Universidad Zaragoza 1542

Master's Thesis

Multi-robot Active Perception for Fast and Efficient Scene Reconstruction

Autor

David Morilla Cabello

Directores

Margarita Chli

Eduardo Montijano Muñoz

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2022

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisors, Eduardo and Margarita, and the researchers, Lucas and Luca, who contributed with their inestimable help in this work. Their inspiring ideas and practicality made this work possible, and their corrections finally polished the final result and my experience as a future researcher. I would also like to thank my colleagues, whose friendship facilitated the move to Zaragoza and made it feel like a second home. Finally, I would like to thank my parents, Mavi and Luis, who supported me in every decision and move and encouraged me to follow my goals.

ABSTRACT

The efficiency of path-planning in robot navigation is crucial in tasks, such as search-and-rescue and disaster surveying, but this is emphasised even more when considering multi-rotor aerial robots due to the limited battery and flight time. In this spirit, this Master Thesis proposes an efficient, hierarchical planner to achieve a comprehensive visual coverage of large-scale outdoor scenarios for small drones. Following an initial reconnaissance flight, a coarse map of the scene gets built in real-time. Then, regions of the map that were not appropriately observed are identified and grouped by a novel perception-aware clustering process that enables the generation of continuous trajectories (*sweeps*) to cover them efficiently. Thanks to this partitioning of the map in a set of tasks, we are able to generalize the planning to an arbitrary number of drones and perform a well-balanced workload distribution among them. We compare our approach to an alternative state-of-the-art method for exploration and show the advantages of our pipeline in terms of efficiency for obtaining coverage in large environments. This work was carried out in person collaboration with the Vision for Robotics Lab (V4RL) in ETH Zürich. The stay enabled to share knowledge and methods in perception and path planning between both laboratories that underlie the results of this thesis. As a result of the collaboration, a scientific article was submitted for publication to the IEEE Robotics and Automation Lettters with option to the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Supplementary video – `https://youtu.be/V2UIrM91oQ8`

Index

Chapter 1

Introduction

Recent advances in robot navigation and perception have enabled the establishment of modern multi-rotor aircraft, i.e., drones, as the best choice for autonomous 3D reconstruction or visual coverage of large-scale outdoor scenarios. Their flexibility allows them to move freely through the environment and observe areas that are not visible from the ground. However, time efficiency is critical for using drones because of their short flight times (due to battery limitations), usually well under 30 minutes. Moreover, certain deployments add extra time requirements, such as search-and-rescue or disaster surveying as the case shown in Figure 1.1, where lives can be endangered. Therefore, the efficiency and effectiveness of the planning algorithms is essential to enable the deployment of drones in large scale outdoor environments. Similarly, using multiple drones as advocated in this work promises to boost the efficiency of the scene-coverage mission.

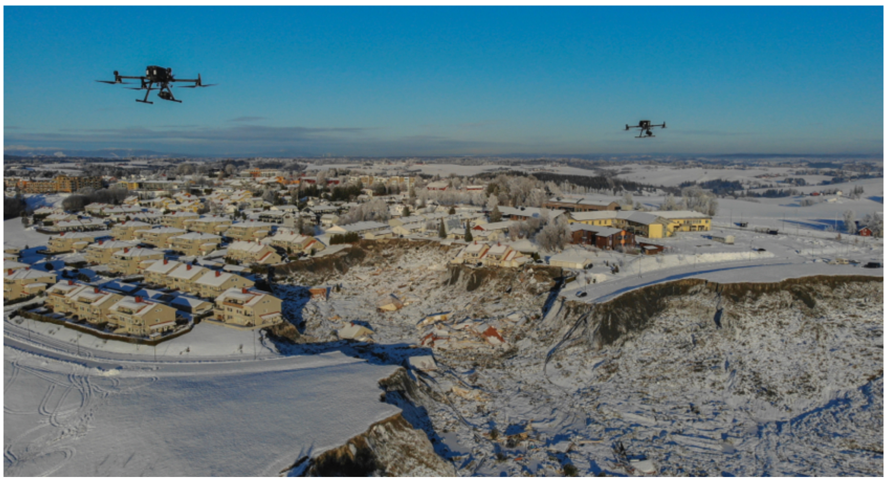


Figure 1.1: Use case of a team of drones for disaster surveying during Gjerdrum landslide in Norway [1]. Drones can traverse large distances fast and reach inaccessible areas.

[1] `enterprise-insights.dji.com/blog/let-drones-search-so-you-can-rescue-norway-landslide-m300`

Deploying drones for mapping a large area from a high altitude is an effective way to obtain a first estimation, as collisions with the environment can be more easily avoided. However, this strategy does not provide informative enough viewpoints for scene coverage and impacts the quality of the scene captures. State-of-the-art exploration approaches [1, 2] often lack in efficiency because of problems such as over-exploring local regions, and abrupt changes in motion due to constant re-planning or the need for revisiting areas.

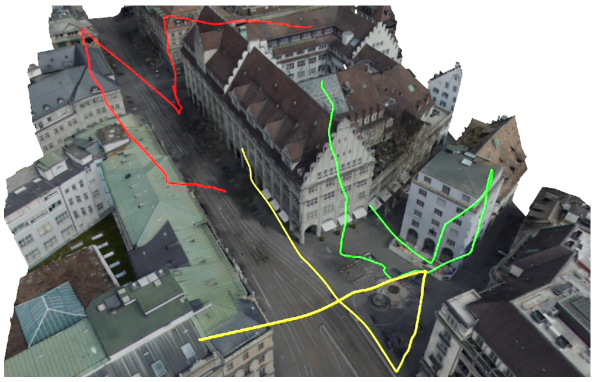


Figure 1.2: Team of drones that *sweep* the area of interest by flying paths generated by the proposed planner in order to achieve fast coverage. Using a rough prior map (e.g. captured in a reconnaissance flight) to identify areas that require further observation, this work generates efficient path planning and workload distribution for a team of drones (three in this example) to cover the scene.

To overcome these limitations, this Master Thesis presents a hybrid solution that uses the best of both types of strategy in a synergetic way. We consider a team of drones with cameras, each performing a fast, reconnaissance flight at a high-altitude capturing a rough map of the area of interest using a coarse real-time mapping pipeline. Although the noise present in the obtained map, we obtain a dense representation that allows reasoning on the scene structure. Based on this map, the proposed method computes a set of drone trajectories for subsequent flights in order to efficiently cover the area of interest completely. This process aims to maximize the use of *sweep* lines to avoid constant changes of the flight direction, while considering the visibility of surfaces as shown in Figure 1.2 and, at the same time, managing the workload distribution amongst the participating drones to minimize the execution time.

1.1 Objectives and scope

The aim of this Master Thesis is to develop a system that is able to achieve visual coverage of previously unknown outdoor scenarios with a team of drones. To accomplish this goal, the following objectives are established:

1. First, a study of state-of-the-art methods for scene reconstruction and exploration with drones. We discuss their applicability to the task of efficient coverage of large-scale outdoor environments and the practicality for real-world deployments.

2. Design of a system that is able to obtain coverage of the scene. We consider real drones capabilities and propose a hierarchical strategy to solve the problem at hand.

3. The implementation of the proposed multi-stage strategy. Each of the stages is treated in a modular way, which allows to consider alternative approaches in each part. The system integrates newly proposed methods and leverages existing techniques to consolidate an end-to-end pipeline.

4. Finally, the evaluation of the proposed pipeline in simulated environments to asses the performance of the system and extent of obtained coverage. The coverage strategy is also compared with a state-of-the-art planner for exploration of unknown environments to showcase the advantages of the proposed method.

The Master Thesis has required the application of path planning and perception techniques together in a joint problem. The student has collaborated with researchers from the RoPeRT group in the Universidad de Zaragoza and the Vision for Robotics Lab (V4RL) in ETH Zürich with expertise in the two fields. This work has lead to the submission of a scientific article for publication to the IEEE Robotics and Automation Lettters with option to the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (included in Appendix X).

The remaining of this Thesis is structured as follows. First, a comprehensive frame of reference is explained for the unexperienced reader in Chapter 2. Then, a more precise literature review is offered in Chapter 3 where current state of research in visual coverage, exploration and reconstructions of scenes is discussed. The proposed pipeline, the theoretical methods and implementation details are explained in Chapter 4. In Chapter 5, the experiments to evaluate the performance of the pipeline for the task of visual coverage of unknown scenes is presented and the results are discussed. Finally, Chapter 6 offers a conclusion and proposes future directions for next work.

Chapter 2

Frame of Reference

2.1 Scene Reconstruction

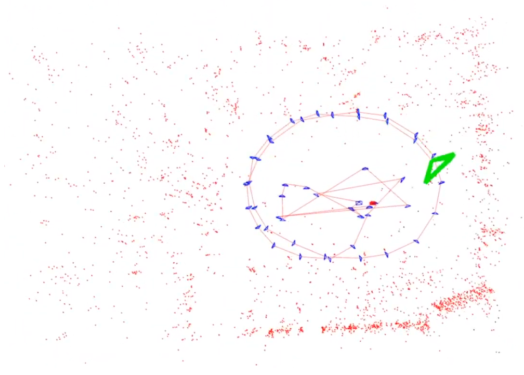
The use of robots for automated acquisition of knowledge about unknown environments is one of the most relevant topics in the roboitcs research community. As in the case of humans, vision sensors provide rich information about the world. Obtaining visual information of the scene can be the ultimate goal, as in photogrametry, or a tool to achieve other goals such as navigation, object search or assessing the structural integrity of a building. Depending on the task, different reconstructions modalities can be pursued:

– Sparse landmarks are represented as points in the space (Figure 2.1a) . They are used in traditional visual Simultaneous Localization and Mapping (SLAM) allow robots to localize themselves in the environment [3]. This reconstructions can be achieved with cameras applying classical Computer Vision methods able to work in real-time. They are used in Structure-from-motion (SfM) and SLAM pipelines [4]. Despite enabling navigation in the space, these reconstructions barely offer any information to reason over the scene.

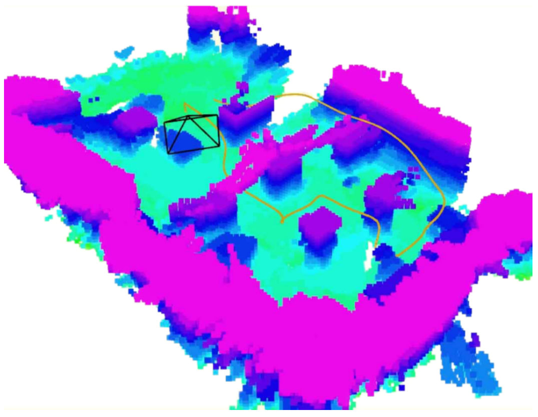
– Occupancy reconstructions are a good solution to improve the navigation through an environment [5] (Figure 2.1b). The space is discretized in 2D or 3D voxels and they are classified as unknown, free or occupied. As SfM and SLAM methods are only able to reconstruct the scene sparsely, alternative sensors such as RGB-D or stereo cameras are used to detect small obstacles. Even though the surfaces are detected as occupied voxels, their visibility is not considered (i.e., frontal observations), which does not ensure that high-level features such as textures or small details can be identified.

– Dense surface reconstructions offer the most meaningful information about the scene (Figure 2.1c). Traditionally, these reconstructions are obtained with classic Computer Vision methods such as Multi-View Stereo (MVS) [6]. Correct visibility of the surfaces (i.e., fronto-parallel views, enough parallax...) to reconstruct are a requirement of these methods. Inspired by the Graphics community, Signed Distance Field (SDF) can be built with stereo or RGB-D cameras and used to obtain precise real-time surface reconstructions with the Marching cubes method [7] (Figure 2.1d).

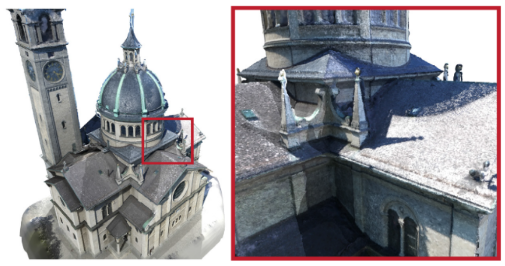
The aim of this thesis is to obtain a comprehensive visual reconstruction of the scene. Thus, we focus on methods that attempt to obtain dense surface reconstructions that enable to reconstruct accurate 3D models. In particular, our pipeline takes inspiration from MVS reconstructions methods to ensure visibility of the scene.



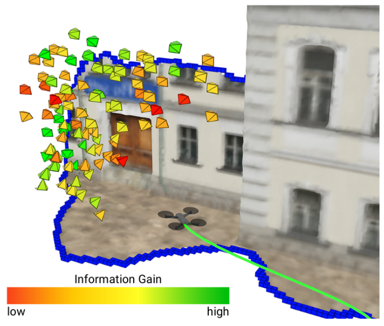
(a) Sparse landmark reconstruction [3]



(b) Occupancy reconstruction [8]



(c) MVS dense surface reconstruction [9]



(d) SDF based surface reconstruction [2]

Figure 2.1: Different map reconstructions. Sparse landmarks (a) and (b) occupancy maps are used for navigation but they are not appropriate to reason about surfaces. Dense reconstructions from MVS pipelines (c) and SDFs (d) include richer information about surfaces.

2.2 Map Representation

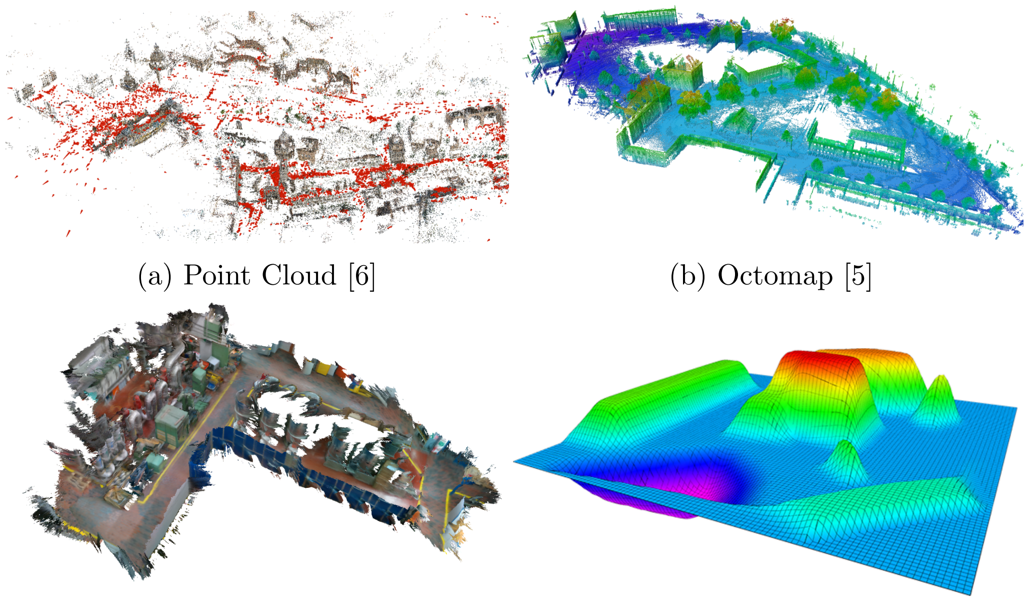
Depending on the task, different map representations might be used. The main considerations are: the meaning of the stored information and their computational efficiency.

- Point Clouds represent points in 3D space by its coordinates and parameters such as intensity or color (Figure 2.2a). The extensive research on this representation resulted in many out-of-the-shelf algorithms that ease the processing data, allowing to extract high-level features such as surfaces and their normals' or identifying objects. This representation is common in landmark reconstructions (SLAM). Point Clouds can quickly become too dense to be computationally efficient. KD-Trees are used for efficient operations on them such as neighbor search.

- Occupancy Maps divide the space in a grid and classify each cell as unknown, occupied or free space (Figure 2.2b). This representation is commonly used in 2D environments. The extension to 3D environments requires the use of hierarchical resolution to reduce the computational complexity such as octrees [5] or reducing the map to a local space around the robot [10].

- Signed Distance Fields (SDF) have recently been applied to robotics as the implicit representation of surfaces is suited for online reconstructions and planning. The two most common SDF are Euclidean SDF (ESDF) and Truncated SDF (TSDF). In both cases, the space is discretized and each cell stores the distance to the nearest object. The distance is positive outside of the object and negative inside. Surfaces are represented by the zero crosses in the map. TSDF differs from ESDF in that the distance measurements are truncated after some value. One of the established systems in the robotics research community is Voxblox [11] (Figure 2.2c). This methods iteratively builds a TSDF by integrating projective depth measurements along the ray between the sensor and the surface. In this case, each cell of the SDF also include a weight which represents the confidence about the depth measurement. The distances integrated in the TSDF are accurate near surface crossings, but accumulates large global errors. In order to obtain a global accurate representation of the space, the TSDF is processed to build a ESDF incrementally, which correctly represents the environment. In the same way of Occupancy Maps, the computational complexity of this representation also grows fast for 3D environments.

– Heightmaps are a great solution to reduce the computational complexity in very large-scale environments (Figure 2.2d). In this case, the environment is represented by a 2D grid where each cell stores its height. Additionally, other measurements such as a weight to represent the confidence of the height or its color can be stored. Thus, the 3D environment is reduced to a 2.5D representation. The main advantage is the lightweight representations of large-scale environments but their disadvantage is the loss of structural information in the vertical dimension of the environment by assuming that all the volume below the highest detected surface is occupied.


(a) Point Cloud [6]

(b) Octomap [5]

(c) Voxblox generated mesh [11]

(d) Heightmap [1]

Figure 2.2: Different map representations. Point Clouds (a) are sparse but there are many available processing tools. Octomap (b) is also useful for navigation but it does not account for the quality of represented surfaces. Voxblox (c) can generate accurate surfaces in real-time and includes implicit information about them which is useful to perform planning based on the surfaces in the map. Despite their simplicity, Heightmaps (d) can be a good solution to scale maps in a computationally efficient way. However, the structural information in the vertical direction is lost.

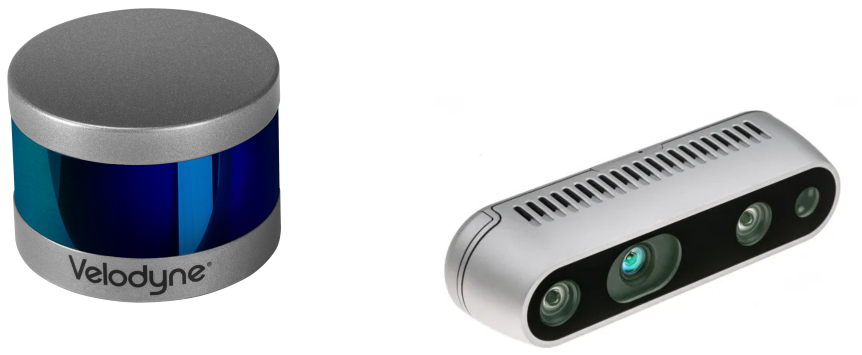
Throughout our system, different representations are used. Voxblox is used as the main representation for it can offer rich information to reason about the geometry of the scene, which renders the necessary observations to correctly cover the scene (Section 3.2). Additionally, Point Clouds are used to cluster the environment as they ease the processing thanks to already existent tools (Section 4.3). Heightmaps were considered also as an alternative representation to Voxblox to manage large-scale environments.

[1]`wiki.ros.org/grid_map`

2.3 Sensors for visual coverage

The most common and extended sensor to obtain visual information are monocular cameras. Their reduced size and low consumption make them suitable to be mounted on small drones. Despite being able to infer the geometric structure of the scene using Structure from Motion (SfM), a camera by itself cannot estimate the real-world scale of a scene [4].

Stereo cameras can obtain this measurement using a real-world prior: the distance between the two cameras (baseline). However, the depth that these cameras can estimate is limited by the parallax of the cameras, which increases with the baseline. The available baseline in small drones is short due to their limited size, limiting the sensing range of stereo cameras.



(a) Velodyne LiDAR [2]

(b) Intel Realsense D435i RGB-D camera [3]

Figure 2.3: Sensors used to reconstruct the real-scale geometry of a scene in order to reason over it using the time-of-flight of infrared light beams. RGB-D cameras generate a dense pointcloud but their limited range and low intensity make them unsuited for outdoor scenarios. LiDARs' measurements are also dense and in 3D but have high weight and consumption for small drones.

RGB-D cameras as the one shown in Figure 2.3b have additional sensors such as infrared to compute ground truth depth measurements to surfaces in the scene by using the time-of-flight of infrared light beams. These cameras work well in indoor environments or close to surfaces but they are affected by distance and outdoor light due to the low intensity of the infrared light compared to the sun and the reflections that different surfaces cause. LiDAR sensors also apply the idea of time-of-flight with higher intensity light beams, enabling outdoor use for distances up to 80 m. In order to obtain dense representations of the environment Velodyne LiDARs' models rotate

[1]`velodynelidar.com`

[1]`intelrealsense.com/depth-camera-d435i/`

a vertical array of 16 or 32 rays to generate a 3D point cloud. These sensors are used in combination with monocular cameras in autonomous driving applications. Small drones, however, have low autonomy due to their batteries and can only carry low payloads which also affect their autonomy. LiDARs have a great consumption and their weight is well above cameras. Furthermore, the measurements of LiDARs become more sparse with the distance, reducing the amount of information that can be extracted. These reasons were the motivation that inspired the appearance of Depth Completion techniques.

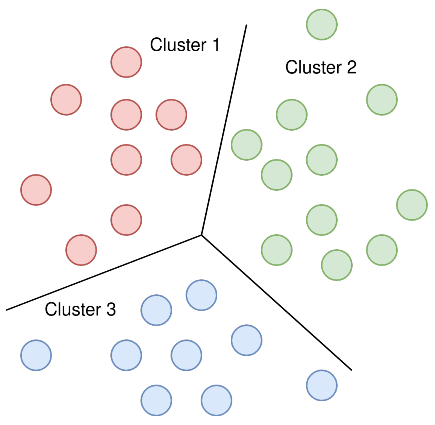
In our system we use an out-of-the-box Deep Learning Depth Completion method to create an initial rough estimate of the environment flying at high altitude [12] (Section 3.2). This method is able to estimate dense depth for all the pixels in a image providing an RGB image and sparse depth measurements. Additionally, we use a stereo camera to enable navigation (Section 4.5). Finally, as the aim of this thesis is the acquisition of visual coverage of a scene, the images from a monocular camera are considered as the last output of the system.

2.4 Clustering the space

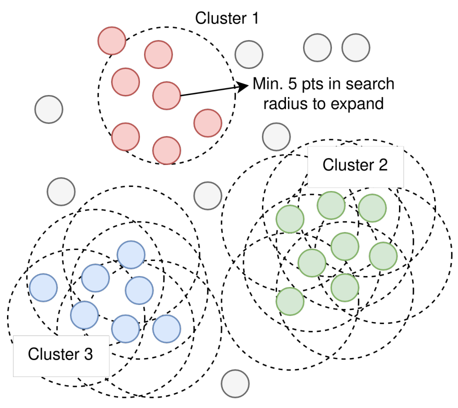
Clustering is the task of identifying groups and assigning elements to them. In this thesis we are interested in clustering regions of the scene to generate tasks that can be assigned to a team of drones. Two basic methods were considered and modified to fit our problem. The basic methods are introduced here and the modification explained in Section 4.3.

– K-Means classifies observations in a number of given clusters, k (Figure 2.4a). The elements are assigned to clusters according to some metric such as the Euclidean distance to the centroid of the clusters. The result are Voronoi cells that separate the space equally.

– Density-based spatial clustering of applications with noise (DBSCAN) [13] is a density-based clustering. It groups points that are near in space and classifies isolated points as noise (Figure 2.4b). This algorithms work by iteratively initializing new clusters and extending them to neighbours if they fulfil the density criteria (i.e., there are enough neighbors to the point to extend).



(a) KMeans



(b) DBSCAN

Figure 2.4: Classical clustering methods. KMeans (a) groups points in a pre-defined number of clusters using the distances between the points. DBScan (b) initializes clusters and expands them while the density criteria (i.e., min. number of points in search radius) is fulfilled for the candidate neighbors.

2.5 Vehicle Routing Problem

The Vehicle Routing Problem is a traditional planning method which arose from logistics and delivery. The problem is the routing of a fleet of one or more vehicles that have to visit a set of tasks only once, minimizing the total distance traversed by all the agents. The method models the problem of a set of tasks represented as nodes, $v \in \mathcal{V}$, in a graph, $\mathcal{G}$ that have to be visited by a number of agents, $k \in \mathcal{K}$. The nodes are connected by edges, $e \in \mathcal{E}$, with an associated weight c_{ij} with $i, j \in \mathcal{V}$ that represents the possible routes between nodes and the cost to traverse them. The solution is proposed by means of Integer Linear Programming (ILP). Define $\mathcal{X} = \{x_{ij}^k\}$, for $i, j \in V$, and $k \in K$, the set of binary variables that indicate whether agent k has traverse the route from i to j or not. The VRP solves:

$$\min_{\mathcal{X}} \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ij}^k, \quad s.t. \tag{2.1a}$$

$$\sum_{k\in K} \sum_{i\in V} x_{ij}^k = 1 \quad \forall j \in V \setminus \{0\} \tag{2.1b}$$

$$\sum_{k\in K} \sum_{j\in V} x_{ij}^k = 1 \quad \forall i \in V \setminus \{0\} \tag{2.1c}$$

$$\sum_{k\in K} \sum_{i\in V} x_{i0}^k = \sum_{j\in V} \sum_{k\in K} x_{0j}^k = |K| \tag{2.1d}$$

$$\sum_{i,j\in S} x_{i,j}^k \leq |S| - 1, \quad \forall S \subset V \setminus \{0\}, S \neq \emptyset \tag{2.1e}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i, j \in V \tag{2.1f}$$

where (2.1a) is the cost function, which denotes the sum of all the cost (i.e., traversed distances) among all the agents for a given assignment, constraints (2.1b) and (2.1c) indicate that drones only visit each location once. Constraints in (2.1d) impose the drones to start and end at the initial point. Constraints (2.1e) are the sub-tour elimination constraints that eliminate all solutions containing tours that return back to the start, without visiting all the tasks. Finally, conditions (2.1f) impose binary conditions on the decision variables. Figure 2.5 shows a toy example of this problem and a possible solution.

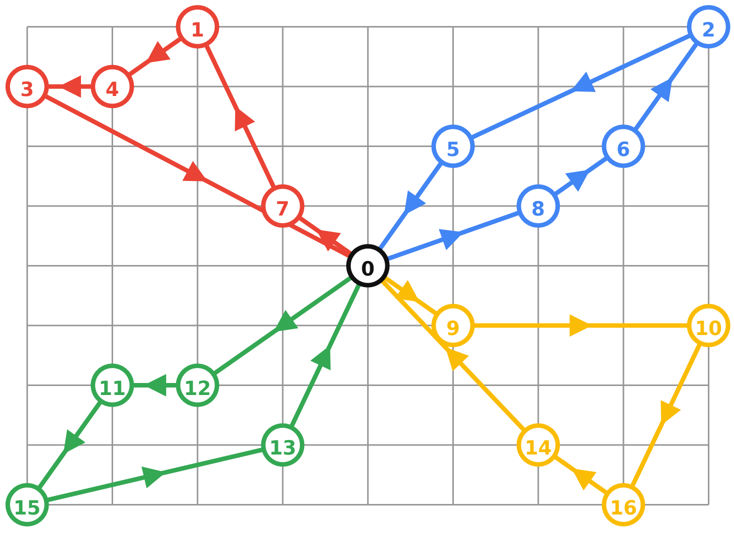


Figure 2.5: Example problem of a VRP problem with a possible solution. The distance traversed to visit all the nodes once and return back is minimize by using four agents .

In our problem, regions to observe in the scene are represented as tasks and assigned to the drones that will solve them by observing them in a defined trajectory. However, basic VRP only considers the minimization of the *total* distance traversed by all the drones. Depending on the structure of the scene and initial point (for example, use as depot node 15 in Figure 2.5), the distance might be minimized by only using one drone. In order to ensure balanced workload distribution among the available agents, we use a modified version, the min-max VRP explained in Section 4.4.

[3] `developers.google.com/optimization/routing/vrp`

Chapter 3

Related Work

Aerial planning for the best path in order to explore a scene has been a topic of extensive research in robotics and computer vision already due to its wide applicability.

3.1 Scene exploration and coverage

With the robotics literature mainly concentrating on online map construction from unknown or partially known, environments, the computer vision literature conversely focuses on highly accurate, but computationally demanding reconstructions. With the outlook of practicality, robotics approaches often focus on fast scene exploration, by eliminating the unknown space as quickly as possible. Frontier exploration methods look for regions, where free and unknown space meet [14]. The exploration is completed by identifying and pushing frontiers separating known and unknown space until all the space is explored. There are different criteria used to decide which frontier to explore next, such as their proximity to the current field of view [15], following a greedy selection strategy [16] or having global planning dictate their selection [1]. All these methods focus on volumetric representations of the map, whereas our approach considers surfaces and their visibility. Other works use Active SLAM in 2D environments for indoors ground robot navigation using landmarks [17, 18] or learning methods [19, 20]. In comparison, we consider aerial robots in 3D outdoor environments to obtain a comprehensive visual coverage.

When considering the reconstruction of surfaces of the scene, sampling-based approaches to address this by proposing a set of configurations (e.g. viewpoints) that get evaluated with respect to their expected information gain. For example, accurate surface reconstructions [21] can be achieved in a Next-Best View fashion [22]. In order to improve the efficiency of the planning, Rapidly-exploring Random Trees are a common approach [22, 21, 2]. To improve the sampling process, [2] applies informed sampling of configurations by reasoning over the available reconstructed model. The method in [23] considers voxels lying on the surface at a frontier, near both the unknown and free space. In general, all of these methods use depth cameras that allow for exploration or reconstruction in indoor and small scenarios. However, the performance in large-scale outdoor scenarios as considered in this work, decreases as the sensor range only allows for observations at a close distance. In [8], it is proposed to use online Multi-View Stereo (MVS) reconstruction in order to incrementally asses the surface reconstruction and plan iteratively in order to improve the reconstructed mesh. In comparison, the proposed approach executes a fast high-altitude reconnaissance flight to obtain a global coarse map as a prior, to provide an insight of the structure of the whole scene at once.

3.2 Use of a prior map

Other works used priors for improving the view selection for 3D reconstruction and generate a global plan. They analyse a prior map obtained from a previous flight in order to plan views that maximize heuristics for 3D reconstruction as parallax angle [24] or matchability [25]. These methods propose an initial distribution of views in the space and optimize them based on the aforementioned heuristics. In [24], the problem is addressed by using submodular optimization to improve the proposed views in the free space and obtain the final trajectory by solving an orienteering problem accounting for a maximum allowed time-budget. Submodular optimization is also used by [9] to plan views based on volumetric representations in a any-time optimization.

As discussed by [8], many of the previous methods obtain their prior from MVS pipelines, which is time consuming and might require long waiting times for processing, which is impractical for time-critical applications. In this work, we obtain a prior map online using depth completion, which allows us to extract good estimates of the views to reconstruct the scene. The work in [8] considers individual views without focusing on the trajectory to connect them, which might generate path redundancies and lower the efficiency of the global plan. In contrast, here we leverage the fact that many of these views can be grouped in a single efficient trajectory in order to cover large parts of the scene, e.g. building facades.

3.3 Multi-robot extension

All of the aforementioned methods assume a single robot. While they can be extended to multi-robot setups by partitioning the area of interest according to the number of robots, this does not ensure efficient enough collaboration between them.

Cooperative frontier based approaches have also been proposed in a centralized [26] and decentralized [27] way. These methods address the coordination problem in frontier based approaches, but suffer from the aforementioned locality problems and do not use global information until the scene is explored. Relevant is the work in [23], which is extended to the multi-robot case by greedily assigning the view configurations [28]. Another approach would be to perform partitioning of the area of interest, however, the complexity of the environment is not known a priori and the load balancing between the robots would not be accounted for. In this sense, [29] proposes continuous region partitioning based on Voronoi components for informative path planning. By considering the whole map and the set of regions to be covered (tasks) as a Vehicle Routing Problem (VRP), the generalization to multiple drones is straightforward in our pipeline, easily accounting for collaboration between them and minimizing the overall mission time.

Chapter 4

Method

Our goal is the efficient mapping of a bounded 3D outdoor space using a team of drones equipped with one monocular camera each. We achieve this by developing a system that computes smooth and straight flights for the drones to reduce the execution time of a mission. These trajectories are dubbed *sweeps*, as the maneuvers can be executed at higher speeds and do not require to change the flight direction.

In order to follow good practices in MVS reconstruction, we also search for trajectories that yield fronto-parallel views of the scene surfaces to maximize the scene coverage and quality of a posterior reconstruction.

4.1 System overview

Our planner is illustrated in Figure 4.2 and the results at different steps of the pipeline are shown in Figure 4.1. First, an initial down-looking (nadir) flight over the area is performed by the drones (Figure. 4.1a). The aim of this reconnaissance flight plan is twofold: to capture a large portion of the top view of the area of interest flying at high speeds, and to obtain a global overview of the scene online. This enables better informed reasoning over the subsequent drone trajectories to complete the coverage due to the detection of missing and poorly observed surfaces in the map (Figure. 4.1b). These surfaces are then grouped into clusters by a novel perception-aware clustering algorithm (Figure. 4.1c), favouring the generation of flights that sweep the scene to better capture these surfaces with efficient maneuvers (Figure. 4.1d). The next step computes global paths of all drones participating in the mission, aiming to minimize the distance travelled and the duration of the mission. This is achieved with a variation of the classical Vehicle Routing Problem (VRP), assigning surface-clusters to the drones (Figure. 4.1e). The processing of the initial map and the global plan are performed by a central server that integrates the measurements obtained in the initial reconnaissance flight. Finally, the flight-plans are assigned to the drones and a

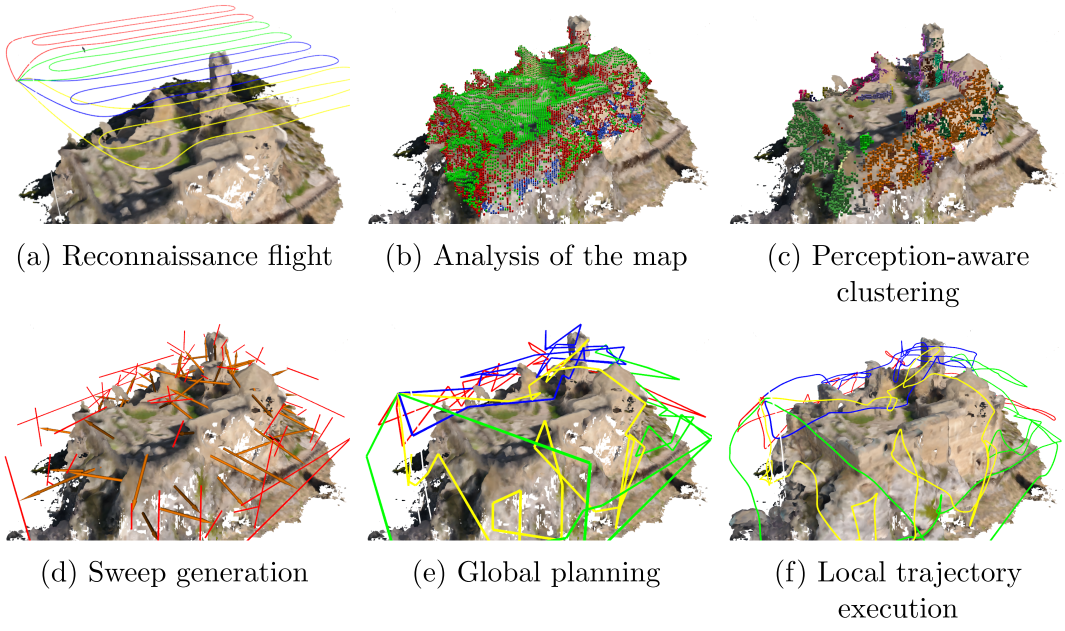

(a) Reconnaissance flight
(b) Analysis of the map
(c) Perception-aware clustering
(d) Sweep generation
(e) Global planning
(f) Local trajectory execution

Figure 4.1: The drones perform a down-looking flight to compute online a coarse initial map shown in (a), which is used to detect poorly observed or missing areas visualized in (b); red voxels correspond to surfaces seen from an oblique point of view (i.e., poorly observed) and blue voxels represent missing areas. Using perception-aware clustering these missing areas get clustered, shown in different colors in (c). The clusters are used to compute *sweeps*, visualized in (d), to observe them efficiently. The orange arrows represent the surface normals and red lines, the computed sweeps. The global paths of each drone are shown in (e), as computed by a VRP aiming to minimize the mission time and favour longer sweeps. These get smoothed out by a local planner to result in the final drone trajectories seen in (f).

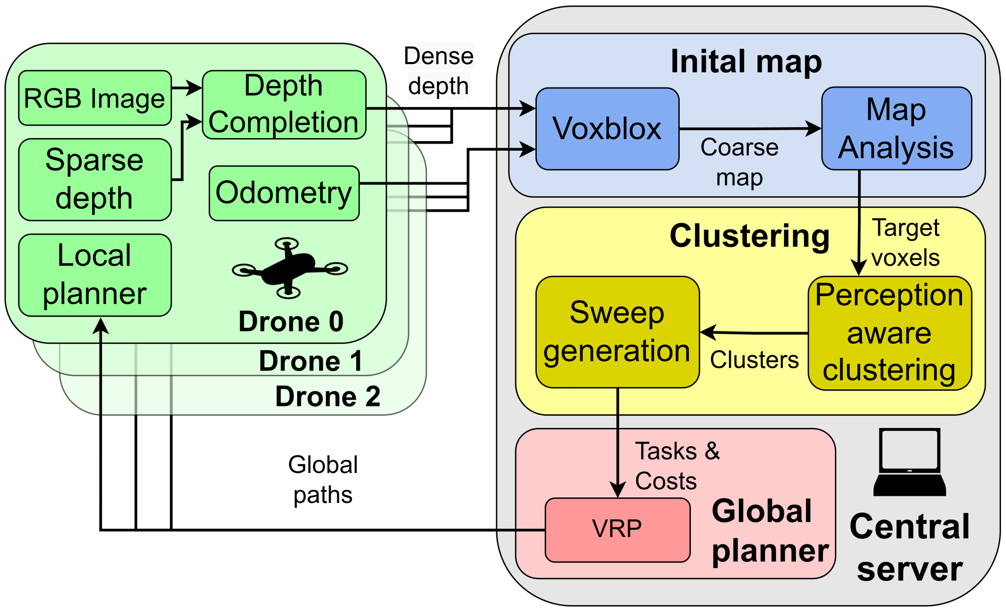


Figure 4.2: Proposed pipeline. The drones send measurements for the initial map integration to a central server. This processes the information to generate an efficient plan for the team of drones, which is communicated back to the drones.

trajectory planner guides the drones smoothly along the sweeps to obtain new relevant views of the scene (Fig. 4.1f). This execution is carried out without the need of exchanging information with the server or between the drones, favoring the deployment of small and low-powered platforms. In practice, one run of the pipeline is enough to cover most of the scene. Only complex concave surfaces, galleries and narrow passages could remain unexplored as they are not detected from the top of the scene. A possible way to explore them would be to integrate the local plans observations into Voxblox to repeat the process until the whole scene is covered.

4.2 Initial map

The reconnaissance flight captures top views of the scene to obtain a first approximation of the map quickly. However, the high altitude, together with the use of monocular cameras onboard the drones render the generation of this map challenging without the use of MVS expensive reconstruction methods. To compute it online, we use a depth completion system [12] onboard the drones that provides dense depth measurements from a sparse input, e.g., SLAM. [3].

We considered heightmaps and SDFs as the map representation. The first is more efficient due to the lower dimensionality. However, heightmaps cannot represent structural variations along the vertical direction. Its lower information would yield worse successive planning than using a 3D representation. SDFs are better for representing surfaces and can be used for planning in the successive flights. Increasing the voxel size for the SDFs proved to adapt the system to larger maps, improving the efficiency with similar results. The depth measurements from the sensor are integrated into a common voxel-based Truncated Signed Distance Field (TSDF) map, that incrementally builds a Euclidean Signed Distance Field (ESDF) map [11], $\mathcal{M}$ (Figure 4.3a). Voxels are organized in a uniform grid, where each voxel, $m \in \mathcal{M}$, contains a distance, d_m, to the closest surface and a weight, w_m, that contains the confidence about the depth measurement of that voxel. Moreover, we denote by $\mathbf{p}_m$ the centroid of the voxel and $\mathbf{n}_m$ its normal vector. Voxels that do not have any measurement have an associated weight equal to w_0.

The initial map is analysed in order to detect voxels that require additional observations. In particular, voxels that belong to a poorly observed surface, $\mathcal{M}_s$, and voxels without measurements (i.e., are unobserved), $\mathcal{M}_u$ (Figure 4.4 and Figure 4.3b).

Surfaces are identified locating the voxels that satisfy

$$w_m > w_0 \text{ and } |d_m| < d_v \ , \tag{4.1}$$

where d_v is the voxel size.

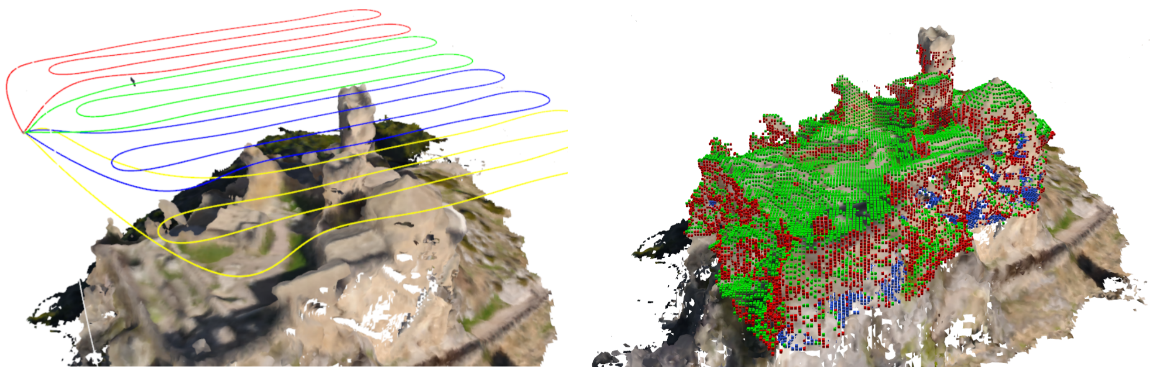

(a) Reconnaissance flight (b) Analysis of the map

Figure 4.3: Result obtained for the initial map and its analysis. During the reconnaissance flights, a depth completion system creates a Voxblox coarse reconstruction in real-time which is analyzed to detect well (green) and poorly (red) observed surfaces and accessible unorbserved areas (blue).

Aligning the sensor's depth direction with the surface normal, as shown in Figure 4.4, is key in enabling accurate and high-quality scene reconstructions. With this in mind, we identify poorly observed surface voxels, $\mathcal{M}_s$, as

$$-\,\mathbf{o}_m \cdot \mathbf{n}_m > \cos(\theta_t)\ , \tag{4.2}$$

where $\mathbf{o}_m$ is the observation direction of the camera for the voxel and θ_t is the threshold angle to consider the observation of the surface valid. We consider $\theta_t = 45°$ as a good indication that the visibility of a surface is poor. During the initial flight, the cameras are looking downward (i.e., $-Z$ axis). Thus, vertical and oblique surfaces are considered poorly observed, while horizontal or low tilted surfaces are considered as correctly observed.

The second step is the analysis of the unobserved voxels. Out of all the unobserved voxels in the map, with weight equal to w_0, we find those that are accessible (i.e., can be observed). Unobserved voxels are accessible if they are surrounded by free space voxels, m_f, defined by

$$w_m > w_0 \text{ and } d_m > d_v. \tag{4.3}$$

The accessible unobserved voxels, $\mathcal{M}_u$, are then formalized as the voxels, such that

$$\exists m_f \in \mathcal{N}_{26}(m), \tag{4.4}$$

where $\mathcal{N}_{26}(m)$ is the set of 26-connected neighbors, around the voxel m. Finally, the set of voxels that need further observations is defined as

$$\mathcal{M}_t = \mathcal{M}_s \cup \mathcal{M}_u\ . \tag{4.5}$$

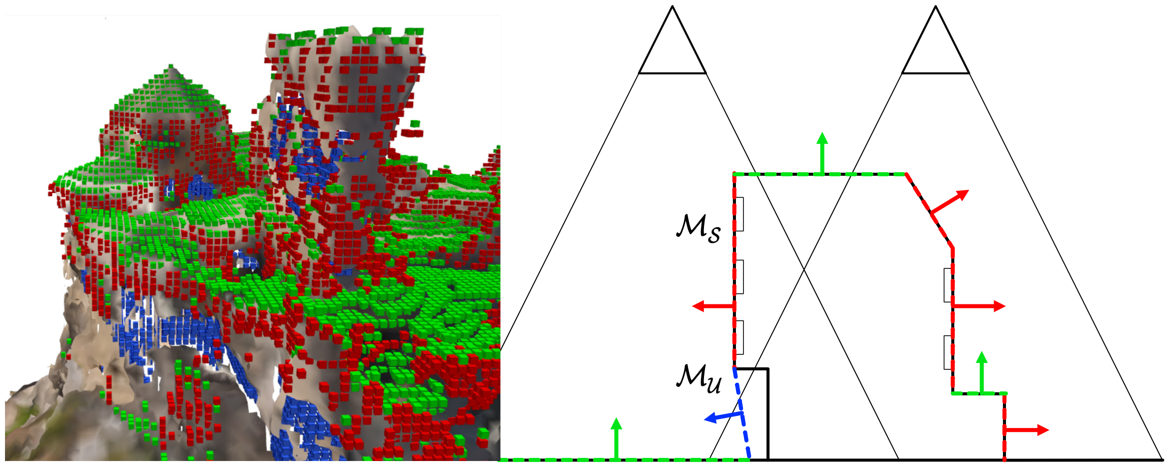


Figure 4.4: The analysis the initial map, visualised from a side view on the right with two down-looking cameras, indicates the quality of the views of the scene seen on the left, obtained in the reconnaissance flight. Voxels on the left are visualised as dashed lines on the right, with arrows indicating the estimated surface normals. Red and green indicate poorly and well captured surfaces, respectively, while blue indicates accessible unknown areas, whose normals are estimated to point towards free space.

Additional heuristics can be used to improve the reasoning over the map. Some of the implemented ideas that we did not consider in the final result reason over the weight value of the voxels. The weight of each voxel is increased with each depth measurement by a constant value or a weighted value inverse to the distance from the sensor, as the confidence of measured depth is reduced with respect to the squared of it distance. Thus, the weight would encode a measurement that informs of surfaces that were observed few times or from a large distance, which could also require further observation.

An alternative is to update the weight of the voxel depending on the observation angle with respect to the current estimated normal direction of the surface. This adds the criteria of perpendicular observations in the confidence of the map. These heuristics were not used because the value of the voxel weight is currently designed for the integration process. More research is required to study its applicability to infer statistical confidence metrics applicable to our case.

4.3 Perception-aware clustering

This step performs clustering over $\mathcal{M}_t$. Different cluster techniques were studied in order to split the working environment between the set of agents. Ideally, the segmentation would generate a balanced workload-distribution among the drones. First K-Means was studied as a naive segmentation. This technique only considers the distances in the map but ignores the structure of it. Thus, opposed surfaces are clustered together but they do not allow the generation of efficient trajectory planning. To improve the generation of meaningful tasks, K-Means was modified to not only account for the distances in position, but also the angle distance between the normals of the voxels with respect to the surface they represent. This worked well for small environments such as simple buildings, separating different facades and assigning them to different drones that can be covered more efficiently. However, in large-scale environments several facades will be assigned to each of the robots. Additionally, this method segments the environment in a fix number of tasks and pushes the problem of trajectory generation to a lower level in the planning stack. Figure 4.5 shows a comparison of the clustering alternatives.

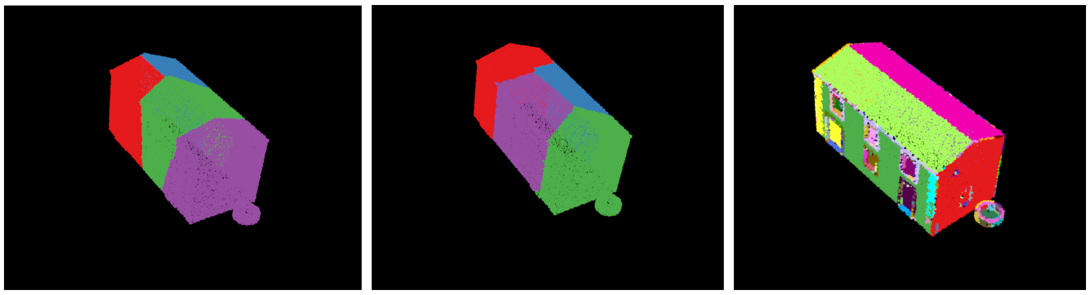

(a) KMeans (b) Normals aware KMeans (c) Perception-aware clustering

Figure 4.5: Alternative clustering methods applied to a ground truth model of a house. Basic KMeans (a) only considers distances and ignores the structure of the scene. Normals aware KMeans (b) consider the general structure but only works for basic structures as the house (notice the the structure is not considered in the well). The proposed Perception-aware clustering is able to separate surfaces depending on their proper observation direction. The edges produce individual clusters due to noisy normals. In our application, the small clusters coming from edges or noisy measurements are merged with the most similar cluster in their vicinity.

We found that considering the generation of efficient trajectories in the clustering offered a better environment segmentation and simplified successive planning steps to a task assignment problem. With this objective, we propose a perception-based clustering to group voxels together, such that can be observed by a drone in a single efficient *sweep* trajectory by considering the distribution of their normals in the cluster (Figure 4.6a). This clustering also aims at generating a natural partition of the scene into a set of *tasks* that can be assigned to a team of drones. In the following, we explain how the clustering works and how sweep paths are generated from them (Figure 4.6b).

The proposed clustering is based on the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method [13]. The basic method groups voxels[1]

[1]The original method refers to points.

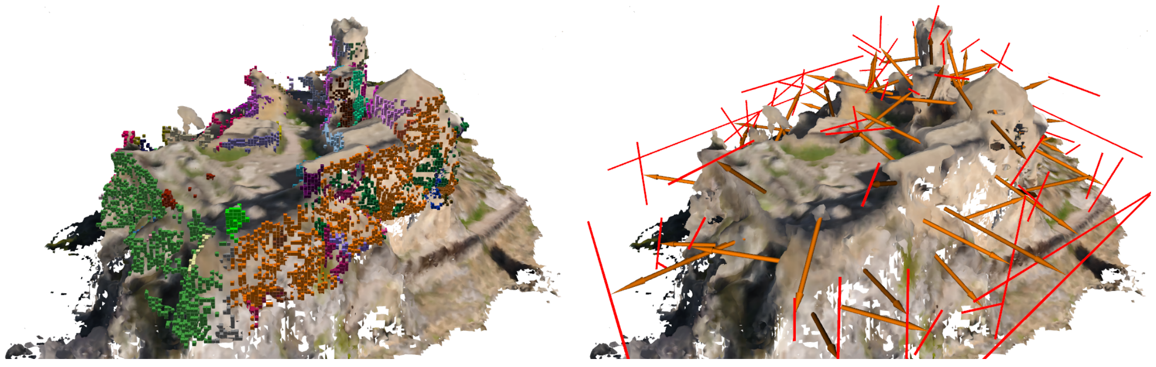

(a) Perception-aware clustering (b) Sweep generation

Figure 4.6: Result for the clustering and sweep generation steps. The voxels represented in (a) are the result of our Perception-aware cluster with each color representing a cluster. Except for the noise, the clustering models surfaces that can be observed in a similar direction. We show in (b) the *sweep* paths generated for each cluster to observe them in an efficient maneuver.

that are closely together in space and identifies as noise isolated voxels in low density regions. It works by iteratively expanding clusters, $\mathcal{C}_i$, to neighboring voxels that fulfill the following *density* condition:

$$|\mathcal{N}_\sigma(\mathbf{p}_m)| > \epsilon \ , \tag{4.6}$$

where $|\mathcal{N}_\sigma(\mathbf{p}_m)|$ is the number of neighboring voxels in a radius σ of the voxel's center, $\mathbf{p}_m$, and ϵ is the minimum number of neighbors to include the voxel in that cluster.

Our goal is to group regions observable from a similar point of view (i.e., surfaces). Thus, we extend DBSCAN by adding a second condition for expansion. This condition checks if the normal of a candidate voxel, $\mathbf{n}_m$, lies within the distribution of normals in the cluster. The normals in $\mathcal{M}_s$ are estimated from the gradient of distances in the ESDF initial map. The normals of unobserved voxels are computed as the average of all the directions that lead from $\mathbf{p}_m$ to free space voxels in $\mathcal{N}_{26}(m)$ (Figure 4.4). We also smooth the estimated normals using neighboring values to filter noise.

In particular, we focus on the distribution of the cosine distance with respect to the mean normal of the cluster, $\mathbf{n}_c$,

$$d_\alpha(\mathbf{n}_m, \mathbf{n}_c) = 1 - \frac{\mathbf{n}_m \cdot \mathbf{n}_c}{\|\mathbf{n}_m\| \, \|\mathbf{n}_c\|}. \tag{4.7}$$

We then compute the average $\mu_d(\mathcal{C}_i)$, and standard deviation $\sigma_d(\mathcal{C}_i)$ of the distances from all the normals of the voxels in the cluster to $\mathbf{n}_c$. The *normal direction* condition checks that the distance of the normal between the candidate voxel and the cluster's distribution is sufficiently small,

$$d_\alpha(\mathbf{n}_m, \mathbf{n}_c) < \min(\mu_d(C_i) + 2\sigma_d(C_i), \tau). \tag{4.8}$$

where τ is a fixed value.

We identify $\mu_d(\mathcal{C}_i)+2\sigma_d(\mathcal{C}_i)$ as the relative tolerance to the cluster's distribution and τ as the absolute tolerance. The aim of the relative tolerance is to adapt the expansion of the cluster to the surface in question, e.g., allowing soft curvatures. On the other hand, the absolute tolerance avoids the cluster to expand through discontinuities such as edges.

Finally, we perform a merging step that fuses small clusters with the most similar neighbor. If no neighbor is found, these voxels are discarded.

Considering that each voxel cluster resembles a surface, a *sweep* is defined as a linear trajectory that is orthogonal to the normal of the cluster (Figure 4.7). Among all the possible sweeps, we find the longest one through the inertia moments of the cluster, $\mathbf{l}_i$. Then, for each voxel in the cluster, we compute the longest distance from the center, projected on this axis,

$$d_i^* = \max_{m\in\mathcal{C}_i} \left|\mathbf{l}_i^T\left(\mathbf{p}_m - \bar{\mathbf{c}}_i\right)\right| \ , \tag{4.9}$$

where $\mathbf{p}_m$ is the centroid of the voxel and $\bar{\mathbf{c}}_i$ the centroid of the cluster. The extension of this distance from the centroid of the cluster in both directions of $\mathbf{l}_i$ generates the path that traverses the cluster through its length. We name both ends of this path, the *entrance* points of the cluster.

In order to guarantee that the whole surface is visible with a single sweep, we compute its height in the direction of the axis perpendicular to the sweep direction

$$\mathbf{h}_i = \mathbf{l}_i \times \mathbf{n}_i. \tag{4.10}$$

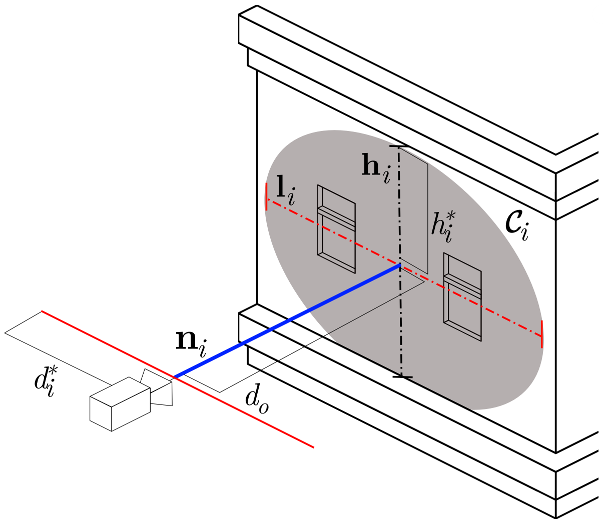
The value of the height is computed in the same way as (4.9) using the axis $\mathbf{h}_i$ instead:

$$h_i^* = \max_{m\in\mathcal{C}_i} \left|\mathbf{h}_i^T\left(\mathbf{p}_m - \bar{\mathbf{c}}_i\right)\right| \ , \tag{4.11}$$

where h_i^* if the half height of the cluster. Then, we use the relationship between the field of view (FoV) angle of the camera and h_i^* to compute the distance that is able to cover the height of the cluster. The observation distance, d_o, along the normal is computed as

$$d_o = \frac{h_i^*}{tan(\frac{\text{FoV}}{2})} \tag{4.12}$$

Finally, if the sweep intersects an obstacle we perform a rotation of the observation direction to refine it (Figure 4.7).



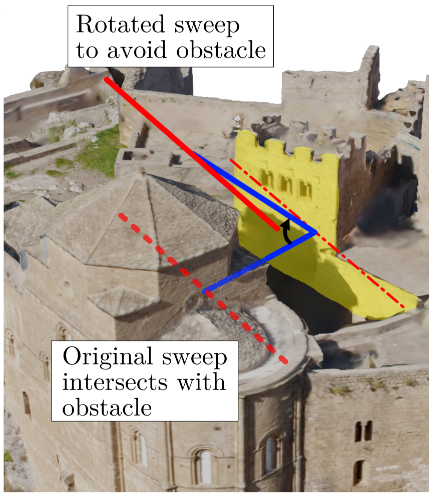

Rotated sweep to avoid obstacle
Original sweep intersects with obstacle

Figure 4.7: Sweep definition and refinement scheme (left). The gray area represents a surface cluster. The dashed red line is the major eigen vector that will be covered by the sweep (red solid line). The green is the normal. An example in a real map is shown on the right where the observation direction, $\mathbf{n}_i$, was adjusted to avoid an obstacle.

4.4 Global planner

In the next step, the objective is to compute high-level paths for the drones to cover all the clusters. We want these paths to minimize the total time of the mission and balance the workload (i.e., traversed distances) between the available drones. We propose to solve this problem with an adaptation of the min-max Vehicle Routing Problem (VRP).

Originally, this algorithm looks for optimal routes for a set of agents, K, that visit once all the locations of a given set, V. Denote by c_{ij} the cost to go from location i to location j, which we consider is the same for all the agents, and define $\mathcal{X} = \{x_{ij}^k\}$, for $i, j \in V$, and $k \in K$, the set of binary variables that indicate whether agent k has traverse the route from i to j or not. Then, the min-max VRP solves

$$\min_{\mathcal{X}} \max_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \quad s.t. \tag{4.13a}$$

$$\sum_{k \in K} \sum_{i \in V} x_{ij}^k = 1 \quad \forall j \in V \setminus \{0\} \tag{4.13b}$$

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \quad \forall i \in V \setminus \{0\} \tag{4.13c}$$

$$\sum_{k \in K} \sum_{i \in V} x_{i0}^k = \sum_{j \in V} \sum_{k \in K} x_{0j}^k = |K| \tag{4.13d}$$

$$\sum_{i,j \in S} x_{i,j}^k \leq |S| - 1, \quad \forall S \subset V \setminus \{0\}, S \neq \emptyset \tag{4.13e}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i, j \in V \tag{4.13f}$$

where (4.13a) is the cost function, which denotes the largest cost among all the agents for a given assignment, constraints (4.13b) and (4.13c) indicate that drones only visit each location once. Constraints in (4.13d) impose the drones to start and end at the initial point. Constraints (4.13e) are the sub-tour elimination constraints. Finally, conditions (4.13f) impose binary conditions on the decision variables.

In order to adapt the VRP to the clusters and their *sweeps*, we propose a definition of the costs, c_{ij}, that considers them. Given two clusters, i and j, we compute the path between them, as the line that join their closest entrance points with distance, d_{ij}, if there are no obstacles. In case there are obstacles, we consider the same path, but flying over the top of the scene. This way we guarantee that all the clusters are reachable from each other, but we favour assignments of the nearby ones. Additionally, to account for the cost of covering each cluster, we add the distance of the sweep to all the costs with it as destination. The distance of the sweep generated for $\mathcal{C}_i$ is $l_i^* = 2d_i^*$, with d_i^* defined in (4.9). Therefore, the cost c_{ij} is defined as

$$c_{ij} = d_{ij} + l_j^*. \tag{4.14}$$

Lastly, to compute the solution of (4.13), we consider an implementation with limited capacities. We simplify the objective to minimize the total cost travelled by all the drones

$$\min_{\mathcal{X}} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \tag{4.15}$$

and we add a capacity constraint for each of them,

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k < c_{\max} \quad \forall k \in K. \tag{4.16}$$

Our solution searches for the minimum value of $c^*_{\max}$ that solves the problem using the bisection method. An example of the obtained global plan is shown in Figure 4.8a.

4.5 Local planner

For the last step of the proposed pipeline, the local planner by Zhou et al. [30] is used to plan in two stages: an initial kinodynamic A* path search based on motion primitives finds a safe, feasible and minimum-time initial path, and a B-spline optimization generates smooth and collision-free trajectories that use gradient information from the ESDF and dynamic constraints.

The kinodynamic A* path search generates a graph by extending the state of the robots with motion primitives that respect the dynamic capabilities of the drone. This is, possible next states are generated by modifying the current state with different

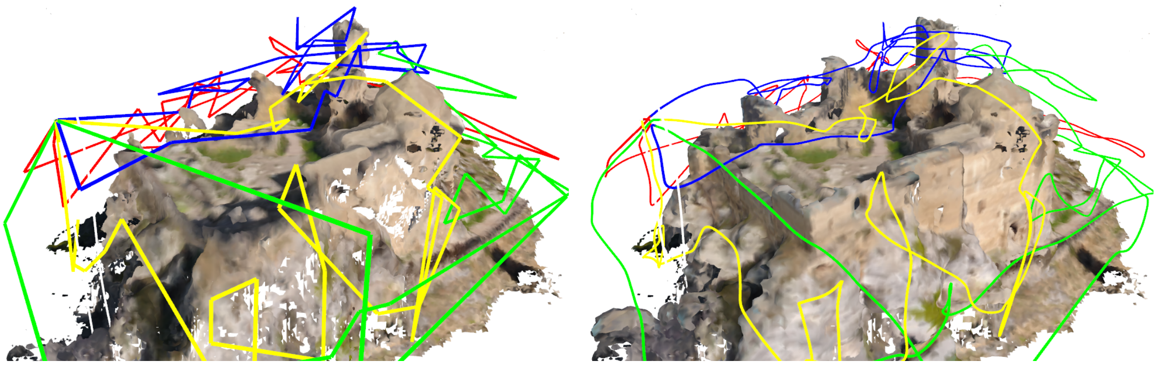

(a) Global plan (b) Local trajectories

Figure 4.8: Global plan and the trajectories executed by the drones. The min-max VRP solves the routing of a team of drones through the generated sweeps to minimize the total time of the mission and balance the distances between the drones (a). Then, a local planner in each drone computes the trajectories to execute the global path smoothly and avoiding collisions (b).

accelerations, modifying both the position and velocity. The possible accelerations are bounded by the maximum acceleration and velocity selected. Next states are evaluated a extended to reach the destination position. Once a path is generated, the intermediate states are introduced in a B-spline optimization setup that smooth the trajectory and corrects to avoid obstacles. This optimization leverages the ESDF representation of the environment, which offers a gradient along the normals of the surfaces to avoid obstacles and B-splines properties to bound the smooth trajectory within the convex hull of the control points. The resulting trajectory following the original global plan is shown in Figure 4.8b.

In order to cover a surface efficiently and effectively, the sweep direction needs to be orthogonal to the observation vector. To enable safe and efficient navigation, while obtaining high quality scene observations, we decouple the problems of navigation and observation. We assume that the observation camera is mounted on an actuated gimbal, which is able to set the yaw and pitch directions. A second sensor, such as a laser ranger or a depth camera is used for navigation.

Chapter 5

Experiments and results

To assess the performance of the proposed method, the pipeline is implemented in a simulation setup and run on photo-realistic outdoor scenarios of varying sizes and difficulty. We run the experiments considering three different algorithms. We name *Ours single* and *Ours multi* the solutions obtained running our pipeline with one and four drones respectively. In the *multi* version, we perform an ablation study to show the difference in visual coverage obtained after the reconnaissance flight and the successive flights resulting from our pipeline. Even when our pipeline is not directly comparable in terms of the sensor setup with other exploration methods that use stereo pairs, the third method uses the planning approach of Kompis et al. [2] for a single and multiple drones, which is among the state-of-the-art planners with available implementation. In the version with multiple drones, the environment was segmented equally among the drones. This comparison is not intended to rank the two methods but to showcase the potential advantages of the proposed planning approach in terms of efficiency. In the following, we explain the simulation setup and the results obtained in the execution.

5.1 Implementation details

The system was tested in a simulation setup for drones and photo-realistic scenarios that include the simulators, the planning methods and control algorithms. It was implemented in Ubuntu 18.04 using the Robotics Operating System (ROS) with Melodic version. It was also ported and tested on Ubuntu 20.04 with ROS Noetic applying changes to ensure libraries compatibility. ROS is based on a publisher-subscriber pattern with the concept of *nodes*. These are processes that run continuously and communicate through *topics* and *services* using different *message* types [1]. A summary of the nodes used in our system and the information flows is depicted in Figure 5.1

[1]`ros.org`

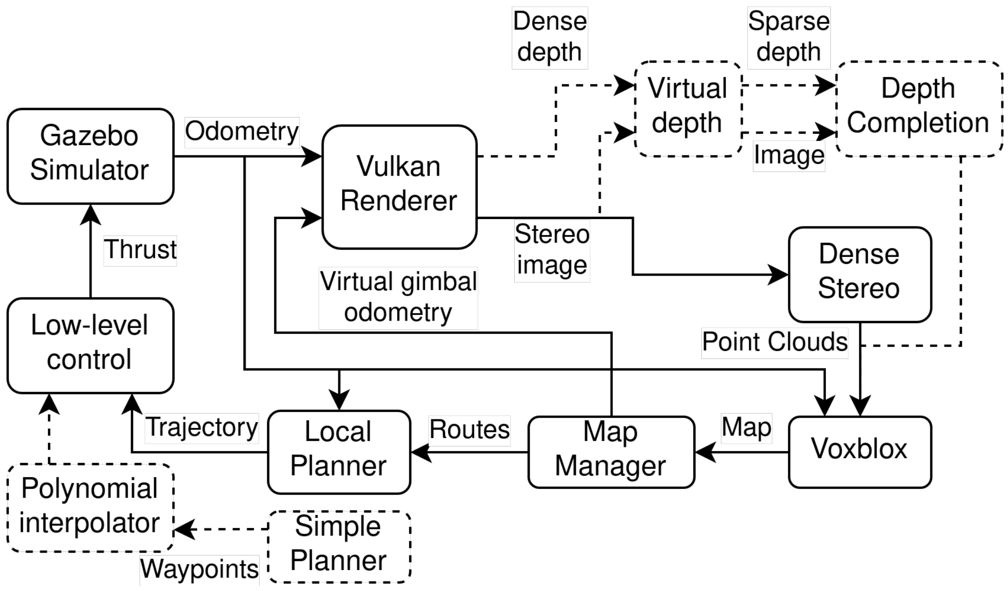


Figure 5.1: Software stack used for the implementation of the pipeline in a simulated environment. The boxes represent nodes in the ROS system and the arrows are the information they communicate. Dashed lines are components only used during the reconnaissance flights.

The Gazebo RotorS simulator is used with ground-truth odometry of the drones [31]. During the initial map construction, flying at high altitude enables the use of accurate RTK GPS systems with small odometry error. The uncertainty in the successive flights can be alleviated by overestimating the observation distance and safety radius. As we target our application to consumer platforms, problems such as aerodynamics or other electrical and mechanical delays are assumed to be solved by their system. The drones are equipped with a monocular camera mounted on an actuated gimbal that can rotate independently of the orientation of the drone. Its resolution is 752×480 and FoV is $80° \times 55°$. The drones' linear and angular maximum velocity and acceleration are set to 2 $\mathrm{m\,s^{-1}}$ and 0.9 $\mathrm{m\,s^{-2}}$, respectively, for fairness with the compared system and to ensure safety at all times. We use a Vulkan Renderer implemented by the V4RL to generate images of photo-realistic models more efficiently than Gazebo. The experiments showcase four different environments of varying sizes and difficulty, namely on the *Bunker*, *Wood Bridge*, *Loarre Castle*, and *Zurich*, visible in Figure 5.2. The transfer of this simulation setup to real-world cases was proved in previous work [12][32].

During the reconnaissance flight, the drones fly at a fixed height over the model in a grid pattern with their cameras looking downward. A Simple Planner was implemented to compute a grid path from the dimensions of the bounded volume to explore. The waypoints in the path are used to compute a continuous trajectory with polynomial interpolation. The trajectory commands are sent to the low-level

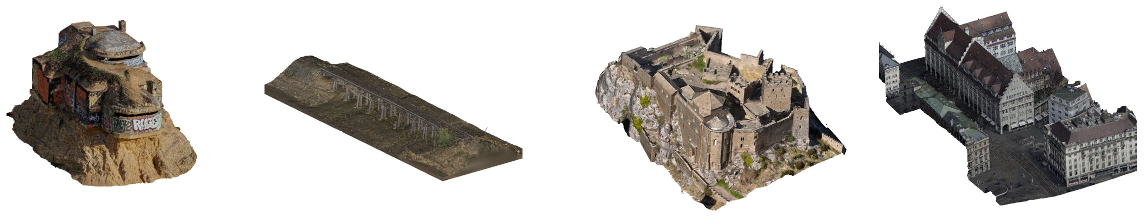

(a) *Bunker* v_s : 0.2
[20m × 20m × 11m]
(b) *Wood Bridge* v_s : 0.2
[7m × 30m × 7m]
(c) *Loarre castle* v_s : 0.3
[40m × 30m × 20m]
(d) *Zurich* v_s : 0.5
[55m × 55m × 30m]

Figure 5.2: The models used in the evaluation indicating their size and the voxel size used in our pipeline (v_s). The voxel size used for Kompis et al. [2] is set to 0.1 to obtain good reconstructions.

control. The controller is composed of: a non-linear Model Predictive Control (MPC) that optimizes the velocity command and a PID attitude controller which transforms the commands to thrust inputs for the drones [33]. The low-level controller is the same used during the successive flights.

From the visual simulator we obtain ground truth depth for the image which is sub-sampled to simulate the output of a SLAM system. The sparse depth and RGB image are introduced to the Depth Completion system implemented in Python using the Deep Learning library Pytorch [12]. The Depth Completion system generate back dense Point Clouds that is integrated into the Voxblox map. The voxel size used for the initial map and planning is 0.2 (*Bunker* and *Wood Bridge*), 0.5 (*Loarre*) and 0.7 (*Zurich*).

The map manager node represents the central server in Figure 4.2. It performs the analysis, clustering, sweep generation and global planning. The parameters for the clustering step depend on the resolution of the prior map (i.e., voxel size v_s). We set $\epsilon = 10v_s$, $\sigma = 6$ (Eq. (4.6)) and $\tau = 0.4$. We also apply an inflation factor over the coarse map of $20v_s$ to the observation and safety distance for the sweep generation.

The global paths are sent to the Local Planner that generates a smooth and obstacle-free trajectory and send the commands to the low-level controller. The Local Planner was modified from [30]. Originally, this system only planned the trajectory from one point to another. We let the Kinodynamic A* to continue the search to the next waypoint after reaching the current. Continuing the search allows to further smooth the path after a waypoint is reached. Due to computational resources required to simulate several drones, the local paths are executed by a single drone sequentially, which starts from and comes back to the same initial point. The simulation runs until all the local trajectories have been executed.

5.2 Planning efficiency

The times for the execution of the plan are shown in Table 5.1. The time for the reconnaissance flight and initial map construction with Voxblox is included in the total and shown below. For the method of Kompis et al., we report the times necessary to achieve the same coverage as our system.

Tabla 5.1: Execution times to complete a scene coverage mission. The reconnaissance flight time, in parenthesis, is included in the total time. *Ours single* refers to our pipeline using one drone, while *Ours multi* indicates the time taken by the longest flight of any drone in a team (four in this case), indicating the end of the mission. For [2], we report the time to reach the same extent of coverage achieved by each of our methods (Table 5.3). In lager maps, [2] is not able to achieve our coverage after one hour of execution and the total coverage by that time is reported. The '$\star$' indicates that the global planner only assigned two drones in this map, as introducing more would not reduce the total time.

Method	Bunker	Wood Bridge	Loarre Castle	Zurich
Kompis et al. [2]	859.71 s	897.08 s	>3600 s [67.08%]	>3600 s [13.91%]
Ours single	491.91 s (183.06 s)	331.16 s (122.54 s)	1474.95 s (329.88 s)	2027.88 s (588.56 s)
Kompis et al. multi [2]	214.43 s	405.98 s	1440.59 s	>3600 s [60.49%]
Ours multi	126.26 s (42.04 s)	172.09* s (53.31 s)	433.74 s (117.11 s)	741.07 s (269.53 s)

The results for *Ours single* and *Ours multi* validate that our setup can generalize to an arbitrary number of drones. When using several drones instead of one, times are a fraction of the number of drones with little overhead. In the case of *Wood bridge*, the global planner assigned the tasks to only two drones even if four were available. Due to the scene structure, adding more drones would not reduce the time of the mission as drones would have to return to the initial point. Compared to Kompis et al., our method is able to completely cover the maps faster in every case. For large maps (i.e., *Loarre Castle* and *Zurich*), [2] is not able to cover the environment after one hour of execution and we report the amount of coverage obtained at that time.

There are two main reasons for this difference. Firstly, the different approach on drone dynamics in the planners. Stop-and-go motions are necessary as the exploration process is incremental. This limits the planning horizon of the system to a local region. In their approach, the drone has to stop in order to acquire each individual view and plan the next (see Figure 5.3). In our case, the drone is able to keep moving while observing a whole surface in a sweep. Notice that our system could potentially use higher velocities and accelerations for large trajectories in free-space, as in the case of the reconnaissance flight. This would further improve the planning efficiency. The second reason is that their planner revisits areas in order to obtain thorough coverage, committing resources to small regions with difficult accessibility. The reason for their low coverage result in the *Zurich* map is explained by their viewpoint proposal method, which leads to larger re-planning times when the scale of the map grows.

We also report the time for the initial map processing. The time for the analysis and clustering steps depend on the size and resolution of the map. The global planning step depends on the number of generated clusters and number of agents. We show the results in Table 5.2 in the smallest and biggest maps: *Bunker* and *Zurich*. The time is always below one minute which is negligible for the total time of the mission.

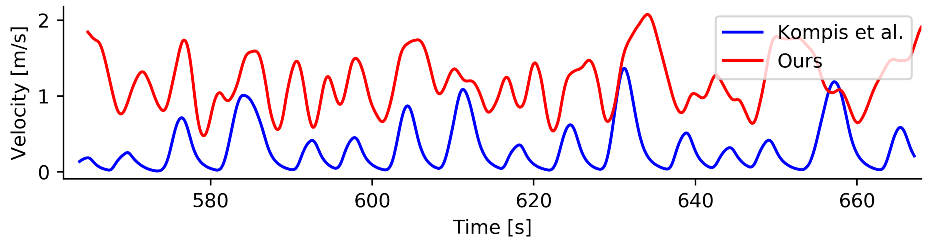


Figure 5.3: Extract of the moving average for the velocity during the simulation in *Loarre Castle* for *Ours single* and [2]. While traditional methods stop to capture a view and plan the next goal, our method is able to keep flying at higher speed.

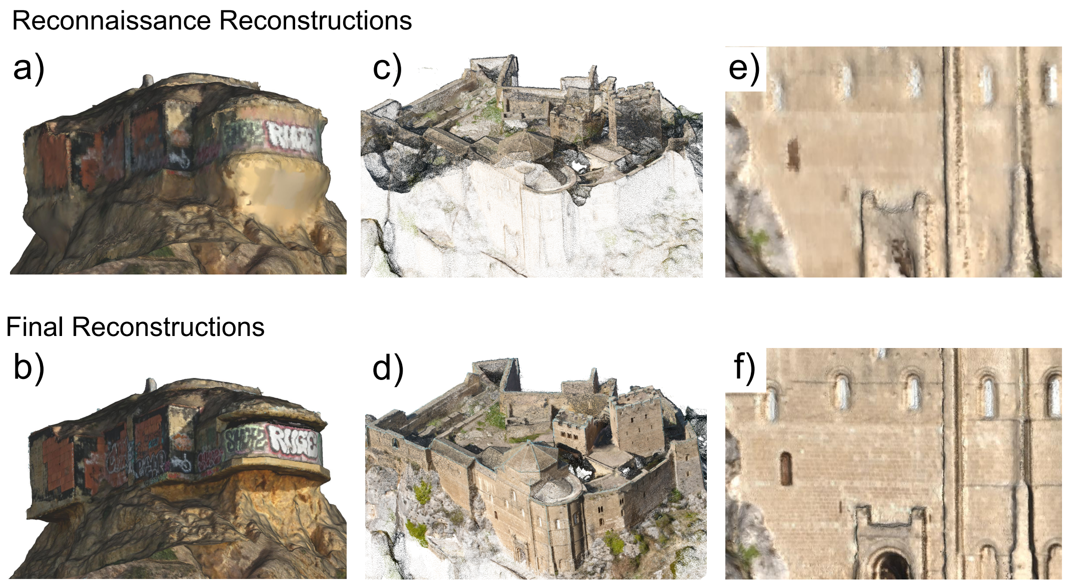


Figure 5.4: Comparison of the coverage quality after the reconnaissance flight (up) and the successive flights (down). Occluded regions under the *Bunker* are not reconstructed (a-b). In addition, even though vertical surfaces such as *Loarre's* walls are covered, their observation yields poor scene reconstructions (c-f).

Tabla 5.2: Computation time for the different map processing steps: analysis, clustering and global planner. Mean and standard deviation for 10 runs.

		Bunker	Zurich
Analysis	-	2.22 ± 0.082 s	6.96 ± 1.05 s
Clustering	-	1.83 ± 0.045 s	7.12 ± 0.297 s
Global planner	Single	0.99 ± 0.053 s	15.43 ± 0.689 s
	Multi	0.94 ± 0.064 s	17.10 ± 0.737 s

5.3 Coverage and surface quality

Besides the efficiency of our planner, we have also assessed that the coverage and the quality of the views are correct. The images captured from the monocular cameras of the drones have been used to generate 3D reconstructions of the scenes using COLMAP. The reconstructed models are compared with the ground-truth (GT) virtual models. We consider that a point in the GT surface was covered if the closest distance to a point from the reconstructed mesh is below a threshold of 0.1m. Our metric is the percentage of covered points in the ground-truth mesh. We also measure the accuracy of the reconstruction as the RMSE of the distances from the reconstructed model to the ground truth mesh. While we are mainly interested in the first two metrics, the accuracy proves that our method can be used to obtain accurate 3D reconstructions of the environment. We also report the coverage, its quality and the reconstruction accuracy from the Voxblox generated mesh of the pipeline in [2]. The voxel size for their reconstruction is the same they use in their experiments, 0.1, which is the threshold used for considering a point covered in our setup. The results are reported in Table 5.3. For Kompis et al., the reported value is the coverage achieved by the completion time of our plan.

The reconnaissance flight (*Recon.*) is able to cover large amount of surfaces. However, the coverage quality is low, yielding poor scene reconstructions (Figure 5.4). After the execution of our pipeline, we obtain images that ensure good observation of surfaces. We can see similar coverage for the case of single and multi-drone approaches as the drones traverse similar sweeps. Compared to Kompis et al., our system is able to achieve more coverage in less time. Notice how the coverage difference is increased with the size of the map. Qualitative results are shown in Figure 5.5 for all the maps. It might be seen that our pipeline misses some areas with difficult accessibility. In return, it is able to cover the overall scene in a fraction of the time. This demonstrates that lot of information can be extracted from the map by planning more efficiently and shows the advantage of using prior knowledge about the scene structure for planning.

Tabla 5.3: For each method, we report the RMSE of the reconstructions and the extent of the coverage for a threshold of 0.1 meters at the completion time of the experiment as reported in Table 5.1. Recon. indicates the metrics from a reconstruction with only the reconnaissance flight.

Method	Bunker	Wood Bridge	Loarre Castle	Zurich
Kompis et al.[2]	0.085 m 75.6 %	0.068 m 55.63 %	0.049 m 42.58 %	0.074 m 9.13 %
Ours single	0.027 m 97.35 %	0.043 m 93.23 %	0.048 m 97.92 %	0.09 m 95.96 %
Kompis et al. multi [2]	0.076 m 89.10 %	0.074 m 60.29 %	0.059 m 50.20 %	0.087 m 20.90 %
Recon. (ablation)	0.04 m 84.75 %	0.039 m 62.93 %	0.063 m 88.54 %	0.147 m 75.34 %
Ours multi	0.026 m 96.36 %	0.039 m 92.37 %	0.043 m 98.64 %	0.086 m 97.34 %

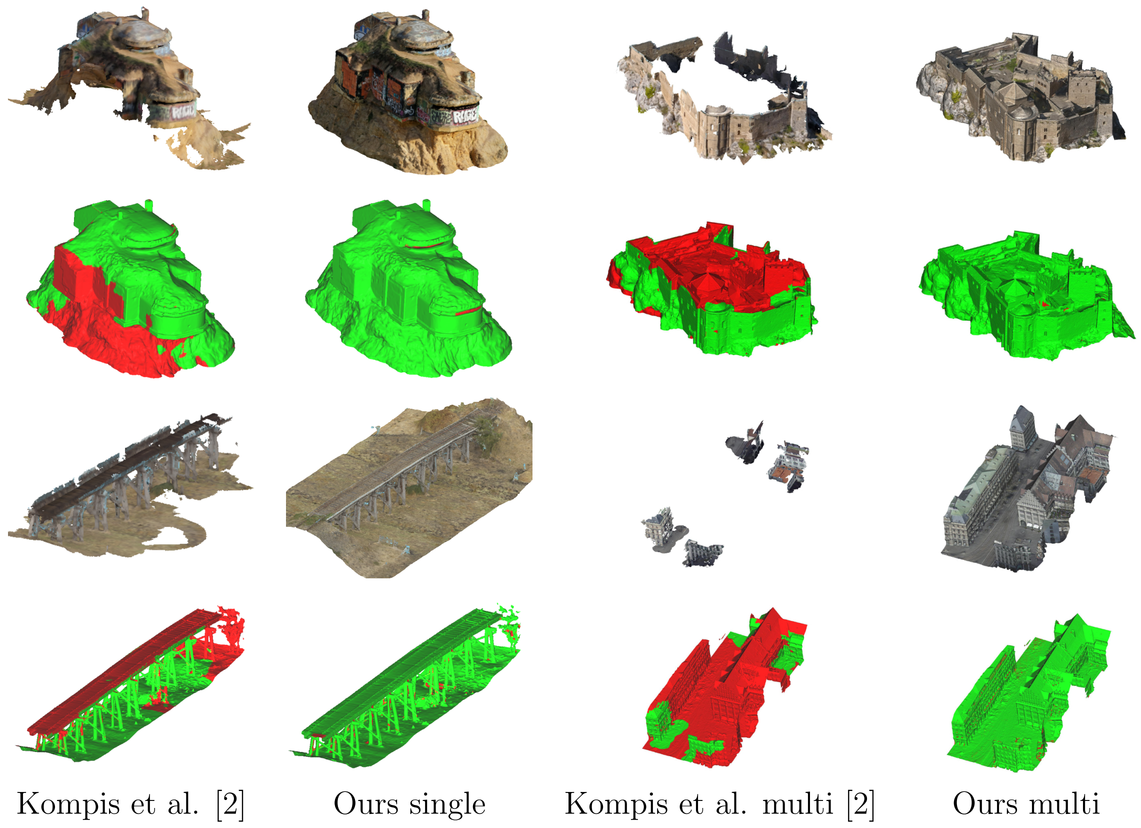


Figure 5.5: Qualitative comparison of the coverage obtained for all the maps considering a fixed time. In green are points in the ground truth (GT) mesh that have been covered during the mission, while red indicates the opposite. Our planner is able to obtain more coverage of the overall scene, despite missing some some details in inaccessible/non-directly visible surfaces. Detailed numbers of the coverage and accuracy of the reconstructions are provided in Table 5.3.

Chapter 6

Conclusion

In order to improve the efficiency in large-scale deployments of drones for visual coverage, this thesis proposes a multi-stage planner that generates long linear trajectories (*sweeps*) that observe a large amount of surface in a continuous motion. We accomplish this by leveraging a prior coarse map to cluster these surfaces and improve the posterior coverage trajectories. This approach is generalised to an arbitrary number of drones, managing the workload distribution between them in order to minimize the completion time of the mission. Comparison with alternative approaches to exploration of scenes show the advantages of our pipeline for large scenarios, where the overall coverage of the scene in a minimal amount of time is necessary. We show that a single run of our pipeline is able to obtain coverage of scenes faster and with great accuracy.

Future work will explore the integration of the proposed pipeline in a real platform, including a mapping framework to ensure safe local navigation and additional coordination systems to deploy a team of autonomous drones in large-scale environments. Besides, exploring the extension to a team of heterogeneous aerial drones (i.e., fixed-wing UAVs for the nadir flight) could improve even further the efficiency of the system by allocating each to different task modalities.

Bibliography

[1] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, 2021.

[2] Yves Kompis, Luca Bartolomei, Ruben Mascaro, Lucas Teixeira, and Margarita Chli. Informed sampling exploration path planner for 3d reconstruction of large scenes. *IEEE Robotics and Automation Letters*, 6:7893–7900, 2021.

[3] Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[4] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[5] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `https://octomap.github.io`.

[6] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016.

[7] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, aug 1987.

[8] Soohwan Song, Daekyum Kim, and Sunghee Choi. View path planning via online multiview stereo for 3-d modeling of large-scale structures. *IEEE Transactions on Robotics*, 38(1):372–390, 2022.

[9] Benjamin Hepp, Matthias Nießner, and Otmar Hilliges. Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction. *ACM Transactions on Graphics*, 38(1), dec 2018.

[10] Vladyslav Usenko, Lukas von Stumberg, Andrej Pangercic, and Daniel Cremers. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222, 2017.

[11] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[12] Lucas Teixeira, Martin R. Oswald, Marc Pollefeys, and Margarita Chli. Aerial single-view depth completion with image-guided uncertainty estimation. *IEEE Robotics and Automation Letters*, 5(2):1055–1062, 2020.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, page 226–231, 1996.

[14] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, page 146, 1997.

[15] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *International Conference on Intelligent Robots and Systems*, pages 2135–2142, 2017.

[16] Daniel Duberg and Patric Jensfelt. Ufoexplorer: Fast and scalable sampling-based exploration with a graph-based planning structure. *IEEE Robotics and Automation Letters*, 7(2):2487–2494, 2022.

[17] Henry Carrillo, Ian Reid, and José A Castellanos. On the comparison of uncertainty criteria for active slam. In *IEEE International Conference on Robotics and Automation*, pages 2080–2087, 2012.

[18] Luca Carlone, Jingjing Du, Miguel Kaouk Ng, Basilio Bona, and Marina Indri. Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311, 2014.

[19] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *IEEE International Conference on Learning Representations (ICLR)*, 2020.

[20] Georgios Georgakis, Bernadette Bucher, Anton Arapin, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. Uncertainty-driven planner for exploration and navigation. In *IEEE International Conference on Robotics and Automation*, 2022.

[21] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[22] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon "next-best-view" planner for 3d exploration. In *IEEE International Conference on Robotics and Automation*, pages 1462–1468, 2016.

[23] Guillaume Hardouin, Fabio Morbidi, Julien Moras, Julien Marzat, and El Mustapha Mouaddib. Surface-driven Next-Best-View planning for exploration of large-scale 3D environments. In *IFAC World Congress*, July 2020.

[24] Mike Roberts, Debadeepta Dey, Anh Truong, Sudipta Sinha, Shital Shah, Ashish Kapoor, Pat Hanrahan, and Neel Joshi. Submodular trajectory optimization for aerial 3d scanning. In *International Conference on Computer Vision*, 2017.

[25] Neil Smith, Nils Moehrle, Michael Goesele, and Wolfgang Heidrich. Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark. *ACM Trans. Graph.*, 37(6), dec 2018.

[26] Anna Mannucci, Simone Nardi, and Lucia Pallottino. Autonomous 3d exploration of large areas: A cooperative frontier-based approach. In Jan Mazal, editor, *Modelling and Simulation for Autonomous Systems*, pages 18–39, Cham, 2018.

[27] Rafael Gonçalves Colares and Luiz Chaimowicz. The next frontier: Combining information gain and distance cost for decentralized multi-robot exploration. In *ACM Symposium on Applied Computing*, page 268–274, 2016.

[28] Guillaume Hardouin, Julien Moras, Fabio Morbidi, Julien Marzat, and El Mustapha Mouaddib. Next-best-view planning for surface reconstruction of large-scale 3d environments with multiple uavs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1567–1574, 2020.

[29] Ayan Dutta, Amitabh Bhattacharya, O Patrick Kreidl, Anirban Ghosh, and Prithviraj Dasgupta. Multi-robot informative path planning in unknown environments through continuous region partitioning. *International Journal of Advanced Robotic Systems*, 17(6):1729881420970461, 2020.

[30] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019.

[31] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[32] Fabiola Maffra, Lucas Teixeira, Zetao Chen, and Margarita Chli. Real-time wide-baseline place recognition using depth completion. *IEEE Robotics and Automation Letters*, 4(2):1525–1532, 2019.

[33] Mina Kamel, Thomas Stastny, Kostas Alexis, and Roland Siegwart. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In Anis Koubaa, editor, *Robot Operating System (ROS) The Complete Reference, Volume 2*. Springer, 2016.

List of Figures

List of Tables

Appendix A

Article

Sweep-Your-Map: Efficient Coverage Planning for Aerial Teams in Large-Scale Environments

David Morilla-Cabello, Luca Bartolomei, Lucas Teixeira, Eduardo Montijano, and Margarita Chli

Abstract*— The efficiency of path-planning in robot navigation is crucial in tasks such as search-and-rescue and disaster surveying, but this is emphasised even more when considering multi-rotor aerial robots due to the limited battery and flight time. In this spirit, this work proposes an efficient, hierarchical planner to achieve a comprehensive visual coverage of large-scale outdoor scenarios for small drones. Following an initial reconnaissance flight, a coarse map of the scene gets built in real-time. Then, regions of the map that were not appropriately observed are identified and grouped by a novel perception-aware clustering process that enables the generation of continuous trajectories (*sweeps*) to cover them efficiently. Thanks to this partitioning of the map in a set of tasks, we are able to generalize the planning to an arbitrary number of drones and perform a well-balanced workload distribution among them. We compare our approach to an alternative state-of-the-art method for exploration and show the advantages of our pipeline in terms of efficiency for obtaining coverage in large environments.
***Video* – `https://youtu.be/V2UIrM91oQ8`**

I. Introduction

Recent advances in robot navigation and perception have enabled the establishment of modern multi-rotor aircraft, i.e., drones, as the best choice for autonomous 3D reconstruction or visual coverage of large-scale outdoor scenarios. Their flexibility allows them to move freely through the environment and observe areas that are not visible from the ground. However, time efficiency is critical for using drones because of their short flight times (due to battery limitations), usually well under 30 minutes. Therefore, the efficiency and effectiveness of the planning algorithms is essential to enable the deployment of drones in large scale outdoor environments. Similarly, using multiple drones as advocated in this work promises to boost the efficiency of the scene-coverage mission.

Deploying drones for mapping a large area from a high altitude is an effective way to obtain a first estimation, as collisions with the environment can be more easily avoided. However, this strategy does not provide informative enough viewpoints for scene coverage and impacts the quality of the scene captures. State-of-the-art exploration approaches [1], [2] often lack in efficiency because of problems such as over-exploring local regions, and abrupt changes in motion due to constant re-planning or the need for revisiting areas.

David Morilla-Cabello and Eduardo Montijano are with the Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Spain `{davidmc, emonti}@unizar.es`

Luca Bartomei, Lucas Teixeira and Margarita Chli are with the Vision for Robotics Lab, Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zurich, Switzerland `{lbartolomei, lteixeira, chlim}@ethz.ch`

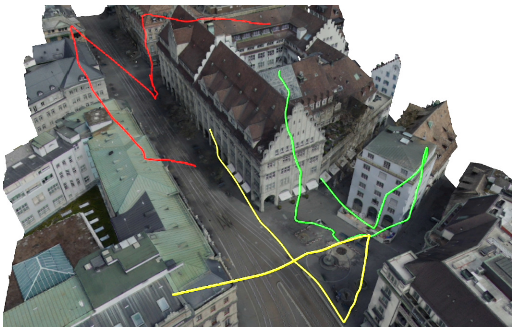


Fig. 1: Team of drones that *sweep* the area of interest by flying paths generated by the proposed planner in order to achieve fast coverage. Using a rough prior map (e.g. captured in a reconnaissance flight) to identify areas that require further observation, this work generates efficient path planning and workload distribution for a team of drones (three in this example) to cover the scene.

To overcome these limitations, this paper presents a hybrid solution that uses the best of both types of strategy in a synergetic way. In this work, we assume a team of drones with cameras, each performing a fast, reconnaissance flight at a high-altitude capturing a rough map of the area of interest using a coarse real-time mapping pipeline. Based on this map, the proposed method computes a set of drone trajectories for subsequent flights in order to efficiently cover the area of interest in completely. This process aims to maximize the use of *sweep* lines to avoid constant changes of the flight direction, while considering the visibility of surfaces and, at the same time, managing the workload distribution amongst the participating drones to minimize the execution time. The main contribution of this paper is the overall perception-aware global planning that is capable of handling the initial, noisy and coarse map as well as enforcing high-speed trajectories.

II. Related Work

Aerial planning for the best path in order to explore a scene has been a topic of extensive research in robotics and computer vision already due to its wide applicability.

A. *Scene exploration and coverage*

With the outlook of practicality, robotics approaches often focus on fast scene exploration, by eliminating the unknown space as quickly as possible. Frontier exploration methods look for regions, where free and unknown space meet [3]. There are different criteria used to decide which frontier to explore next, such as their proximity to the current field of view [4], following a greedy selection strategy [5] or having global planning dictate their selection [1]. All these methods focus on volumetric representations of the map, whereas our approach considers surfaces and their visibility.

Other works use Active SLAM in 2D environments for indoors ground robot navigation using landmarks [6], [7] or learning methods [8], [9]. In comparison, we consider aerial robots in 3D outdoor environments to obtain a comprehensive visual coverage.

When considering the reconstruction of surfaces, sampling-based approaches propose viewpoints based on their expected information gain. For example, accurate surface reconstructions [10] can be achieved in a Next-Best View fashion [11]. In order to improve the efficiency of the planning, Rapidly-exploring Random Trees are a common approach [10], [11]. To improve the sampling process, [2] applies informed sampling of configurations by reasoning over the available reconstructed model. The method in [12] considers voxels lying on the surface at a frontier. In general, all of these methods use depth cameras that allow for exploration or reconstruction in indoor and small scenarios. The performance in large-scale outdoor scenarios as considered in this work, decreases as the sensor range only allows for close observations. In [13], online Multi-View Stereo (MVS) is used to incrementally asses the surface reconstruction. In comparison, the proposed approach executes a fast high-altitude reconnaissance flight to obtain a global coarse map as a prior and provide an insight of the structure of the whole scene at once.

B. Use of a prior map

Other works used priors for improving the view selection for 3D reconstruction and generate a global plan. They analyse a prior map obtained from a previous flight in order to plan views that maximize heuristics for 3D reconstruction as parallax angle [14] or matchability [15]. In [14], the problem is addressed by using submodular optimization to improve the proposed views in the free space and obtain the final trajectory by solving an orienteering problem accounting for a maximum allowed time-budget. Submodular optimization is also used by [16] to plan views based on volumetric representations in a any-time optimization.

As discussed by [13], many of the previous methods obtain their prior from MVS pipelines, which is time consuming and might require long waiting times for processing. In this work, we obtain a prior map online using depth completion to extract good estimates of the views to reconstruct the scene. The work in [13] considers individual views without focusing on the trajectory to connect them, which might generate path redundancies. In contrast, we leverage the fact that many of these views can be grouped in a single efficient trajectory in order to cover large parts of the scene, e.g., building facades.

C. Multi-robot extension

All of the aforementioned methods assume a single robot. While they can be extended to multi-robot setups by partitioning the area of interest according to the number of robots, this does not ensure efficient enough collaboration between them. Cooperative frontier based approaches have also been proposed in a centralized [17] and decentralized [18] way. These methods address the coordination problem in frontier based approaches, but suffer from the aforementioned locality problems. The work in [12] extends to the multi-robot case by greedily assigning the view configurations [19]. The work in [20] distributes the workload through continuous region partitioning based on Voronoi components. By considering the whole map and the set of regions to be covered (tasks) as a Vehicle Routing Problem (VRP), the generalization to multiple drones is straightforward in our pipeline, easily accounting for collaboration between them and minimizing the overall mission time.

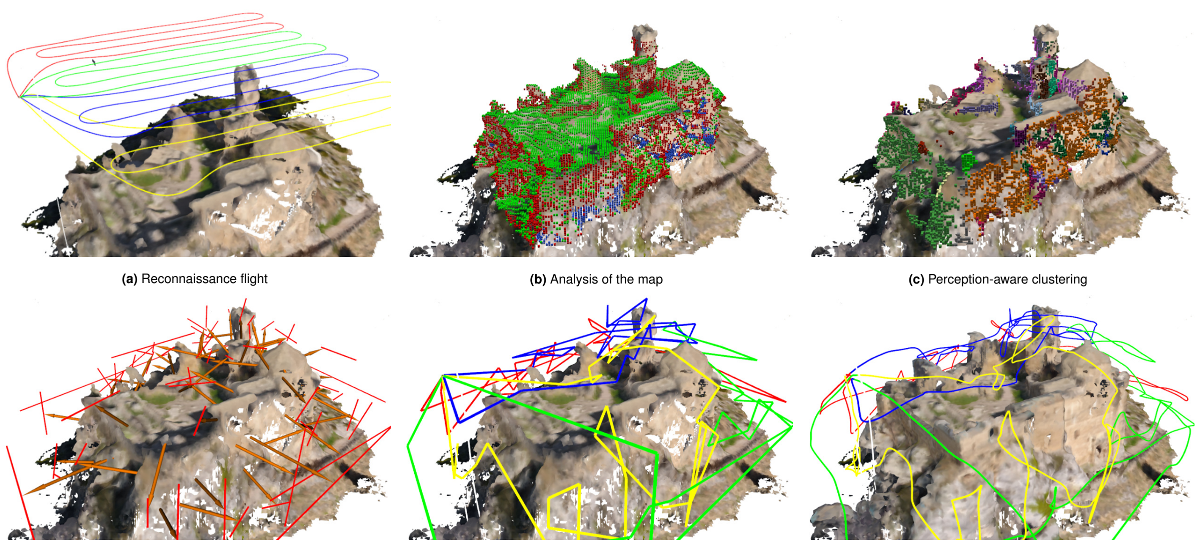
III. Method

Our goal is the efficient mapping of a bounded 3D outdoor space using a team of drones equipped with one monocular camera each. We achieve this by developing a system that computes smooth and straight flights for the drones to reduce the execution time of a mission. These trajectories are dubbed *sweeps*, as the maneuvers can be executed at higher speeds and do not require to change the flight direction.

In order to follow good practices in MVS reconstruction, we also search for trajectories that yield fronto-parallel views of the scene surfaces to maximize the scene coverage and quality of a posterior reconstruction.

A. System overview

Our planner is illustrated in Figure 3 and the results at different steps of the pipeline are shown in Figure 2. First, an initial down-looking (nadir) flight over the area is performed by the drones (Figure. 2a). The aim of this reconnaissance flight plan is twofold: to capture a large portion of the top view of the area of interest flying at high speeds, and to obtain a global overview of the scene online. This enables better informed reasoning over the subsequent drone trajectories to complete the coverage due to the detection of missing and poorly observed surfaces in the map (Figure. 2b). These surfaces are then grouped into clusters by a novel perception-aware clustering algorithm (Figure. 2c), favouring the generation of flights that sweep the scene to better capture these surfaces with efficient maneuvers (Figure. 2d). The next step computes global paths of all drones participating in the mission, aiming to minimize the distance travelled and the duration of the mission. This is achieved with a variation of the classical Vehicle Routing Problem (VRP), assigning surface-clusters to the drones (Figure. 2e). The processing of the initial map and the global plan are performed by a central server that integrates the measurements obtained in the initial reconnaissance flight. Finally, the flight-plans are assigned to the drones and a trajectory planner guides the drones smoothly along the sweeps to obtain new relevant views of the scene (Fig. 2f). This execution is carried out without the need of exchanging information with the server or between the drones, favoring the deployment of small and low-powered platforms. In practice, one run of the pipeline is enough to cover most of the scene. Only complex concave surfaces, galleries and narrow passages could remain



(a) Reconnaissance flight **(b)** Analysis of the map **(c)** Perception-aware clustering

(d) Sweep generation **(e)** Global planning **(f)** Local trajectory execution

Fig. 2: The drones perform a down-looking flight to compute online a coarse initial map shown in (a), which is used to detect poorly observed or missing areas visualized in (b); red voxels correspond to surfaces seen from an oblique point of view (i.e., poorly observed) and blue voxels represent missing areas. Using perception-aware clustering these missing areas get clustered, shown in different colors in (c). The clusters are used to compute *sweeps*, visualized in (d), to observe them efficiently. The orange arrows represent the surface normals and red lines, the computed sweeps. The global paths of each drone are shown in (e), as computed by a VRP aiming to minimize the mission time and favour longer sweeps. These get smoothed out by a local planner to result in the final drone trajectories seen in (f).

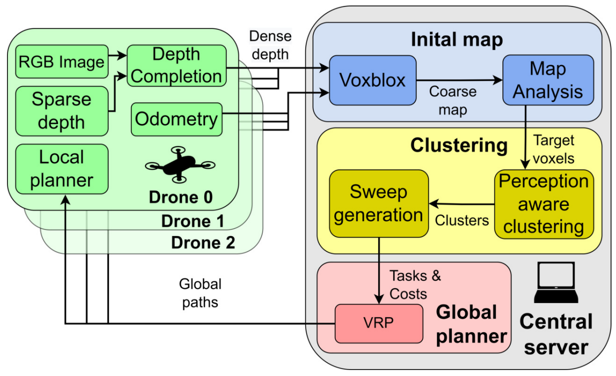


Fig. 3: Proposed pipeline. The drones send measurements for the initial map integration to a central server. This processes the information to generate an efficient plan for the team of drones, which is communicated back to the drones.

unexplored as they are not detected from the top of the scene. A possible way to explore them would be to integrate the local plans observations into Voxblox to repeat the process until the whole scene is covered.

B. Initial map

The reconnaissance flight captures top views of the scene to obtain a first approximation of the map quickly. However, the high altitude, together with the use of monocular cameras onboard the drones render the generation of this map challenging without the use of MVS expensive reconstruction methods. To compute it online, we use a depth completion system [21] onboard the drones that provides dense depth measurements from a sparse input, e.g., SLAM.

The depth measurements are integrated into a common voxel-based Truncated Signed Distance Field (TSDF) map, that incrementally builds a Euclidean Signed Distance Field (ESDF) map [22], $\mathcal{M}$. Voxels are organized in a uniform grid, where each voxel, $m \in \mathcal{M}$, contains a distance, d_m, to the closest surface and a weight, w_m, that contains the confidence about the depth measurement of that voxel. Moreover, we denote by $\mathbf{p}_m$ the centroid of the voxel and $\mathbf{n}_m$ its normal vector. Voxels that do not have any measurement have an associated weight equal to w_0.

The initial map is analysed in order to detect voxels that require additional observations. In particular, voxels that belong to a poorly observed surface, $\mathcal{M}_s$, and voxels without measurements (i.e., are unobserved), $\mathcal{M}_u$.

Surfaces are identified locating the voxels that satisfy

$$w_m > w_0 \text{ and } |d_m| < d_v \ , \tag{1}$$

where d_v is the voxel size.

Aligning the sensor's depth direction with the surface normal, as shown in Figure 4, is key in enabling accurate and high-quality scene reconstructions. With this in mind, we identify poorly observed surface voxels, $\mathcal{M}_s$, as

$$-\mathbf{o}_m \cdot \mathbf{n}_m > \cos(\theta_t) \ , \tag{2}$$

where $\mathbf{o}_m$ is the observation direction of the camera for the voxel and θ_t is the threshold angle to consider the observation of the surface valid. We consider $\theta_t = 45°$ as a good indication that the visibility of a surface is poor. During the initial flight, the cameras are looking downward (i.e., $-Z$ axis). Thus, vertical and oblique surfaces are considered poorly observed, while horizontal or low tilted surfaces are considered as correctly observed.

The second step is the analysis of the unobserved voxels. Out of all the unobserved voxels in the map, with weight equal to w_0, we find those that are accessible (i.e., can be observed). Unobserved voxels are accessible if they are

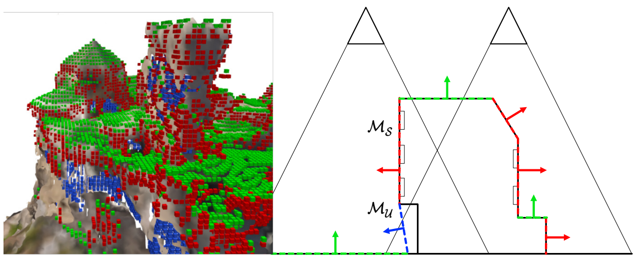


Fig. 4: The analysis the initial map, visualised from a side view on the right with two down-looking cameras, indicates the quality of the views of the scene seen on the left, obtained in the reconnaissance flight. Voxels on the left are visualised as dashed lines on the right, with arrows indicating the estimated surface normals. Red and green indicate poorly and well captured surfaces, respectively, while blue indicates accessible unknown areas, whose normals are estimated to point towards free space.

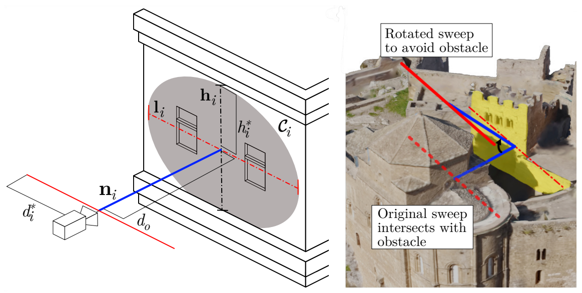


Fig. 5: Sweep definition and refinement scheme (right). The gray area represents a surface cluster. The dashed red line is the major eigen vector that will be covered by the sweep (red solid line). The green is the normal. An example in a real map is shown on the left where the observation direction, $\mathbf{n}_i$, was adjusted to avoid an obstacle.

surrounded by free space voxels, m_f, defined by

$$w_m > w_0 \text{ and } d_m > d_v. \tag{3}$$

The accessible unobserved voxels, $\mathcal{M}_u$, are then formalized as the voxels, such that

$$\exists m_f \in \mathcal{N}_{26}(m), \tag{4}$$

where $\mathcal{N}_{26}(m)$ is the set of 26-connected neighbors, around the voxel m. Finally, the set of voxels that need further observations is defined as

$$\mathcal{M}_t = \mathcal{M}_s \cup \mathcal{M}_u \ . \tag{5}$$

C. Perception-aware clustering

This step performs a novel perception-aware clustering over $\mathcal{M}_t$. In particular, voxels get grouped together, such that can be observed by a drone in a single efficient *sweep* trajectory by considering the distribution of their normals in the cluster. This clustering also aims at generating a natural partition of the scene into a set of *tasks* that can be assigned to a team of drones. In the following, we explain how the clustering works and how sweep paths are generated from them.

The proposed clustering is based on the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method [23]. The basic method groups voxels[1] that are closely together in space and identifies as noise isolated voxels in low density regions. It works by iteratively expanding clusters, $\mathcal{C}_i$, to neighboring voxels that fulfill the following *density* condition:

$$|\mathcal{N}_\sigma(\mathbf{p}_m)| > \epsilon \ , \tag{6}$$

where $|\mathcal{N}_\sigma(\mathbf{p}_m)|$ is the number of neighboring voxels in a radius σ of the voxel's center, $\mathbf{p}_m$, and ϵ is the minimum number of neighbors to include the voxel in that cluster.

Our goal is to group regions observable from a similar point of view (i.e., surfaces). Thus, we extend DBSCAN by adding a second condition for expansion. This condition checks if the normal of a candidate voxel, $\mathbf{n}_m$, lies within the distribution of normals in the cluster. The normals in $\mathcal{M}_s$ are estimated from the gradient of distances in the ESDF initial map. The normals of unobserved voxels are computed as the average of all the directions that lead from $\mathbf{p}_m$ to free space voxels in $\mathcal{N}_{26}(m)$ (Figure 4). We also smooth the estimated normals using neighboring values to filter noise.

[1]The original method refers to points.

In particular, we focus on the distribution of the cosine distance with respect to the mean normal of the cluster, $\mathbf{n}_c$,

$$d_\alpha(\mathbf{n}_m, \mathbf{n}_c) = 1 - \frac{\mathbf{n}_m \cdot \mathbf{n}_c}{\|\mathbf{n}_m\| \, \|\mathbf{n}_c\|}. \tag{7}$$

We then compute the average $\mu_d(\mathcal{C}_i)$, and standard deviation $\sigma_d(\mathcal{C}_i)$ of the distances from all the normals of the voxels in the cluster to $\mathbf{n}_c$. The *normal direction* condition checks that the distance of the normal between the candidate voxel and the cluster's distribution is sufficiently small,

$$d_\alpha(\mathbf{n}_m, \mathbf{n}_c) < \min(\mu_d(C_i) + 2\sigma_d(C_i), \tau). \tag{8}$$

where τ is a fixed value.

We identify $\mu_d(\mathcal{C}_i) + 2\sigma_d(\mathcal{C}_i)$ as the relative tolerance to the cluster's distribution and τ as the absolute tolerance. The aim of the relative tolerance is to adapt the expansion of the cluster to the surface in question, e.g., allowing soft curvatures. On the other hand, the absolute tolerance avoids the cluster to expand through discontinuities such as edges.

Finally, we perform a merging step that fuses small clusters with the most similar neighbor. If no neighbor is found, these voxels are discarded.

Considering that each voxel cluster resembles a surface, a *sweep* is defined as a linear trajectory that is orthogonal to the normal of the cluster (Figure 5). Among all the possible sweeps, we find the longest one through the inertia moments of the cluster, $\mathbf{l}_i$. Then, for each voxel in the cluster, we compute the longest distance from the center, projected on this axis,

$$d_i^* = \max_{m \in \mathcal{C}_i} \left| \mathbf{l}_i^T \left(\mathbf{p}_m - \bar{\mathbf{c}}_i \right) \right| \ , \tag{9}$$

where $\mathbf{p}_m$ is the centroid of the voxel and $\bar{\mathbf{c}}_i$ the centroid of the cluster. The extension of this distance from the centroid of the cluster in both directions of $\mathbf{l}_i$ generates the path that traverses the cluster through its length. We name both ends of this path, the *entrance* points of the cluster.

In order to guarantee that the whole surface is visible with a single sweep, we compute its height in the direction of the axis perpendicular to the sweep direction

$$\mathbf{h}_i = \mathbf{l}_i \times \mathbf{n}_i. \tag{10}$$

The value of the height is computed in the same way as (9) using the axis $\mathbf{h}_i$ instead:

$$h_i^* = \max_{m \in \mathcal{C}_i} \left| \mathbf{h}_i^T \left(\mathbf{p}_m - \bar{\mathbf{c}}_i \right) \right| \ , \tag{11}$$

where h_i^* if the half height of the cluster. Then, we use the relationship between the field of view (FoV) angle of the camera and h_i^* to compute the distance that is able to cover the height of the cluster. The observation distance, d_o, along the normal is computed as

$$d_o = \frac{h_i^*}{tan(\frac{\text{FoV}}{2})} \tag{12}$$

Finally, if the sweep intersects an obstacle we perform a rotation of the observation direction to refine it (Figure 5).

D. Global planner

In the next step, the objective is to compute high-level paths for the drones to cover all the clusters. We propose to solve this problem with an adaptation of the min-max Vehicle Routing Problem (VRP).

Originally, this algorithm looks for optimal routes for a set of agents, K, that visit once all the locations of a given set, V. Denote by c_{ij} the cost to go from location i to location j, which we consider is the same for all the agents, and define $\mathcal{X} = \{x_{ij}^k\}$, for $i,j \in V$, and $k \in K$, the set of binary variables that indicate whether agent k has traverse the route from i to j or not. Then, the min-max VRP solves

$$\min_{\mathcal{X}} \max_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \quad s.t. \tag{13a}$$

$$\sum_{k \in K} \sum_{i \in V} x_{ij}^k = 1 \quad \forall j \in V \setminus \{0\} \tag{13b}$$

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \quad \forall i \in V \setminus \{0\} \tag{13c}$$

$$\sum_{k \in K} \sum_{i \in V} x_{i0}^k = \sum_{j \in V} \sum_{k \in K} x_{0j}^k = |K| \tag{13d}$$

$$\sum_{i,j \in S} x_{i,j}^k \leq |S| - 1, \quad \forall S \subset V \setminus \{0\}, S \neq \emptyset \tag{13e}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i,j \in V \tag{13f}$$

where (13a) is the cost function, which denotes the largest cost among all the agents for a given assignment, constraints (13b) and (13c) indicate that drones only visit each location once. Constraints in (13d) impose the drones to start and end at the initial point. Constraints (13e) are the sub-tour elimination constraints. Finally, conditions (13f) impose binary conditions on the decision variables.

In order to adapt the VRP to the clusters and their *sweeps*, we propose a definition of the costs, c_{ij}, that considers them. Given two clusters, i and j, we compute the path between them, as the line that join their closest entrance points with distance, d_{ij}, if there are no obstacles. In case there are obstacles, we consider the same path, but flying over the top of the scene. This way we guarantee that all the clusters are reachable from each other, but we favour assignments of the nearby ones. Additionally, to account for the cost of covering each cluster, we add the distance of the sweep to all the costs with it as destination. The distance of the sweep generated for $\mathcal{C}_i$ is $l_i^* = 2d_i^*$, with d_i^* defined in (9). Therefore, the cost c_{ij} is defined as

$$c_{ij} = d_{ij} + l_j^*. \tag{14}$$

Lastly, to compute the solution of (13), we consider an implementation with limited capacities. We simplify the objective to minimize the total cost travelled by all the drones

$$\min_{\mathcal{X}} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \tag{15}$$

and we add a capacity constraint for each of them,

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k < c_{\max} \quad \forall k \in K. \tag{16}$$

Our solution searches for the minimum value of $c_{\max}^*$ that solves the problem using the bisection method.

E. Local planner

For the last step of the proposed pipeline, the local planner by Zhou et al. [24] is used to plan in two stages: an initial kinodynamic A* path search based on motion primitives finds a safe, feasible and minimum-time initial path, and a B-spline optimization generates smooth and collision-free trajectories that use gradient information from the ESDF and dynamic constraints.

In order to cover a surface efficiently and effectively, the sweep direction needs to be orthogonal to the observation vector. To enable safe and efficient navigation, while obtaining high quality scene observations, we decouple the problems of navigation and observation. We assume that the observation camera is mounted on an actuated gimbal, which is able to set the yaw and pitch directions. A second sensor, such as a laser ranger or a depth camera is used for navigation.

IV. Experiments and results

To assess the performance of the proposed method, the pipeline is run on photo-realistic outdoor scenarios of varying sizes and difficulty, namely on the *Bunker*, *Wood Bridge*, *Loarre Castle*, and *Zurich* models visible in Figure 8. The transfer of this simulation setup to real-world cases was proved in previous work [21] [25]. The Gazebo RotorS simulator is used with ground-truth odometry of the drones. During the initial map construction, flying at high altitude enables the use of accurate RTK GPS systems with small odometry error. The uncertainty in the successive flights can be alleviated by overestimating the observation distance and safety radius. As we target our application to consumer platforms, problems such as aerodynamics or other electrical and mechanical delays are assumed to be solved by their system. The drones are equipped with a monocular camera

TABLE I: Execution times to complete a scene coverage mission. The reconnaissance flight time, in parenthesis, is included in the total time. *Ours single* refers to our pipeline using one drone, while *Ours multi* indicates the time taken by the longest flight of any drone in a team (four in this case), indicating the end of the mission. For [2], we report the time to reach the same extent of coverage achieved by each of our methods (Table III). In lager maps, [2] is not able to achieve our coverage after one hour of execution and the total coverage by that time is reported. The '$\star$' indicates that the global planner only assigned two drones in this map, as introducing more would not reduce the total time.

Method	Bunker	Wood Bridge	Loarre Castle	Zurich
Kompis et al. [2]	859.71 s	897.08 s	>3600 s [67.08%]	>3600 s [13.91%]
Ours single	491.91 s (183.06 s)	331.16 s (122.54 s)	1474.95 s (329.88 s)	2027.88 s (588.56 s)
Kompis et al. multi [2]	214.43 s	405.98 s	1440.59 s	>3600 s [60.49%]
Ours multi	126.26 s (42.04 s)	172.09* s (53.31 s)	433.74 s (117.11 s)	741.07 s (269.53 s)

mounted on an actuated gimbal that can rotate independently of the orientation of the drone. Its resolution is 752×480 and FoV is $80° \times 55°$. The drones' linear and angular maximum velocity and acceleration are set to 2 $\mathrm{m\,s^{-1}}$ and 0.9 $\mathrm{m\,s^{-2}}$, respectively, for fairness with the compared system and to ensure safety at all times. During the reconnaissance flight, the drones fly at a fixed height over the model in a grid pattern with their cameras looking downward. The voxel size used for the initial map and planning is 0.2 (*Bunker* and *Wood Bridge*), 0.5 (*Loarre*) and 0.7 (*Zurich*).

The parameters for the clustering step depend on the resolution of the prior map (i.e., voxel size v_s). We set $\epsilon = 10v_s$, $\sigma = 6$ (Eq. (6)) and $\tau = 0.4$. We also apply an inflation factor over the coarse map of $20v_s$ to the observation and safety distance for the sweep generation. Due to computational resources required to simulate several drones, the local paths are executed by a single drone sequentially, which starts from and comes back to the same initial point. The simulation runs until all the local trajectories have been executed.

We run the experiments considering three different algorithms. We name *Ours single* and *Ours multi* the solutions obtained running our pipeline with one and four drones respectively. In the *multi* version, we perform an ablation study to show the difference in visual coverage obtained after the reconnaissance flight and the successive flights resulting from our pipeline. Even when our pipeline is not directly comparable in terms of the sensor setup with other exploration methods that use stereo pairs, the third method uses the planning approach of Kompis et al. [2] for a single and multiple drones, which is among the state-of-the-art planners with available implementation. In the version with multiple drones, the environment was segmented equally among the drones. This comparison is not intended to rank the two methods but to showcase the potential advantages of the proposed planning approach in terms of efficiency.

A. Planning efficiency

The times for the execution of the plan are shown in Table I. The time for the reconnaissance flight and initial map construction with Voxblox is included in the total and shown below. For the method of Kompis et al., we report the times necessary to achieve the same coverage as our system.

The results for *Ours single* and *Ours multi* validate that our setup can generalize to an arbitrary number of drones. When using several drones instead of one, times are a fraction of the number of drones with little overhead. In the case of *Wood bridge*, the global planner assigned the tasks to only two drones even if four were available. Due to the scene structure, adding more drones would not reduce the time of the mission as drones would have to return to the initial point. Compared to Kompis et al., our method is able to completely cover the maps faster in every case. For large maps (i.e., *Loarre Castle* and *Zurich*), [2] is not able to cover the environment after one hour of execution and we report the amount of coverage obtained at that time.

There are two main reasons for this difference. Firstly, the different approach on drone dynamics in the planners. Stop-and-go motions are necessary as the exploration process is incremental. This limits the planning horizon of the system to a local region. In their approach, the drone has to stop in order to acquire each individual view and plan the next (see Figure 6). In our case, the drone is able to keep moving while observing a whole surface in a sweep. Notice that our system could potentially use higher velocities and accelerations for large trajectories in free-space, as in the case of the reconnaissance flight. This would further improve the planning efficiency. The second reason is that their planner revisits areas in order to obtain thorough coverage, committing resources to small regions with difficult accessibility. The reason for their low coverage result in the *Zurich* map is explained by their viewpoint proposal method, which leads to larger re-planning times when the scale of the map grows.

We also report the time for the initial map processing. The time for the analysis and clustering steps depend on the size and resolution of the map. The global planning step depends on the number of generated clusters and number of agents. We show the results in Table II in the smallest and biggest maps: *Bunker* and *Zurich*. The time is always below one minute which is negligible for the total time of the mission.

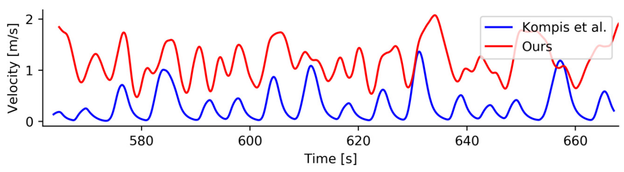


Fig. 6: Extract of the moving average for the velocity during the simulation in *Loarre Castle* for *Ours single* and [2]. While traditional methods stop to capture a view and plan the next goal, our method is able to keep flying at higher speed.

B. Coverage and surface quality

Besides the efficiency of our planner, we have also assessed that the coverage and the quality of the views are correct. The images captured from the monocular cameras of the drones have been used to generate 3D reconstructions of the scenes using COLMAP. The reconstructed models are compared with the ground-truth (GT) virtual models.

TABLE II: Computation time for the different map processing steps: analysis, clustering and global planner. Mean and standard deviation for 10 runs.

		Bunker	Zurich
Analysis	-	2.22 ± 0.082 s	6.96 ± 1.05 s
Clustering	-	1.83 ± 0.045 s	7.12 ± 0.297 s
Global planner	Single	0.99 ± 0.053 s	15.43 ± 0.689 s
	Multi	0.94 ± 0.064 s	17.10 ± 0.737 s

TABLE III: For each method, we report the RMSE of the reconstructions, the extent of the coverage for a threshold of 0.1 meters at the completion time of the experiment as reported in Table I. Recon. indicates the metrics from a reconstruction with only the reconnaissance flight.

Method	Bunker	Wood Bridge	Loarre Castle	Zurich
Kompis et al. [2]	0.085 m 75.6 %	0.068 m 55.63 %	0.049 m 42.58 %	0.074 m 9.13 %
Ours single	0.027 m 97.35 %	0.043 m 93.23 %	0.048 m 97.92 %	0.09 m 95.96 %
Kompis et al. multi [2]	0.076 m 89.10 %	0.074 m 60.29 %	0.059 m 50.20 %	0.087 m 20.90 %
Recon. (ablation)	0.04 m 84.75 %	0.039 m 62.93 %	0.063 m 88.54 %	0.147 m 75.34 %
Ours multi	0.026 m 96.36 %	0.039 m 92.37 %	0.043 m 98.64 %	0.086 m 97.34 %

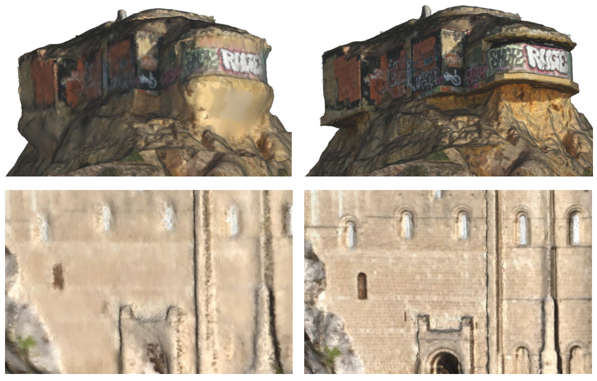


Fig. 7: Comparison of the coverage quality after the reconnaissance flight (left) and the successive flights (right). Occluded regions under the *Bunker* are not reconstructed (up). In addition, even though vertical surfaces such as *Loarre's* walls are covered, their observation yields poor scene reconstructions (down).

We consider that a point in the GT surface was covered if the closest distance to a point from the reconstructed mesh is below a threshold of 0.1m. Our metric is the percentage of covered points in the ground-truth mesh. We also measure the accuracy of the reconstruction as the RMSE of the distances from the reconstructed model to the ground truth mesh. While we are mainly interested in the first two metrics, the accuracy proves that our method can be used to obtain accurate 3D reconstructions of the environment. We also report the coverage, its quality and the reconstruction accuracy from the Voxblox generated mesh of the pipeline in [2]. The voxel size for their reconstruction is the same they use in their experiments, 0.1, which is the threshold used for considering a point covered in our setup. The results are reported in Table III. For Kompis et al., the reported value is the coverage achieved by the completion time of our plan.

The reconnaissance flight (*Recon.*) is able to cover large amount of surfaces. However, the coverage quality is low, yielding poor scene reconstructions (Figure 7). After the execution of our pipeline, we obtain images that ensure good observation of surfaces. We can see similar coverage for the case of single and multi-drone approaches as the drones traverse similar sweeps. Compared to Kompis et al., our system is able to achieve more coverage in less time. Notice how the coverage difference is increased with the size of the map. Qualitative results are shown in Figure 8 for all the maps. It might be seen that our pipeline misses some areas with difficult accessibility. In return, it is able to cover the overall scene in a fraction of the time. This demonstrates that lot of information can be extracted from the map by planning more efficiently and shows the advantage of using prior knowledge about the scene structure for planning.

V. Conclusion

In order to improve the efficiency in large-scale deployments of drones for visual coverage, this article proposes a multi-stage planner that generates long linear trajectories (*sweeps*) that observe a large amount of surface in a continuous motion. We accomplish this by leveraging a prior coarse map to cluster these surfaces and improve the posterior coverage trajectories. This approach is generalised to an arbitrary number of drones, managing the workload distribution between them in order to minimize the completion time of the mission. Comparison with alternative approaches to exploration of scenes show the advantages of our pipeline for large scenarios, where the overall coverage of the scene in a minimal amount of time is necessary. We show that a single run of our pipeline is able to obtain coverage of scenes faster and with great accuracy.

Future work will explore the integration of the proposed pipeline in a real platform, including a mapping framework to ensure safe local navigation and additional coordination systems to deploy a team of autonomous drones in large-scale environments. Besides, exploring the extension to a team of heterogeneous aerial drones (i.e., fixed-wing UAVs for the nadir flight) could improve even further the efficiency of the system by allocating each to different task modalities.

References

[1] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2021.

[2] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed sampling exploration path planner for 3d reconstruction of large scenes," *IEEE Robotics and Automation Letters*, vol. 6, 2021.

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, 1997.

[4] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *International Conference on Intelligent Robots and Systems*, 2017.

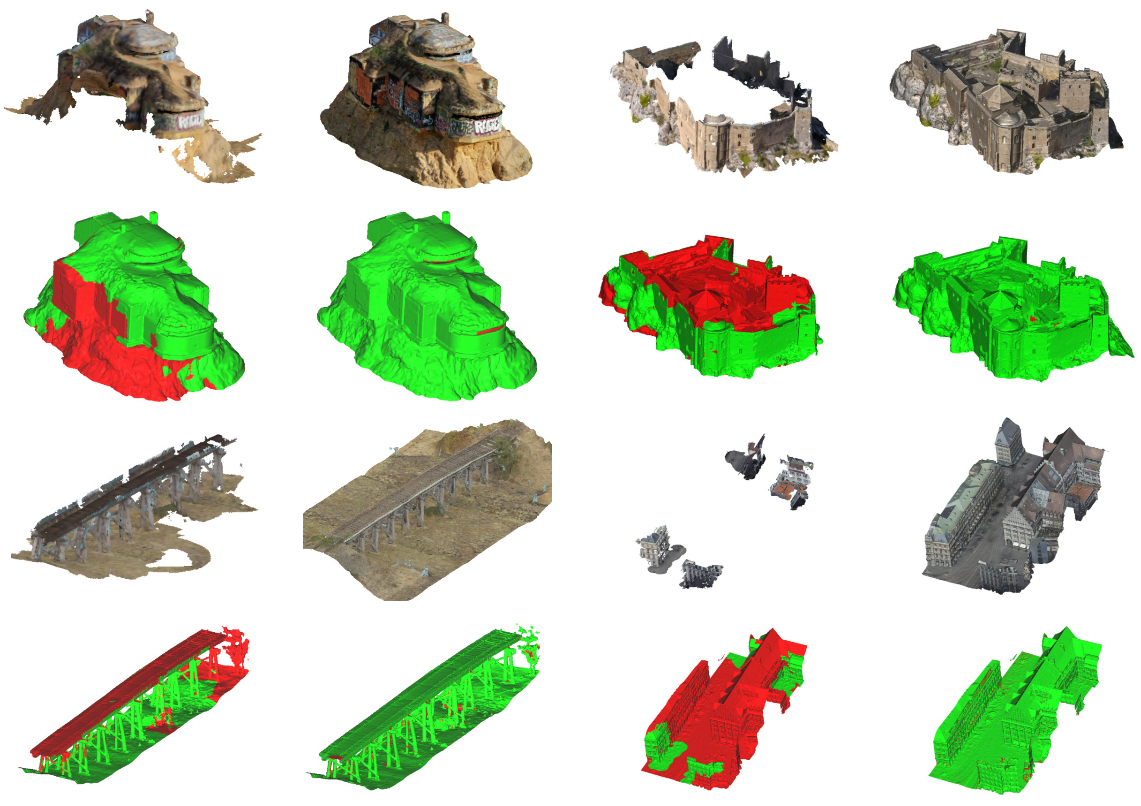


Fig. 8: Qualitative comparison of the coverage obtained for all the maps considering a fixed time. In green are points in the ground truth (GT) mesh that have been covered during the mission, while red indicates the opposite. Our planner is able to obtain more coverage of the overall scene, despite missing some some details in inaccessible/non-directly visible surfaces. Detailed numbers of the coverage and accuracy of the reconstructions are provided in Table III.

[5] D. Duberg and P. Jensfelt, "Ufoexplorer: Fast and scalable sampling-based exploration with a graph-based planning structure," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 2022.

[6] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active slam," in *IEEE Int. Conf. on Robotics and Automation*, 2012.

[7] L. Carlone, J. Du, M. Kaouk Ng, B. Bona, and M. Indri, "Active slam and exploration with particle filters using kullback-leibler divergence," *Journal of Intelligent & Robotic Systems*, vol. 75, no. 2, 2014.

[8] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *IEEE Int. Conf. on Learning Representations (ICLR)*, 2020.

[9] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, and K. Daniilidis, "Uncertainty-driven planner for exploration and navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2022.

[10] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.

[11] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *IEEE Int. Conf. on Robotics and Automation*, 2016.

[12] G. Hardouin, F. Morbidi, J. Moras, J. Marzat, and E. M. Mouaddib, "Surface-driven Next-Best-View planning for exploration of large-scale 3D environments," in *IFAC World Congress*, July 2020.

[13] S. Song, D. Kim, and S. Choi, "View path planning via online multiview stereo for 3-d modeling of large-scale structures," *IEEE Transactions on Robotics*, vol. 38, no. 1, 2022.

[14] 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 *Int. Conf. on Computer Vision*, 2017.

[15] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich, "Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark," *ACM Trans. Graph.*, vol. 37, no. 6, dec 2018.

[16] B. Hepp, M. Nießner, and O. Hilliges, "Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction," *ACM Trans. Graph.*, vol. 38, no. 1, dec 2018.

[17] A. Mannucci, S. Nardi, and L. Pallottino, "Autonomous 3d exploration of large areas: A cooperative frontier-based approach," in *Modelling and Simulation for Autonomous Systems*, J. Mazal, Ed., Cham, 2018.

[18] R. G. Colares and L. Chaimowicz, "The next frontier: Combining information gain and distance cost for decentralized multi-robot exploration," in *ACM Symposium on Applied Computing*, 2016.

[19] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. M. Mouaddib, "Next-best-view planning for surface reconstruction of large-scale 3d environments with multiple uavs," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2020.

[20] A. Dutta, A. Bhattacharya, O. P. Kreidl, A. Ghosh, and P. Dasgupta, "Multi-robot informative path planning in unknown environments through continuous region partitioning," *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, 2020.

[21] L. Teixeira, M. R. Oswald, M. Pollefeys, and M. Chli, "Aerial single-view depth completion with image-guided uncertainty estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.

[22] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *Int. Conf. on Intelligent Robots and Systems*, 2017.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Int. Conf. on Knowledge Discovery and Data Mining*, 1996.

[24] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, 2019.

[25] F. Maffra, L. Teixeira, Z. Chen, and M. Chli, "Real-time wide-baseline place recognition using depth completion," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, 2019.