$

$$-\bar{\boldsymbol{A}} \leq K_{vel}\left(K_{pos}\left(\boldsymbol{u}_i(j|k)) - \boldsymbol{x}_{i_{(1:3)}}(j|k)\right) - \boldsymbol{x}_{i_{(4:6)}}(j|k)\right) \leq \bar{\boldsymbol{A}},$$
$$\forall j \in \mathbb{N}_0^{N-1}, \quad \forall i \in \mathbb{N}_1^{N_d}, \tag{3.47c}$$

$$-\bar{\boldsymbol{V}} \leq \boldsymbol{x}_{i_{(4:6)}}(j|k) \leq \bar{\boldsymbol{V}}, \quad \forall j \in \mathbb{N}_0^N, \quad \forall i \in \mathbb{N}_1^{N_d}, \tag{3.47d}$$

$$x_{i_{(1:2)}}(j|k) \in \mathcal{D}_i \cap \mathcal{D}_{i+1}, \quad \forall j \in \mathbb{N}_0^N, \quad \forall i \in \mathbb{N}_1^{N_d-1}, \tag{3.47e}$$

$$x_{N_{d_{(1:2)}}}(j|k) \in \mathcal{D}_{N_d}, \quad \forall j \in \mathbb{N}_0^N, \tag{3.47f}$$

$$\|x_{i+1_{(1:2)}} - x_{i_{(1:2)}}\|_2^2 \geq \underline{l}^2, \quad \forall i \in \mathbb{N}_0^{N_d-1}, \tag{3.47g}$$

$$\|x_{i+1_{(1:2)}} - x_{i_{(1:2)}}\|_2^2 \leq \bar{l}^2, \quad \forall i \in \mathbb{N}_0^{N_d-1}, \tag{3.47h}$$

$$x_{i_{(3)}}(j|k) \geq \underline{\mathrm{z}}_i + z_{margin}, \quad \forall j \in \mathbb{N}_0^N, \quad \forall i \in \mathbb{N}_1^{N_d}, \tag{3.47i}$$

$$\boldsymbol{x}_{i_{(4:6)}}(N|k) = \mathbf{0}^{3\times 1}, \quad \forall i \in \mathbb{N}_1^{N_d}, \tag{3.47j}$$

$$x(0|k) = x(k) \tag{3.47k}$$

where all inequalities and equalities are element-wise, $\mathbb{N}_a^b = \{n \in \mathbb{N} \mid a \leq n \leq b\}$ and $\bar{A}$, $\bar{V}$ are the vehicle's maximum acceleration and velocity vectors, assumed here for simplicity to be identical for all the vehicles. The FHOCP (3.47) presents linear constraints for the system's dynamics (3.47b), acceleration (3.47c), velocity (3.47d) and position (3.47e), (3.47f), while the nonlinear constraint (3.35) defining the maximum and minimum tether lengths, is imposed through the non-convex quadratic constraint (3.47g) and the convex one (3.47h). Finally, linear constraints are imposed on the altitude of the vehicle (3.47i), on the terminal state (3.47j) and on the initial condition (3.47k). Note that although $\underline{\mathrm{z}}_i$ is nonlinear in the drones' states (see (3.25)), its value is calculated a each time step, before solving the problem, and assumed constant until the next iteration, therefore (3.47i) is linear. Thus, the FHOCP $\mathscr{P}(x(k), \mathcal{S}(k), P_t(k))$ is a non-convex QCQP, where the non-convex constraint is (3.47g). Although non-convex, the problem structure can be exploited to efficiently find a local minimizer with general-purpose nonlinear optimization algorithms, such as Interior Point OPTimizer [76]. The solution of the FHOCP (3.47) is $U^*(x(k), \mathcal{S}(k), P_t(k))$. Furthermore, $k^*(k) \leq k$ denotes the latest sampling instant leading to a feasible FHOCP $\mathscr{P}(x(k^*(k)), \mathcal{S}(k^*(k)), P_t(k^*(k)))$.

Due to the time-varying nature of state constraints, the recursive feasibility of the problem is no more guaranteed with standard receding horizon strategies. Thus to guarantee the existence of a feasible problem at each time step, the FHOCP (3.47) is embedded in the following receding horizon strategy, similarly to the one presented in [58]: **Algorithm 1**: Receding horizon strategy

1) At time k compute the set $\mathcal{S}(k)$ containing the safe sets for all the pairs of drones.

2) **if** $\mathscr{P}(x(k), \mathcal{S}(k), P_t(k))$ is feasible, apply to the system the first control input in the optimal sequence $U^*(x(k), \mathcal{S}(k), P_t(k))$. Store the set $\mathcal{S}(k)$ as $\mathcal{S}(k^*(k+1))$ and set $k^*(k+1) = k$.

else solve $\mathscr{P}(x(k), \mathcal{S}(k^*(k)), P_t(k))$ and apply to the system the first control input in the optimal sequence $U^*(x(k), \mathcal{S}(k^*(k)), P_t(k))$. Set $k^*(k+1) = k^*(k)$.

3) set $k = k + 1$ and go to **1)**.

The time-invariant nature of the environment considered in this work implies that the safe sets $\mathcal{D}_j(k)$ depend only on the system state $x(k)$. The proposed MPC approach is not a static state-feedback law, but a dynamic controller with internal states $k^*(k)$ and inputs $x(k)$ and $P_t(k)$:

$$\begin{aligned} u(k) &= \kappa(x(k), k^*(k), P_t(k)) \\ k^*(k+1) &= \eta(x(k), k^*(k)) \end{aligned} \tag{3.48}$$

where functions $\kappa : \mathbb{R}^{6N_d} \times \mathbb{Z} \times \mathbb{R}^{3N_d} \to \mathbb{R}^{3N_d}$ and $\eta : \mathbb{R}^{6N_d} \times \mathbb{Z} \to \mathbb{Z}$ are implicitly defined by Algorithm 1 leading to the following closed loop system:

$$\begin{aligned} x(k+1) &= Ax(k) + B\kappa(x(k), k^*(k), P_t(k)) \\ k^*(k+1) &= \eta(x(k), k^*(k)) \end{aligned} \tag{3.49}$$

The role of variable $k^*(k)$ is to guarantee the existence of a feasible FHOCP at each time step despite the time-varying nature of the safe convex set $\mathcal{S}(k)$.

Remark 4. *The guaranteed existence of a feasible FHOCP holds only when unknown but static obstacles are considered in the problem. When the environment is dynamic, additional assumptions must be considered and it is currently subject of our research.*

Lemma 2. *Assume that at time $k = k_0$, the FHOCP* (3.47) *is feasible and $x(k_0)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}$, $(\boldsymbol{P}_{i+1_{(1:2)}}(k_0) - \boldsymbol{P}_{i_{(1:2)}}(k_0)) \in \boldsymbol{S}_{free}$, $\forall i \in \mathbb{N}_1^{N_d-1}$ i.e. the drones and the tethers are initially in an obstacle-free region. Then, by applying Algorithm 1, the trajectory of the close loop system* (3.49) *is such that $x(k)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}$, $(\boldsymbol{P}_{i+1_{(1:2)}} - \boldsymbol{P}_{i_{(1:2)}}) \in \boldsymbol{S}_{free}$, $\forall k > k_0$.*

Proof. Lemma 2 is proven for $k = k_0 + 1$; since k_0 is a generic sampling instant, the result will then follow by induction. At time $k = k_0$ the FHOCP (3.47) is feasible and $k^*(k_0 + 1)$ is set equal to k_0. $\forall k \geq k_0$ denotes the optimal input sequence $U^*(k) = [u^*(0|k)^T, \ldots, u^*(N-1|k)^T]$ obtained by solving either $\mathscr{P}(x(k), \mathcal{S}(k), P_t(k))$ or $\mathscr{P}(x(k), \mathcal{S}(k^*(k)), P_t(k))$ and leading to the optimal state trajectory $X^*(k) = [x^*(1|k)^T, \ldots, x^*(N|k)^T]$, where $x^*(N|k)$ is a steady state for system (3.32) (see constraint (3.47j)). At time $k_0 + 1$, there are only two possibilities:
A) If $\mathscr{P}(x(k_0+1), \mathcal{S}(k_0+1), P_t(k_0+1))$ is feasible, $x(k_0+1)\mathbb{P}_{xy} \in \mathcal{S}(k)$, see (3.47e) (3.47j).
B) Conversely if $\mathscr{P}(x(k_0 + 1), \mathcal{S}(k_0 + 1), P_t(k_0 + 1))$ is not feasible, the problem $\mathscr{P}(x(k_0 + 1), \mathcal{S}(k^*(k_0 + 1)), P_t(k_0 + 1))$ is solved, where a sub-optimal solution can be easily built by considering the tail of $U^*(k^*(k_0 + 1)) = U^*(k_0)$, i.e. $[u^*(1|k^*(k_0 + 1))^T, \ldots, u^*(N|k^*(k_0 + 1)^T, 0^{1\times 3N_d}]$, since the terminal state of $X^*(k^*(k_0+1)) = X^*(k_0)$ is a steady state for the system (3.32) (see (3.47j)). Thus, in this case $x(k_0+1)\mathbb{P}_{xy} \in \mathcal{S}(k^*(k_0+1))$. Therefore in both cases **A)** and **B)** $x(k_0 + 1)\mathbb{P}_{xy} \in \mathcal{S}(j)$, with $j \leq k_0 + 1$. Now, by construction each convex polytope $\mathcal{D}_j(k) \in \mathcal{S}(k)$ is an under-approximation of the obstacle-free region containing two drones and, due to its convexity, also the segment $(\boldsymbol{P}_{i+1_{(1:2)}} - \boldsymbol{P}_{i_{(1:2)}})$ representing the tether. It was thus proven that $x(k_0)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}$, $(\boldsymbol{P}_{i+1_{(1:2)}}(k_0) - \boldsymbol{P}_{i_{(1:2)}}(k_0)) \in \boldsymbol{S}_{free} \Rightarrow x(k_0 + 1)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}$, $(\boldsymbol{P}_{i+1_{(1:2)}}(k_0 + 1) - \boldsymbol{P}_{i_{(1:2)}}(k_0 + 1)) \in \boldsymbol{S}_{free}$. □

Remark 5. *Lemma 1 guarantees an obstacle free-trajectory for the control-oriented model* (3.32)*, however the real system exhibits a mismatch with respect to the model* (3.32)*. This mismatch can be estimated and bounded with suitable uncertainty quantification techniques (see e.g. [77, 78]) and it can be considered with suitable robust approaches (see e.g. [79, 80]).*

3.2.4 Results

In order to test the described procedures, several simulations were carried out in the same virtual environment, on a Quad-Core Intel Core i7 (2.8 GHz, 16 GB) on MATLAB 2021a, considering a system of three drones

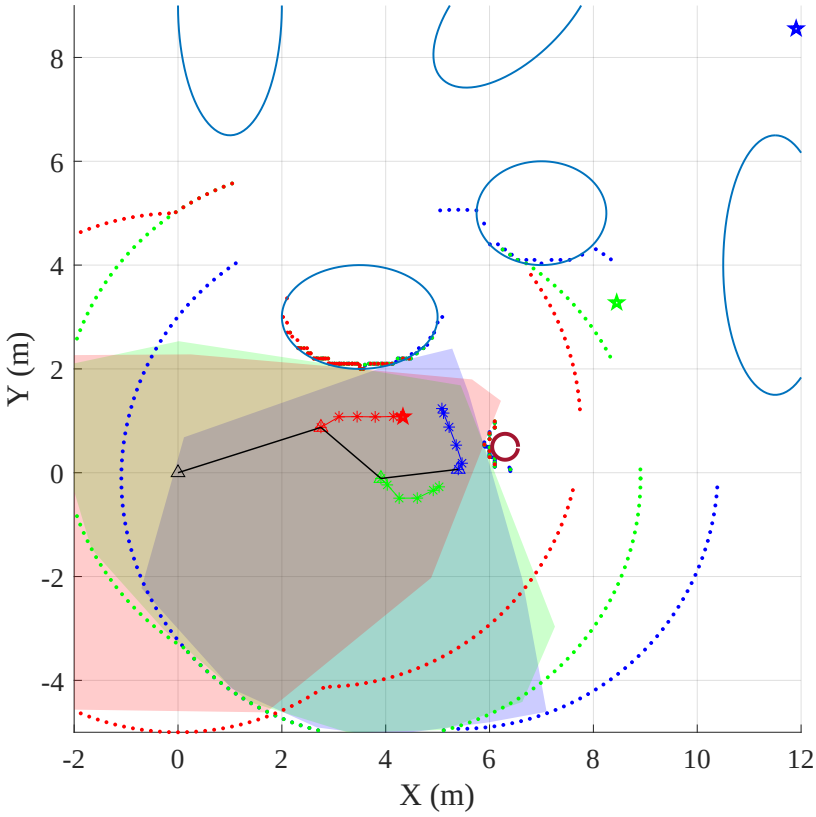

Figure 3.15: *A representation of the planner in action. Stars denote the tracked optimal configuration $\boldsymbol{C}^*$, whereas the lines with the '*' symbol are the predicted trajectories over the horizon N. Points represent the merged LiDAR readings $\mathcal{L}_{i,i+1}(k)$ while the colored surfaces are the polytopes $\mathcal{D}_j(k)$. Blue and red ellipses portray known and unknown obstacles, respectively.*

connected to a ground station. Results of our simulations captured in Fig. 3.16 show that the configuration planning procedure finds admissible solutions for the optimal configuration $\boldsymbol{C}^*$ in various scenarios. Furthermore, simulations of the on-line configuration tracking algorithm with a sampling time $T_s = 0.5$ s show that the system tends to the desired state, while each follower tries to position itself behind the next drone during the flight, see Fig. 3.16. This is the case in the presence of previously unknown obstacles too. Even though adding an obstacle in an arbitrary position could in principle make a previously optimal configuration not admissible, when this happens the system can converge to a slightly different one where the leader drone still ends up in $\boldsymbol{P}_{goal}$. Fig. 3.15 shows an example of this, where an obstacle is introduced near the point $(6.5,\ 0.5)$ after choosing the configuration, whose presence is revealed to the UAVs by the LiDAR readings. It induces the second drone in the chain to stop slightly farther than its originally intended destination. It should also be noted that the MPC problem described in Section 3.2.3 exhibits a non-convex quadratic constraint, and has an average solution time of 0.23 s . Though it depends on the hardware, such a solution time allows applications of this framework in safety critical environments. Furthermore, the solution process can be decentralized and tackled by each drone's on board computer. In fact, the nature of constraints is such that they only affect pairs of connected drones at most, thus the problem can be solved cooperatively by each agent calculating the next best control action for itself, as long as it has access to the positions and LiDAR readings of the previous and next drones to formulate suitable obstacle avoidance constraints.

3.2.5 Conclusion

An approach to navigate systems of tethered multicopters in a partially known environment has been presented, where an off-line approach computes the optimal configuration to reach a target considering the known obstacles, and a real-time MPC algorithm allows to reach the desired configuration despite the presence of unexpected obstacles. A novel approach to approximate the free-space with a convex polytope able to guarantee that the vehicles and tethers remain in an obstacle-free area has been used, together with a strategy able to guarantee the existence of a feasible problem at each time step and consequently guarantee obstacle avoidance. The current research activities aim to include uncertainty and model mismatch quantification to robustify the approach, distribute the problem between the different vehicles and include dynamic obstacles.

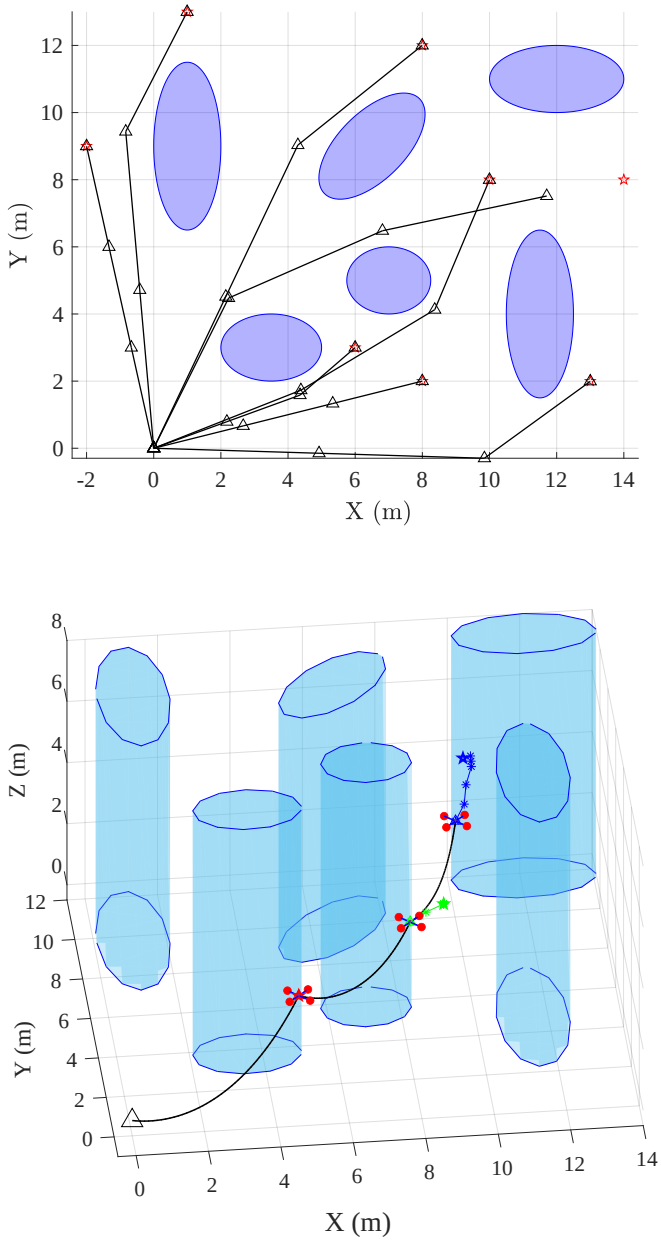

Figure 3.16: *A representation (top) of several runs of the off-line optimization. Each depicted configuration is the optimal solution of problem* (3.38) *with the goal of the leader drone (red star) as input. Note that for* $\boldsymbol{P}_{goal,1:2} = (14, 8)$ *the target is unreachable, due to the limitations on tether length. 3D view (bottom) of a simulation.*

3.3 Reactive Path Planning for Collaborative Tasks

A tethered connection between a drone and the ground can transfer things other than power. Some systems use UAVs to deliver fluids with precision, for tasks such as spray-painting [81], coating, cleaning [82] and watering. In such scenarios an autonomous and collaborative planner is necessary to guide the aircraft in an ideal position and keep the tether away from obstacles. In this work, we focus on the spray-painting task and we present a reactive motion planner for a drone that enables such task. A human operator holds one end of a hose to simulate the action, while the other end is fixed to the ground. A single drone is connected to the hose around its middle point, and its autonomous controller is tasked with allowing the maximum freedom of movement to the human operator while ensuring that the hose does not touch obstacles. This implies that the drone must steer clear of obstacles too. The project presented here is based upon a previous research effort, described in [83], and extends it by enabling the system to work in an unknown outdoor environment.

While several controllers for unconstrained drones have been designed and deeply researched in literature, the autonomous control of tethered or otherwise constrained UAVs is still developing. Initially, some results have been published regarding the control of drones that physically interact with the environment [84] and that use tethers specifically [85]. This opened the road to the exploration of multi-drone tethered systems, where the physical coupling of the vehicles plays a non-negligible role on the system's stability. Such systems can potentially be used for transporting loads [86].

A niche research area has developed around systems of tethered multicopters that draw power through the tether [55]. This configuration is offered as a solution to the problem of limited freedom of movement: by connecting multiple drones in series, the overall system enjoys more flexibility and can reach behind obstacles. Controllers have been proposed for autonomous system-level motion planning, based on different variations of the same system, with fixed length tethers [87] and reeling ones [23]. Another valid idea is to conceptualize the system as an over-actuated robotic arm and apply standard robotics path planning techniques [64]. These projects entail multiple aircraft, always guided by a centralized controller. Tethered drones have also been deployed for human interaction tasks. One important project in this regard is AERO-GUIDE [88, 89], where an autonomous UAV guides a blindfolded human through space by pulling the cable that connects them and controlling the force exchanged through it. The work in the present discussion stems from a previous project, where

a spray-painting task is simulated by a human [83]. As such, it is also based on the same theoretical framework, that of RMPs (Riemannian Motion Policies) [90].

With respect to the existing literature, and especially the previous iteration of this project [83], we provide the following contributions:

- the design of an improved controller, combining RMPs with the real-time readings of a LiDAR sensor, thus solving the necessity of a precise map of the environment. This not only enables the system to work in outdoor, previously unexplored environments, but it also allows the system to react to slowly moving obstacles;
- a method for estimating the state of the tether connecting the drone to the end effector, so that collisions with obstacles can be avoided, together with a comparison of two implementations;
- a technical description of the practical challenges in the realization of a working prototype of the system and our proposed solutions.

Throughout the discussion, we focus on the specific use case of spray-painting, tough the same considerations apply to similar tasks such as solar panel cleaning, coating, roof snow removal and firefighting, some of which can already be carried out with drones [91, 92]. Even though we focus on one use case, the planner we propose, like many others existing in literature, is broadly applicable to the wide class of tethered UAV systems and collaborative task, whether the end effector is a human or another robot. Furthermore, it can be used in tethered systems of multiple drones described for example in [23, 87], for which a fully decentralized solution does not exist yet, to the best of the authors' knowledge.

In the following section the problem is defined and hypotheses are introduced, together with notation. In section 3.3.2 we describe the system and in 3.3.3 the designed controller, then we present test results obtained in simulation and through experiments (section 3.3.4). We summarize our conclusions in section 3.3.5.

3.3.1 Problem Formulation

We design a controller for a system composed of a tether, representing a hose through which paint flows towards an end effector, a human operator holding such end effector and a single drone, fixed to the rope around its mid-point. The tether is fixed to the ground at the other end. The aim of our automatic controller is to guide the drone in space, without human intervention, to achieve the following list of goals:

- to facilitate the task for the human operator, by allowing him the maximum freedom of movement and holding part of the weight of the tether;
- to prevent collisions between the drone and obstacles, as well as between the tether and obstacles;
- to allow the system to operate in a previously unknown environment, either indoor or outdoor, by sampling it with a LiDAR sensor. The environment can contain slowly moving obstacles;
- to position the drone such that it is possible for its sensors to estimate the state of the rope by measuring the position of the end effector;

During the development of the controller, some hypotheses were introduced. Namely, we assume that obstacles have constant cross-section vertically and the human operator is always visible within the frame of the camera mounted on the drone, and his position can therefore be estimated. The position of the robot is defined as $\boldsymbol{q} \in \mathbb{R}^3$ We describe the position of the end effector as the point in 3D space $\boldsymbol{p}_e \in \mathbb{R}^3$, and the position of the fixed end of the rope as $\boldsymbol{p}_s \in \mathbb{R}^3$. We assume without loss of generality that $\boldsymbol{p}_s$ is constant through time.

3.3.2 Models

The Drone

We employ a hexacopter both in simulations and in experimental validation, though the planner is sufficiently high-level to be independent of the aircraft's configuration. The drone has low-level attitude and altitude controllers and it accepts position, velocity and acceleration references, such that it can autonomously hover in place or integrate an acceleration into a reference trajectory and then track it. The planner thus generates motion policies in terms of accelerations.

The Rope

To manipulate the rope, the system needs to know the shape of the rope. We estimate the shape by involving a rope model [REF]. The robe is modeled as a sequence of nodes with mass connected by springs. We evaluate the forces, e.g., gravity and external forces, added to the node one by one. Then, we update the rope state with position-based verlet integration. It integrates accelerations to positions while considering position-based constraints for representing collisions with the obstacles or the ground.

3.3.3 Method

Odometry

To hover and move through space autonomously and precisely, the drone performs odometry based on its on-board IMU and LiDAR sensor readings. The two are fused in a framework called FAST-LIO [93] (Fast LiDAR-Intertial Odometry), based on a computationally efficient extended Kalman filter approach, and resulting in a quick and rather precise estimation of the vehicle's position in the environment.

End Effector Detection

The real configuration of the rope is *a priori* unknown, but its knowledge is necessary to design motion policies that prevent its collision with obstacles, therefore it must be measured or estimated. Given that a rough estimate is sufficient for obstacle avoidance, we measure the position of the end of the rope the human is holding and combine it with the positions of the drone and the other, fixed end of the tether, then we exploit the model introduced in section 3.3.2 to estimate the positions of the lumped masses. Note that this approach can correctly estimate the configuration of the rope when it is in contact with no obstacles or with known ones, like the ground plane, but not with unknown ones. The estimate is therefore reliable as long as collisions are avoided.

Several solutions were considered to tackle the estimation problem, ruling out force sensors installed on the drone and measuring the force exchanged with the rope, as this would only yield an inaccurate estimate of the direction where the end effector lies with respect to the drone, but not the distance, which is crucial for establishing the position in 3D. Vision-based methods were employed and compared instead, whereby the end effector position is estimated by recording it with a camera. Because the human might occlude the visual line of sight, the position of the operator is taken as a proxy for the position of the end effector, assuming the latter can be obtained by adding a fixed offset to the former. This makes the estimation more robust during operation at the cost of accuracy, but our experimental validation shows that such loss does not significantly impact the operation of the system, as the planner not only prevents contact between rope and obstacles, but it also keeps a certain safety margin between them. We tested two frameworks in particular: one based on You Only Look Once (YOLO) [94] human detection and another one based on AprilTags [95].

YOLO The first technique consists in identifying the human operator in the image frame with a suitably trained YOLO neural network and back-projecting its position in 3D. Note that the projection from a single pixel in a 2D image to a point in 3D space is under-determined, therefore it is only possible to obtain a direction along which the object it represent lies, not the correct distance. To recover it, it is necessary to calculate a scale factor, *i. e.* the ratio between its height in the physical world and in the image frame at a known distance. Supposing, for example, a person of height H appears h pixels tall in the image frame when standing at distance l from the camera, the scale factor is:

$$s = \frac{hl}{H}. \tag{3.50}$$

The height in the image frame h is one of the outputs of the YOLO detection procedure, but the network cannot discriminate between a human figure completely contained in the image frame or just a part of it. As a consequence, the operator should be always inside the frame and completely visible to obtain a reliable estimate of his distance. Given the scale factor, it is possible to back-project the position of any object of known size from its appearance in the image. Adopting the standard convention of a camera reference system where X_c points to the right, Y_c points down and Z_c inside the image, and assuming that the human stands parallel to the image plane, the spatial coordinates of the middle point of the human operator with respect to the frame are:

$$X = \left(\frac{(w_{min} + w_{max})}{2} - c_x \right) \frac{Z}{f_x} \tag{3.51}$$

$$Y = \left(\frac{(h_{min} + h_{max})}{2} - c_y \right) \frac{Z}{f_y} \tag{3.52}$$

$$Z = \frac{sH}{h_{max} - h_{min}}, \tag{3.53}$$

where w_{min}, w_{max} represent the horizontal extrema of the rectangular bounding box containing the operator and h_{min}, h_{max} the vertical ones. The remaining symbols represent the camera intrinsic parameters calculated during calibration: horizontal and vertical focal lengths (f_x, f_y) and coordinates of the optical center (c_x, c_y). If the camera is mounted on the drone at an angle, *i. e.* its frame is such that Z_c is not parallel to the ground, but inclined by θ with respect to the horizontal plane, then the effect of such angle on the projection must be taken into account (see Figure 3.17), and

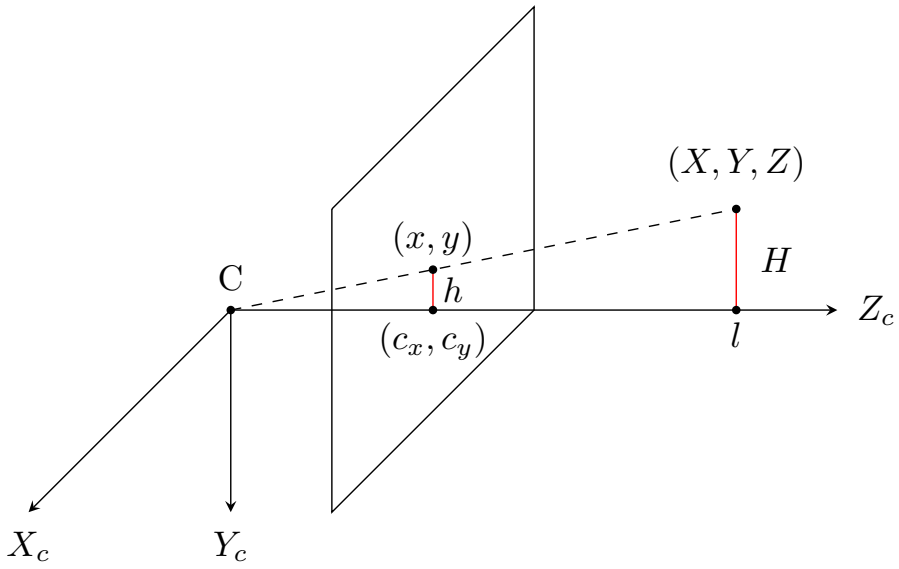

Figure 3.17: *Back-projection of the position of the human operator into 3D space.*

the estimate of the distance becomes

$$Z = \frac{sH}{h_{max} - h_{min}} \cos(\theta). \tag{3.54}$$

It is worth noting that the adoption of a neural network-based approach for image analysis and object detection requires the use of dedicated, specialized hardware like a Gprahics Processing Unit (GPU) to run the estimation at a sufficient rate without overwhelming the CPU.

AprilTags The second technique applies the same mathematical principles to estimate the distance of a set of visual markers with known physical sizes. At the cost of placing markers on the human operator in advance, one can obtain an estimate of his distance from the camera by employing a readily available software application [95]. Differently from the previous approach, the AprilTag library does not rely on neural networks, but rather on line and corner detection algorithms. Furthermore, it provides estimates of both the position and the orientation of a marker in space. To counter the fact that a marker might become invisible as a consequence of a rotation, it is possible to define marker bundles, *i. e.* sets of markers whose dimensions and relative positions are known. Mounting such a marker bundle on a helmet ensures the position of the operator can always be estimated, as

long as the markers are within the image frame and appear sufficiently big in it (see Figure 3.20). This method also theoretically allows a more precise tracking of the end effector, by applying the markers directly on it.

RMPs

Riemannian Motion Policies are a motion planning framework, recently developed in the context of robotic manipulation [90]. They are essentially differential equations that tie together the position, velocity and acceleration vectors of a robot, ans as such can be interpreted as maps: given a certain state (position and acceleration), a desired acceleration can be obtained by designing the shape and parameters of the differential equation. Applying this approach is particularly convenient because:

- several policies can be designed for separate tasks (goal-following, obstacle avoidance, ...) and then mathematically combined through a weighted *summation* that results in a consistent global behavior. The weights in the sum can be tuned to activate some policies in certain sub-spaces (*i. e.* only preventing the robot from accelerating in the direction perpendicular to an obstacle, but not in the parallel one) and to prioritize some behaviors over others (*i. e.* avoiding obstacles is more important than tracking a position reference);
- each policy can be formulated in in the space where it is most convenient to do so (either the task space $\mathcal{X}$ or the robot configuration $\mathcal{Q}$), and then combined through the *pullback* operation.

Supposing that $\boldsymbol{q} \in \mathcal{Q} \subseteq \mathbb{R}^d$ is the system's configuration in a $d-$dimensional configuration space, that $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ is a configuration in the $n-$dimensional task space, and that there exists a differentiable function $\phi : \mathcal{Q} \to \mathcal{X}$, then we can write $\boldsymbol{x} = \phi(\boldsymbol{q})$ and call $\boldsymbol{J} = \frac{\partial \phi}{\partial \boldsymbol{x}}$ the Jacobian matrix. We can then define a *natural form* as the pair $(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$ where the matrices $\boldsymbol{M}$ and $\boldsymbol{f}$ represent the dynamic of the system in the equation:

$$\boldsymbol{M}(\boldsymbol{x}, \dot{\boldsymbol{x}})\ddot{\boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0. \tag{3.55}$$

The same equation can be expressed in the *canonical form* $(\boldsymbol{M}, \boldsymbol{h})_{\mathcal{X}}$:

$$\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0, \tag{3.56}$$

with $\boldsymbol{h} = \boldsymbol{M}^{-1}\boldsymbol{f}$, and in the *policy form* $(\boldsymbol{M}, \pi)_{\mathcal{X}}$:

$$\ddot{\boldsymbol{x}} = -\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \pi, \tag{3.57}$$

π being the *acceleration policy*. To transform a policy from configuration space $\mathcal{Q}$ to task space $\mathcal{X}$, the *pullback* operation is defined as

$$pull_\phi(\boldsymbol{M}, \boldsymbol{f})_\mathcal{X} = (\boldsymbol{J}^T \boldsymbol{M} \boldsymbol{J}, \boldsymbol{J}^T(\boldsymbol{f} + \boldsymbol{M}\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}))_\mathcal{Q}. \tag{3.58}$$

Once all the p policies are expressed in the same space, each contributing a certain desired acceleration π_i with $i = 1, \ldots, p$ based on the system's position and velocity, they can be summed to obtain a single acceleration:

$$\tilde{\pi} = \left(\sum_i^p \boldsymbol{W}_i \right)^{-1} \sum_i^p \boldsymbol{W}_i \pi_i, \tag{3.59}$$

where the matrices $\boldsymbol{W}_i$ are suitably designed and tuned weights, also called priority metrics. We can then define the single policies to design the behavior of the system. Some have the same form as those in [83], with modified numerical values to better suit the different flying platform.

Goal Reaching The task of reaching a goal in space can be designed by writing a potential function $\psi(\boldsymbol{x})$ with a minimum in the desired position. The related acceleration can be obtained naturally as the negative gradient of such function $\ddot{\boldsymbol{x}} = -\frac{\partial \psi(\boldsymbol{x})}{\partial \boldsymbol{x}}$, pushing the system towards the destination. A suitable choice is therefore defining the function to be minimized as the difference between the current position and the desired one $\boldsymbol{x}_o = \phi(\boldsymbol{q}) = \boldsymbol{q} - \boldsymbol{q}_o$, where the desired position's x and y coordinates are

$$\boldsymbol{q}_{o,xy} = \boldsymbol{p}_{1,xy} + d_1 \frac{\boldsymbol{q}_{xy} - \boldsymbol{p}_{1,xy}}{\left|\left| \boldsymbol{q}_{xy} - \boldsymbol{p}_{1,xy} \right|\right|} \tag{3.60}$$

and the height along the z axis is

$$q_{o,z} = p_{1,z} + d_1 \cos(45°) = p_{1,z} + \frac{d_1}{\sqrt{2}} \tag{3.61}$$

This defines the goal of the drone as the point in space d_1 meters away from the end effector, obtained by traveling $\frac{d_1}{\sqrt{2}}$ meters along the direction between the current end effector position $p_1 \in \mathbb{R}^3$ and the current drone position q along the xy plane, and $\frac{d_1}{\sqrt{2}}$ upwards from there. The height differential is necessary so that the camera, mounted at a $45°$ angle, keeps the end effector as close as possible to the center of the image frame, so that it is possible to estimate its position. A convenient negative gradient is then defined as:

$$\frac{\partial \psi(\boldsymbol{x}_o)}{\partial \boldsymbol{q}} = k_\psi \tanh(\lambda_\psi \phi(\boldsymbol{q})). \tag{3.62}$$

The hyperbolic tangent form has the advantage of keeping the acceleration constant when the distance from the goal is big, and quickly reducing it to zero when such distance is small. The maximum acceleration is determined by k_ψ and λ_ψ specifies how quickly acceleration decreases once the drone approaches the target.

A straight-forcing policy is also added to manipulate where the drone will converge, so as to align its position along the direction connecting the end effector to the point $\boldsymbol{p}_N$ where the rope is fixed:

$$\boldsymbol{x}_s = \phi(\boldsymbol{q}) = \boldsymbol{q} - \boldsymbol{q}_s \tag{3.63}$$

$$\boldsymbol{q}_s = \boldsymbol{p}_1 + d_1 \frac{\boldsymbol{p}_N - \boldsymbol{p}_1}{||\boldsymbol{p}_N - \boldsymbol{p}_1||}. \tag{3.64}$$

The associated potential is identical to that of the previous policy

Obstacle Avoidance We design a set of policies to prevent collisions both between obstacles and the drone and between obstacles and the rope. A natural way to do so is to define as task state a vector $\boldsymbol{x}$ pointing from the point on obstacle to be avoided to the drone or rope, and then once again define the acceleration as the negative gradient of a function of the task state: $\pi(\boldsymbol{x}) = -\frac{\partial \psi(\boldsymbol{x})}{\partial \boldsymbol{x}}$. The related priority metric can be set for simplicity as $\boldsymbol{W} = \boldsymbol{I}$, although it is also possible to formulate it as a function of robot's state $\boldsymbol{W}(\boldsymbol{x}, \dot{\boldsymbol{x}})$. As such, the policy could be forced to only activate for the components of the robot's speed that are approaching the obstacle and not for the parallel ones, thus preventing unnecessary speed reduction [96]. Note that the obstacle shape is unknown and only partially observable through the LiDAR sensor, therefore a sensible strategy is to avoid, at each time step, contact with the closest detected obstacle by accelerating in the opposite direction if the distance is too small. Another feasible strategy is to obtain the same acceleration by applying the same considerations to a greater number of detected obstacle points, but the only advantage of doing this would be to reduce oscillating behavior when the drone is confined in a tight corridor-like space between two obstacles.

For the drone collision avoidance policy, we therefore design the task map as

$$\boldsymbol{x}_d = \boldsymbol{\delta}_d = \boldsymbol{q} - \boldsymbol{q}_c, \tag{3.65}$$

where $\boldsymbol{\delta}_d \in \mathbb{R}^3$ represents the vector pointing from the closest obstacle point $\boldsymbol{q}_c \in \mathbb{R}^3$ measured by the drone's LiDAR sensor to the current posi-

tion of the drone $\boldsymbol{q} \in \mathbb{R}^3$. The acceleration policy is defined as

$$\pi_d = -(\zeta_d + (1 - \zeta_d)\dot{\boldsymbol{q}}^2)\frac{k_d \boldsymbol{\delta}_d}{||\boldsymbol{\delta}_d||} \tag{3.66}$$

$$\zeta_d = \frac{1}{2}(1 - \tanh(\lambda_d(||\boldsymbol{\delta}_d|| - l_d))), \tag{3.67}$$

where ζ_d is a switching function parameterized by λ_d, supporting two different behaviors when the robot is closer to the obstacle than a certain tunable threshold l_d and when it is further away. In the former case $||\boldsymbol{\delta}_d|| \ll l_d$, therefore $\zeta_d \approx 1$ and the resulting acceleration is $\pi_d = -k_d \frac{\boldsymbol{\delta}_d}{||\boldsymbol{\delta}_d||}$, *i. e.* an acceleration of magnitude k_d pointing away from the obstacle. In the latter case $||\boldsymbol{\delta}_d|| \gg l_d$, therefore $\zeta_d \approx 0$ and the resulting acceleration is further scaled by the squared speed $\dot{\boldsymbol{q}}^2$, which enables the robot to hover at l_d from the obstacle when the speed approaches zero. The priority metric $\boldsymbol{W}_d$ is also a function of the robot's speed:

$$\boldsymbol{W}_d = \frac{1}{2}(1 - \tanh(\lambda_d \dot{\boldsymbol{q}} \cdot \boldsymbol{\delta}_d)). \tag{3.68}$$

This weighs the policy proportionally to the projection of velocity component along the obstacle-drone direction $\dot{\boldsymbol{q}} \cdot \boldsymbol{\delta}_d$, scaling down the priority when the speed is not directed towards the obstacle.

For rope-obstacle avoidance, we define policies of the same form with small differences. One policy per rope model node is defined, whose task state is

$$\boldsymbol{x}_r = \boldsymbol{\delta}_d^i = \boldsymbol{p}_i - \boldsymbol{q}_{c,i}, \tag{3.69}$$

where $\boldsymbol{\delta}_d^i$ is the vector pointing to the current position $\boldsymbol{p}_i$ of the $i-$th node in the rope model, from the obstacle point $\boldsymbol{q}_{c,i}$ closest to it. For each task, the related policy π_r^i has the same shape as (3.66), depending on a switching function ζ_r mirroring (3.67), including analogous parameters with possibly different numerical values. The priority metric is instead the combination of two components, $\boldsymbol{W}_r = \boldsymbol{W}_{r,1}\boldsymbol{W}_{r,2}$. While $\boldsymbol{W}_{r,1}$ retains the same form of (3.68), we add a second criterion for weighting:

$$\boldsymbol{W}_{r,2} = \frac{1}{2}(\tanh(\lambda_r(d_r - s_r)) + 1)\boldsymbol{I} \tag{3.70}$$

$$d_r = d_{1,max} - ||\boldsymbol{q} - \boldsymbol{p}_1||. \tag{3.71}$$

This penalizes the stretching of the rope between the drone and the end effector past a certain threshold $d_{1,max}$, as parameterized through λ_r and s_r.

Energy regulation After combining the defined policies through summation, obtaining a system formulated as $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0$, it is necessary to ensure that the system converges and does so in a reasonable time. We do so by adding the term $\alpha_{\mathcal{H}_e}\dot{\boldsymbol{x}}$ to the equation of the system, yielding

$$\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \alpha_{\mathcal{H}_e}\dot{\boldsymbol{x}} = 0. \tag{3.72}$$

By relying on the concept of energized system introduced in [97], we define the energy Lagrangian $\mathcal{L}_e$ with energy $\mathcal{H}_e$ and the related equation of motion:

$$\mathcal{L}_e = \frac{1}{2}\,||\dot{\boldsymbol{x}}|| \tag{3.73}$$

$$\mathcal{H}_e = \frac{\partial \mathcal{L}_e^T}{\partial \dot{\boldsymbol{x}}}\dot{\boldsymbol{x}} - \mathcal{L}_e \tag{3.74}$$

$$\boldsymbol{M}_e\ddot{\boldsymbol{x}} + \boldsymbol{f}_e = 0. \tag{3.75}$$

By imposing that the derivative of energy over time is null, we obtain

$$\dot{\mathcal{H}}_e = \dot{\boldsymbol{x}}^T\left[\boldsymbol{M}_e(-\boldsymbol{h} - \alpha_{\mathcal{H}_e}\dot{\boldsymbol{x}}) + \boldsymbol{f}_e\right] = 0 \tag{3.76}$$

$$\alpha_{\mathcal{H}_e} = \frac{\dot{\boldsymbol{x}}^T\left[\boldsymbol{f}_e - \boldsymbol{M}_e\boldsymbol{h}\right]}{\dot{\boldsymbol{x}}^T\boldsymbol{M}_e\dot{\boldsymbol{x}}}. \tag{3.77}$$

For a more comprehensive discussion of energized systems, damping and the effects on convergence, the reader is referred to [83] and [96].

3.3.4 Validation

The Platform

We validated our controller both in simulations and in experiments in the real world, employing the custom built hexarotor drone visible in figure 3.18. The platform was endowed with low-level attitude and altitude controllers, and able to track position and trajectory references. Besides the onboard standard PixHawk, Proportional-Integral-Derivative cascade flight controller with IMU, necessary for telemetry and flight stabilization, several other essential components where installed on the aircraft:

- an Ouster OS1 LiDAR sensor, to detect the environment (see Table 3.2 for technical specifications). Note that this sensor comes with its own IMU mounted inside. The LiDAR is not only necessary to localize obstacles for collision avoidance, but also to identify features in

the environment and aide the estimation of odometry. Such estimation was carried out through the publicly available FAST-LIO framework [93], fusing together LiDAR and IMU data, and proved particularly accurate and stable, especially outdoors. This is probably due to indoor environments being delimited on the sides mostly by vertical, feature-less walls, which translates into the estimator not identifying displacement along the vertical direction as accurately. Given the sensor's position on the top of the drone, the arms and propellers of the aircraft partially blocked its view, therefore it was necessary to mask out readings too close to the drone and to take into account the lack of data in the obstructed areas. The relatively limited vertical field of view also made it impossible to detect obstacles directly below the drone. To prevent unwanted landings, the drone was therefore endowed with a model of the environment only consisting of a ground plane, with height equal to that of the drone when it took off. This allows it to correctly simulate the behavior of the rope while it rests on the ground and to avoid the ground as an obstacle;

- a Firefly S FFY-U3-16S2C-S color camera, with a wide angle lens (technical specifications in Table 3.3). As previously mentioned, the camera was mounted below the drone, facing forwards and tilted $45°$ downwards. Such placement forces the drone to hover higher than the operator, which in turn also prevents the rope from dragging along the ground. The field of view amplitude is not critical, as the drone is controlled to keep the human operator in the center of the frame, but it should be wide enough to capture his movement before the controller is able to react. Frame rate is not a limiting factor, but resolution is: the higher the resolution, the farther away the operator (and the markers) can goo while still being correctly detected. As a consequence, the desired operator distance from the camera poses a lower bound on the necessary camera resolution.

- an Intel® Neural Compute Stick 2, to run the YOLO neural network for human detection in real time. While in practice it is possible to run the model on the aircraft's CPU together with the controllers and the rest of the software, its limited computational power would be saturated at all times, possibly leading to unwanted behavior. It is therefore necessary to install this dedicated hardware, at the cost of one USB3.0 port, additional development effort, and non-negligible installation, set-up and configuration time. We note that the Neural Compute Stick 2 sometimes exhibited overheating, inducing connec-

Range	45m
Minimum range	0.3m
Range resolution	0.01m
Operating frequency	20Hz
Horizontal field of view	360°
Vertical field of view	45°(−22.5° to +22.5°)

Table 3.2: *Operating parameters of the Ouster OS1 LiDAR.*

Resolution	1440×1080
Megapixel	1.6MP
Focal length	3.5mm
Lens format	$1/2.5$”
F Aperture	$F1.8$

Table 3.3: *Operating parameters of the Firefly camera with the selected lens.*

tion problems, especially in the debug phases, when running while the drone was not flying. The air pushed by the propellers during flight is enough to solve the problem during operation. Given the hardware and the specific neural network, the detection of humans in the image frame ran at roughly 1 Hz.

Simulations were executed on a laptop with an Intel® Core™ i7-8750HQ processor, running ROS 1 and the Gazebo simulator on Ubuntu 20.04.

The Tests

We mainly tested two aspects of our system: its ability to detect the end effector and establish an estimate of the sate of the rope, and its ability to navigate space according to the designed policies while avoiding obstacles.

Obstacle Avoidance To test this policy in simulation, we manufactured an environment with a single building, then guided the end effector around a corner, so that the vehicle would follow it and on one hand be pushed to align with the operator and the fixed point of the rope by the straight-forcing policy, and on the other prevented from doing so by the collision avoidance one. Figure 3.19 represents a snapshot of the behavior of the system at the equilibrium, demonstrating that the drone does hover at l_d meters from the obstacles. The dynamic evolution of the system during the simulation clearly shows that the drone approaches the obstacle while following the end effector, and then gets pulled away from it by the repulsive acceleration generated by the obstacle avoidance policy.

Figure 3.18: *The Kiwi platform deployed in the tests.*

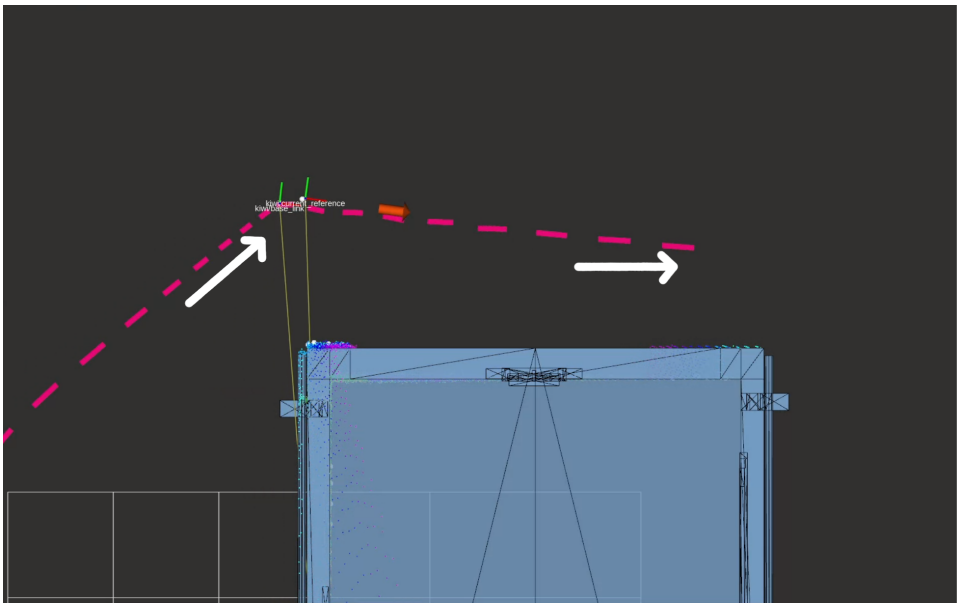

Figure 3.19: *A top-view snapshot of the drone avoiding the obstacle in light blue. Arrows indicate the drone's trajectory (left) and the end effector's trajectory (right). The rope is the red dashed line.*

A similar experiment was also carried out in the physical world, where the drone followed the human operator navigating around a tank (see Figure 3.20). As in the previous case, the drone exhibited the desired behavior, by following the operator at a certain distance and avoiding getting too close to the obstacle.

End Effector Position Estimation The position estimation was tested both indoors and outdoors. In the first test (Figure 3.22, the drone hovered at a fixed position in mid air and a human operator with a helmet walked around it. A pilot rotate the drone to keep the operator inside the image frame, while the onboard computer performed both AprilTag detection and YOLO-based estimation of his position in space. Both offer a reasonably accurate estimation of the position, which was compared in this test against a ground truth provided by a high frame-rate Vicon® motion capture system. Nevertheless, they differ quite significantly: due to imperfect detection of the operator in the picture, the bounding box containing it is not always precise, and this influences the calculation of the distance between him and the camera. Furthermore, this method is not robust to more humans entering the picture and to partial detection, should the operator

Figure 3.20: *The tank experiment.*

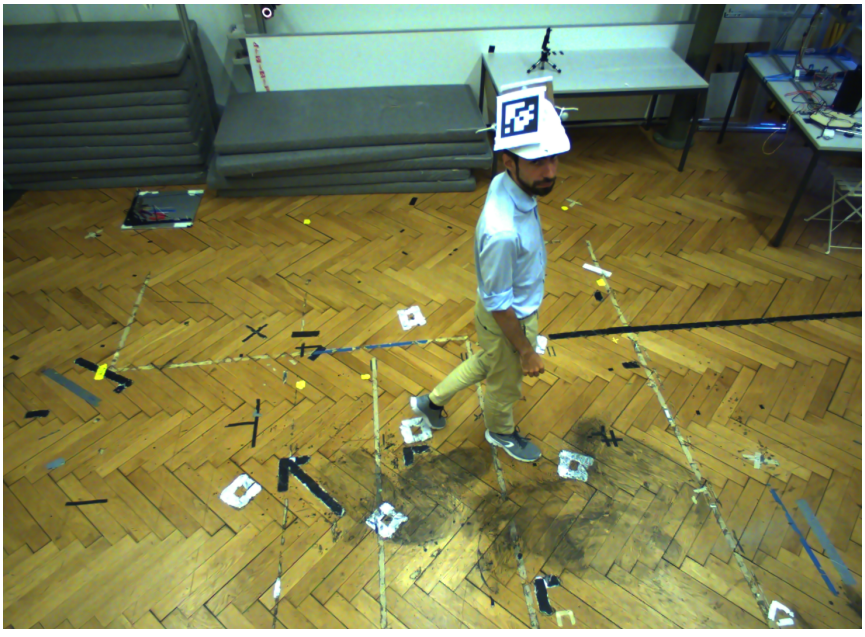

Figure 3.21: *Indoor test of AprilTags and YOLO.*

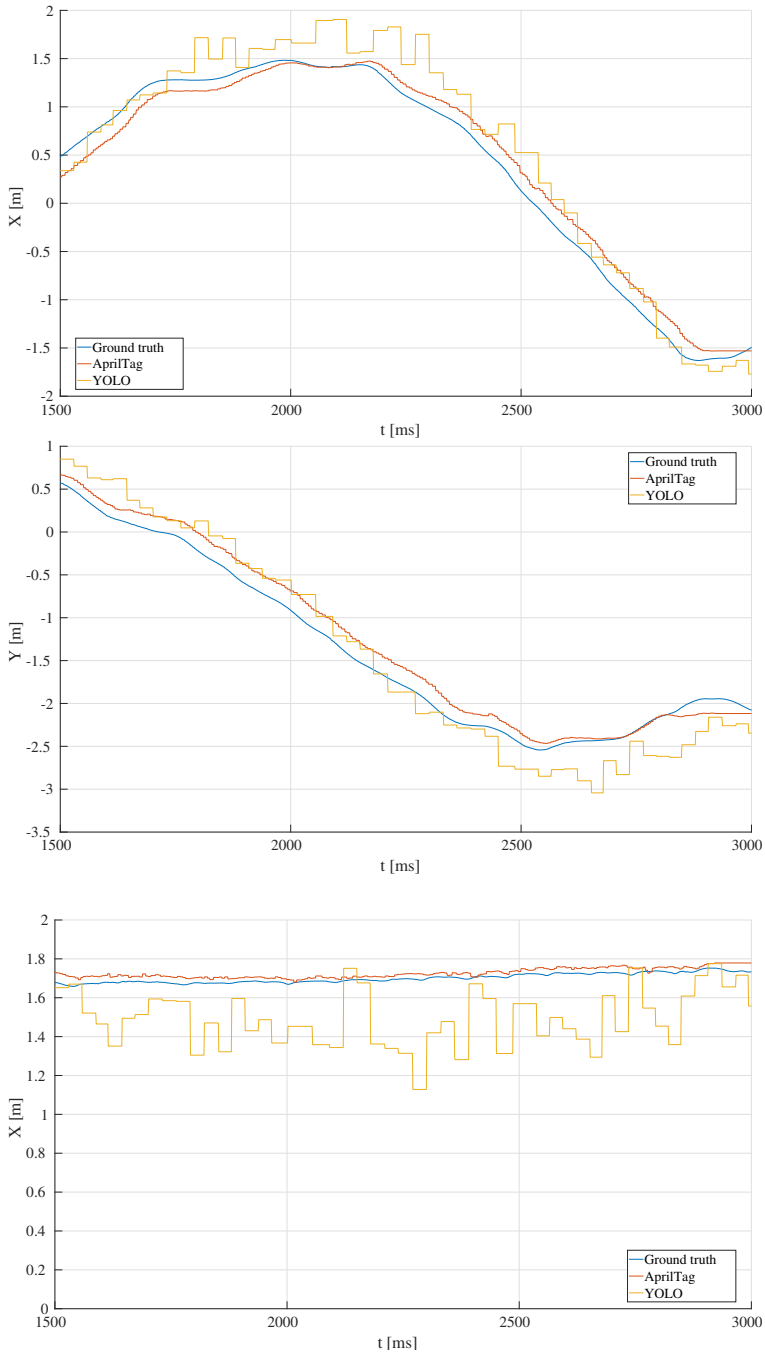

Figure 3.22: *Testing AprilTags and YOLO, the three graphs depict the three coordinates: x top, y middle, z bottom. Note that YOLO estimation is reasonably accurate, but jumps frequently.*

only be partially in the image frame. In this case, the human would appear smaller in the frame, inducing the controller to conclude that he is further away than he really is and moving the drone closer to him, potentially causing harm. The same would happen if the operator were, for example, to crouch or climb a ladder. Conversely, the AprilTag detection method offers the possibility of targeting the end effector directly instead of the operator, provided it is not obstructed and markers are big enough. We elected not to do this in order to gain a longer detection range with the bigger markers on the helmet. In addition to that, AprilTags bundles can be detected at 7Hz to 10Hz, while YOLO is roughly one order of magnitude slower given our hardware, resulting in a much jumpier signal, which can negatively affect the generation of a reference position for the drone. Given all of these considerations, the only real advantage the YOLO approach could bring is a higher maximum detectable distance, which is not worth all of the other technical drawbacks, including the necessary additional hardwer. Hence, we decided to rely on AprilTag detection for the following tests.

The detection and tracking capabilities were then validated in a following experiment, where the helmet with markers was installed on a moving cart and the estimate of its position was used to obtain a reference for the drone as described in section 3.3.3, before being deployed in the one described in the tank experiment in the previous section. These tests verified that the controller was capable of steering the drone to keep the the target near the center of the image frame at a fixed distance. Nevertheless, they highlighted an oscillating behavior induced by the non-smoothness of the target position reference estimation. This became especially apparent when the helmet was still and the drone hovering around a fixed position: noise in the estimated position of the marker bundle was reflected directly in the generation of a position reference for the drone. This was solved by introducing a simple filter that would not update the reference position $\boldsymbol{q}_0$ if it was very close (< 20cm) to the last generated reference position.

It should also be noted that during subsequent tests, where a human operator with the helmet simulated spray-painting a tank, the lens of the camera slowly unscrewed, losing focus in the process. We incidentally ended up verifying that the AprilTag detection is also surprisingly robust to focus loss: the detection of the markers was accurate even in the conditions shown in Figure 3.23.

Figure 3.23: *An out of focus image, where the detector was nevertheless able to pick up the marker on the helmet.*

3.3.5 Conclusion and Future Work

We presented a RMPs-based controller for the high-level motion planning of a human-collaborative drone for the spray-painting task in an unknown environment. We described the theoretical background and design choices, highlighting different policies for the goal tracking, straight forcing and collision avoidance behaviors. We also provided two methods for detecting the end effector and estimating its position, with the aim of using such measure to estimate the state of the whole rope, and compared their performance. We tested the proposed controller both in simulations and in a real world, realistic scenario, validating our hypotheses and design choices and demonstrating a working prototype. Some limitations still exist, such as the inability to detect obstacles below the drone, which could in some cases induce collisions between them and the rope. Future developments of this project could entail a working demonstration of the spray-painting task, and will likely address the issue of invisible obstacles through the use of a map of the explored environment.

Part II

UAV-based Sensing and Image Processing for Structural Health Monitoring

As previously established, drones are very agile and versatile. They can potentially transform a sensor into a flying one. In the hands of a skilled professional, this can reduce inspection times by and order of magnitude or more; in fact, UAVs have already started revolutionizing the field of visual inspections by allowing an operator to quickly inspect structures from privileged points of view. Gathering hundreds of pictures of a tower or bridge from all angles could take hours and expensive machinery, without a drone. The next step in actualizing the potential of this technology is their automation: by enabling drones to perform inspection tasks without human intervention, costs and times can be driven down further. The road to full autonomy is long and complex, touching several issues simultaneously, like mapping, precise localization and mission planning, and the different approaches necessary for different kinds of inspections. Nevertheless, several research efforts are under way world-wide to cover it. The following chapters detail some of these efforts in vision-based modal analysis and energy efficiency assessment.

CHAPTER 4

Vision-Based Modal Analysis[1]

4.1 Vision-Based Modal Analysis of the Built Environment with Multiple Drones

The advancement of autonomous and drone-assisted inspection techniques is especially important in developed countries. Their aging infrastructure poses serious safety risks and steep maintenance costs. Owing to recent economic crises, large investments for infrastructural renewal in the foreseeable future are in general not planned, therefore management and maintenance are critical. It is calculated that both the USA and the combined countries of the European Union spend each year a hundred billion dollars in road maintenance alone [98]: infrastructure costs still account for a substantial fraction of national budgets; It follows that any method to reduce costs is at least worth investigating.

Within the maintenance and inspection field, we focus on vibration-based SHM, which is commonly performed with accelerometers. These sensors measure accelerations in three dimensions (3D), which can be used to estimate modal parameters such as natural frequencies and modal shapes

[1] This work was previously published by the author, largely in the same form, in [20]. It is reported here with the permission of the editor and copyright holder, Elsevier.

[99]. Nevertheless, a wide network of synchronized sensors is necessary to identify modal shapes with a high spacial resolution, and accelerometers are expensive to maintain. Additionally, most existing structures were built without an embedded sensor network, thus performing the analysis would entail installing accelerometers ad-hoc, a cumbersome and expensive task. Vision-based modal analysis shines in these scenarios: cameras can capture videos of relatively large sections of a structure, thus collecting more data than a single accelerometer [100]. Furthermore, they measure displacement rather than acceleration, although they do so more accurately along the two directions orthogonal to the camera axis than along the parallel one [101]. This typically results in a more accurate displacement measurement than the one obtained by integrating acceleration twice, due to the possible noise-induced drift the integration process. In this work, we investigate the potential of vision-based SHM with multiple drones and provide the following contributions. First, we introduce a mechanism through which multiple drone-mounted cameras are synchronized and data can be gathered about large sections of a structure at a time; second, we validate the new approach experimentally on a small cantilever structure and evaluate the accuracy with respect to both a fixed camera and accelerometers, as well as the capability to detect and isolate alterations, such as added mass or a faulty connection to the supporting structure. In the process, we show that markers are not necessary to produce an accurate measurement, since the approach can exploit visual features of the structure, such as a junction, a bolt or a welding spot.

The remainder of this chapter is organized as follows. After a brief overview of the literature, we describe the problem in section 4.1.1 and present our method in section 4.1.2. Section 4.1.3 details the experimental set-up, and the following one presents the results of our experiments. The final section contains our conclusions and a discussion of the limitations of the approach in its current state, and it points to future development paths.

Literature Overview

Vibration modal analysis has been developed as a tool for SHM decades ago [102,103]. It is defined as the process of determining the dynamic characteristics of a mechanical system in terms of natural frequencies, modal shapes and damping factors. These can be used to formulate a modal model of the structure, and their variation over time can be tracked as an indicator of its health. Modal analysis is traditionally performed with accelerometers, but the decreasing costs of cameras and computational power has pushed the development of computer-vision methods [104]. It has been shown that,

by measuring displacements with a fixed camera, it is possible to estimate the first modal frequencies and recreate the modal shapes of a beam that is excited by an external input in a real world scenario. In [105], the displacement of a structure across time is measured by identifying features in the first frames of the recorded video and tracking their motion in the following frames with the Lucas-Kanade [106] algorithm, based on optical flow calculation. One advantage of such a technique is that there is no need to install markers on the structure. Similarly, some studies [107, 108] attained subpixel accuracy in displacement measurement using markers on the target structure, thanks to pattern matching techniques adopted in image processing, both in laboratory and in realistic conditions. This fact suggests that markers could be exploited where accuracy is critical, such as in scenarios where the camera is far away from the structure or the oscillations are small enough to be hardly distinguishable from measurement noise. Furthermore, the authors of [109] have shown that natural frequencies can be estimated reliably with inexpensive consumer grade cameras, at least in laboratory conditions. While applications of these techniques in realistic scenarios are possible, further data processing is often necessary to detect oscillations of low amplitude [110, 111]. Most of the real buildings where these techniques are tested are bridges or slender structures [108, 111–113], which tend to exhibit wider oscillations than other structures. The same techniques can be applied to measurement of frequencies and mode shapes of cables [114], with applications to cable-stayed bridges and stadium roofs, among others. Natural frequencies, in particular, can be used in the estimation of cable tension, provided that its length, mass and elasticity are known [115–117]. Photogrammetry and other optical methods involving multiple viewpoints have been employed as well for structural dynamics measurement [118], especially with Digital Image Correlation (DIC). Such techniques can achieve very high precision 3D measurements and support stitching of multiple views together [119], but they often rely on the presence of a custom-painted speckled pattern on the surface of the target structure [120].

Drones too have been studied and developed as tools for SHM. Recent projects deploy them as platforms for digitalization and Building Information Modeling [121], and even mobile contact sensors for non-destructive testing [122, 123]. Some technologies have been developed for the contact-based inspection of reinforced concrete [123]. In this scenario, a UAV is fitted with electrical resistance sensors, in order to detect cracks and corrosion in steel under the surface. Other applications include inspection of oil and gas pipes [124]. The main application of UAVs, though, is in visual

inspection, where they excel due to their freedom of movement along the three spacial dimensions [24], the ability to hover in place and the variety of cameras they can host. When equipped with thermal cameras, they can be used for thermography, offering for example extremely quick surveys of the status of solar panel fields [125]. Regular RGB cameras are exploited for photogrammetry [126] and acquiring photographs that are subsequently processed through machine learning techniques [25], aiming at automatically detecting surface cracks, concrete spalling, rust, humidity and other surface-level issues.

Several video-based techniques for SHM using regular cameras have been proposed, and the issue of multiple camera synchronization is well known in literature. The fixed nature of regular cameras makes it possible to use wired connections, thus frame synchronization is achieved with trigger signals sent to all sensors at the same time. This is possible through simple configurations where all cameras are connected to a central clock [127] and also in more complex networks relying on the Ethernet standard [128]. Regardless of the technology, the error is as low as a few microseconds. Such a low error enables approaches like DIC for vibration measurement [129]. Unfortunately, due to their reliance on a physical connection between the cameras, these solutions cannot be applied to drone-mounted sensors.

Still, some drone-mounted video-based techniques have been proposed to monitor the condition of buildings. The authors of [130] deployed a commercially available drone to measure the displacement of a structure, both in laboratory and on a real railway bridge. They performed natural feature tracking to estimate the displacement of the targets, and compensated egomotion by estimating it with respect to background features. Results were compared to measurements taken with a fixed camera + LED system and found to be of good quality, though some simplifying hypotheses were introduced in the design phase of the experiment. Furthermore, the authors noted that their technique of egomotion compensation is not robust. The authors of [131] aimed at estimating modes of vibration of a scale model of a wind turbine, obtaining promising results through DIC. The same technique was exploited in [132] to estimate two-dimensional strain. In [14], a single commercially available drone was used to measure the oscillating motion of both a six-story building model and a real pedestrian bridge. Displacement was measured through marker detection and Kanade-Lucas-Tomasi (KLT) tracking across video frames. Measurement quality was found to be remarkable even in the more realistic case on the pedestrian bridge, whose motion was forced by people jumping on the deck. Some limitations of the approach are mentioned, namely: the low number of acquisitions that

can be made with a single battery, the issue of detecting natural frequencies within the bandwidth of the motion of the drone, the issue of positional stability with respect to the target in GPS-denied environments, and the issue of applying and maintaining a significant forcing input to a larger, more rigid structure. At least for the first modes of vibration, the comparatively low Nyquist frequency introduced by cameras with respect to accelerometers is usually not an issue. In fact commonly available cameras easily reach at least 30 Frames Per Second (FPS), whereas the first natural frequencies appear in the $0 - 10$ Hz bandwidth.

Scope

In the state of the art, we note the absence of drone-enabled vision-based vibration modal analysis techniques that can capture the response of large spans of structure at once with high spacial accuracy: [130] achieves high accuracy but focuses on a single area of the building, whereas [14] calculates the whole modal shape of a pedestrian bridge, but it does so by sequentially repeating the displacement measurement with a single drone-mounted camera on several sections while an external force is continuously applied to the deck. This trade-off between accuracy, acquisition time and completeness is due to the fact that when a single aircraft is deployed, it must hover rather close to the structure to measure its displacement with sufficient accuracy through the video analysis, thus losing the possibility of capturing the whole building at once. By using multiple drones, instead, it is possible to perform in parallel accurate measurements of larger stretches at the same time, thus also overcoming time limitations due to battery charge and removing the need to maintain a continuous external forcing input for a prolonged amount of time. Furthermore, to the best of the authors' knowledge no solutions exist in literature to the problem of drone-mounted camera synchronization. As previously noted, the non-stationary nature of drones makes it impractical to physically connect them (even though some research in this direction is being carried out, see [23, 55]), therefore a fundamentally different solution must be found.

In this context, we present an experimental feasibility study of a novel system to perform video-based modal analysis through multiple cameras mounted on collaborative drones, that measure natural frequencies and estimate modal shapes. Our approach exploits the idea of overlapping the target areas measured by two different cameras to perform *a posteriori* signal synchronization. With respect to [14], we show experimentally that markers on the structure are not necessary to measure displacement with sufficient accuracy to extract modal shapes.

Using multiple UAVs at the same time can provide a more complete view of the response of a structure to an arbitrary input excitation with limited error propagation, which is hardly possible when using a single device, addressing the mentioned research gap. Deploying several drones at a time can also help to reconstruct the 3D motion of the structure by capturing it from several different points of view, thus overcoming another limitation of a single camera, *i.e.* the reduced accuracy at which displacement is measured along the direction of the camera's axis. In this regard, using several cameras with different orientations with respect to the structure can effectively perform the same type of measurement an accelerometer network offers.

We show promising preliminary results in a test case where two drones are employed. Using commercially available UAVs similar to those we adopted, this procedure can already be applied to small-size structures without incurring in complexity scaling issues. For example, with three aircraft hovering 2.5 m away from the target surface and recording footage with 1920×1080 resolution, it is already possible to capture the oscillations of structures less than 10 m long, such as barriers, antennas, parapets, small metallic pedestrian bridges or portions of larger structures.

4.1.1 Problem Description

The problem at hand is to obtain the modal parameters of a structure, namely natural frequencies ω_i and modal shapes $\mathbf{\Phi}_i$, where $i = 1, ..., n$ is an index that identifies the different modes, up to the n-th. This is to be done through vision only, without relying on contact. Our main objectives are three:

- Identifying natural frequencies and modal shapes through displacements measured using cameras mounted on drones;
- Establishing if our method is sufficiently accurate to detect changes in modal parameters due to damage of the structure;
- Testing the viability of multiple drones to obtain and fuse information collected across the whole structure.

4.1.2 Proposed Method

The procedure for estimating the modal parameters of the structure is summarized in these steps:

1. Recording the behaviour of the structure and identifying features in the collected frames;
2. Measuring the structure's displacement by tracking the identified features across frames;
3. Estimating natural frequencies from the measured displacement;
4. Estimating the modal shape for each identified natural frequency.

We adopt the classic pinhole camera model to describe our visual sensors, hence we can write the following relationship between the 3D position of objects in the real world $\mathbf{P}$ with respect to a global reference frame and their positions in the image plane $\mathbf{p}$:

$$s\mathbf{p} = K\,[R|v]\,\mathbf{P}. \tag{4.1}$$

Here s is the scale factor, R and v are the rotation matrix and translation vector that operate the projection onto the 2D plane and K represents the intrinsic parameters of the camera. The lens distortion is corrected after camera calibration. Our experiments verify that such a model does not negatively affect the accuracy of the measurements in a significant way.

Recording and Feature Detection

In this study, the structure is excited with a kinetic hammer and its oscillation is recorded by the cameras. In order to track the displacement of an object, *i.e.* the movement of the targets across pixels in consecutive frames, we first apply the Harris corner detection algorithm [133] to find features within the gray-scale version of the first frame. Some of those features are manually selected for tracking: automation of this step will be the object of further study. This procedure is based on an estimation of the bi-dimensional gradient of the color in each frame and can reliably distinguish homogeneous areas, edges, and corners. Such an algorithm is compatible with markers featuring many corners such as ArUco [134], like the ones in Figure 4.1, but also works without them. We initially chose to exploit the markers to ease the feature selection procedure, maximize robustness of the measurements and streamline the presentation of the algorithm. Nevertheless, we also repeated the measurement by tracking different features, belonging to the surface of the structure. We show in section 4.1.4 that markers are not strictly necessary to track structural displacement. In fact, tracking naturally occurring features on the surface of the target yields equally accurate results, as visible in Figure 4.2.

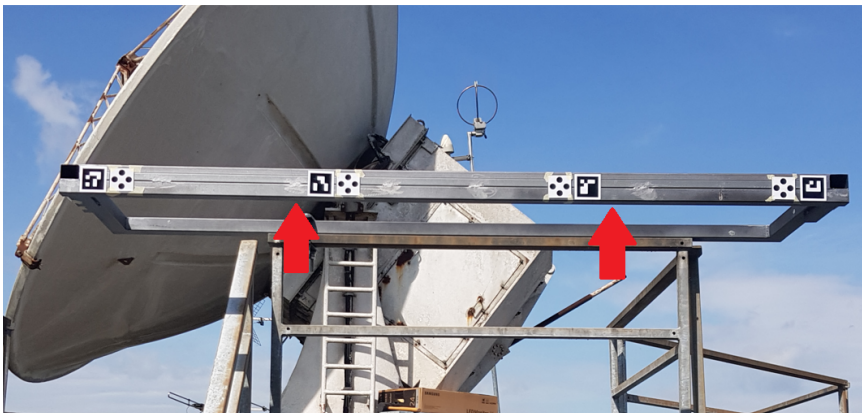

Figure 4.1: *The structure used for vibration tests. Spot markers are present alongside ArUco ones, but they were not exploited in these procedures. The arrows indicate where the accelerometers (not visible in this picture) were placed.*

Displacement measurement

The second step of motion tracking is estimating how the selected features move across frames. To make the measurement robust, the chosen feature is the mean position of the four corners. Tracking is performed through the KLT algorithm [106], which solves an optimization problem to estimate the displacement d of single pixels in consecutive pictures as optical flow. Optical flow is defined as the pattern of motion that objects perform in a scene caused by the relative movement between the camera and the scene itself. This method works reliably under the hypothesis that out-of-plane motion is negligible with respect to the mean distance between the camera and the target. Furthermore, if it is necessary to estimate the vibration amplitude in addition to the frequencies, the optical axis of the camera must also be orthogonal to the plane of motion of the target. This introduces limitations both on the amplitude and speed of the measurable motion with respect to the frame rate, which are not an issue in our specific case, where the amplitude is limited to a few pixels. Finally, a scale factor must be computed to express the amplitude of motion in physical units. It is necessary to rely on a known physical distance between two features to establish a relationship between lengths in pixels and in millimeters. To this end we exploit the corners of the markers, knowing their dimensions. For a fixed camera, this scale factor can be computed on the first frame and used on all of them, but the same is not true for flying cameras. Due to their non-null movement in the out-of-plane direction, the scale factor may change across time. Nevertheless the distance between the features is fixed, hence the scale factor can

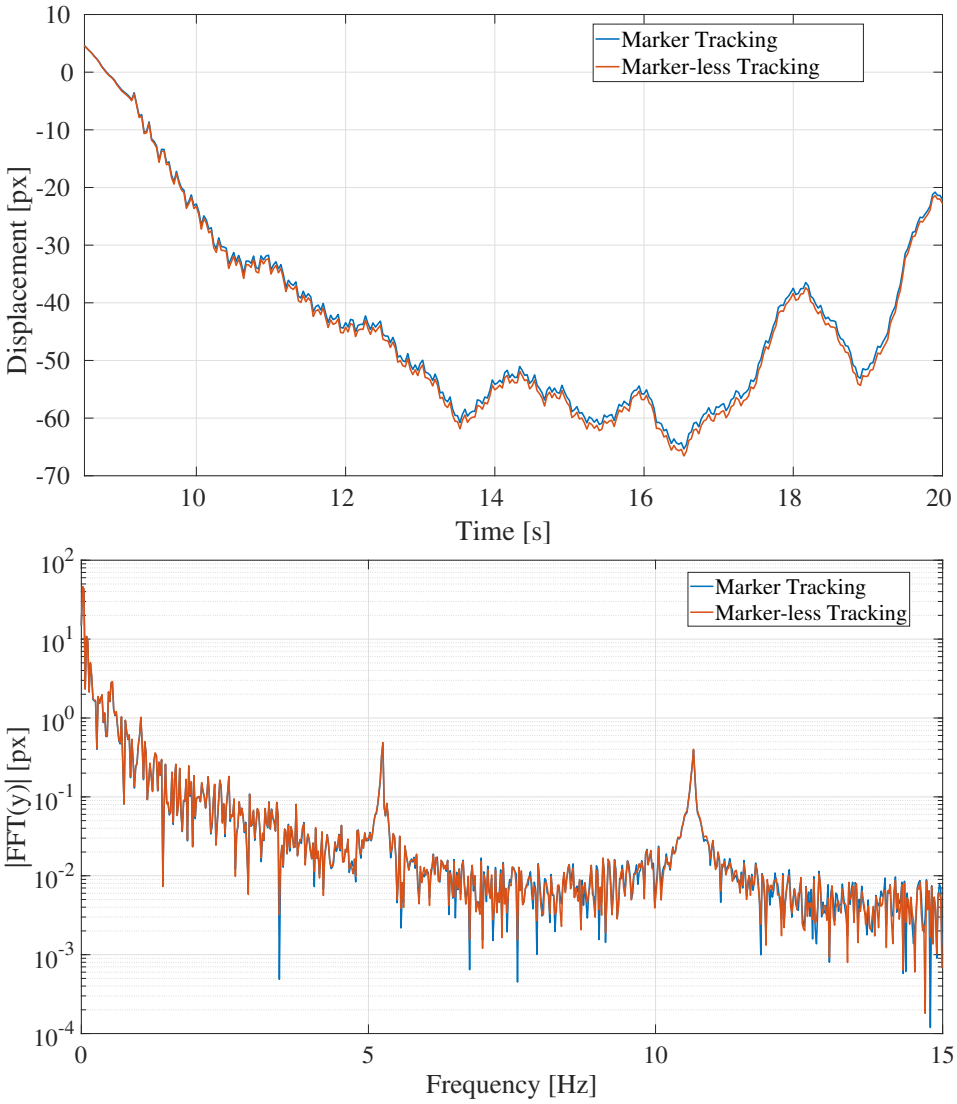

Figure 4.2: *Non-filtered time history and frequency spectrum of the averaged vertical displacement of features on a marker and naturally occurring on the structure itself, such as a junction. The plots overlap to the point of being almost indistinguishable. The time history is dominated by the relative motion between drone and structure, but the frequency components are well separated.*

be computed at every frame as:

$$s(t) = \frac{|\boldsymbol{P}_1 - \boldsymbol{P}_2|}{|\boldsymbol{p}_1(t) - \boldsymbol{p}_2(t)|}, \tag{4.2}$$

where t is the discrete-time index, $|\boldsymbol{P}_1 - \boldsymbol{P}_2|$ expresses the physical distance between two features, which is constant across time, and $|\boldsymbol{p}_1(t) - \boldsymbol{p}_2(t)|$ describes the same distance in pixels, in the image plane. The scale factor not only converts the camera plane displacement, measured in pixels, into an absolute displacement, measured for example in millimeters; it is also exploited to compensate the out of plane motion of the camera, since the physical distance between the features is constant across time. In the absence of markers, there are at least two ways to solve the scale problem:

- exploiting the known dimensions of some component of the structure, such as the size of a joint or bolt. This requires some previous knowledge of the structure or additional measurement to be carried out;
- calculating the physical distance between two features appearing in the image plane with trigonometry. This requires measuring the distance between the camera and the surface in real time during the recording, which is possible through LiDAR or ultrasonic sensors, for example. Such sensors are often present on UAVs for collision avoidance.

Estimating Natural Frequencies

We apply a Fast Furier Transform (FFT) to obtain the spectrum of frequency components of the signal. In the case of drone-mounted cameras, noise originated by the movement of the aircraft and the action of the motors must be accounted for. In commercially available drones, ours included, cameras are mounted on active gimbals, capable of compensating some of these disturbances. Furthermore, we verify experimentally that the noise does not contain frequency components that could corrupt the signal (see Figure 4.3). The effect of the rotating propellers is not visible because of its high frequency: according to the manufacturer's claims, the motor rotation frequency of small drones while hovering is higher than 80 Hz. On the other hand, the drone motion relative to the ground presents frequency components that are below the natural ones of the structure. As a consequence, we can apply a high-pass filter to separate the natural frequencies of the structure from the frequency components introduced by the motion of the camera. A reasonable choice in our case is to select a cutoff frequency of 2 Hz, as it is lower than the first natural frequency of the structure at hand, but higher than the components related to the camera movement. We exploit here a fundamental assumption, which will be recalled multiple times throughout the project, and is verified for the considered structure (see section 4.1.3): the natural frequencies of the target do not overlap with

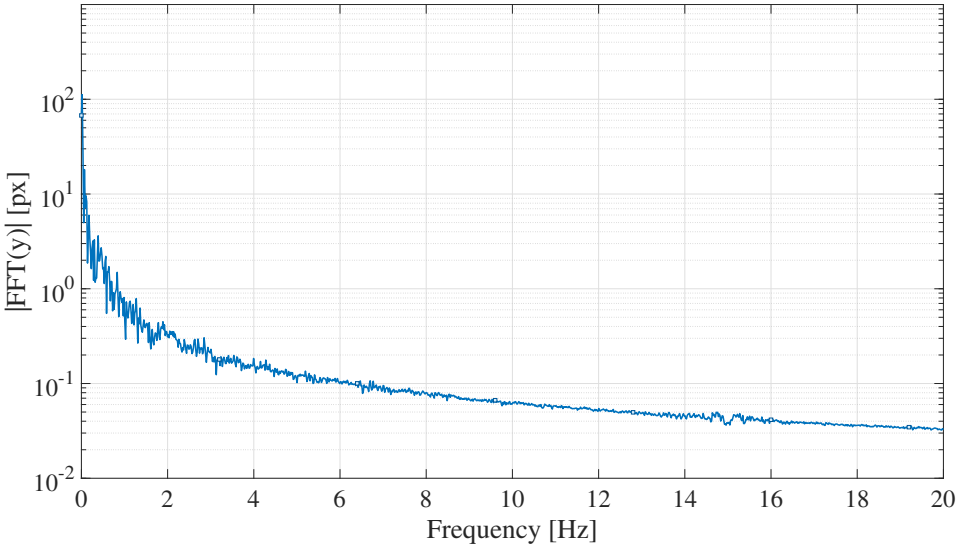

Figure 4.3: *Spectrum of the measurement of a fixed feature, representing the frequency components of measurement noise only. By comparison with Figure 4.4, we conclude that the peaks at* 5.06 *Hz and* 11.16 *Hz represent the oscillating behavior of the structure in response to excitation. This is also consistent with an estimated rotational speed of the propellers during hovering of* 5000 *revolutions per minute, equivalent to more than* 80 *Hz.*

the main spectral components of camera displacement. If this were not the case, the relative displacement can still be compensated, for example by accurately measuring the motion of the camera with respect to the still background and subtracting it from the one of the structure relative to the camera [130]. Figure 4.4 exemplifies the effect of filtering on the time history of relative displacement between the structure and the camera: the spectral components of camera displacement are canceled and the resulting signal is a time-history of the structure's vibration alone. Furthermore, from the obtained FFT, the natural frequencies ω_i of the target structure can be measured. Figure 4.3 displays the spectrum of the signal that is obtained if the structure is not excited, further confirming that the peaks visible in tests with excitation represent the behavior of the structure only.

Estimating Modal Shapes

In order to obtain the modal shapes, we need to know the pair-wise phase delays between features across the length of the structure, therefore, we take the signals two at a time and estimate the transfer function between them. Evaluating the phase of such transfer function at the natural frequencies

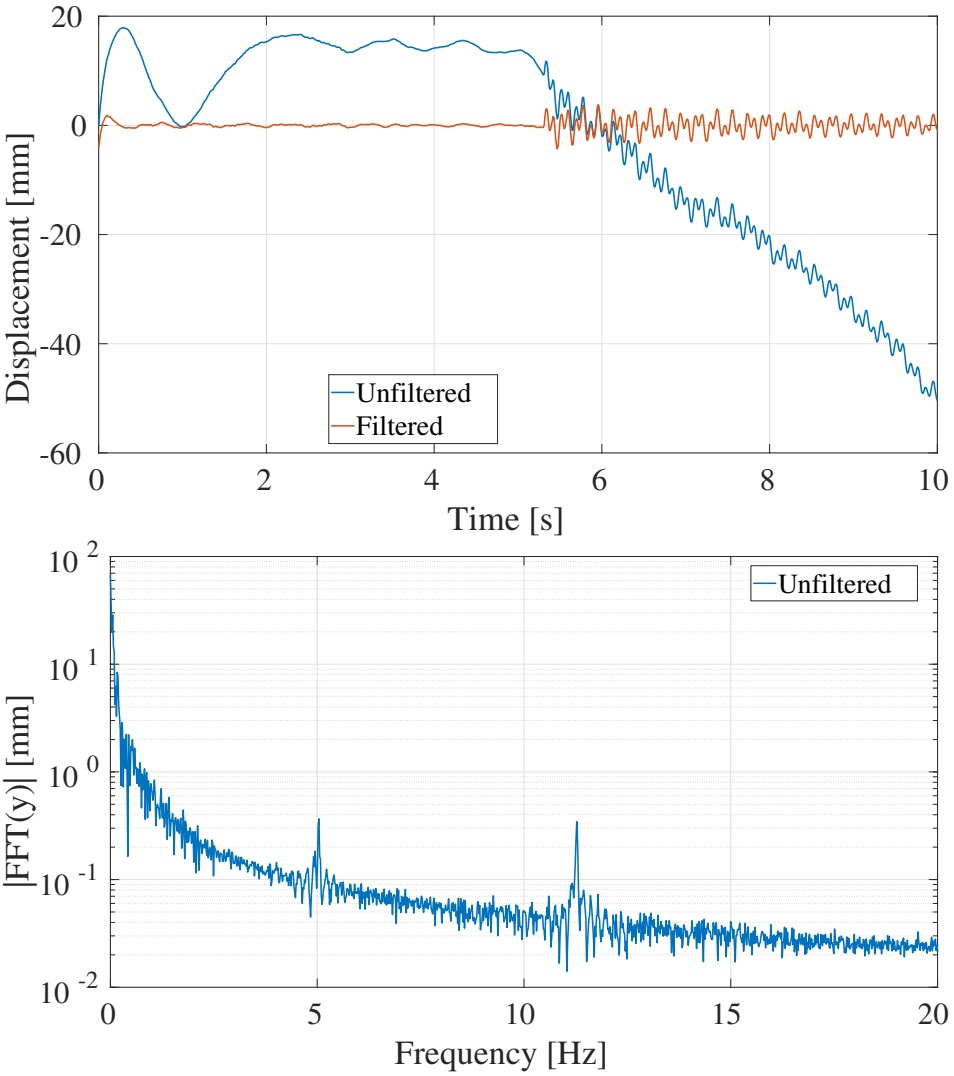

Figure 4.4: *Vertical displacement (top) of a feature as measured by a flying camera, in the first ten seconds of a test. The hammer impacts around the* 5.5 *s mark. Notice how the unfiltered measurement is dominated by the relative movement of the camera. In the frequency domain, these effects are distinguishable, as conveyed by the FFT (bottom). The first two natural frequencies of the structure (*5.06 *Hz and* 11.16 *Hz) are clearly visible.*

yields the phase delay between each feature pair:

$$\Delta\phi_{i,a-b} = \arg\left(G_{a-b}(j\omega_i)\right), \tag{4.3}$$

where j is the imaginary unit, $G_{a-b}(j\omega_i)$ is the value of the continuous-time transfer function between the signals describing the displacement of features a and b, evaluated at point $j\omega_i$.

To calculate pair-wise phase delays between features appearing in different video sources, we first synchronize them. This is achieved through cross-correlation of the high-pass filtered displacement measurements of a feature or marker appearing in both videos, in the overlapping region. This step is necessary for data collected with commercially available drones, for which hardware synchronization of the cameras is not possible.

We then estimate the transfer function between signals extracted from the two different but synchronized recordings. To better describe such a method, let us consider two cameras, without loss of generality. In a generic scenario, not only the initial time instant of the two recordings might be different, but also their frame rate. If however at least one feature is recorded by both cameras, we can artificially over-sample the two time-histories recorded at the same feature to the least common multiple of their frame rates, then apply a high-pass filter to isolate the oscillation of the structure from the relative motion of the cameras. Then, the cross-correlation between the two processed signals is calculated to estimate their relative delay in time, indicated with $\hat{\tau}$. We expect this delay to be always well defined, because the two signals correspond to the displacement of the same feature. Finally, a correction is applied to one of the two original signals to compensate the estimated time delay $\hat{\tau}$, to synchronize the two video sources. Then, the relative displacements among all features, including those not appearing in the overlapping regions of the two cameras, can be estimated. Thus, this method makes it possible to directly estimate phase delays also between features that are recorded by different cameras. Figure 4.5 summarizes the proposed procedure.

4.1.3 Experimental set-up

To determine the accuracy of the estimates, at least for portions of structures of limited size, we compare those obtained from flying cameras with those taken with accelerometers and with a fixed, high frame-rate camera. In order to test the capability of the proposed approach to provide information about possible alterations of the structure or faults, we then repeat and compare the measurements in different conditions. First, we identify the modal parameters in the reference situation, corresponding to an undamaged state. Then we identify the same parameters in modified scenarios obtained by gradually adding more weights to the structure. Finally we

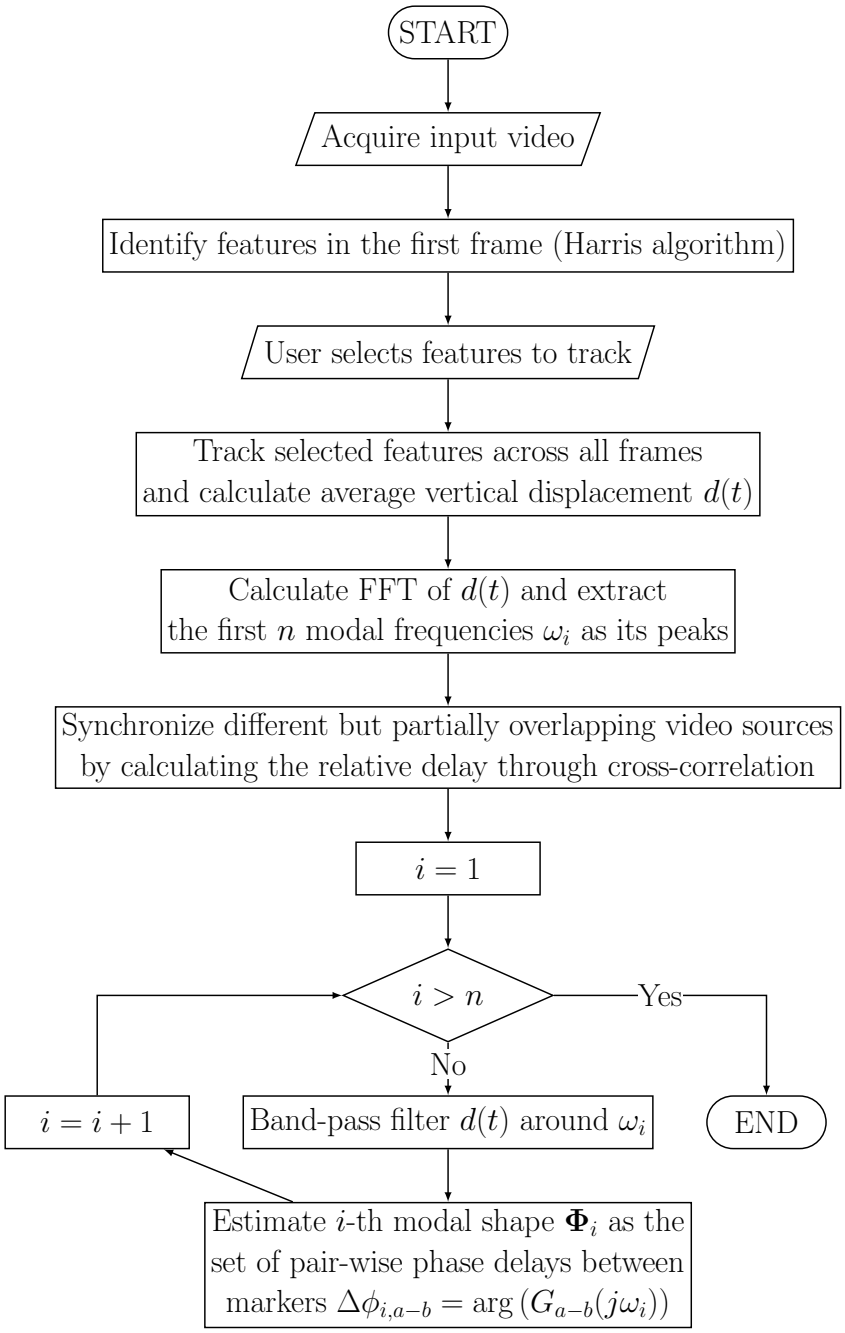

Figure 4.5: *Layout of the proposed methodology.*

simulate a faulty scenario, by restoring the initial weight and loosening one of the fixings.

Instrumentation

The accelerometers are PCB 333B30 monoaxial sensors with full scale range of ± 50g, acquired with NI 9234 board at 2048 Hz. The bandwidth of these sensors and the acquisition system is $1 - 700$ Hz, while the range of interest for this test is $1 - 100$ Hz. Motivated by their high accuracy, we take accelerometers acquisitions as ground truth to evaluate the results obtained with the cameras. Accelerometers were placed on top of the structure to measure the vertical displacement, as indicated by the red arrows in Figure 4.1.

The fixed camera is a FLIR Blackfly S BFS-U3-16S2M, a monochrome camera, with a resolution of 1440x1080 px and the framerate used for this experiment is 200 FPS. The camera was placed on a tripod at a distance of about 5 m to capture the entire structure. The focal length is 12 mm, to have a FOV large enough for the entire structure.

The first UAV is a DJI Mavic Pro equipped with a 12 MP camera, capable of shooting full HD video (1920×1080) at 96 FPS and HD video (1280×720) at 120 FPS. The second is a DJI Mavic 2 Enterprise Dual, whose camera captures 4K Ultra HD footage (3840×2160) at 30 FPS. Both drones are commercial, and they are equipped with a stabilizing gimbal for the camera. This device is controlled by the on board computer and actively compensates part of the disturbance induced by the aircraft motion, by decoupling its attitude with respect to that of the camera.

The resolutions of the cameras are different, which influences the minimum structural displacement they can record. To ensure that every camera can capture the oscillation of the target in every test, they are at a suitable distance from the structure. The resulting scale factors, for all cameras and all tests, were within the $0.5 - 1.8\ 10^{-3} \frac{m}{px}$ (millimeters per pixel) range, which is enough to capture oscillations with amplitude of a few millimeters (see Figure 4.4). By the Nyquist-Shannon sampling theorem, frame-rates also introduce an upper limit of $\frac{FPS}{2}$ on the measurable frequencies. In our case, the lowest frame-rate is 30 FPS, corresponding to the Mavic 2 drone, therefore we limit our analysis to natural frequencies below 15 Hz, which pertain to the first two modes.

Test structure and procedure

The structure for our experiments has two main components: a railing, welded on a metal base fixed to the ground, and a cantilever section on top (see Figure 4.1). The two parts are made of steel and attached with two clamps. Both are made of hollow tubes with square cross-section. The cantilever was designed to have suitable natural frequencies. In particular, lower bounds on the frequency values were introduced by the frequency content of drone movement, in the $0-1$ Hz range. The Nyquist rates of the two flying cameras, on the other hand, posed an upper bound. As a consequence, the cantilever was designed to have frequencies within the $3-12$ Hz bandwidth. The cantilever was welded into a rectangular, 2000×550 mm shape with a mass of 20.5 kg. A finite element model of the structure was developed to simulate its behavior and determine the values of the natural frequencies for several different mass distributions.

Four square, 55×55 mm markers are attached on the cantilever, two near the corners and two near the center, equally spaced along the structure (see Figure 4.1). All the tests are performed with the same procedure: an impulse is generated with a rubber hammer on the top to excite all frequencies. The impact point is at the center of the protruding end of the cantilever, and the hammer impacts the structure vertically. The fixed camera was placed 5 m away from the structure, facing it directly, whereas drones were hovered at approximately 1 to 2 m from the cantilever, to compensate for their wider field of view.

A total of 13 tests were carried out in the span of two hours, with consistent lighting conditions. The duration of each test is on average of 30 s, which is enough to include the whole response of the structure, up to the moment when it stops vibrating. Consistency was kept by keeping the static camera and the accelerometers fixed between tests and by manually piloting the drones during the tests so as to keep them as stationary as possible.

4.1.4 Experimental Results

Measuring Modal Parameters

The first objective of the procedure is to accurately measure the natural frequencies of the structure, and to evaluate the accuracy obtained with the drones as compared with the accelerometers, which we take as ground truth. Hence, a test was designed where the accelerometers were fixed onto the structure as shown in Figure 4.1, the fixed camera was placed in front of the target at a distance of 5 m and the DJI Mavic Pro was flown at a 1.5 m distance, hovering in place. The drone was positioned slightly above the

Mode	Accel.	Fixed Camera (err.)	Flying Camera (err.)
$i = 1$	5.06 Hz	5.01 Hz (1.0%)	5.05 Hz (0.2%)
$i = 2$	11.16 Hz	11.14 Hz (0.2%)	11.17 Hz (0.1%)

Table 4.1: *Estimated natural frequencies*

structure in order not to obstruct the visual of the fixed camera. The comparison of the FFTs of the accelerometers, fixed camera and flying camera in the relevant bandwidth is reported in Figures 4.6 (fixed and flying cameras) and 4.7 (accelerometers) and in Table 4.1. Error is calculated as relative difference with respect to the accelerometer value:

$$e_i = 100 \frac{|\omega_{i,accelerometer} - \omega_{i,camera}|}{\omega_{i,accelerometer}}. \tag{4.4}$$

The two measured modes are within the bounds specified in the design phase; in particular, their values are sufficiently close to those obtained in finite-element simulations: 4.88 Hz and 10.66 Hz. Furthermore, the error with respect to the accelerometers is at most 0.2% for the flying camera, and slightly higher for the fixed one, notwithstanding its higher frame rate and the absence of motion.

It is possible to extract modal shapes from the data, comparing the movement of the four different markers and their relative pairwise phase delays $\Delta\phi_{i,a-b}$. The phase delay provided by the two accelerometers placed on the protruding end of the cantilever is approximately zero for both investigated frequencies (see Figure 4.7).

The pair-wise transfer functions and phase delays between the four measurements recorded by the DJI Mavic Pro were estimated as described in Section 4.1.2. Table 4.2 details the calculated delays, where each column represents the estimated phase delay between a marker and the next one. The phase delays are all very close to zero, which indicates, in accordance with the information provided by the accelerometers, that both modes correspond to a vertical rigid displacement of the front side of the cantilever. This behavior is again compatible with the one observed in the finite-element simulations. The two are likely due to the railing oscillating with respect to the ground and the cantilever vibrating with respect to the railing itself. The cantilever does not exhibit bending modes at such low frequencies.

We verified experimentally that measurements taken by tracking naturally occurring features on the surface of the structure are as accurate as those obtained by exploiting markers. In order to do so, we selected as

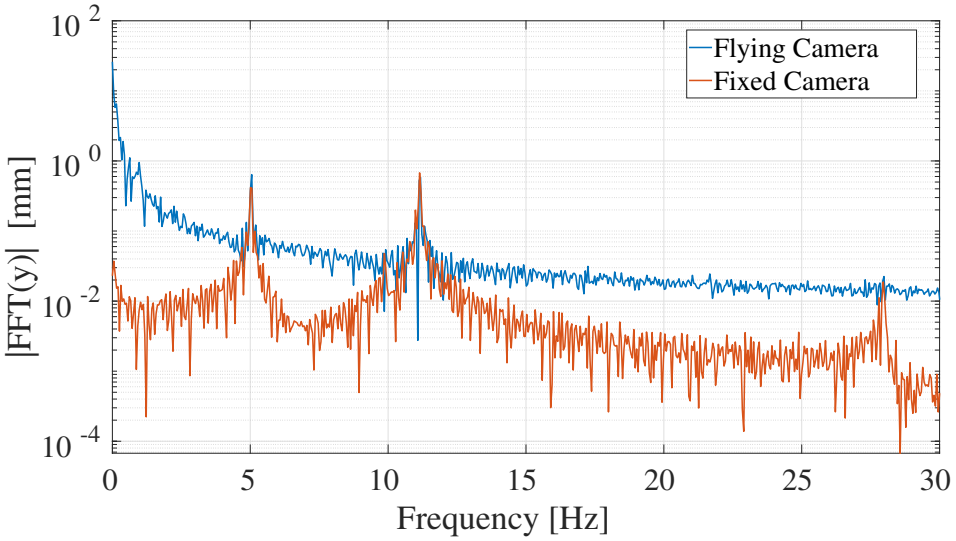

Figure 4.6: *FFT of the displacement measurement, comparing flying camera and fixed camera. The main difference between the sensors is not the accuracy of the frequency peaks, but the baseline noise. This is the reason why the third natural frequency is not identifiable with the flying camera.*

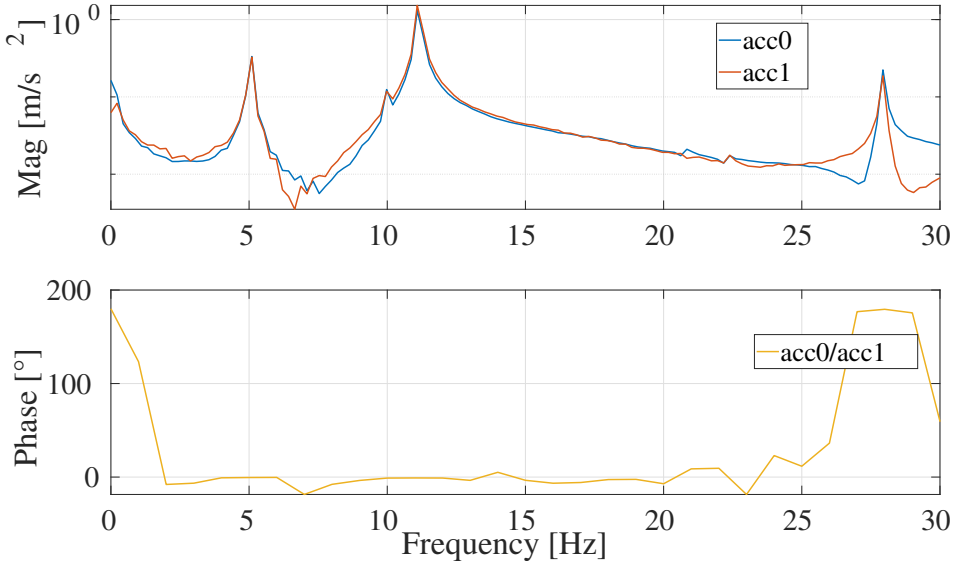

Figure 4.7: *The magnitude of the FFT of the accelerometers placed to measure the vertical displacement (top) and their relative phase delay (bottom), obtained from the transfer function estimated with Welch's averaged periodogram method.*

Mode	ω_i	$\Delta\phi_{i,1-2}$	$\Delta\phi_{i,1-3}$	$\Delta\phi_{i,1-4}$
$i = 1$	5.05 Hz	$-1.10°$	$0.61°$	$-0.24°$
$i = 2$	11.17 Hz	$0.75°$	$-0.22°$	$0.56°$

Table 4.2: *Phase delays estimated with a single camera*

features the edges of the soldering points (see Figure 4.8 for an example). These results are summarized in Figure 4.2, which shows that the measured displacement and its frequency spectrum are nearly identical in the two cases.

Figure 4.8: *Comparison between using markers and features that are inherent in the structure. Triangles are the corners of the marker, while squares belong to a soldering spot on the structure.*

Synchronization of Two Drones

We next investigate the acquisition of the use of data collected by two drones, each one with a partial view of the whole structure and with an overlapping region with respect to the other aerial vehicle. For this reason a second test was carried out, where only the two UAVs described in Section 4.1.3 were used to record the displacement. The structure was still excited with a rubber hammer and the drones were hovering in front of the structure at a 1 m distance (see Figure 4.9). The first aircraft recorded at 96 FPS, the second at 30 FPS. After calculating and correcting the delay between

Figure 4.9: *Setup of the second test. Drones are positioned such that each one is unable to capture the whole structure.*

ω_i	$\Delta\phi_{i,1-2}$ (err.)	$\Delta\phi_{i,1-3}$ (err.)	$\Delta\phi_{i,1-4}$ (err.)
5.05 Hz	$-1.17°$ $(0.07°)$	$-1.87°$ $(2.48°)$	$-2.73°$ $(2.49°)$
11.17 Hz	$-3.02°$ $(3.77°)$	$-2.21°$ $(1.99°)$	$-2.23°$ $(2.79°)$

Table 4.3: *Phase delays estimated with two drone-mounted cameras and synchronization*

the two videos, through oversampling and cross-correlation of the displacement of the overlapping marker (as described in Section 4.1.2), the pairwise relative phase delays between markers can be computed. Synchronized displacements were treated as if they were recorded by the same camera, and yielded the results shown in Table 4.3. The error is calculated as the absolute value of the difference with respect to the phase delays estimated with a single camera. Phase delays between non-consecutive markers can be computed as

$$\Delta\phi_{i,a-c} = \Delta\phi_{i,a-b} + \Delta\phi_{i,b-c}, \tag{4.5}$$

and they are comparable to the ones obtained with a single flying camera.

Detection of Overload and Tampering

In order to ascertain the ability of this procedure to detect variations in modal parameters, a third series of tests was carried out. The same excitation technique was adopted, while recording and measuring with a single

drone, as in the first scenario. Three tests were carried out after gradually adding masses to the protruding end of the cantilever. In particular, we added three 4.0 kg masses, one at a time, and repeated the measurement. While such a significant modification of the modal parameters may be unrealistic, we show our procedure still correctly identifies the change and the measurement of the parameters is commensurate with that obtained with accelerometers.

Another test was conducted after removing the additional masses and loosening one of the clamps that held together the structure. Results were compared to the base case to find out if the measured modal parameters differ significantly, and they are collected in Table 4.4. Figure 4.10 shows

Mode	ω_i	$\Delta\phi_{i,1-2}$	$\Delta\phi_{i,2-3}$	$\Delta\phi_{i,3-4}$
$i=1$	4.96 Hz	$3.51°$	$0.50°$	$-2.75°$
$i=2$	8.19 Hz	$4.32°$	$0.81°$	$-4.95°$

Table 4.4: *Estimated phase delays after tampering*

the effect of gradually adding mass to the structure. As expected, the natural frequencies decrease, since they are, in general, proportional to the square root of the ratio between the modal stiffness and mass. It should also be noted that while the two frequencies decrease, they maintain the same ratio. No significant differences emerge in the modal shape analysis: the main oscillations are still related to vertical rigid displacement of the protruding end of the cantilever and no bending takes place.

The effect of a tampered clamp, on the other hand, is noticeably different (see Figure 4.10). Here the natural frequencies are still impacted, but the second one drops to 8.19 Hz, while the first is 4.96 Hz, thus the ratio is not maintained. This decrease in frequency is likely due to an overall reduction of the stiffness of the structure. The modal shape is still the same, though some phase delays increase with respect to the base scenario, indicating that bending of the cantilever beam is probably starting to happen due to the loosened clamp, with a modal shape that is not symmetric anymore.

4.1.5 Conclusion, Limitations and Future Developments

We described and tested a method to conduct vision-based modal analysis of small structures or portions of structures through the use of measurements acquired simultaneously by multiple airborne cameras. We measured the natural frequencies and calculated the modes of vibration of a metal

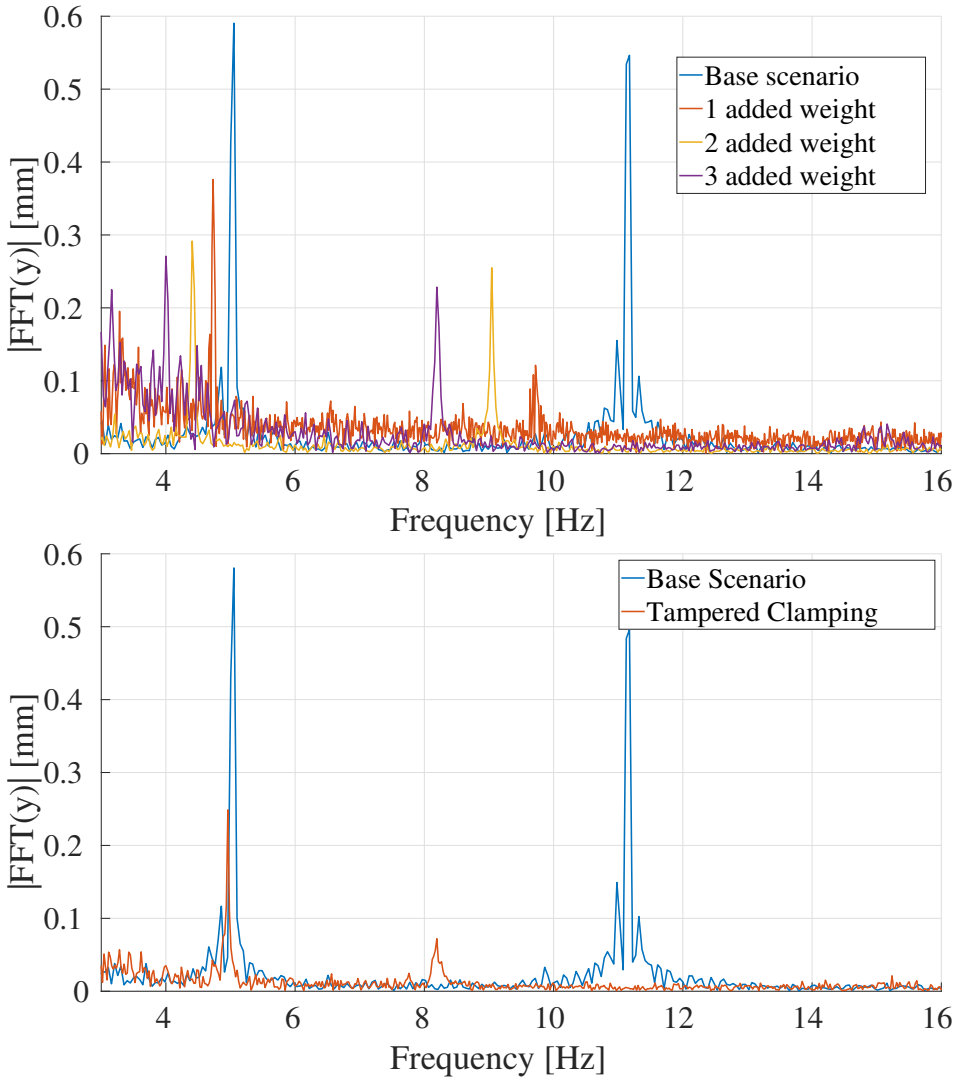

Figure 4.10: *Effect of increasing mass (top) and of loosening a clamp (bottom) on natural frequencies.*

structure and showed that accurate displacement measurements can be carried out based on features occurring on the surface of the target structure, without markers. We also tested the ability of such method to detect variations in modal parameters. The results are encouraging, despite the limited bandwidth due to rather low FPS cameras, with respect to accelerometers. We deem this solution feasible for structures of limited size and convenient when accelerometers are not already present and their installation is diffi-

cult due to the structure being precarious or hard to reach. Even though the method, in its current state, is not yet suited to large structures which would need a large number of UAVs making the solution complex, we think that it can be developed further to extend both its range of applicability and the accuracy of the estimates. For example, the range of frequencies that it is possible to study can be extended towards the lower end of the spectrum, where camera movement noise is significant, by devising a more complex way of filtering ego-motion, such as measuring camera displacement with respect to a still background or estimating it based on data collected from the drone's IMU. Furthermore, higher autonomy can be reached both in feature selection, through specifically designed algorithms, and in aircraft deployment, by employing autonomous UAVs. The technology for achieving this is already available while the regulatory framework is not sufficiently developed yet. A technical limit to further scaling is the relationship between camera resolution, field of view and oscillation amplitude, which however is favorable for slender structures and cables. Motion blur due to the camera hovering can in principle impact measurement accuracy, but we verified that its contribution in our tests is negligible, as displacements with amplitude below 1 px are identified. Finally, in case the surface of the target structure does not offer a sufficient number of features, it is possible to spray-paint suitable patterns from a distance, also exploiting drones [81]. As long as environmental conditions are favorable (limited wind, suitable lighting and absence of reflections), the procedure can be applied as it stands to some real world tasks, including modal analysis of small, flexible structures such as barriers, antennas, parapets and small metallic bridges. This method can be suitable also for vision-based estimation of tension force in cables, such as those in stadiums, cable-ways, power transmission systems and cable-stayed bridges. There is ongoing research within our group in this field with positive results.

CHAPTER 5

Energy Efficiency Assessment

5.1 Building digitalization, diagnostics and energy efficiency assessment

State of the art

Another area of application of robotics and artificial intelligence is building energy efficiency evaluation. Starting from the need for general rules that guarantee good results in terms of safety and precision, some relevant works can be considered as good protocols for the general activity [135–137]. A vehicle with a camera, RGB or infrared, and remotely-controlled by a licensed operator, can monitor the state of a civil construction, focusing the attention on potential defects without risks and using only non-destructive procedures. These operations are normally executed manually by a technician and all the considerations are obtained based on visual inspections, that considers the presence of a potential problem in a building after the failure has already occurred. Such a delayed and risky operation leads to rather high maintenance costs. In [138] the drone technology is presented as the potential new standard in asset inspections, based on the results obtained in the oil and gas industry, where the time required for in-

spection operations were reduced from 8 weeks to 5 days on average. In fact, an estimated cost saving using drone-based assets was predicted in a range between 50% and 90% and the monitoring is going to be extended to other sectors [138].

The aim of this project is to further the development of fully autonomous drones for inspection by selecting, integrating and testing existing techniques to cover the inspection pipeline from data collection to diagnosis.

Path planning The first aspect analyzed pertains to the best path for the drone to collect a reliable picture data set. The flight path shall consider the specs of the drone, in particular the resolution of the camera installed and other limits of the device such as the focal length and the type of lens. For a complete data acquisition, a flight path able to cover all the building is required. Despite many path planning algorithms being present in literature [139], the identification of the best one still depends on the specifics of the tasks to a great extent. On the market, approaches are limited at cases where the object is contained in a plane. However, buildings and infrastructure are 3D objects and can rarely be approximated as 2D. In any case, some good rules can be extracted in order to perform a good flight and obtain a good result. Considering each facade of a building as a separate 2D surface, the most common and efficient approach is the strip-method, where the drone is conducted by the pilot in a zig-zag movement, either vertical or horizontal [136]. The former is suited for collecting information on tall buildings buildings, the latter on wide ones. It is also shown that a vertical flight tends to reduce the clarity and the quality of the images due to the unfavorable movement of the lens, while the strip horizontal method is most effective when the speed of the drone is low. Another type of flight path is the spiral movement, which is particularly indicated for preliminary recognition and 3D model reconstruction. The attitude of the thermal camera during flight operations is another important aspect to be taken into account because, since all the system works analysing irradiation emissions of the objects, this output varies by changing the position of the drone [137]. For thermal images, it is thus important to consider the relative angle between the drone and the building. In fact, by an analysis of a photo-voltaic field performed by positioning the drone in different places, different results up to 30% have been reported in [140]. Finally, adapting the operations to various environment conditions is important for safety and repeatability. Wind speed and optimal lighting have to be considered: the first aspect can affect the motion of the drone, the second aspect can affect the quality of the collected images. In addition, if the flight path is realized

with the purpose of performing a 3D image-based reconstruction, the trajectory to be considered is strongly affected by the triangulation matching between the points of each picture collected [141].

3D and Thermal Modelling Computer-aided 3D reconstruction is mainly diffused in medical and archaeological fields. In the latter domain, the application of these techniques provides various benefits, such as limiting the destructive nature of excavating to the possibility of enriching the archaeological research [142]. 3D reconstruction can be performed with different information sources, where images offer the advantages of carrying more information than LiDAR scans (in the form of color) and can be obtained with cameras, which are cheaper. This comes at the cost of somewhat reduced geometrical accuracy [143]. In our application, the 3D model was obtained through the application of SfM: a photogrammetry technique that reconstructs an object into 3D environment starting from 2D pictures through triangulation of features [144].

To calculate and extract the features required by the SfM algorithm, the Scale-Invariant Feature Transform (SIFT) [145] is employed. The attention in our work is focused on obtaining a complete 3D preview of the site of interest, in order to subsequently analyze, qualitatively and quantitatively, the thermal behavior of the surfaces involved. However, differently from what is done with RGB pictures, it is not possible to obtain an accurate 3D output by directly matching thermal (infrared) 2D images due to their low resolution (commonly around 640x480 pixels against 4056x3040 pixels of a normal RGB photo). Thus, another solution that links the two types of information is necessary. Two techniques available in the literature can accomplish this goal: the bi-camera method and a photo-texturing based on a scaling factor. The first consists in process images that are collected simultaneously by two connected cameras, one acquiring RGB and the other one thermal images. The calibration of the system is possible because their relative poses are known [146]. By considering the rotation matrix R_{RGB}, describing the rotation between the RGB sensor and the center of the camera pair, the relative rotation matrix of the Infrared camera can be found out by considering a geometrical transformation based on the system stability condition. Another constraint is the center of perspective of the camera. Since the system is composed by two cameras simultaneously acting, it is not possible to consider as center the one of two of them. However, by considering the fixed position, the translation vector t is assumed constant along all the transformation. This approach has been tested by some of the authors on the Trifoglio building of Politecnico di Milano [146]. The

geometric model was automatically generated by a software working with Terrestrial Laser Scanning and the bi-camera approach was used for the thermal model rebuilding. At the end of the test, the result was considered not acceptable due to the lack of information that was not captured by the cameras. In addition, it is important to remark that all the tests were performed considering an approach that was suitable for ground data gathering. The hypotheses made about the stability of the transformation matrix may be not met in full when the system is adopted for a drone acquisition.

The second technique exploits the a-priori known position of some fixed points, called baselines. Those points are captured by the RGB images and reported also in the Infra-Red (IR) pictures. After the generation of the 3D mesh with the RGB images, the IR photos are used to create a texture covering it and properly scaled by matching the position of the baselines. In this approach, the low resolution of IR images is not an issue. On the other hand, the position of the baselines still requires some a-priori knowledge and (limited) user intervention.

Scope

The main goal of research presented in this chapter has been to develop and test a procedure for collecting RGB and thermal pictures of a building with drone-mounted cameras, to obtain a 3D model of such building comprising both the visual appearance and the heat flow information regarding it. This required the integration of techniques from different disciplines. Besides validating experimentally the overall toolchain and obtaining significant insight on the thermal behavior of the building at hand, one further goal was to assess the requirements and feasibility of the full automation of the inspection process. The procedure is divided into four stages:

1. flight paths optimization to collect the most informative data, considering the characteristics of the deployed UAV and the camera installed;
2. 3D reconstruction based on the collected 2D pictures, through SfM. Thermal information is also integrated and properly associated to the relevant sections of the building, resulting in a point-cloud model;
3. identification of facade elements, in particular distinguishing opaque and transparent parts, in order to derive suitable thermal balance equations;
4. comparison of the measured energy-efficiency related values with the ones obtained from a heat balance model describing the climatic and

geometrical parameters of the inspected building, in order to detect issues and generally assess the validity of such models against measured data.

The procedure has been tested on an office and laboratory building based in Spino d'Adda (Italy) used by Politecnico di Milano and the Italian Space Agency as research facility.

5.1.1 Data collection and elaboration pipeline

Flight path planning

Path planning should take into account several factors, including the geometry of the building and national, regional and local regulations (see [147]). These characteristics are included in the planning of the flight path used to acquire all the pictures. The presence of obstacles and prohibited areas must also be accounted for. For example, in our experimental application a no-fly zone is set in the south of the area where other activities take place.

The main objective of the flight plan is collecting suitable color pictures to reconstruct an accurate 3D model using photogrammetry. For this reason, the distance between camera and object, the distance between points where pictures are taken and the intrinsic parameters of the camera have to be addressed. We relied on Ground Sampling Distance (G, measured in $[\frac{cm}{px}]$), calculated as:

$$G = \frac{W_s H}{F_R W_{im}} 100, \tag{5.1}$$

where W_s is the width of the sensor, H is the object-camera distance, F_r is the focal length of the camera and W_{im} is the image width.

Overlap in the images is warranted at the same points to be triangulated correctly and the 3D reconstruction to be complete. For Structure from Motion, images should ideally overlap by 65% to 85%; lower values lead to shorter flights, but they also negatively impact the resolution of the model and introduce holes caused missing triangulations. On the other hand, high values of overlapping ensure redundancy and a reliable 3D reconstruction at the cost of a longer flight, potentially incurring limits posed by the battery capacity.

A higher number of pictures is warranted around the corners between different facades, so that the edge is captured accurately. We therefore implemented a procedure which, starting from a rough 3D model of the building to inspect, establishes a finite number of candidate points in space, from which a picture should be taken, by:

1. calculating the ideal object-camera distance H for planar surfaces, based on an arbitrary choice of the Ground Sampling Distance G;

2. finding a suitable number of candidate points for each facade of the building, given a certain value for the desired overlap between pictures;

3. adding further points around the corners connecting the facades, at a distance of H_r.

A path is then obtained by connecting the points by the strip method [136]. Alternatively, one can exploit graph theory to minimize flight time, see for example [21].

3D model reconstruction

Once the mission has been carried out, a set of 2D picture pairs, color and infrared (RGB and IR), is available to build the 3D representation. At this point it is wise to calculate some measure of their quality, such as resolution and sharpness, and discard the worst ones, in order to maximize the accuracy of the reconstruction. SfM is then applied to the set of color images, whereby feature detection, matching and triangulation yields a 3D point cloud representing the inspected building. Such technique also identifies the relative positions from which the pictures were taken. The resulting model, though, only estimates the relative distances between features, not the correct scale factor. To complete the modeling phase, the scale factor is estimated based on the GPS tags that characterize each photo, localizing the position in space they were taken from. Should the accuracy of the GPS localization be insufficient for the task at hand, it is also possible to identify a set features that appear in several images (both RGB and IR) and whose real distance is known, and exploit those to assign a proper scale factor to the resulting 3D model. Regardless of the employed mechanism, estimating the scale factor endows the model with a one to one correspondence with the modeled building.

Subsequently, as anticipated the correspondence between features in the color and thermal images is exploited to establish a relationship between the appearance of each feature in the RGB picture and its estimated temperature in the IR one. The end result is a properly scaled point cloud, where each point carries at once geometrical, color and thermal information. It can therefore be represented in a matrix $B \in \mathbb{R}^{n\times 7}$, where n is the number of identified points. Three values store the position of the point,

three values store its color and the last one carries its measured temperature. This last value can be obtained from the mono-dimensional thermal image by interpolation between the colors representing the maximum and minimum temperatures, which are set by the user.

Facade modelling

To determine the presence of defects and failures in the thermal properties of the envelope, we consider one facade at a time. To this end, the point cloud is segmented spatially in order to separate the different facades. Each of them, in fact, has a different thermal behavior depending on its dimensions and materials. In order to perform the segmentation, basic clustering solutions such as k-means fail [48] due to the structured nature of the point cloud, while geometric fitting methods work well. In this case M-Estimator Sample Consensus (MSAC) is applied, which is an upgrade of RANdom Sample Consensus (RANSAC), proposed by Fischler and Bolles [148,149]. It is a general parameter estimation approach designed to make a distinction of the data based on some fixed characteristics. MSAC can be considered as an iterative re-sampling technique that generates candidate solutions by using the minimum number observations (data points) required to estimate the underlying model parameters. Compared to conventional sampling techniques that use as much of the data as possible to obtain an initial solution and then proceed the research of the outliers, MSAC, like RANSAC, uses the smallest set possible and proceeds to enlarge this set with consistent data points.

After the first segmentation step it is possible to reduce the complexity of the model by discarding points that are marked as outliers by MSAC. It is also possible to further sub-divide each facade into its composing elements, such as walls and windows, by applying the same algorithm with properly scaled parameters (see Figure 5.6 in our experimental results). Additionally, the number of points can be greatly reduced by tiling each facade with square tiles of arbitrary side length l and substituting all of the points whose normal projection is inside the perimeter of the tile with a single point obtained as their average. This step is often necessary for computational reasons when working with SfM, as the algorithm tends to find numerous features, leading to a very dense point cloud. A suitable value of l should thus be chosen as a trade-off between accuracy of the resulting model and computational advantage.

Expected temperature

We compared the measurement of the thermal behavior of the building with the expected one, in order to detect issues as discrepancies between the two. Ideally, a detailed digital twin of the building should be used as a model, but this is often unavailable for old buildings, which represent the major part of the built environment.. Still, it is possible to model the exchange of heat between a facade and the surrounding environment through heat flow equations. At steady state, the following flow balance holds:

$$q_{SWR} + q_{LWR} + q_{conv} + q_{cond} = 0, \tag{5.2}$$

where q_{SWR} and q_{LWR} represent the heat exchanged through radiation, pertaining respectively to short and long wave radiation, q_{conv} is the quantity of energy exchanged by convection between the facade and the air, q_{cond} describes heat exchanged through conduction between the internal and external sides of the wall (see Figure 5.1).

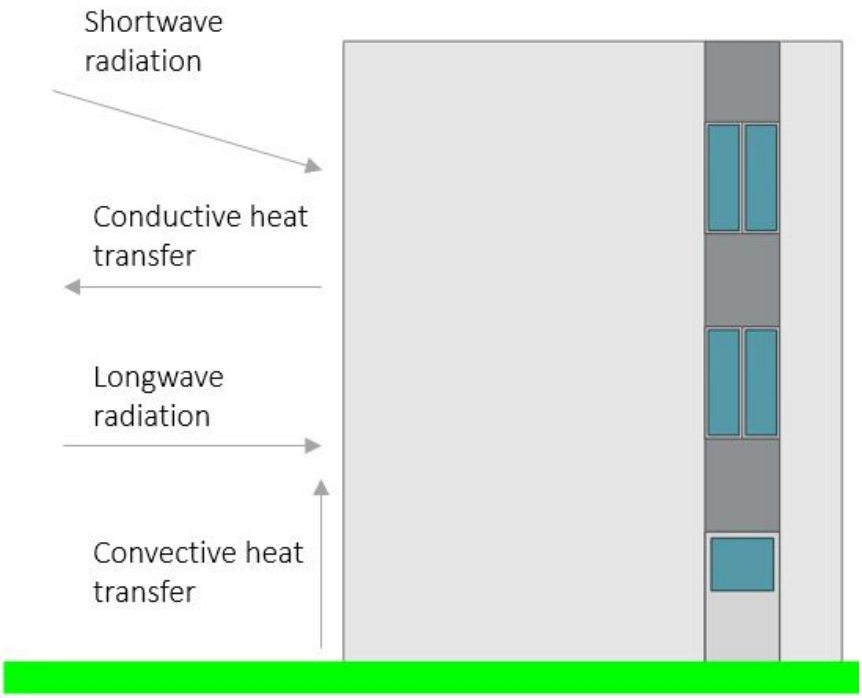

Figure 5.1: *Thermal balance*

Thanks to the geometry of the building at hand, some assumptions can be introduced for long-wave radiation exchange calculation, namely that

facades exchange heat with the ground, the air and the sky, whereas the roof does not exchange with the ground. In general:

$$q_{LWR} = q_{ground} + q_{sky} + q_{air} \tag{5.3}$$

$$q_{ground} = \frac{\epsilon\sigma F_{gnd}(T_{es}^4 - T_{air}^4)}{T_{es} + T_{air}}(T_{air} + T_{es}) \tag{5.4}$$

$$q_{sky} = \frac{\epsilon\sigma F_{sky} B(T_{es}^4 - T_{sky}^4)}{T_{es} + T_{sky}}(T_{sky} + T_{es}) \tag{5.5}$$

$$q_{air} = \frac{\epsilon\sigma F_{sky}(1 - B)(T_{es}^4 - T_{air}^4)}{T_{es} + T_{air}}(T_{air} + T_{es}), \tag{5.6}$$

were $q_{ground}, q_{air}, q_{sky}$ denote the thermal flows to the ground, sky (radiation) and air (conduction) respectively. T_{es}, T_{air}, T_{sky} describe the temperatures of the external surface of the building, the surrounding air and the sky. Finally, σ is the Stefan-Boltzmann constant and ϵ is the view factor. Such factor accounts for the angle between the building surface and the other heat-exchanging element, being maximum when the two are facing each other in parallel with no obstacles in between. Air temperature is considered constant along the height of the building. 5.3 describes a heat balance, stating that the low wave radiation component comprises three heat flow contributions: heat from the ground, from the sky and from surrounding air. Equations 5.4 through 5.6 specify how these are calculated.

The presence of shortwave radiation is one of the most influential factors for a thermal balance. For this reason, since the external temperature of a surface under direct irradiation is subject to important variations in the time, it is necessary to plan the acquisition phase in the part of the day when the sun radiation is minimal. In addition, the radiation coming from the sun is influenced by many aspects such as the meteorological conditions, the presence of obstacles and the pollution level. At a design stage, values according to predetermined models can be used to obtain a daily average value.

Convection is calculated as:

$$q_{conv} = h(T_{es} - T_{air}). \tag{5.7}$$

Simple Combined Correlation is adopted to calculate the convective coefficient h, as specified in ISO6946, based on air velocity V_z and the characteristics of the external layer:

$$h = D + EV_z + FV_z^2. \tag{5.8}$$

Surface Type	D	E	F
External plaster	10.22	3.1	0.0
Impermeabilization	10.79	4.192	0.0

Table 5.1: *Standard values for convection coefficient estimation*

W_s	Sensor width	28 mm
H	Object-camera distance	5 m
F_r	Camera focal distance	24 mm
H_{im}(RGB)	RGB image height	3040 pixels
W_{im}(RGB)	RGB image width	4056 pixels
H_{im}(IR)	IR image height	480 pixels
W_{im}(IR)	IR image width	640 pixels

Table 5.2: *Camera Parameters*

Typical values for parameters D, E and F, employed in this study, are specified in Table 5.1. Should a local measurement of V_z not be available, the value recorded by the closest meteorological station can be adopted as an estimate.

For calculating conduction heat exchange, internal temperature is assumed uniform and constant. In general, the heat transfer through conduction is calculated as:

$$q_{cond} = U(T_{int} - T_{ext}) \tag{5.9}$$

Where thermal transmittance U, calculated as the inverse sum of thermal impedances of the strata making up the wall:

$$U = \frac{1}{R} = \frac{1}{\sum R_i} \tag{5.10}$$

Equations 5.2-5.10 can finally be evaluated to obtain the value of the temperature of the external surface, T_{es}, to be compared with the measured one for each point in the 3D cloud.

5.1.2 Experimental Results

Data acquisition

An upper limit on ground sampling distance of $\bar{G} = 1$ cm/px was set, together with an overlap of 80%. Given the parameters of the cameras described in Table 5.2, mounted on a DJI Mavic 2 Enterprise Dual, the resulting distances are described in Table 5.3. We elected to take pictures around corners ad a distance $H_r = 6$ m, every $22.5°$. Figure 5.2 depicts part of the

Table 5.3: *Flight planning parameters*

Camera	G	D_w [m]	H_f [m]	L_f [m]	H_f [m]
RGB	0.14	6.00	4.00	1.17	0.87
IR	0.91	6.00	4.00	1.17	0.88

planned path. Several similar paths are stacked vertically at a distance of 1 m to cover the whole building. A total of 328 image pairs were captured:

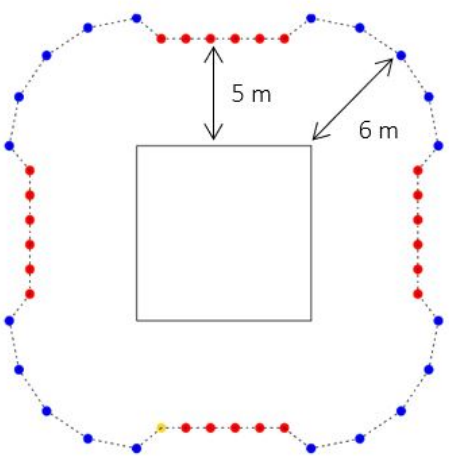

Figure 5.2: *Top view of a section of the planned path. The starting point is depicted in yellow, while red points are generated to capture the facades and blue ones to capture corners.*

to each 4056×3040 color picture, a 640×480 thermal one is associated. The SfM algoithm was applied with the help of Agisoft Metashape [150]. Given the low accuracy of the GPS, 7 features were also identified in several pictures and their distances measured to obtain the correct scale factor (Figure 5.3).

The resulting sparse point cloud of the building is shown in Figure 5.4.

By applying the model simplification method described earlier with a tile side length of $l = 10$ cm and filtering outliers, the 3D model is scaled from 942'660 points to only 137'594, depicted in Figure 5.5.

As far as MSAC parameters are concerned, the maximum distance between a point and the plane was set as 0.25 m, the maximum angle at $1°$, the maximum iterations at 10000 and the confidence factor at 0.99.

In order to obtain an estimate for the radiance and transmittance parameters, the atmosphere was modelled with MODTRAN® [151] as a unique

Figure 5.3: *Positioning of the markers*

Figure 5.4: *Sparse point cloud of the case study.*

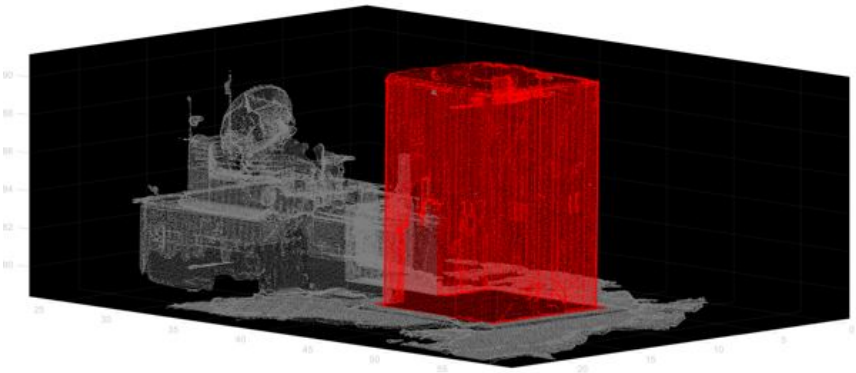

Figure 5.5: *Refined point cloud. The area where segmentation was applied is highlighted in red.*

Input	Value	Unit of measure
Atmoshperic	Mid-latitude winter	-
Water column	500	Atm/cm
O_3 column	0.004	Atm/cm
CO_2	400	ppm
CO	0.15	ppm
CH_4	1.8	ppm
T_{ground}	10	°C
Ground albedo	0.25	-
Aerosol model	Rural	-

Table 5.4: *MODTRAN parameters*

homogeneous layer. The specified values and obtained values are shown in table 5.4. It is important to mention that temperature, water column and ozone input required by the code were calculated considering the latitude where the site is in relation with the altitude. As a result, the heat exchange due to irradiation was estimated to be $140\ W/m^2$.

The temperature inside the building was regulated to be constant, with a value of 20 °C, and was assumed to be uniform throughout the building. Furthermore, the building was kept at this temperature for several hours in order to reach steady state. The composition of each layer of the walls heavily influences their thermal impedance. Since it was impossible to make an invasive analysis to characterize the layers composing the envelope, conduction parameters were estimated based on the year of construction of the building and the techniques and regulations of the time. The office centre of Spino d'Adda was realized in the 1970s, the external wall features a 4 cm air gap through two layers of bricks, while the roof contains 10 cm of

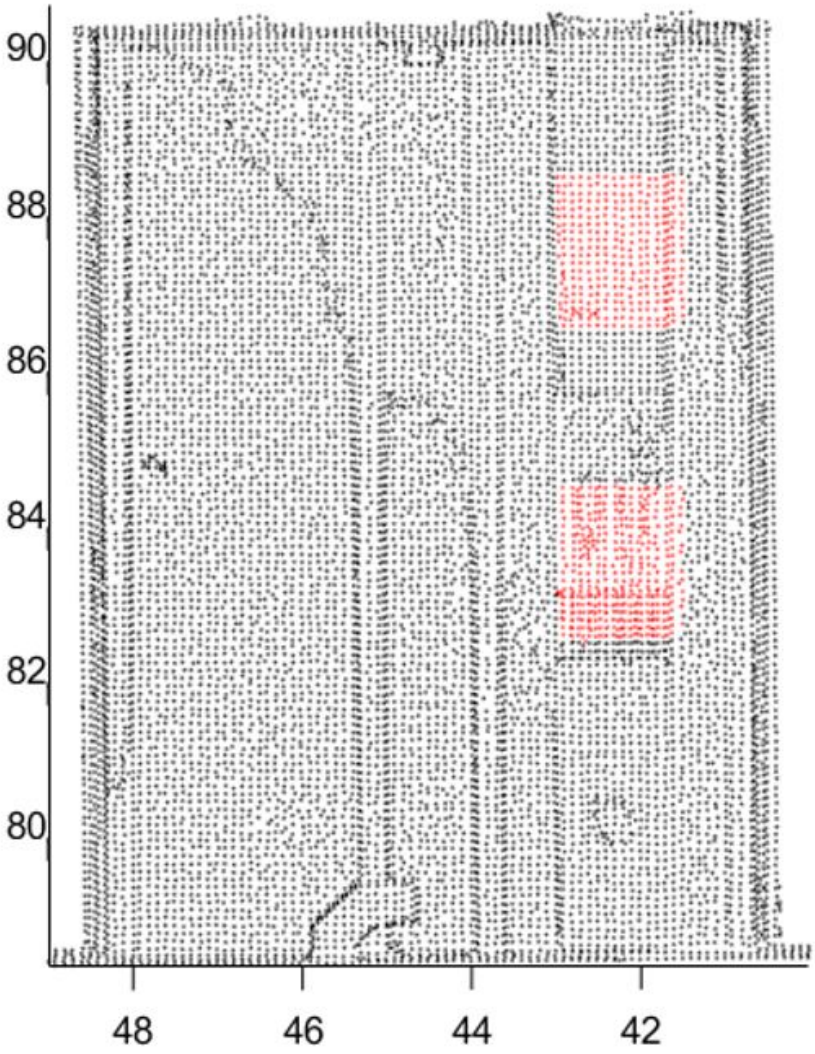

Figure 5.6: *MSAC can also sub-divide facades into their components. Two windows of the east facade highlighted in red.*

	Composition	Width	U
Wall	Double brick with air gap	32 cm	$1.1 \pm 0.2 \frac{W}{m^2K}$
Window	Double clear float and argon gap	2.4 cm	$2.5 \pm 0.2 \frac{W}{m^2K}$
Roof	Masonry slab and insulation layer	38 cm	$0.53 \pm 0.2 \frac{W}{m^2K}$

Table 5.5: *Conduction coefficients*

thermal insulation and an impermeabilization layer, also visible from the top of the building. Resulting coefficients are presented in Table 5.5.

Qualitative analysis of the thermal behavior

In the considered building a qualitative analysis of the thermal images already highlights defects and failures inside the envelope through color contrast between adjacent areas, as visible in Figure 5.7. It is possible to notice several areas considered potentially interesting from an analytical point of view. In the east facade (area 2), an important difference in terms of temperature in the areas close to the windows is noticeable. It is reasonable to suppose that in that area, during a renovation, part of the wall was demolished in order to install the windows and substituted with a layer of material with thermal properties different from those of the rest of the wall. On the other hand, a solid dark margin running along all the perimeter of every single window demonstrates that no air leakage is present. Particular attention was devoted to the edges of the envelope, both between two facades and between a facade and the roof, where thermal bridges can occur due to the bi-axial heat transfer and due to the junction between different materials. The only edge where the temperature is higher than expected is the south-east one (marked as 3 in Figure 5.7), where a copper drainpipe is installed. Such material reflects significantly more light than the surrounding wall, likely biasing the temperature measurement. The top of the envelope (marked as 5 in Figure 5.7) reveals an important thermal defect in the roof. A significant thermal bridge causes a sizeable heat exchange, raising potential problems in the internal side of the offices. The shape and intensity of the thermal anomaly supports the diagnosis of a general failure of the waterproof layer: due to the roof not being sloped, rainwater is not dispersed effectively and prolonged stagnation has probably infiltrated the roof layers, affecting the rest of the elements and reducing their thermal insulation performance. Another thermal defect that can be seen is at the bottom of the envelope, between the sidewalk and the opaque east facade (area 6).

The presence of a short segment where the measured temperature is much higher than the rest of the envelope suggests that there is a thermal bridge caused by a possible problem in the realization phase. It is also apparent that door and window frames exhibit a high temperature, conveyed by the bright orange colour. This is probably due to the aluminum alloy composing them, which has poor thermal insulation properties with respect to the walls. It should also be stressed that such alloys tend to partially reflect light and could thus induce a bias in the temperature measurement, even if the insulation is effective. Table 5.6 summarizes these considerations.

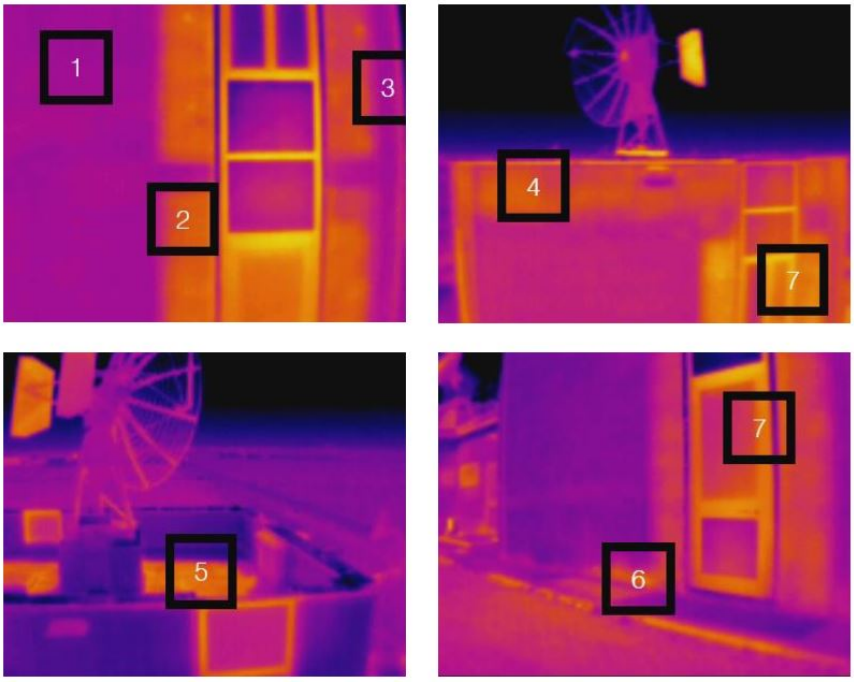

Figure 5.7: *Thermal issues highlighted by analysis of the pictures. The drainpipe in the south-east corner is visible in the top-right image, to the left of area 4.*

Quantitative analysis and comparison with the thermal model The issues highlighted in the previous section through qualitative analysis of the thermal images can also be detected by comparing the model with the measured temperatures. Given the measured temperature is also possible to estimate the thermal transmittance U_{es} as measured by the IR camera. A quantitative analysis also enables the measurement of the extent to which the detected defects impair the insulation of the building. This difference can be taken as a marker of severity to plan suitable interventions. Numerical data is

Area	Temperature $[°C]$			Transmittance $U[W/m^2K]$			Diagnosis
	T_{ex}	T_{es}	Diff.	U_{ex}	U_{es}	% Var.	
1	12.8	12.3	-0.5	1.14	1.12	-2%	-
2	12.8	15.3	2.5	1.14	2.04	79%	Thermal bridge
3	12.8	15.9	3.1	1.14	2.35	106%	Measurement anomaly
4	12.8	15.2	2.4	1.14	2.02	77%	Thermal bridge
5	13.6	16.6	3.0	1.52	2.81	85%	Water leak
6	12.8	14.8	2.0	1.14	1.85	62%	Thermal bridge

Table 5.6: *Summary of the quantitative analysis. T_{es}, U_{es} are the measured temperature and thermal transmittance of the external surface, whereas T_{ex}, U_{ex} are the expected values predicted by the model.*

gathered in Table 5.6.

The area marked as 1 in figure 5.7 is used as reference to test that the model correctly predicts the thermal behavior of the building under normal circumstances. The absolute error in temperature is deemed reasonably small. The analysis also captures all the issues raised in the previous section, evidencing a difference in temperature of at least 2 °C between measured and expected value in all relevant areas. These correspond to significant variations in thermal transmittance of the surfaces. Such deviations from the expected behavior in the direction of higher transmittance, ranging from 62% to 106%, point to a severe decay of the thermal performance of the building, regarding especially the roof, where the water leak problem could potentially affect the serviceability of the structure in general. The data pertaining to area 4 in Figure 5.7 adds to the diagnosis of a severely damaged roof, even if the parapet wall does not facilitate the flow of heat from the inside of the building.

Discussion

The identification of defects and measurement of their severity through comparison with a model of the building can be exploited to rank the issues by urgency and devise an intervention plan aimed at solving them. Furthermore, a classification based on severity can also facilitate an objective and repeatable description of the state of the building according to national guidelines and international standards, making comparisons between different buildings possible and reliable.

Given the nature of IR pictures and the appearance of thermal defects within them, damage detection could also be performed through Machine Learning techniques. In particular, an algorithm can be trained to recognize defects in pictures starting from the set of available pictures (i.e. the mea-

sured surface temperatures) and the model predictions, which together constitute a set of labeled data. This idea can also be expanded to exploit geometrical information originating from the 3D model reconstruction through SfM.

Furthermore, the monetary and pilot training costs of adopting UAVs to take the pictures is well worth the gains: they are easy to use, they significantly extend the reach and speed of movement of the cameras, they relieve human operators of the necessity to climb structures or use cranes, thus decreasing risks by a substantial margin, and they offer the possibility of fully automating inspections, at least in private environments, thus driving the costs further down. In short, they make inspections faster, safer and cheaper. We expect them to become the industry standard in the next decade.

5.1.3 Conclusion and Future Work

We presented a UAV-based pipeline for the 3D modeling of a structure and its thermal behavior assessment through IR pictures, comprising a procedure for generating the best candidate points for data gathering, the reconstruction of the model through SfM, its segmentation with MSAC, the modeling of the thermal behavior of the structure and the comparison with the measured data. The end result is a complete process from mission planning to diagnosis and intervention planning which offers great automation potential. The clarity of the obtained results and the ease of data collection lead us to conclude that drones should become even more integrated in regular inspection and maintenance, as they can greatly reduce the costs and time requirements for performing such tasks.

While the automation of some of the steps has already been undertaken in our laboratory (see [21]), we aim at automating the entire pipeline in the near future, in particular the damage detection tasks through data analysis, with the overall objective of designing a completely autonomous drone network for the inspection of civil infrastructure.

Acronyms

B

BILP	Binary Integer Linear Problem. 29, 36
BIM	Building Information Modeling. 10

C

CAD	Computer Aided Design. 16
CPU	Central Processing Unit. 89, 95
CVRP	Constrained Vehicle Routing Problem. I, III, 11–13, 16, 24, 30, 32, 34–38, 41–44, 155

D

DIC	Digital Image Correlation. 111, 112
DJI	Da-Jiang Innovations. 2, 123–125, 142
DOF	Degrees of Freedom. 48

F

FFT	Fast Furier Transform. 118–120, 125, 126
FHOCP	Finite Horizon Optimal Control Problem. 70, 71, 78–80
FOV	Field of View. 16, 123
FPS	Frames Per Second. 113, 123, 127, 130

G

GPS	Global Positioning System. 71, 113, 138, 143
GPU	Gprahics Processing Unit. 89

I

IMU	Inertial Measurement Unit. 71, 87, 131
IR	Infra-Red. 136, 138, 142, 143, 148–150

K

KLT	Kanade-Lucas-Tomasi. 112, 116

L

LED	Light-Emitting Diode. 112
LiDAR	Light Detection And Ranging. I, III, 2, 3, 5, 45, 47–50, 52, 55–57, 60–62, 65, 71, 74–77, 81, 82, 85–87, 118, 135, 155
LP	Linear Program. 76

M

MILP	Mixed-Integer Linear Problem. 29, 36
MPC	Model Predictive Control. I, III, 4, 5, 46, 66, 70, 79, 82
MSAC	M-Estimator Sample Consensus. 139, 143, 146, 150
mTSP	multi-agent Traveling Salesman Problem. 11

N

NLP	Non-Linear Program. 74

P

POI	Points of Interest. 10–13, 15–19, 27, 32–34, 42, 155

Q

QCQP	Quadratically Constrained Quadratic Program. 78
QP	Quadratic Program. 54, 61, 62, 65

R

RANSAC RANdom Sample Consensus. 139
RGB Red Green and Blue. 2, 112, 133, 135, 136, 138, 142, 143
RMP Riemanniann Motion Policy. I, 4, 85, 103
RRT Rapidly-exploring Random Tree. 47

S
SEC Subtour Elimination Constraints. 11, 30
SfM Structure from Motion. 11, 135, 136, 138, 139, 143, 150
SHM Structural Health Monitoring. 2, 109–112
SIFT Scale-Invariant Feature Transform. 135
STEM System of TEthered Multicopters. 45, 46, 48, 54, 65, 66, 155

T
TSP Traveling Salesman Problem. I, 11, 13, 21, 22, 27, 32, 34, 35, 41–43, 155

U
UAV Unmanned Aerial Vehicle. I, 1–3, 9–12, 14, 15, 19–23, 42, 43, 45, 46, 48, 49, 52, 55, 57, 82, 84, 85, 107, 111, 114, 118, 123, 127, 131, 136, 150
UCLA University of California - Los Angeles. 18, 19, 28, 33, 36–41, 43
USA United States of America. 109

Y
YOLO You Only Look Once. 87, 88, 95, 98, 101

List of Figures

List of Tables

Bibliography

[1] T. Rakha and A. Gorodetsky, "Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones," *Automation in Construction*, vol. 93, no. May, pp. 252–264, 2018.

[2] A. Hodges, "Alan turing: the enigma," in *Alan Turing: The Enigma*, Princeton University Press, 2014.

[3] P. D. SAS, "Parrot drones," 2022.

[4] DJI, "Dji drones," 2022.

[5] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2016.

[6] V. Puri, A. Nayyar, and L. Raja, "Agriculture drones: A modern breakthrough in precision agriculture," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 507–518, 2017.

[7] U. R. Mogili and B. Deepak, "Review on application of drone systems in precision agriculture," *Procedia computer science*, vol. 133, pp. 502–509, 2018.

[8] L. Tang and G. Shao, "Drone remote sensing for forestry research and practices," *Journal of Forestry Research*, vol. 26, no. 4, pp. 791–797, 2015.

[9] M. S. Innocente and P. Grasso, "Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems," *Journal of Computational Science*, vol. 34, pp. 80–101, 2019.

[10] L. La Bella, *Drones and entertainment*. The Rosen Publishing Group, Inc, 2016.

[11] C. Kima, H. Moon, and W. Leea, "Data management framework of drone-based 3d model reconstruction of disaster site," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, 2016.

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

[13] M. L. Mauriello and J. E. Froehlich, "Towards automated thermal profiling of buildings at scale using unmanned aerial vehicles and 3d-reconstruction," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pp. 119–122, 2014.

[14] V. Hoskere, J.-W. Park, H. Yoon, and B. F. Spencer Jr, "Vision-based modal survey of civil infrastructure using unmanned aerial vehicles," *Journal of Structural Engineering*, vol. 145, no. 7, p. 04019062, 2019.

[15] B. B. Kocer, T. Tjahjowidodo, M. Pratama, and G. G. L. Seet, "Inspection-while-flying: An autonomous contact-based nondestructive test using uav-tools," *Automation in Construction*, vol. 106, p. 102895, 2019.

[16] D. Lattanzi and G. Miller, "Review of robotic infrastructure inspection systems," *Journal of Infrastructure Systems*, vol. 23, no. 3, pp. 1–16, 2017.

[17] G. Silano, T. Baca, R. Penicka, D. Liuzza, and M. Saska, "Power line inspection tasks with multi-aerial robot systems via signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, 2021.

[18] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Nikolakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[19] T. Rakha, A. Liberty, A. Gorodetsky, B. Kakillioglu, and S. Velipasalar, "Heat mapping drones: an autonomous computer-vision-based procedure for building envelope inspection using unmanned aerial systems (uas)," *Technology| Architecture+ Design*, vol. 2, no. 1, pp. 30–44, 2018.

[20] M. Bolognini, G. Izzo, D. Marchisotti, L. Fagiano, M. P. Limongelli, and E. Zappa, "Vision-based modal analysis of built environment structures with multiple drones," *Automation in Construction*, vol. 143, p. 104550, 2022.

[21] M. Bolognini and L. Fagiano, "A scalable hierarchical path planning technique for autonomous inspections with multicopter drones," in *2021 European Control Conference (ECC)*, pp. 787–792, IEEE, 2021.

[22] M. Bolognini, L. Fagiano, and M. P. Limongelli, "An autonomous, multi-agent UAV platform for inspection of civil infrastructure," in *Proceedings of the 10th International Conference on Structural Health Monitoring of Intelligent Infrastructure, SHMII 10*, 2021.

[23] M. Bolognini and L. Fagiano, "Lidar-based navigation of tethered drone formations in an unknown environment," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9426–9431, 2020.

[24] L. Duque, J. Seo, and J. Wacker, "Synthesis of unmanned aerial vehicle applications for infrastructures," *Journal of Performance of Constructed Facilities*, vol. 32, no. 4, p. 04018046, 2018.

[25] B. F. Spencer Jr, V. Hoskere, and Y. Narazaki, "Advances in computer vision-based civil infrastructure inspection and monitoring," *Engineering*, vol. 5, no. 2, pp. 199–222, 2019.

[26] V. Hoskere, Y. Narazaki, T. Hoang, and B. Spencer Jr, "Vision-based structural inspection using multiscale deep convolutional neural networks," *arXiv preprint arXiv:1805.01055*, 2018.

[27] G. Morgenthal and N. Hallermann, "Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014.

[28] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 731–747, 2018.

[29] M. Shanmugavel, A. Tsourdos, B. White, and R. Å»bikowski, "Co-operative path planning of multiple uavs using dubins paths with clothoid arcs," *Control Engineering Practice*, vol. 18, no. 9, pp. 1084–1092, 2010.

[30] V. P. Tran, M. Garratt, and I. R. Petersen, "Switching time-invariant formation control of a collaborative multi-agent system using negative imaginary systems theory," *Control Engineering Practice*, vol. 95, p. 104245, 2020.

[31] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi, "Submodular trajectory optimization for aerial 3d scanning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5324–5333, 2017.

[32] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6423–6430, IEEE, 2015.

[33] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, "Uniform coverage structural inspection path–planning for micro aerial vehicles," in *2015 IEEE international symposium on intelligent control (ISIC)*, pp. 59–64, IEEE, 2015.

[34] A. Donzé, "On signal temporal logic," in *International Conference on Runtime Verification*, pp. 382–383, Springer, 2013.

[35] S. Daftry, C. Hoppe, and H. Bischof, "Building with drones: Accurate 3d facade reconstruction using mavs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3487–3494, IEEE, 2015.

[36] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[37] B. L. Brumitt and A. Stentz, "Grammps: A generalized mission planner for multiple mobile robots in unstructured environments," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 2, pp. 1564–1571, IEEE, 1998.

[38] E. I. Grøtli and T. A. Johansen, "Path planning for uavs under communication constraints using splat! and milp," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 265–282, 2012.

[39] H. Ergezer and K. Leblebicioğlu, "3d path planning for multiple uavs for maximum information collection," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 737–762, 2014.

[40] M. Salaris, A. Riva, and F. Amigoni, "Multirobot coverage of linear modular environments," *arXiv preprint arXiv:2001.02906*, 2020.

[41] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[42] C. Lee, K. Lee, and S. Park, "Robust vehicle routing problem with deadlines and travel time/demand uncertainty," *Journal of the Operational Research Society*, vol. 63, no. 9, pp. 1294–1306, 2012.

[43] I. Sungur, F. Ordónez, and M. Dessouky, "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty," *Iie Transactions*, vol. 40, no. 5, pp. 509–523, 2008.

[44] F. Ordóñez, "Robust vehicle routing," in *Risk and optimization in an uncertain world*, pp. 153–178, INFORMS, 2010.

[45] J.-A. Fernández-Madrigal and J. González, "Multihierarchical graph search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 103–113, 2002.

[46] "Matlab toolbox containing code used in simulations." `https://it.mathworks.com/matlabcentral/fileexchange/95533-multidrone-robust-planning`. Accessed: 2021-10-27.

[47] J. Dentler, S. Kannan, M. A. O. Mendez, and H. Voos, "A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors," in *2016 IEEE Conference on Control Applications (CCA)*, pp. 519–525, 2016.

[48] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[49] MATLAB, *version 9.9.0 (R2020b)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[50] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde University*, 2017.

[51] "GitHub repository of code used in simulations." `https://github.com/MicheleBolognini/MultidroneRobustPlanning`. Accessed: 2021-07-14.

[52] M. Xiao and H. Nagamochi, "An exact algorithm for tsp in degree-3 graphs via circuit procedure and amortization on connectivity structure," *Algorithmica*, vol. 74, no. 2, pp. 713–741, 2016.

[53] D. Eppstein, "The traveling salesman problem for cubic graphs," 2007.

[54] M. N. Boukoberine, Z. Zhou, and M. Benbouzid, "Power supply architectures for drones-a review," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 5826–5831, IEEE, 2019.

[55] L. Fagiano, "Systems of tethered multicopters: modeling and control design," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4610–4615, 2017.

[56] Elistair, "Company website: http://elistair.com/, last accessed september 2016," 2020.

[57] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *2011 50th IEEE conference on decision and control and European control conference*, pp. 7488–7493, IEEE, 2011.

[58] D. Saccani and L. Fagiano, "Autonomous uav navigation in an unknown environment via multi-trajectory model predictive control," in *2021 European Control Conference (ECC)*, pp. 1577–1582, IEEE, 2021.

[59] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous uav navigation using reinforcement learning," *arXiv preprint arXiv:1801.05086*, 2018.

[60] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based uav path planning," *Knowledge-Based Systems*, vol. 158, pp. 54–64, 2018.

[61] B. Kosarnovsky and S. Arogeti, "A string of tethered drones - system dynamics and control," in *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–6, 2019.

[62] M. Tognon and A. Franchi, "Nonlinear observer for the control of bi-tethered multi aerial robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1852–1857, 2015.

[63] M. Tognon, "Attitude and tension control of a tethered formation of aerial vehicles," Master's thesis, UniversitÃ degli Studi di Padova, 2014.

[64] M. Caruso, P. Gallina, and S. Seriani, "On the modelling of tethered mobile robots as redundant manipulators," *Robotics*, vol. 10, no. 2, p. 81, 2021.

[65] L. Fagiano, "Systems of tethered multicopters: Modeling and control design," *IFAC - PapersOnLine*, vol. 50, no. 1, pp. 4610–4615, 2017.

[66] M. Hwangbo, J. Kuffner, and T. Kanade, "Efficient two-phase 3d motion planning for small fixed-wing uavs," (Rome, Italy), pp. 1035–1041, 2007.

[67] Y. Kitamura, F. Kishino, T. Tanaka, and M. Yachida, "Real-time path planning in a dynamic 3-D environment," No. 2, (Osaka, Japan), pp. 925–931, 1996.

[68] F. Duchon, A. Babinec, and M. Kajan, "Path planning with modified a* algorithm for a mobile robot," *Modeling of Mechanical and Mechatronic Systems*, vol. 96, pp. 59–69, 2014.

[69] Y. Li, Z. Littlefield, and K. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[70] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," (St. Louis, MO, USA), pp. pp. 500–505, 1986.

[71] R. Zapata and P. Lepinay, "Flying among obstacles," *Proceedings of the Third European Workshop on Advanced Mobile Robots*, pp. 81–88, 1999.

[72] J. C. Zufferey and D. Floreano, "Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 137–146, 2006.

[73] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," in *17th IFAC World Congress*, (Seoul, Korea), pp. 6974–6979, 2008.

[74] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Transactions on Automatic Control*, vol. 61, pp. 1111–1116, April 2016.

[75] S. Formentin and M. Lovera, "Flatness-based control of a quadrotor helicopter via feedforward linearization," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 6171–6176, IEEE, 2011.

[76] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[77] M. Milanese, J. Norton, H. Piet-Lahanier, and É. Walter, *Bounding approaches to system identification*. Springer Science & Business Media, 2013.

[78] M. Lauricella and L. Fagiano, "Set membership identification of linear systems with guaranteed simulation accuracy," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5189–5204, 2020.

[79] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[80] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*, pp. 207–226, Springer, 1999.

[81] A. Uryasheva, M. Kulbeda, N. Rodichenko, and D. Tsetserukou, "Dronegraffiti: autonomous multi-uav spray painting," in *ACM SIGGRAPH 2019 Studio*, pp. 1–2, 2019.

[82] R. L. Dahlstrom, "State of technology: Cleaning and coating uav systems-industrial spray painting drones," in *Offshore Technology Conference*, OnePetro, 2020.

[83] L. Shi, M. Pantic, O. Andersson, M. Tognon, R. Siegwart, and R. H. Jacobsen, "Reactive motion planning for rope manipulation and collision avoidance using aerial robots," *IROS - IN PRESS*, 2022.

[84] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *2013 European Control Conference (ECC)*, pp. 1805–1810, IEEE, 2013.

[85] L. A. Sandino, M. Bejar, K. Kondak, and A. Ollero, "Advances in modeling and control of tethered unmanned helicopters to enhance hovering performance," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 3–18, 2014.

[86] M. Tognon and A. Franchi, "Dynamics, control, and estimation for aerial robots tethered by cables or bars," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 834–845, 2017.

[87] B. Kosarnovsky and S. Arogeti, "Geometric and constrained control for a string of tethered drones," *Robotics and Autonomous Systems*, vol. 133, p. 103609, 2020.

[88] M. Tognon, R. Alami, and B. Siciliano, "Physical human-robot interaction with a tethered aerial vehicle: Application to a force-based human guiding problem," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 723–734, 2021.

[89] M. Allenspach, Y. Vyas, M. Rubio, R. Siegwart, and M. Tognon, "Human-state-aware controller for a tethered aerial robot guiding a human by physical interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2827–2834, 2022.

[90] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.

[91] A. S. Vempati, M. Kamel, N. Stilinovic, Q. Zhang, D. Reusser, I. Sa, J. Nieto, R. Siegwart, and P. Beardsley, "Paintcopter: An autonomous uav for spray painting on three-dimensional surfaces," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2862–2869, 2018.

[92] H. Ando, Y. Ambe, A. Ishii, M. Konyo, K. Tadakuma, S. Maruyama, and S. Tadokoro, "Aerial hose type robot by water jet for fire fighting," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1128–1135, 2018.

[93] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[95] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[96] M. Xie, K. Van Wyk, A. Li, M. A. Rana, Q. Wan, D. Fox, B. Boots, and N. Ratliff, "Geometric fabrics for the acceleration-based design of robotic motion," *arXiv preprint arXiv:2010.14750*, 2020.

[97] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, "Optimization fabrics," *arXiv preprint arXiv:2008.02399*, 2020.

[98] I. T. Forum, "Transport infrastructure investment and maintenance." Accessed: 2022-06-17.

[99] A. Sabato, C. Niezrecki, and G. Fortino, "Wireless mems-based accelerometer sensor boards for structural vibration monitoring: a review," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 226–235, 2016.

[100] Y. Xu, J. M. Brownjohn, and F. Huseynov, "Accurate deformation monitoring on bridge structures using a cost-effective sensing system combined with a camera and accelerometers: Case study," *Journal of Bridge Engineering*, vol. 24, no. 1, p. 05018014, 2019.

[101] M. A. Sutton, J. H. Yan, V. Tiwari, H. Schreier, and J.-J. Orteu, "The effect of out-of-plane motion on 2d and 3d digital image correlation measurements," *Optics and Lasers in Engineering*, vol. 46, no. 10, pp. 746–757, 2008.

[102] L. M. Khoo, P. R. Mantena, and P. Jadhav, "Structural damage assessment using vibration modal analysis," *Structural Health Monitoring*, vol. 3, no. 2, pp. 177–194, 2004.

[103] Z.-F. Fu and J. He, *Modal analysis*. Elsevier, 2001, ISBN 978-0750650793.

[104] C.-Z. Dong and F. N. Catbas, "A review of computer vision-based structural health monitoring at local and global levels," *Structural Health Monitoring*, vol. 20, no. 2, pp. 692–743, 2021.

[105] C.-Z. Dong, O. Celik, and F. N. Catbas, "Marker-free monitoring of the grandstand structures and modal identification using computer vision methods," *Structural Health Monitoring*, vol. 18, no. 5-6, pp. 1491–1509, 2019.

[106] C. Tomasi and T. Kanade, "Detection and tracking of point," *International Journal of Computer Vision*, vol. 9, pp. 137–154, 1991.

[107] D. Feng and M. Q. Feng, "Experimental validation of cost-effective vision-based structural health monitoring," *Mechanical Systems and Signal Processing*, vol. 88, pp. 199–211, 2017.

[108] D. Feng and M. Q. Feng, "Vision-based multipoint displacement measurement for structural health monitoring," *Structural Control and Health Monitoring*, vol. 23, no. 5, pp. 876–890, 2016.

[109] H. Yoon, H. Elanwar, H. Choi, M. Golparvar-Fard, and B. F. Spencer Jr, "Target-free approach for vision-based structural system identification using consumer-grade cameras," *Structural Control and Health Monitoring*, vol. 23, no. 12, pp. 1405–1416, 2016.

[110] Y. Yang, C. Dorn, T. Mancini, Z. Talken, G. Kenyon, C. Farrar, and D. Mascareñas, "Blind identification of full-field vibration modes from video measurements with phase-based video motion magnification," *Mechanical Systems and Signal Processing*, vol. 85, pp. 567–590, 2017.

[111] V. Fioriti, I. Roselli, A. Tatì, R. Romano, and G. De Canio, "Motion magnification analysis for structural monitoring of ancient constructions," *Measurement*, vol. 129, pp. 375–380, 2018.

[112] J. G. Chen, T. M. Adams, H. Sun, E. S. Bell, and O. Büyüköztürk, "Camera-based vibration measurement of the world war i memorial bridge in portsmouth, new hampshire," *Journal of Structural Engineering*, vol. 144, no. 11, p. 04018207, 2018.

[113] J. J. Lin, A. Ibrahim, S. Sarwade, and M. Golparvar-Fard, "Bridge inspection with aerial robots: Automating the entire pipeline of visual data capture, 3d mapping, defect detection, analysis, and reporting," *Journal of Computing in Civil Engineering*, vol. 35, no. 2, p. 04020064, 2021.

[114] Y. Ji and C. Chang, "Nontarget stereo vision technique for spatiotemporal response measurement of line-like structures," *Journal of engineering mechanics*, vol. 134, no. 6, pp. 466–474, 2008.

[115] S.-W. Kim, B.-G. Jeon, N.-S. Kim, and J.-C. Park, "Vision-based monitoring system for evaluating cable tensile forces on a cable-stayed bridge," *Structural Health Monitoring*, vol. 12, no. 5-6, pp. 440–456, 2013.

[116] X. Zhao, K. Ri, and N. Wang, "Experimental verification for cable force estimation using handheld shooting of smartphones," *Journal of Sensors*, vol. 2017, 2017.

[117] D. Feng, T. Scarangello, M. Q. Feng, and Q. Ye, "Cable tension force estimate using novel noncontact vision-based sensor," *Measurement*, vol. 99, pp. 44–52, 2017.

[118] J. Baqersad, P. Poozesh, C. Niezrecki, and P. Avitabile, "Photogrammetry and optical methods in structural dynamics - a review," *Mechanical Systems and Signal Processing*, vol. 86, pp. 17–34, 2017. Full-field, non-contact vibration measurement methods: comparisons and applications.

[119] F. Chen, X. Chen, X. Xie, X. Feng, and L. Yang, "Full-field 3d measurement using multi-camera digital image correlation system," *Optics and Lasers in Engineering*, vol. 51, no. 9, pp. 1044–1052, 2013.

[120] K. Patil, V. Srivastava, and J. Baqersad, "A multi-view optical technique to obtain mode shapes of structures," *Measurement*, vol. 122, pp. 358–367, 2018.

[121] D. Liu, X. Xia, J. Chen, and S. Li, "Integrating building information model and augmented reality for drone-based building inspection," *Journal of Computing in Civil Engineering*, vol. 35, no. 2, p. 04020073, 2021.

[122] P. J. Sanchez-Cuevas, A. Gonzalez-Morgado, N. Cortes, D. B. Gayango, A. E. Jimenez-Cano, A. Ollero, and G. Heredia, "Fully-actuated aerial manipulator for infrastructure contact inspection: Design, modeling, localization, and control," *Sensors*, vol. 20, no. 17, p. 4708, 2020.

[123] P. Pfändler, K. Bodie, U. Angst, and R. Siegwart, "Flying corrosion inspection robot for corrosion monitoring of civil structures–first results," in *SMAR 2019-Fifth Conference on Smart Monitoring, Assessment and Rehabilitation of Civil Structures-Program*, pp. We–4, SMAR, 2019.

[124] M. Á. Trujillo, J. R. Martínez-de Dios, C. Martín, A. Viguria, and A. Ollero, "Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry," *Sensors*, vol. 19, no. 6, p. 1305, 2019.

[125] P. Zhang, L. Zhang, T. Wu, H. Zhang, and X. Sun, "Detection and location of fouling on photovoltaic panels using a drone-mounted infrared thermography system," *Journal of Applied Remote Sensing*, vol. 11, no. 1, p. 016026, 2017.

[126] Y. S. Lee, D. G. Lee, Y. G. Yu, and H. J. Lee, "Application of drone photogrammetry for current state analysis of damage in forest damage areas," *Journal of Korean Society for Geospatial Information System*, vol. 24, no. 3, pp. 49–58, 2016.

[127] P. Hu, X. Hao, J. Li, C. Cheng, and A. Wang, "Design and implementation of binocular vision system with an adjustable baseline and high synchronization," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 566–570, IEEE, 2018.

[128] A. Noda, Y. Yamakawa, and M. Ishikawa, "Frame synchronization for networked high-speed vision systems," in *SENSORS, 2014 IEEE*, pp. 269–272, IEEE, 2014.

[129] M. N. Helfrick, C. Niezrecki, P. Avitabile, and T. Schmidt, "3d digital image correlation methods for full-field vibration measurement," *Mechanical systems and signal processing*, vol. 25, no. 3, pp. 917–927, 2011.

[130] H. Yoon, J. Shin, and B. F. Spencer Jr, "Structural displacement measurement using an unmanned aerial system," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 3, pp. 183–192, 2018.

[131] A. Khadka, B. Fick, A. Afshar, M. Tavakoli, and J. Baqersad, "Non-contact vibration monitoring of rotating wind turbines using a semi-autonomous uav," *Mechanical Systems and Signal Processing*, vol. 138, p. 106446, 2020.

[132] C. Lee, W. A. Take, and N. A. Hoult, "Optimum accuracy of two-dimensional strain measurements using digital image correlation," *Journal of Computing in Civil Engineering*, vol. 26, no. 6, pp. 795–803, 2012.

[133] K. G. Derpanis, "The harris corner detector," *York University*, vol. 2, 2004.

[134] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[135] A. Entrop and A. Vasenev, "Infrared drones in the construction industry: designing a protocol for building thermography procedures," *Energy procedia*, vol. 132, pp. 63–68, 2017.

[136] T. Rakha and A. Gorodetsky, "Review of unmanned aerial system (uas) applications in the built environment: Towards automated building inspection procedures using drones," *Automation in Construction*, vol. 93, pp. 252–264, 2018.

[137] T. Rakha, A. Liberty, A. Gorodetsky, B. Kakillioglu, and S. Velipasalar, "Campus as a lab for computer vision-based heat mapping drones: A case study for multiple building envelope inspection using unmanned aerial systems (UAS)," 2018.

[138] U. Weichenhain, "Cutting costs in infrastructure maintenance with uavs," 2019.

[139] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of uav 3d path planning," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pp. 2376–2381, IEEE, 2014.

[140] C. Buerhop-Lutz and H. Scheuerpflug, "Inspecting pv-plants using aerial, drone-mounted infrared thermography system," 3rd Southern African Solar Energy Conference, South Africa, 11-13 May, 2015., 2015.

[141] C. Eschmann, C.-M. Kuo, C.-H. Kuo, and C. Boller, "Unmanned aircraft systems for remote building inspection and monitoring," 2012.

[142] H. Ham, J. Wesley, and H. Hendra, "Computer vision based 3d reconstruction: A review," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 4, p. 2394, 2019.

[143] L. Inzerillo, G. Di Mino, and R. Roberts, "Image-based 3d reconstruction using traditional and uav datasets for analysis of road pavement distress," *Automation in Construction*, vol. 96, pp. 457–469, 2018.

[144] Djurdjani and D. Laksono, "Open source stack for structure from motion 3d reconstruction: A geometric overview," in *2016 6th International Annual Engineering Seminar (InAES)*, pp. 196–201, 2016.

[145] T. Lindeberg, "Scale invariant feature transform," 2012.

[146] L. Barazzetti, S. Erba, M. Previtali, E. Rosina, and M. Scaioni, "Mosaicking thermal images of buildings," in *Videometrics, Range Imaging, and Applications XII; and Automated Visual Inspection*, vol. 8791, p. 879108, International Society for Optics and Photonics, 2013.

[147] ENAC, "Regolamento enac 2020: Mezzi a pilotaggio remoto," 2020. Last accessed 5 September 2021.

[148] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[149] K. G. Derpanis, "Overview of the ransac algorithm," *Image Rochester NY*, vol. 4, no. 1, pp. 2–3, 2010.

[150] Agisoft, "Metashape." `https://www.agisoft.com/`, 2022.

[151] S. S. Inc., "Modtran." `http://modtran.spectral.com/`, 2022.